Analysis of Approximation and Uncertainty in Optimization

by

Dana Andrew Mastin

B.S., Electrical Engineering, University of Arizona, 2007
S.M., Elect. Eng. and Comp. Sci., Massachusetts Institute of Technology, 2009

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology

February 2015

Signature of Author: ______________________________

Department of Electrical Engineering and Computer Science
January 16, 2015

Certified by: ______________________________

Patrick Jaillet
Dugald C. Jackson Professor of Electrical Engineering and Computer Science
Co-Director, Operations Research Center
Thesis Supervisor

Accepted by: ______________________________

Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Committee on Graduate Students

Analysis of Approximation and Uncertainty in Optimization

by Dana Andrew Mastin

Submitted to the Department of Electrical Engineering and Computer Science
on January 16, 2015 in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Electrical Engineering and Computer Science

Abstract

We study a series of topics involving approximation algorithms and the presence of uncertain data in optimization. On the first theme of approximation, we derive performance bounds for rollout algorithms. Interpreted as an approximate dynamic programming algorithm, a rollout algorithm estimates the value-to-go at each decision stage by simulating future events while following a heuristic policy, referred to as the base policy. We provide a probabilistic analysis of knapsack problems, proving that rollout algorithms perform significantly better than their base policies.

Next, we study the average performance of greedy algorithms for online matching on random graphs. In online matching problems, vertices arrive sequentially and reveal their neighboring edges. Vertices may be matched upon arrival and matches are irrevocable. We determine asymptotic matching sizes obtained by a variety of greedy algorithms on random graphs, both for bipartite and non-bipartite graphs.

Moving to the second theme of uncertainty, we analyze losses resulting from uncertain transition probabilities in Markov decision processes. We assume that policies are computed using exact dynamic programming with estimated transition probabilities, but the system evolves according to different, true transition probabilities. Given a bound on the total variation error of estimated transition probability distributions, we derive a general tight upper bound on the loss of expected total reward.

Finally, we consider a randomized model for minmax regret in combinatorial optimization under cost uncertainty. This problem can be viewed as a zero-sum game played between an optimizing player and an adversary, where the optimizing player selects a solution and the adversary selects costs with the intention of maximizing the regret of the player. We analyze a model where the optimizing player selects a probability distribution over solutions and the adversary selects costs with knowledge of the player's distribution. We show that under this randomized model, the minmax regret version of any polynomial solvable combinatorial problem is polynomial solvable, both for interval and discrete scenario representations of uncertainty.

Thesis Supervisor: Patrick Jaillet
Title: Dugald C. Jackson Professor of Electrical Engineering and Computer Science

Acknowledgements

My graduate school experience has been rewarding in large part due to my interactions with excellent people. It has been a true pleasure to work with my advisor, Patrick Jaillet. Patrick has been incredibly attentive and supportive of my research interests, and I am continually inspired by his professionalism, rigor, and kindness. This thesis simply would not have been possible without his guidance.

I was lucky to have a similarly outstanding thesis committee. Dimitri Bertsekas and Costis Daskalakis have greatly influenced me through their research, teaching, and advice. In particular, Dimitri helped form my interest in rollout algorithms, and Costis offered many interesting ideas on game theory. Joining my committee more recently, Peter Chin has been a source of helpful feedback and enjoyable discussions.

I am thankful for illuminating discussions with Warren Powell, who encouraged me to look at error bounds in dynamic programming. I also thank John Fisher III and Jeremy Kepner for their general advice early in my graduate studies, which has repeatedly proved useful.

Being a part of Patrick's group and LIDS has afforded me the opportunity to collaborate with a wonderful group of fellow students. I am especially grateful to Xin Lu, with whom I had the privilege of sharing an office. Xin was an endless source of helpful comments on research and offered perspectives on problems that I never would have considered. I also had many fun and useful interactions with Swati Gupta, Dawsen Hwang, Maokai Lin, and Vahideh Manshadi.

Many other individuals have been helpful in one way or another. I thank Mark Abramson, Luca Bertazzi, Alan Frieze, Steve Kolitz, Gabor Lippner, Chin Hon Tan, and Nick Wormald for feedback and advice. The administrative staff at LIDS, EECS, and Draper has been very supportive; I am grateful to Rachel Cohen, Gail Dourian, Janet Fischer, and Brian Jones.

My graduate school years have been greatly enriched by relationships outside of school. I am thankful to Rachel Ernst, Tara Grillos, Maggie Jarmolowski, Kirsten Obermiller, Ian Slattery, Eveleen Sung, and many others. Finally, I am grateful to my family: my parents, Gary and Jenny, and my sister, Alisa. They have offered steadfast support in as many ways as possible.

This work has been supported by an NDSEG Fellowship with the Office of Naval Research, an NSF Graduate Research Fellowship, NSF grant 1029603, ONR grant N00014-12-1-0033, AFOSR grant FA9550-12-1-0136, and NASA ESTO's AIST program under grant number NNX12H81G.

Contents

List of Figures

List of Algorithms

List of Tables

Chapter 1

Introduction

OPTIMIZATION is a fundamental topic in computer science, mathematics, and operations research. With the development of a formal theory of computation, the last century has witnessed the emergence of what are now considered basic paradigms of mathematical optimization. Algorithms drawn from the theory of linear and integer programming, combinatorial optimization, and dynamic programming, for instance, have proved invaluable for solving a wide variety of optimization problems. Consider the conventional 0–1 knapsack problem: given a set of items with values and weights, how does one select a subset of items that maximizes the total value without exceeding a total weight limit? It is well known that this can be solved with a simple dynamic programming recursion [34]. Also consider the maximum bipartite matching problem: given a bipartite graph with a set of job vertices, a set of worker vertices, and a set of edges indicating compatible workers and jobs, how does one maximize the number of jobs completed if each worker can complete one job and jobs cannot be split among workers? This can be solved with the Hopcroft-Karp algorithm or, alternatively, formulated as a linear programming problem and solved with the simplex method [76, 128].

When faced with a particular optimization problem, one may find that it can be solved in practice using standard approaches; for example, it is a linear programming problem, or an instance of a well-known combinatorial problem. Indeed, standard approaches give a recipe for obtaining a provably optimal solution. In many applications, however, it is necessary to accept a suboptimal solution – that is, a solution given by an approximation algorithm. The most obvious motivation for approximation algorithms is computational complexity. It is unlikely that polynomial-time algorithms will be found for NP-hard problems [69, 112]. When faced with large instances of NP-hard problems, one must resort to suboptimal techniques where efficient algorithms are available. Space complexity poses similar limitations. For processing large data with limited memory, as modeled by streaming models of computation, approximation is often needed [11, 124].

Approximation algorithms are motivated outside of computational complexity. In online optimization problems, decisions must be made as the problem is revealed, and these decisions are irrevocable [42]. This makes it impossible for an online algorithm to guarantee an optimal solution (though an algorithm can be optimal based on an online criterion). Online optimization is well studied, and any online algorithm can in fact be interpreted as an approximation algorithm for the offline (i.e. not online) problem.

Consider the following variation on the maximum bipartite matching problem. Assume that jobs are not known upfront but arrive online such that the edges for each job are revealed when it arrives. Each job must immediately be assigned to an available worker or dropped. An algorithm for this online problem is equivalent to an algorithm for the offline problem that performs a single pass through job vertices and makes assignment decisions.

Algorithm implementation complexity is yet another motivation for approximation algorithms. Greedy algorithms, which are the most basic type of approximation algorithm, make locally optimal decisions and are often easier to implement than optimal algorithms. A natural greedy algorithm for the 0–1 knapsack problem, for example, is to sort the items in non-increasing profit to weight ratio and add each item (if possible) in this order [89]. In addition to being intuitively pleasing, this algorithm is nearly trivial to implement and runs in polynomial time. On more elaborate problems, implementation complexity becomes increasingly important when choosing an algorithm.

An immediate question regarding approximation algorithms is, what can be proved about the quality of the approximation? Specifically, can an approximation algorithm guarantee a solution with value that has some bounded deviation from the optimal value? The first theme of this thesis is to answer this question in a variety of novel scenarios. Approximation algorithms are often measured from the perspective of worst-case analysis [143, 146]. For instance, a variation on the greedy algorithm that we have described for the 0–1 knapsack problem (the decreasing density greedy algorithm) guarantees a solution with value at least half as large as the optimal value [69]. Our tool of choice will instead be average-case analysis. We will assume that the problem structure and/or parameters is generated by some probabilistic process, and we will look at expected performance. There are merits to both types of analysis. Worst-case analysis may be more appropriate for high risk situations, while average-case analysis can give a more realistic depiction of typical performance. Ultimately, it is desirable to understand algorithm performance from both perspectives, and our contributions provide new average-case results where the analogous worst-case results are already known.

Our first topic on the theme of approximation is rollout algorithms. Rollout algorithms are a general class of approximation algorithms that can be used on a problem admitting a dynamic programming formulation [28, 30]. They combine lookahead with a greedy algorithm to repeatedly estimate optimal decisions. In practice, rollout algorithms have worked well on a variety of problems, but there are few theoretical results guaranteeing the quality of the solutions they generate [98, 129, 141]. There are, however, some worst-case results known for rollout algorithms on the 0–1 knapsack problem [26, 125]. Complementing this research, we look at the average-case performance of rollout algorithms for both the 0–1 knapsack problem and the subset sum problem. We give a probabilistic analysis of two types of rollout algorithms for these problems, showing strong performance bounds.

Our second topic on approximation is greedy online matching on random graphs. In online matching problems, the graph is revealed over time, and matches must be made as the graph appears. Greedy matching algorithms have been studied extensively in the offline setting (i.e. when the entire graph is known) using both worst-case analysis and average-case analysis [55, 56, 82]. Matching algorithms have also been studied in the online setting, but mainly using worst-case analysis [2, 83]. We thus consider greedy online matching from the average-case perspective, using random graphs. We determine the asymptotic matching sizes obtained by a variety of greedy algorithms on both Erdős-Rényi random graphs and random regular graphs, and for bipartite and non-bipartite models.

Beyond approximation, the second theme of this thesis addresses another practical aspect of optimization. Most standard optimization approaches assume that problem data are known exactly. Unsurprisingly, applications frequently arise where data/parameters are uncertain. Uncertainty can be characterized in a variety of ways: parameters can be modeled as being drawn from a probability distribution, contained in an interval (i.e. constrained by known lower and upper bounds), or selected from a finite set of alternatives. Existence of uncertainty can complicate optimization dramatically. In the knapsack problem, for example, uncertainty in item weights makes it difficult to ensure feasibility of potential solutions, and uncertainty in item values directly undermines the objective of finding the most valuable subset.

Yet, the presence of uncertainty may not necessarily require the use of new or sophisticated algorithms, particularly if solution properties for a given problem and algorithm do not depend heavily on the uncertain parameters. This motivates the general study of sensitivity analysis in optimization. Sensitivity analysis seeks to determine how significantly solutions or objective values change as problem parameters are perturbed. While these properties can be analyzed in specific instances using simulation, more general results can be gained with theoretical analysis. There are simple results in linear programming, for example, demonstrating how sensitive objective values are to changes in constraints and objective coefficients [33].

When parameter uncertainties are formidable, it is necessary to ask how to optimize while intelligently accounting for uncertainty. This is addressed with the fields of stochastic programming and robust optimization [31, 37, 93, 131]. Stochastic programming is mainly dedicated to settings where uncertain parameters are modeled as random variables with known probability distributions. The prototypical stochastic programming problem is the two-stage stochastic program with recourse: decisions are made in the first stage to maximize the current value and expected future value, and corrective actions are taken in the second stage after uncertain data are realized and observed. Robust optimization, on the other hand, usually only considers a single optimization stage, and it models uncertainty by describing sets of possibilities for uncertain parameters. This naturally leads to the use of worst-case robust objectives, such as the minmax criterion.

Optimizing in the presence of uncertainty is the second theme of this thesis, and we start with a topic in sensitivity analysis. We consider the presence of uncertain transition probabilities in a Markov decision process. The Markov decision process (MDP) is a standard framework for dynamic optimization, where optimization occurs over a sequence of alternating decisions and random outcomes [27, 123]. We assume that the MDP is solved using dynamic programming, but with estimated transition probabilities. Given a bound on the total variation error for estimated transition probabilities, we determine a general tight upper bound on the loss of expected total reward.

We then move to a topic on robust optimization, combinatorial optimization with uncertainty in cost coefficients. This problem can be viewed as a game played between an optimizing player, who selects a solution from a solution set, and an adversary, who selects costs from an uncertainty set. The two natural choices of objective criteria for the optimizing player are minmax and minmax regret [127, 145]. Problems under both criteria have been studied extensively, but mainly using a deterministic model. We consider a randomized model under the minmax regret criterion, where the optimizing player selects a probability distribution over solutions, and the adversary selects a probability distribution over costs. We show that minmax regret problems are computationally easier to solve in this randomized model than in the deterministic model.

1.1 Organization and Summary of Contributions

We give a topic-by-topic discussion of our results and contributions as follows. The thesis is divided into four main chapters. On the theme of approximation, Chapters 2 and 3 present our results for rollout algorithms and greedy online matching. For optimization under uncertainty, Chapters 4 and 5 give our results for uncertainty in Markov decision processes and robust optimization. A conclusion is given in Chapter 6.

1.1.1 Chapter 2: Rollout Algorithms for Knapsack Problems

Rollout algorithms were introduced by Tesauro and Galperin [140] as online Monte-Carlo search techniques for computer backgammon. Their application to combinatorial optimization and stochastic dynamic programming was formalized by Bertsekas, Tsitsiklis, and Wu [30], and Bertsekas and Castañon [28], respectively. The motivation for the rollout approach comes from problems that can be solved using classical dynamic programming, but for which determining the value function (or value-to-go function) is computationally infeasible. A rollout algorithm estimates the value function at each decision stage by simulating future events while following a heuristic policy, referred to as the base policy. In most cases, the rollout algorithm is guaranteed to perform at least as well as its base policy, but there have been very few results proving a strict improvement in performance. Nevertheless, rollout algorithms have demonstrated excellent computational performance on a wide variety of problems including vehicle routing, fault detection, and sensor scheduling [98, 129, 141].

Fortunately, some results showing a strict improvement in worst-case performance are known for the 0–1 knapsack problem, where the decreasing density greedy (DDG) algorithm is used as a base policy [26, 125]. The DDG algorithm takes the best of two solutions: the one obtained by adding items in order of non-increasing profit to weight ratio, as long as they fit, and the solution resulting from adding only the feasible item with highest profit. The first iteration of the rollout algorithm in this setting tries adding every item to the knapsack and using the DDG algorithm with the remaining items and capacity, and then takes the best of these solutions. This first iteration is equivalent to performing partial enumeration, a general technique for obtaining polynomial time approximation schemes [89]. It can be shown that running a single iteration of the rollout algorithm improves the approximation guarantee from $1/2$ (the bound provided by the base policy) to $2/3$. However, there is a tight example showing that it is not possible to guarantee improved performance with additional iterations [26].

This work motivates a complementary study of average-case performance of rollout algorithms for knapsack problems, which we provide in this chapter [105]. We use a stochastic model that was introduced by Borgwardt and Tremel [41] to study greedy algorithms for the subset sum problem. We analyze two rollout methods that we refer to as the exhaustive rollout and consecutive rollout, both of which employ a simple greedy base policy. Using a graphical interpretation of solutions generated by the algorithms, we prove that both methods yield a significant improvement in expected performance after a single iteration of the rollout algorithm relative to the base policy. The bounds are meaningful for a small number of items, and they also indicate the asymptotic behavior of the algorithms as the number of items approaches infinity.

We also consider rollout algorithms on a much simpler problem, that of finding a shortest path in a binary decision tree. We show that a rollout algorithm has interesting behavior on this problem. Namely, when running every iteration of the rollout algorithm (rather than just the first), average-case performance is strong, but worst-case performance is arbitrarily bad. We believe that this property may hold for rollout algorithms on other problems.

1.1.2 Chapter 3: Greedy Online Matching on Random Graphs

The online bipartite matching problem was introduced in the seminal paper of Karp, Vazirani, and Vazirani [83]. In this problem, we are given a bipartite graph where vertices in one partition, corresponding to bins, are given up front, and vertices in the other partition, corresponding to balls, arrive online. (In our introduction above, we described the bin vertices as workers and the ball vertices as arriving jobs.) When each ball arrives, its neighboring edges are revealed, and it must immediately be either matched with an unmatched neighboring bin or dropped (left unmatched). Each bin may be matched to at most one ball, and decisions are irrevocable. The goal is to maximize the number of matched balls.

From a worst-case perspective, it is well known that the greedy algorithm, which matches each ball to a random unmatched neighboring bin (if possible), always achieves a matching size at least as large as $1/2$ the size of the maximum matching. Karp, Vazirani, and Vazirani introduced the Ranking algorithm, which picks a random permutation of bins up front and matches each ball to its unmatched neighboring bin that is ranked highest in the permutation [83]. They showed that this algorithm guarantees a matching size at least $1 - 1/e \approx 0.632$ of the size of the maximum matching in expectation, and that this is the best possible worst-case guarantee.

In terms of average-case analysis, greedy matching algorithms have been studied widely, but mainly for the basic offline matching problem. Accordingly, in this chapter, we study a series of greedy online matching algorithms on Erdős-Rényi graphs and random regular graphs [104]. We use the differential equation method, pioneered by Kurtz [97] and Wormald [148], as well as other probabilistic arguments to derive expressions for the asymptotic matching sizes produced by various algorithms. In addition to bipartite graphs, we define a model for online matching on non-bipartite graphs, and we consider the average performance of greedy algorithms in this setting.

On the Erdős-Rényi binomial graph $\mathcal{G}(n, n, p)$ (the random graph with n vertices in each partition and edges occurring independently with probability p), we show that for all monotonic functions $p = p(n)$, the greedy algorithm has a performance ratio of at least 0.837. The performance ratio here is defined asymptotically as the ratio of the expected matching size given by the algorithm to the expected maximum matching size. We show that under the $\mathcal{G}(n, n, p)$ model, the performance of the greedy algorithm is equivalent to the Ranking algorithm, so our results indicate that the Ranking algorithm has a performance ratio of at least 0.837. For random 2-regular graphs, we show that the performance ratio for greedy is equal to 0.877. We present similar results for non-bipartite graphs, showing that the greedy algorithm achieves a performance ratio of at least 0.837 on $\mathcal{G}(n, p)$ and has performance ratio of at least 0.869 on random 2-regular graphs. We also analyze a series of other algorithms, including a vertex-weighted matching algorithm and a rollout algorithm on random bipartite graphs.

1.1.3 Chapter 4: Uncertain Transition Probabilities in Markov Decision Processes

A general T-stage Markov decision process is defined by a sequence of stages where, for each stage, there is a set of states, a transition probability function mapping states, decisions, and future states to probabilities, and a reward function mapping states and decisions to rewards [27, 123]. The goal (most commonly) is to find a decision rule, or policy, that maximizes total expected reward. Using dynamic programming, MDPs are solved by stepping backward in time, determining optimal expected future values of available decisions and storing them as value functions.

The topic of uncertainty in MDPs has been fairly well studied, in part due to motivations in approximate dynamic programming. The classical result of Singh and Yee [134] bounds losses incurred by uncertain value functions, and their proof also generalizes to situations with uncertain rewards. For uncertain transition probabilities, various robust frameworks have been developed, such as the Markov decision process with imprecise transition probabilities [78] and the bounded-parameter Markov decision process [71].

Our work on uncertain transition probabilities in MDPs bounds losses that are incurred when conventional dynamic programming is used [103]. We consider a nonstationary T-stage MDP with nonnegative rewards. We assume that policies are computed using exact dynamic programming with the estimated transition probabilities, but that the system evolves according to different, true transition probabilities. Given a bound on the total variation error of estimated transition probability distributions, we derive a general upper bound on the loss of expected total reward. Our proof approach is to analyze the growth of errors incurred by stepping backward in time while computing value functions; this requires bounding a multilinear program. We show that the loss bound is tight and that it generalizes to discounted infinite horizon models.

1.1.4 Chapter 5: Randomized Minmax Regret for Combinatorial Optimization

Robust frameworks have been developed for various types of optimization, including linear programming, convex optimization, and discrete optimization [24, 25, 31]. There has been a particular emphasis on the study of combinatorial optimization problems with uncertainty in cost coefficients [84, 93]. This has been studied under two types of uncertainty sets: discrete scenario uncertainty and interval uncertainty. In discrete scenario uncertainty (see, e.g. the book by Kouvelis and Yu [93]), an entire list of cost coefficients is given for each scenario, and a single scenario is assumed to be true. For interval uncertainty (e.g. Kasperski [84]), cost coefficients independently lie within known lower and upper bounds.

For any type of uncertainty set, the two basic choices of robust criteria are minmax and minmax regret. Under the minmax criterion, the goal (assuming a minimization problem) is to select a solution that gives the best upper bound on objective cost over all possible costs from the uncertainty set. For the minmax regret criterion, the goal is instead to select the solution that minimizes the maximum possible regret, defined as the difference between the cost of the selected solution and the optimal solution.

Our interest lies in the minmax regret criterion for combinatorial optimization under cost uncertainty, which can be viewed as a two-stage game played between an optimizing player and an adversary. In the first stage, the optimizing player selects a deterministic solution. In the second stage, the adversary observes the selected solution and chooses costs from the uncertainty set with the intention of maximizing the optimizing player's regret. The goal of the optimizing player is thus to select a solution that least allows the adversary to generate regret. For both interval and discrete scenario representations of uncertainty, the minmax regret versions of most polynomial solvable problems are NP-hard [8]. A variation on this model, first suggested by Bertsimas et al. [35] for the minmax criterion, is to allow the optimizing player to select a probability distribution over solutions and require the adversary to select costs based on knowledge of the player's distribution, but not its realization. They showed that this randomized model makes some robust minmax problems polynomial solvable that are otherwise NP-hard in the deterministic model [35].

We consider this randomized model under the minmax regret criterion [106]. Remarkably, we show that the minmax regret version of any polynomial solvable combinatorial problem is polynomial solvable. This holds true for both interval and discrete scenario representations of uncertainty. Our algorithm is based on solving two linear programs, one to find marginal probabilities of selecting elements, and another to map marginal probabilities to a mixed strategy. Using the randomized model, we show new proofs of existing approximation algorithms for the deterministic model based on primal-dual approaches. We also determine integrality gaps of minmax regret formulations, giving tight lower bounds on performance gains from randomization. Finally, we prove that minmax regret problems are NP-hard under general convex uncertainty.

1.1.5 Chapter 6: Conclusion

In this chapter, we review our results and methods. We highlight the most important contributions of our work and enumerate key ideas for future research.

Chapter 2

Rollout Algorithms for Knapsack Problems

ROLLOUT algorithms provide a natural and easily implemented approach for approximately solving many discrete and dynamic optimization problems. Their motivation comes from problems that can be solved using classical dynamic programming, but for which determining the value function (or value-to-go function) is computationally infeasible. The rollout technique estimates these values by simulating future events while following a simple greedy/heuristic policy, referred to as the base policy. In most cases, the rollout algorithm is ensured to perform as well as its base policy [30]. As shown by many computational studies, the performance is often much better than the base policy and sometimes near optimal [28].

Theoretical results showing a strict improvement of rollout algorithms over base policies have been limited to average-case asymptotic bounds on the breakthrough problem [27] and a worst-case analysis of the 0–1 knapsack problem [26]. The latter work motivates a complementary study of rollout algorithms for knapsack-type problems from an average-case perspective, which we provide in this chapter. Our goals are to give theoretical evidence for the utility of rollout algorithms and to contribute to the knowledge of problem types and features that make rollout algorithms work well. We anticipate that our proof techniques may be helpful in achieving performance guarantees on similar problems.

We use a stochastic model taken directly from the literature that has been used to study a wide variety of greedy algorithms for the subset sum problem [41]. This model is extended in a natural manner for our analysis of the 0–1 knapsack problem [89, 102]. We analyze two rollout techniques that we refer to as the exhaustive rollout and the consecutive rollout, both of which use the same base policy. During each iteration of the exhaustive rollout, the algorithm decides which one of the available items should be added to the knapsack. The consecutive rollout algorithm sequentially processes the items and at each iteration decides if the current item should be added to the knapsack. The base policy is a simple greedy algorithm that adds items until an infeasible item is encountered.

For both techniques, we derive bounds showing that the expected performance of the rollout algorithms is strictly better than the performance obtained by only using the base policy. For the subset sum problem, this is demonstrated by measuring the gap between the total value of packed items and capacity. For the 0–1 knapsack problem, the difference between total profits of the rollout algorithm and base policy is measured. The bounds are valid after only a single iteration of the rollout algorithm and hold for additional iterations.

The organization of the chapter is as follows. In the remainder of this section we review related work. We introduce our notation in Section 2.2. Section 2.3 describes the stochastic models in detail and derives important properties of the blind greedy algorithm, which is the algorithm that we use for a base policy. Results, examples, and proofs are shown for the exhaustive and consecutive rollout algorithms in Sections 2.4 and 2.5, respectively. In Section 2.6, we analyze the rollout approach on binary decision trees to glean some insights about average-case and worst-case behavior of rollout algorithms. A concluding discussion is given in Section 2.7. Section 2.8 contains omitted integral evaluations.

2.1 Related Work

Rollout algorithms were introduced by Tesauro and Galperin [140] as online Monte-Carlo search techniques for computer backgammon. The application to combinatorial optimization was formalized by Bertsekas et al. [30]. They gave conditions under which the rollout algorithm is guaranteed to perform as well as its base policy, namely if the algorithm is *sequentially consistent* or *sequentially improving*, and presented computational results on a two-stage maintenance and repair problem. The application of rollout algorithms to approximate stochastic dynamic programs was provided by Bertsekas and Castañon [28], where they showed extensive computational results on variations of the quiz problem. Rollout algorithms have since shown strong computational results on a variety of problems including vehicle routing (Secomandi [129]), fault detection (Tu and Pattipati [141]), and sensor scheduling (Li et al. [98]).

Beyond simple bounds derived from base policies, the only theoretical results given explicitly for rollout algorithms are average-case results for the breakthrough problem (Bertsekas [27]) and worst-case results for the 0–1 knapsack problem (Bertazzi [26]). In the breakthrough problem, the objective is to find a valid path through a directed binary tree, where some edges are blocked. If the free (non-blocked) edges occur with a given probability, independent of other edges, a rollout algorithm has a larger probability of finding a free path in comparison to a greedy algorithm [27]. Performance bounds for the 0–1 knapsack problem were recently shown by Bertazzi [26], who analyzed the rollout approach with variations of the decreasing density greedy (DDG) algorithm as a base policy. Similar bounds are also known due to the partial enumeration results of Sahni [125]. The DDG algorithm takes the best of two solutions: the one obtained by adding items in order of non-increasing profit to weight ratio, as long as they fit, and the solution resulting from adding only the item with highest profit. Bertazzi showed that from a worst-case perspective, running the first iteration of a rollout algorithm (specifically, what we will refer to as the exhaustive rollout algorithm) improves the approximation guarantee from 1/2 (bound provided by the base policy) to 2/3.

While results on approximating the subset sum problem initially focused on worst-case analysis, the performance of various greedy algorithms on random instances was analyzed from a computational perspective by Martello and Toth [101]. An early probabilistic analysis of the subset sum problem was given by d'Atri and Puech [48]. Using a discrete version of the model used in our work, they analyzed the expected performance of greedy algorithms with and without sorting. They showed an exact probability distribution for the gap obtained by the algorithms and gave asymptotic expressions for the probability of obtaining a non-zero gap. These results were refined by Pferschy [118], who gave precise bounds on expected gap values for greedy algorithms.

A very extensive analysis of greedy algorithms for the subset sum problem was given by Borgwardt and Tremel [41]. They introduced the continuous model that we use in this chapter and derived probability distributions of gaps for a variety of greedy algorithms. In particular, they showed performance bounds for various prolongations of a greedy algorithm where a different strategy is used on the remaining items after the greedy policy is no longer feasible. They also analyzed cases where items are ordered by size prior to use of the greedy algorithms.

In the area of probabilistic 0–1 knapsack problems, Szkatula and Libura [136] investigated the behavior of greedy algorithms, similar to the blind greedy algorithm used in our work, for the 0–1 knapsack problem with fixed capacity. They found recurrence equations describing the weight of the knapsack after each iteration and solved the equations for the case of uniform weights. In later work [137], they studied asymptotic properties of greedy algorithms, including conditions for the knapsack to be filled almost surely as the number of items approaches infinity.

There has been some work on asymptotic properties of the decreasing density greedy algorithm (DDG) for probabilistic 0–1 knapsack problems. Diubin and Korbut [53] showed properties of the asymptotical tolerance of the algorithm, which characterizes the deviation of the solution from the optimal value. Similarly, Calvin and Leung [43] showed convergence in distribution between the value obtained by the DDG algorithm and the value of the knapsack linear relaxation. Finally, it is worth noting other probabilistic studies of knapsack problems including the work of Dean et al. [49] on adaptive polices, Lueker's [99] analysis of online algorithms, and the expected polynomial time algorithm of Beier and Vöcking [21].

2.2 Notation

Since this chapter involves more notation than our other chapters, we summarize the notation here. We must keep track of ordering in our analysis, so we use sequences in place of sets and slightly abuse notation to perform set operations on sequences. These operations will mainly involve index sequences, and our index sequences will always contain unique elements. Sequences are denoted by bold letters. If we wish for $\boldsymbol{S}$ to be the increasing sequence of integers ranging from 2 to 5, we write $\boldsymbol{S} = \langle 2,3,4,5\rangle$. We then have $2 \in \boldsymbol{S}$, while $1 \notin \boldsymbol{S}$. We also say that $\langle 2,5\rangle \subseteq \boldsymbol{S}$ and $\boldsymbol{S} \setminus \langle 3\rangle = \langle 2,4,5\rangle$. The concatenation of sequence $\boldsymbol{S}$ with sequence $\boldsymbol{R}$ is denoted by $\boldsymbol{S} : \boldsymbol{R}$. If $\boldsymbol{R} = \langle 1,7\rangle$, then $\boldsymbol{S} : \boldsymbol{R} = \langle 2,3,4,5,1,7\rangle$. A sequence is indexed by an index sequence if the index sequence is shown in the subscript. Thus, $\boldsymbol{a_S}$ indicates the sequence $\langle a_2, a_3, a_4, a_5\rangle$. For a sequence to satisfy equality with another sequence, equality must be satisfied element by element, according to the order of the sequence. We use the notation $\boldsymbol{S^i}$ to denote the sequence $\boldsymbol{S}$ with item i moved to the front of the sequence: $\boldsymbol{S^3} = \langle 3,2,4,5\rangle$.

The notation $\mathbb{P}(\cdot)$ indicates probability, and $\mathbb{E}[\cdot]$ indicates expectation. We define $\overline{\mathbb{E}}[\cdot] := 1 - \mathbb{E}[\cdot]$. For random variables, we will use capital letters to denote the random variable (or sequence) and lowercase letters to denote specific instances of the random variable (or sequence). The probability density function for a random variable X is denoted by $f_X(x)$. For random variables X and Y, we use $f_{X|Y}(x|y)$ to denote the conditional density of X given the event $Y = y$. When multiple variables are involved, all variables on the left side of the vertical bar are conditioned on all variables on the right side of vertical bar. The expression $f_{X,Y|Z,W}(x,y|z,w)$ should be interpreted as $f_{(X,Y)|(Z,W)}((x,y)|(z,w))$ and not $f_{X,(Y|Z),W}(x,(y|z),w)$, for example. Events are denoted by the calligraphic font, such as $\mathcal{A}$, and the disjunction of two events is shown by the symbol $\vee$. We often write conditional probabilities of the form $\mathbb{P}(\cdot|X = x, Y = y, \mathcal{A})$ as $\mathbb{P}(\cdot|x,y,\mathcal{A})$. The notation $\mathcal{U}[a,b]$ indicates the density of a uniform random variable on interval $[a,b]$. The indicator function is denoted by $\mathbb{1}(\cdot)$, and the positive part of an expression is denoted by $(\cdot)_+$. Finally, we use the symbol $\leftarrow$ for assignment and $O(\cdot)$ asymptotic growth[1]. A summary of symbols that we use throughout the chapter is given as follows.

2.2.1 List of Chapter Symbols

$\leftarrow$	assignment
$\vee$	disjunction
$(\cdot)_+$	positive part
A	weight of the last packed item, $A := W_{K-1}$
B	knapsack capacity
$\mathcal{C}_j$	event that item j is the critical item
$\overline{\mathcal{C}_1}$	event that the first item is not critical
$\overline{\mathcal{C}_{1n}}$	event that neither the first nor the last item is critical
$\mathcal{D}_j$	event that item j is the drop critical item
$\mathcal{E}$	event that $g + w_{K-1} < 1$
G	gap given by the BLIND-GREEDY algorithm

[1] We write $f(n) = O(g(n))$ if and only if there exists some constant $C > 0$ such that $\lim_{n\to\infty} \left|\frac{f(n)}{g(n)}\right| \leq C$ [130].

$H(\cdot)$	harmonic number
$\mathbb{1}(\cdot)$	indicator function
$\boldsymbol{I}$	index sequence, $\boldsymbol{I} := \langle 1, 2, \ldots, n \rangle$
K	critical item index (first item that is infeasible for Blind-Greedy to pack)
L_1	drop critical item index (first infeasible item for Blind-Greedy when first item skipped)
$L_j,\ j \geq 2$	insertion critical item index (first infeasible item for Blind-Greedy when j inserted first)
M	number of remaining items, $M := n - K$
n	number of items
$O(\cdot)$	asymptotic growth
P_i	profit of item i (for the 0–1 knapsack problem)
$\boldsymbol{P_S}$	sequence of item profits, indexed by sequence $\boldsymbol{S}$
Q	profit of last packed item, $Q := P_{K-1}$
$\mathcal{U}[x, y]$:	uniform distribution with support $[x, y]$
V_1	drop gap (gap when the first item is skipped)
$V_j,\ j \geq 2$	insertion gap (gap when item j is inserted first)
V_j^u	upper bound on V_j
V_*	minimum gap; gap obtained after the first iteration of the rollout algorithm
V_*^u	upper bound on V_*
W_i	weight of item i
$\boldsymbol{W_S}$	sequence of item weights, indexed by sequence $\boldsymbol{S}$
$Z_j,\ j \geq 2$	insertion gain (gain when item j is inserted first)
Z_j^l	lower bound on Z_j
Z_*	maximum gain; gain obtained after the first iteration of the rollout algorithm
Z_*^l	lower bound on Z_*

2.3 Stochastic Model and Blind Greedy Algorithm

In the 0–1 knapsack problem, we are given a sequence of items $\boldsymbol{I} = \langle 1, 2, \ldots, n \rangle$, where each item $i \in \boldsymbol{I}$ has a weight $w_i \in \mathbb{R}_+$ and profit $p_i \in \mathbb{R}_+$. Given a knapsack with capacity $b \in \mathbb{R}_+$, the goal is to select a subset of items with maximum total profit such that the total weight does not exceed the capacity. This is given by the following integer linear program.

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} p_i x_i \\ \text{s.t.} \quad & \sum_{i=1}^{n} w_i x_i \leq b, \\ & x_i \in \{0, 1\}, \qquad i = 1, \ldots, n. \end{aligned} \tag{2.1}$$

The subset sum problem refers to the 0–1 knapsack problem with $p_i = w_i$ for all $i \in \boldsymbol{I}$.

We use the stochastic subset sum model given in [41] and a variation of this model for the 0–1 knapsack problem. In their subset sum model, for a specified number of items n, item weights W_i and the capacity B are drawn independently from the following distributions:

$$\begin{aligned} W_i &\sim \mathcal{U}[0,1], \quad i = 1, \ldots, n, \\ B &\sim \mathcal{U}[0,n]. \end{aligned} \tag{2.2}$$

Our stochastic knapsack model simply assigns item profits that are independently and uniformly distributed,

$$P_i \sim \mathcal{U}[0,1], \quad i = 1, \ldots, n. \tag{2.3}$$

These values are also independent with respect to the weights and capacity.

For evaluating performance, we only consider cases where $\sum_{i=1}^{n} W_i > B$. In all other cases, any algorithm that tries adding all items is optimal. Since it is difficult to understand the stochastic nature of optimal solutions, we use $\mathbb{E}[B - \sum_{i \in \boldsymbol{S}} W_i | \sum_{i=1}^{n} W_i > B]$ as a performance metric for the subset sum problem, where $\boldsymbol{S}$ is the sequence of items selected by the algorithm of interest. This is the same metric used in [41], where it is noted with a simple symmetry argument that for all values of n,

$$\mathbb{P}\left(\sum_{i=1}^{n} W_i > B\right) = \frac{1}{2}. \tag{2.4}$$

For the 0–1 knapsack problem, we directly measure the difference between the rollout algorithm profit and the profit given by the base policy, which we refer to as the *gain* of the rollout algorithm.

For both the subset sum problem and the 0–1 knapsack problem, we use the Blind-Greedy algorithm, shown in Algorithm 1, as a base policy. The algorithm simply adds items (without sorting) until it encounters an item that exceeds the remaining capacity, then stops. Throughout the chapter, we will sometimes refer to Blind-Greedy simply as the greedy algorithm.

Blind-Greedy may seem inferior to a greedy algorithm that first sorts the items by weight or profit to weight ratio, and then adds them in non-decreasing value. Surprisingly, for the subset sum problem, it was shown in [41] that the algorithm that adds items in order of non-decreasing weight (referred to as Greedy 1S) performs equivalently to Blind-Greedy. Of course, we cannot say the same about the 0–1 knapsack problem. A greedy algorithm that adds items in decreasing profit to weight ratio is likely to perform much better. Applying our analysis to a sorted greedy algorithm requires work beyond the scope of this chapter.

In analyzing Blind-Greedy, we refer to the index of the first item that is infeasible as the *critical item*. Let K be the random variable for the index of the critical

```
Input: Item sequence $\boldsymbol{I} = \langle 1, \ldots, n \rangle$ with profit sequence $\boldsymbol{p_I}$ and weight sequence $\boldsymbol{w_I}$,
    capacity $b$.
Output: Feasible solution sequence $\boldsymbol{S}$, value $U$.
 1: Initialize solution sequence $\boldsymbol{S} \leftarrow \langle\rangle$, remaining capacity $\bar{b} \leftarrow b$, and value $U \leftarrow 0$.
 2: for $i = 1$ to $n$ (each item) do
 3:     if $w_i \leq \bar{b}$ (item weight does not exceed remaining capacity) then
 4:         Add item $i$ to solution sequence, $\boldsymbol{S} \leftarrow \boldsymbol{S} : \langle i \rangle$.
 5:         Update remaining capacity $\bar{b} \leftarrow \bar{b} - w_i$, and value $U \leftarrow U + p_i$.
 6:     else
 7:         Stop and return $\boldsymbol{S}$, $U$.
 8:     end if
 9: end for
10: Return $\boldsymbol{S}$, $U$.
```

Algorithm 1. Blind-Greedy

item, where $K = 0$ indicates that there is no critical item (meaning $\sum_{i=1}^{n} W_i \leq B$). Equivalently, assuming $\sum_{i=1}^{n} W_i > B$, the critical item index satisfies

$$\sum_{i=1}^{K-1} W_i \leq B < \sum_{i=1}^{K} W_i. \tag{2.5}$$

We will refer to items with indices $i < K$ as packed items. We then define the *gap* of Blind-Greedy as

$$G := B - \sum_{i=1}^{K-1} W_i, \tag{2.6}$$

for $K > 0$. The gap is relevant to both the subset sum problem and the 0–1 knapsack problem. We use it as a performance measure only on the subset sum problem, however, since the capacity B is a natural upper bound on the optimal value, and the gap is thus an upper bound on the difference between a given solution value and the optimal value. There is no analogous upper bound on optimal value for the 0–1 knapsack problem, so for this problem we define the *gain* of the rollout algorithm as

$$Z := \sum_{i \in \boldsymbol{R}} P_i - \sum_{i=1}^{K-1} P_i, \tag{2.7}$$

where $\boldsymbol{R}$ is the sequence of items selected by the rollout algorithm. A central result of [41] is the following, which does not depend on the number of items n.

Theorem 2.1 (Borgwardt and Tremel, 1991)**.** *Independent of the critical item index $K > 0$, the probability distribution of the gap obtained by* Blind-Greedy *satisfies*

$$\mathbb{P}\left(G \le g \,\middle|\, \sum_{i=1}^{n} W_i > B \right) = 2g - g^2, \quad 0 \le g \le 1, \tag{2.8}$$

$$\mathbb{E}\left[G \,\middle|\, \sum_{i=1}^{n} W_i > B \right] = \frac{1}{3}. \tag{2.9}$$

Many studies measure performance using an approximation ratio (bounding the ratio of the value obtained by some algorithm to the optimal value) [26, 89]. While this metric is generally not tractable under the stochastic model, we can observe a simple lower bound on the ratio of expectations of the value given by Blind-Greedy to the optimal value for the subset sum problem. A natural upper bound on the optimal solution is B, and the solution value given by Blind-Greedy is equal to $B - G$. Thus, by Theorem 2.1 and linearity of expectation, the ratio of expected values is at least $\frac{\mathbb{E}[B-G]}{\mathbb{E}[B]} = 1 - \frac{2}{3n}$. For $n \ge 2$, this value is at least $\frac{2}{3}$, which is the best worst-case approximation ratio derived in [26]. A similar comparison for the 0–1 knapsack problem is not possible because there is no simple bound on the expected optimal solution value.

We describe some important properties of the Blind-Greedy solution that will be used in later sections and that provide a proof of Theorem 2.1. For the proofs in this section, as well as other sections, it is helpful to visualize the Blind-Greedy solution sequence on the nonnegative real line as shown in Figure 2.1.

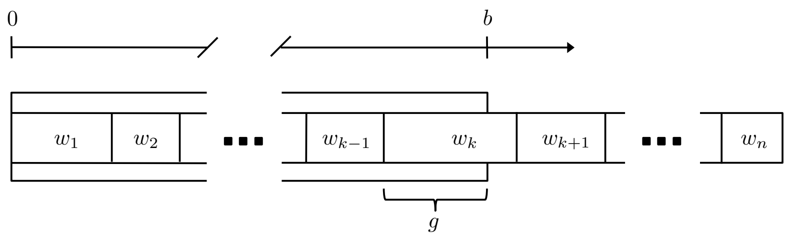


Figure 2.1. Sequence given by Blind-Greedy on the nonnegative real line where $G = g$, $B = b$, and $\boldsymbol{W_S} = \boldsymbol{w_s}$. Each item $\ell = 1, \dots, n$ occupies the interval $\left[\sum_{i=1}^{\ell-1} w_i, \sum_{i=1}^{\ell} w_i\right)$, and the knapsack is given on the interval $[0, b]$. The gap g is the difference between the capacity and the total weight of the packed items.

Previous work on the stochastic model has demonstrated that the critical item index is uniformly distributed on $\{1, 2, \dots, n\}$ for cases of interest (i.e., $\sum_{i=1}^{n} W_i > B$)

[41]. In addition to this property, we show that the probability that a given item is critical is independent of the weights of all other items. In other sections we follow the convention of associating the index k with the random variable K. The index ℓ is used in this section to make the proofs clearer.

Lemma 2.1. *For each item $\ell = 1, \ldots, n$, for all subsequences of items $\boldsymbol{S} \subseteq \boldsymbol{I} \setminus \langle \ell \rangle$ and all weights $\boldsymbol{w_S}$, the probability that item ℓ is critical is*

$$\mathbb{P}(K = \ell | \boldsymbol{W_S} = \boldsymbol{w_S}) = \frac{1}{2n}. \tag{2.10}$$

Proof. Assume that we are given the weights of all items $\boldsymbol{W_I} = \boldsymbol{w_I}$. We can divide the interval $[0, n]$ into $n+1$ segments as a function of item weights as shown in Figure 2.1, so that the ℓth segment occupies the interval $\left[\sum_{i=1}^{\ell-1} w_i, \sum_{i=1}^{\ell} w_i\right)$ for $\ell = 1, \ldots, n$, and the last segment is on $[\sum_{i=1}^{n} w_i, n]$. The probability that item ℓ is critical is the probability that B intersects the ℓth segment. Since B is distributed uniformly over the interval $[0, n]$, we have

$$\mathbb{P}(K = \ell | \boldsymbol{W_I} = \boldsymbol{w_I}) = \frac{w_\ell}{n}, \tag{2.11}$$

showing that this event only depends on w_ℓ. Integrating over the uniform density of w_ℓ gives the result. ■

An important property of this stochastic model, which is key for the rest of our development, is that conditioning on the critical item index only changes the weight distribution of the critical item; all other item weights remain independently distributed on $\mathcal{U}[0, 1]$.

Lemma 2.2. *For any critical item $K > 0$ and any subsequence of items $\boldsymbol{S} \subseteq \boldsymbol{I} \setminus \langle K \rangle$, the weights $\boldsymbol{W_S}$ are independently distributed on $\mathcal{U}[0, 1]$, and W_K independently follows the distribution*

$$f_{W_K}(w_K) = 2w_K, \quad 0 \leq w_K \leq 1. \tag{2.12}$$

Proof. For any item $\ell = 1, \ldots, n$, consider the subsequence of items $\boldsymbol{S} = \boldsymbol{I} \setminus \langle \ell \rangle$. Using Bayes' theorem, the conditional joint density for $\boldsymbol{W_S}$ is given by

$$\begin{aligned} f_{\boldsymbol{W_S}, W_\ell | K}(\boldsymbol{w_S}, w_\ell | \ell) &= \frac{\mathbb{P}(K = \ell | \boldsymbol{W_S} = \boldsymbol{w_S}, W_\ell = w_\ell)}{\mathbb{P}(K = \ell)} f_{\boldsymbol{W_S}}(\boldsymbol{w_S}) f_{W_\ell}(w_\ell) \\ &= \frac{w_\ell / n}{1/(2n)} f_{\boldsymbol{W_S}}(\boldsymbol{w_S}) \\ &= 2w_\ell f_{\boldsymbol{W_S}}(\boldsymbol{w_S}), \quad 0 \leq w_\ell \leq 1, \end{aligned} \tag{2.13}$$

where we have used the results of Lemma 2.1. This holds for the $K = \ell$ and $\ell = 1, \ldots, n$, so we replace the index ℓ with K in the expression. ■

We can now analyze the gap obtained by Blind-Greedy for $K > 0$. This gives the following lemma and a proof of Theorem 2.1.

Lemma 2.3. *Independent of the critical item index $K > 0$, the conditional distribution of the gap obtained by* Blind-Greedy *satisfies*

$$f_{G|W_K}(g|w_K) = \mathcal{U}[0, w_K]. \tag{2.14}$$

Proof. For any $\ell = 1, \ldots, n$ and any $\boldsymbol{W_I} = \boldsymbol{w_I}$, the posterior distribution of B given the event $K = \ell$ satisfies

$$f_{B|\boldsymbol{W_I},K}(b|\boldsymbol{w_I}, \ell) = \mathcal{U}\left[\sum_{i=1}^{\ell-1} w_i, \sum_{i=1}^{\ell} w_i\right], \tag{2.15}$$

since we have a uniform random variable B that is conditionally contained in a given interval. Now using the definition of G in (2.6),

$$f_{G|W_\ell,K}(g|w_\ell, \ell) = \mathcal{U}[0, w_\ell]. \tag{2.16}$$

■

Proof of Theorem 2.1. Using Lemma 2.3 and the distribution for W_K from Lemma 2.2, we have for $K > 0$,

$$f_G(g) = \int_0^1 f_{G|W_K}(g|w_K) f_{W_K}(w_K) \mathrm{d}w_K = \int_g^1 \frac{1}{w_K} 2w_K \mathrm{d}w_K = 2 - 2g, \tag{2.17}$$

where we have used that $G \leq W_K$ with probability one. This serves as a simpler proof of the theorem from [41]; their proof is likely more conducive to their analysis. ■

Finally, we need a modified version of Lemma 2.2, which will be used in the subsequent sections.

Lemma 2.4. *Given any critical item $K > 0$, gap $G = g$, and any subsequence of items $\boldsymbol{S} \subseteq \boldsymbol{I} \setminus \langle K \rangle$, the weights $\boldsymbol{W_S}$ are independently distributed on $\mathcal{U}[0, 1]$, and W_K is independently distributed on $\mathcal{U}[g, 1]$.*

Proof. Fix $K = \ell$ for any $\ell > 0$. The statement of the lemma is equivalent to the expression

$$f_{\boldsymbol{W_S},W_\ell|G,K}(\boldsymbol{w_S}, w_\ell|g, \ell) = \frac{1}{1-g} f_{\boldsymbol{W_S}}(\boldsymbol{w_S}), \quad g \leq w_\ell \leq 1. \tag{2.18}$$

Note that

$$f_{G|\boldsymbol{W_S},W_\ell,K}(g|\boldsymbol{w_S}, w_\ell, \ell) = \mathcal{U}[0, w_\ell], \tag{2.19}$$

which can be shown by the same argument for Lemma 2.3. Then,

$$\begin{aligned} f_{\boldsymbol{W_S},W_\ell|G,K}(\boldsymbol{w_S},w_\ell|g,\ell) &= \frac{f_{G|\boldsymbol{W_S},W_\ell,K}(g|\boldsymbol{w_S},w_\ell,\ell)f_{\boldsymbol{W_S},W_\ell|K}(\boldsymbol{w_S},w_\ell|\ell)}{f_{G|K}(g|\ell)} \\ &= \frac{f_{G|\boldsymbol{W_S},W_\ell,K}(g|\boldsymbol{w_S},w_\ell,\ell)f_{\boldsymbol{W_S}}(\boldsymbol{w_S})f_{W_\ell|K}(w_\ell|\ell)}{f_{G|K}(g|\ell)} \\ &= \frac{1}{w_\ell}\frac{2w_\ell}{2-2g}f_{\boldsymbol{W_S}}(\boldsymbol{w_S}), \quad g \le w_\ell \le 1, \end{aligned} \tag{2.20}$$

where we have used Lemma 2.2, (2.19), and Theorem 2.1. ■

2.4 Exhaustive Rollout

The Exhaustive-Rollout algorithm is shown in Algorithm 2. It takes as input a sequence of items $\boldsymbol{I}$ and capacity b. At each iteration, indexed by t, the algorithm considers all items in the available sequence $\overline{\boldsymbol{I}}$. It calculates the value obtained by moving each item to the front of the sequence and applying the Blind-Greedy algorithm. The algorithm then adds the item with the highest estimated value (if it exists) to the solution. We implicitly assume a consistent tie-breaking method, such as giving preference to the item with the lowest index. The next iteration then proceeds with the remaining sequence of items. An example progression of the algorithm is given in the first subsection of this section.

```
Input: Item sequence $\boldsymbol{I} = \langle 1,\dots,n\rangle$ with profit sequence $\boldsymbol{p_I}$ and weight sequence $\boldsymbol{w_I}$,
    capacity $b$.
Output: Feasible solution sequence $\boldsymbol{S}$, value $U$.
 1: Initialize $\boldsymbol{S} \leftarrow \langle\rangle$, $\overline{\boldsymbol{I}} \leftarrow \boldsymbol{I}$, $\overline{b} \leftarrow b$, $U \leftarrow 0$.
 2: for $t = 1$ to $n$ do
 3:     for $i \in \overline{\boldsymbol{I}}$ (each item in remaining item sequence) do
 4:         Let $\overline{\boldsymbol{I}}^{\boldsymbol{i}}$ denote the sequence $\overline{\boldsymbol{I}}$ with $i$ moved to the first position.
 5:         Estimate value of sequence, $(\cdot, U_i) = $ BLIND-GREEDY$(\boldsymbol{w}_{\overline{\boldsymbol{I}}^{\boldsymbol{i}}}, \overline{b})$.
 6:     end for
 7:     if $\max_i U_i > 0$ then
 8:         Determine item with max estimated value, $i^* \leftarrow \operatorname{argmax}_i U_i$.
 9:         Add item $i^*$ to solution sequence, $\boldsymbol{S} \leftarrow \boldsymbol{S} : \langle i^*\rangle$, $\overline{\boldsymbol{I}} \leftarrow \overline{\boldsymbol{I}} \setminus \langle i^*\rangle$.
10:         Update remaining capacity, $\overline{b} \leftarrow \overline{b} - w_i$, and value, $U \leftarrow U + p_i$.
11:     end if
12: end for
13: Return $S$, $U$.
```

Algorithm 2. Exhaustive-Rollout

Figure 2.2 shows simulated performance of Exhaustive-Rollout on the subset sum problem as a function of the number of items n as well as for the first, second,

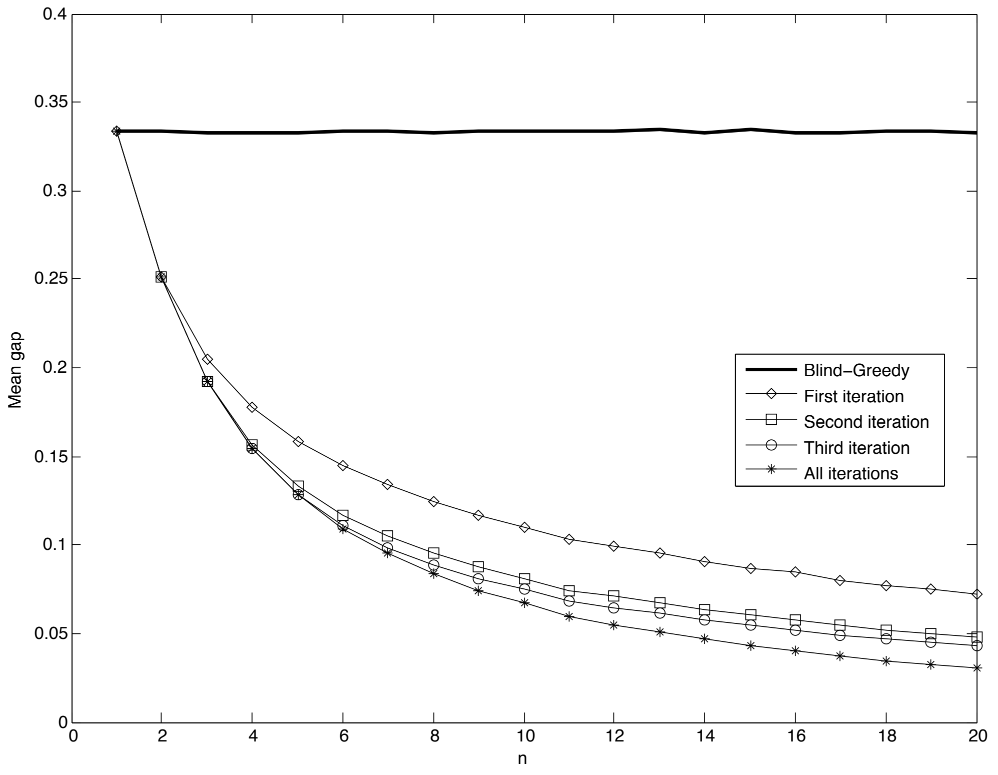


Figure 2.2. Simulated performance of the Blind-Greedy algorithm and the Exhaustive-Rollout algorithm on the subset sum problem as a function of the number of items n. Mean gaps, averaged over 10^5 simulations, are shown for the Blind-Greedy algorithm as well as for the first, second, third, and all iterations of the Exhaustive-Rollout algorithm. "All iterations" indicates that for each value of the number of items n, exactly n iterations are run. For all Exhaustive-Rollout plots, the mean gaps shown indicate the gaps obtained after running the indicated number of iterations and using Blind-Greedy thereafter.

third, and all iterations. "All iterations" means that n iterations are run, where n is the number of items. The gaps shown are not a result of using only the sequence $\boldsymbol{S}$ from Algorithm 2 after the number of iterations indicated; they are the gaps given by the sequence $\boldsymbol{S}$ and using Blind-Greedy thereafter. The figure shows that the most significant reduction in the gap compared to the Blind-Greedy algorithm follows from the first iteration of the rollout algorithm. However, further reductions in the gap are still made with additional iterations.

In our analysis, we only consider the first iteration of Exhaustive-Rollout, which tries using Blind-Greedy after moving each item to the front of the sequence and takes the best of these solutions. This gives an upper bound for the subset sum gap and a lower bound on the 0–1 knapsack problem gain following from additional iterations. The technical condition for these bounding properties to hold is that the base policy/algorithm is *sequentially consistent*, as defined in [30]. It is easy to verify that Blind-Greedy satisfies this property. For the subset sum problem, let $V_*(n)$ denote the gap obtained after a single iteration of Exhaustive-Rollout on the stochastic model with n items. We have the following bounds.

Theorem 2.2. *For the subset sum problem, the gap $V_*(n)$, obtained by running a single iteration of* Exhaustive-Rollout*, satisfies*

$$\mathbb{E}\left[V_*(n) \middle| \sum_{i=1}^{n} W_i > B\right] \quad \leq \quad \frac{1}{n(2+n)} + \frac{1}{n}\sum_{m=0}^{n-2}\frac{9+2m}{3(3+m)(4+m)}. \tag{2.21}$$

Corollary 2.1.

$$\mathbb{E}\left[V_*(n) \middle| \sum_{i=1}^{n} W_i > B\right] \quad \leq \quad \frac{1}{n(2+n)} + \frac{1}{n}\log\left[\left(\frac{3+2n}{5}\right)\left(\frac{7}{5+2n}\right)^{1/3}\right]. \tag{2.22}$$

Theorem 2.3.

$$\lim_{n\to\infty}\mathbb{E}\left[V_*(n) \middle| \sum_{i=1}^{n} W_i > B\right] = 0, \qquad \mathbb{E}\left[V_*(n) \middle| \sum_{i=1}^{n} W_i > B\right] = O\left(\frac{\log n}{n}\right). \tag{2.23}$$

A plot of the bounds and simulated results is shown in Figure 2.3(a). For the knapsack problem, let $Z_*(n)$ denote the gain given by a single iteration of Exhaustive-Rollout. The expected gain is bounded by the two following theorems, where $H(n)$ is the nth harmonic number, $H(n) := \sum_{\ell=1}^{n}\frac{1}{\ell}$.

Theorem 2.4. *For the knapsack problem, the gain $Z_*(n)$, obtained by running a single iteration* Exhaustive-Rollout*, satisfies*

$$\begin{aligned}
&\mathbb{E}\left[Z_*(n) \middle| \sum_{i=1}^{n} W_i > B\right] \geq 1 + \frac{2}{n(n+1)} - \frac{2H(n)}{n^2} \\
&+\frac{1}{n}\sum_{m=0}^{n-2}\left(\sum_{j=1}^{m+1} T(j,m) + \big((186+472m+448m^2+203m^3+45m^4+4m^5)\right. \\
&-(244+454m+334m^2+124m^3+24m^4+2m^5)H(m+1) \\
&\left.-(48+88m+60m^2+18m^3+2m^4)(H(m+1))^2\big)\,\frac{1}{(m+1)(m+2)^3(m+3)^2}\right),
\end{aligned} \tag{2.24}$$

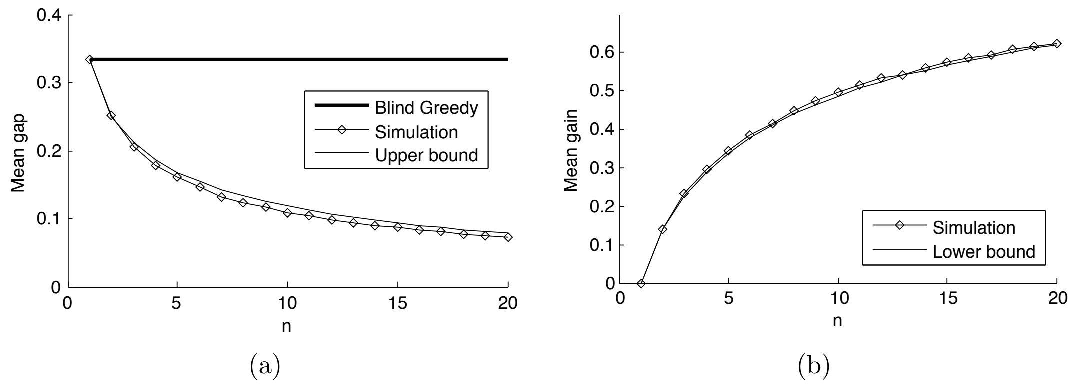


Figure 2.3. Performance bounds and simulated values for (a) expected gap $\mathbb{E}[V_*(n)|\cdot]$ and (b) expected gain $\mathbb{E}[Z_*(n)|\cdot]$ after running a single iteration of Exhaustive-Rollout on the subset sum problem and 0–1 knapsack problem, respectively. For each n, the mean values are shown for 10^5 simulations.

where

$$\begin{aligned} T(j,m) \quad := \quad & \frac{2\left(-4+j-4m+jm-m^2-\left(j+(2+m)^2\right)H(j)\right)}{j(-3+j-m)(-2+j-m)(1+m)(2+m)} \\ & + \frac{2(j+(2+m)^2)H(3+m)}{j(-3+j-m)(-2+j-m)(1+m)(2+m)}. \end{aligned} \tag{2.25}$$

Theorem 2.5.

$$\lim_{n\to\infty} \mathbb{E}\left[Z_*(n) \,\middle|\, \sum_{i=1}^n W_i > B\right] = 1, \tag{2.26}$$

$$1 - \mathbb{E}\left[Z_*(n) \,\middle|\, \sum_{i=1}^n W_i > B\right] = O\left(\frac{\log^2 n}{n}\right). \tag{2.27}$$

The expected gain approaches unit value at a rate slightly slower than the convergence rate for the subset sum problem. This is likely a result of the fact that in the subset sum problem, the algorithm is searching for an item with one criterion: a weight approximately equal to the gap. For the 0–1 knapsack problem, however, the algorithm must find an item satisfying two criteria: a weight smaller than the gap and a profit approximately equal to one. The gain is plotted with simulated values in Figure 2.3(b). While the bound in Theorem 2.4 does not admit a simple integral bound, omitting the nested summation term $\sum_{j=1}^m T(j,m)$ gives a looser but valid bound. We show an example evolution of the algorithm and proofs of the theorems for both problems in the remainder of this section.

2.4.1 Example

We show an example of EXHASUTIVE-ROLLOUT for the subset sum problem (recall that for the subset sum problem, $\boldsymbol{p_I} = \boldsymbol{w_I}$). Let $n = 6$, implying that $\boldsymbol{I} = \langle 1, 2, 3, 4, 5, 6\rangle$. Consider the weights $\boldsymbol{w_I} = \langle 0.3,\ 0.8,\ 0.6,\ 0.4,\ 0.7,\ 0.1\rangle$ and a capacity $b = 2$. An optimal solution sequence is $\boldsymbol{S} = \langle 1, 3, 4, 5\rangle$, which gives a value equal to the capacity of 2. Running BLIND-GREEDY gives a value of 1.7 (and a gap of 0.3), which we illustrate as follows.

$\boldsymbol{I}$	1	2	3	4	5	6	value
$\boldsymbol{w_I}$	0.3	0.8	0.6	0.4	0.7	0.1	
cum.	0.3	1.1	1.7	*2.1*	*2.8*	*2.9*	1.7

This tabular illustration shows the item sequence in the first row, the corresponding weights in the second row, and the cumulative weight – obtained by summing all weights up through the current item – in the last row. The cumulative weights greater than the capacity of 2 are italicized since they correspond to infeasible packings. The value in the last column is determined by the largest feasible cumulative weight, in this case 1.7.

In applying EXHAUSTIVE-ROLLOUT, we initially set $\overline{\boldsymbol{I}} = \boldsymbol{I}$. For $t = 1$ and $i = 1$, we have $\overline{\boldsymbol{I}}^{\boldsymbol{1}} = \boldsymbol{I}$, so the value is equal to the value of the BLIND-GREEDY algorithm. For $t = 1$ and $i = 2$, we set $\boldsymbol{J} = \overline{\boldsymbol{I}}^{\boldsymbol{2}}$, where $\boldsymbol{J}$ henceforth indicates the current sequence that the rollout algorithm is evaluating. This is shown below, following our tabular notation. The value obtained is 1.7.

$\boldsymbol{J}$	2	1	3	4	5	6	value
$\boldsymbol{w_J}$	0.8	0.3	0.6	0.4	0.7	0.1	
cum.	0.8	1.1	1.7	*2.1*	*2.8*	*2.9*	1.7

For $t = 1$ and $i = 3$, $\boldsymbol{J} = \overline{\boldsymbol{I}}^{\boldsymbol{3}}$ and the value is 1.7.

$\boldsymbol{J}$	3	1	2	4	5	6	value
$\boldsymbol{w_J}$	0.6	0.3	0.8	0.4	0.7	0.1	
cum.	0.6	0.9	1.7	*2.1*	*2.8*	*2.9*	1.7

For $t = 1$ and $i = 4$, $\boldsymbol{J} = \overline{\boldsymbol{I}}^{\boldsymbol{4}}$ and the value is 1.5.

$\boldsymbol{J}$	4	1	2	3	5	6	value
$\boldsymbol{w_J}$	0.4	0.3	0.8	0.6	0.7	0.1	
cum.	0.4	0.7	1.5	*2.1*	*2.8*	*2.9*	1.5

For $t = 1$ and $i = 5$, $\boldsymbol{J} = \overline{\boldsymbol{I}}^{\boldsymbol{5}}$ and the value is 1.8.

$\boldsymbol{I}$	5	1	2	3	4	6	value
$\boldsymbol{w_I}$	0.7	0.3	0.8	0.6	0.4	0.1	
cum.	0.7	1.0	1.8	*2.4*	*2.8*	*2.9*	1.8

For $t = 1$ and $i = 6$, $\boldsymbol{J} = \overline{\boldsymbol{I}}^{\boldsymbol{6}}$ and the value is 1.8.

$\boldsymbol{I}$	6	1	2	3	4	5	value
$\boldsymbol{w_I}$	0.1	0.3	0.8	0.6	0.4	0.7	
cum.	0.1	0.4	1.2	1.8	*2.2*	*2.9*	1.8

Both $i = 5$ and $i = 6$ give a value of 1.8 for the first iteration ($t = 1$), which is better than the Blind-Greedy value of 1.7. We break the tie in favor of $i = 5$, so we conclude the first iteration by selecting $i^* = 5$ and moving this item to the solution sequence.

We begin the second iteration with the updated sequences $\boldsymbol{S} = \langle 5 \rangle$ and $\overline{\boldsymbol{I}} = \langle 1, 2, 3, 4, 6 \rangle$. For $t = 2$ and $i = 1$, $\boldsymbol{J} = \overline{\boldsymbol{I}}^{\boldsymbol{1}}$ and the value is 1.8, which we know from the previous iteration, and is shown below.

$\boldsymbol{S}$	5	$\boldsymbol{J}$	1	2	3	4	6	value
$\boldsymbol{w_S}$	0.7	$\boldsymbol{w_J}$	0.3	0.8	0.6	0.4	0.1	
cum.	0.7		1.0	1.8	*2.4*	*2.8*	*2.9*	1.8

Note that we have modified the table to include both the solution sequence $\boldsymbol{S}$ and the current sequence $\boldsymbol{J}$; the cumulative weight includes weight from the solution sequence. We skip the iteration for $i = 2$ because the second item is packed before the critical item (this iteration gives the same value of 1.8). For $t = 2$ and $i = 3$, $\boldsymbol{J} = \overline{\boldsymbol{I}}^{\boldsymbol{3}}$ and the value is 1.6.

$\boldsymbol{S}$	5	$\boldsymbol{J}$	3	1	2	4	6	value
$\boldsymbol{w_S}$	0.7	$\boldsymbol{w_J}$	0.6	0.3	0.8	0.4	0.1	
cum.	0.7		1.3	1.6	*2.4*	*2.8*	*2.9*	1.6

For $t = 2$ and $i = 4$, $\boldsymbol{J} = \overline{\boldsymbol{I}}^{\boldsymbol{4}}$ and the value is 1.4.

$\boldsymbol{S}$	5	$\boldsymbol{J}$	4	1	2	3	6	value
$\boldsymbol{w_S}$	0.7	$\boldsymbol{w_J}$	0.4	0.3	0.8	0.6	0.1	
cum.	0.7		1.1	1.4	*2.2*	*2.8*	*2.9*	1.4

Since we have already added item 5 to the solution sequence, the next item to consider in $\overline{\boldsymbol{I}}$ is item 6. For $t = 2$ and $i = 6$, $\boldsymbol{J} = \overline{\boldsymbol{I}}^{\boldsymbol{6}}$ and the value is 1.9.

$\boldsymbol{S}$	5	$\boldsymbol{J}$	6	1	2	3	4	value
$\boldsymbol{w_S}$	0.7	$\boldsymbol{w_J}$	0.1	0.3	0.8	0.6	0.4	
cum.	0.7		0.8	1.1	1.9	*2.5*	*2.9*	1.9

The highest value found in the second iteration is for $i = 6$, giving a value of 1.9, so we have $i^* = 6$, and we move the sixth item to the solution set.

The third iteration starts with updated sets $\boldsymbol{S} = \langle 5, 6 \rangle$ and $\overline{\boldsymbol{I}} = \langle 1, 2, 3, 4 \rangle$. From the previous iteration, we know that for $t = 3$ and $i = 1$, $\boldsymbol{J} = \overline{\boldsymbol{I}}^{\boldsymbol{1}}$ and the value is 1.9.

$\boldsymbol{S}$	5	6	$\boldsymbol{J}$	1	2	3	4	value
$\boldsymbol{w_S}$	0.7	0.1	$\boldsymbol{w_J}$	0.3	0.8	0.6	0.4	
cum.	0.7	0.8		1.1	1.9	*2.5*	*2.9*	1.9

We again skip the case $i = 2$ since the second item is packed before the critical item. For $t = 3$ and $i = 3$, $\boldsymbol{J} = \overline{\boldsymbol{I}}^{\boldsymbol{3}}$ and the value is 1.7.

$\boldsymbol{S}$	5	6	$\boldsymbol{J}$	3	1	2	4	value
$\boldsymbol{w_S}$	0.7	0.1	$\boldsymbol{w_J}$	0.6	0.3	0.8	0.4	
cum.	0.7	0.8		1.4	1.7	*2.5*	*2.9*	1.7

For $t = 3$ and $i = 4$, $\boldsymbol{J} = \overline{\boldsymbol{I}}^{\boldsymbol{4}}$ and the value is 1.5.

$\boldsymbol{S}$	5	6	$\boldsymbol{J}$	4	1	2	3	value
$\boldsymbol{w_S}$	0.7	0.1	$\boldsymbol{w_J}$	0.4	0.3	0.8	0.6	
cum.	0.7	0.8		1.2	1.5	*2.3*	*2.9*	1.5

Thus for the third iteration, the highest value of 1.9 is given by $i^* = 1$, and we move the first item to the solution set.

The fourth iteration starts with the sets $\boldsymbol{S} = \langle 5, 6, 1\rangle$ and $\overline{\boldsymbol{I}} = \langle 2, 3, 4\rangle$. Rather than illustrating the fourth and additional iterations, we can observe that with the remaining capacity of 0.9, the best that can be done with the remaining weight sequence $\boldsymbol{w}_{\overline{\boldsymbol{I}}} = \langle 0.8, 0.6, 0.4\rangle$ is to insert item 2, giving a total value of 1.9. In fact, this is the solution given by running only the first two iterations of the rollout algorithm. We conclude that running all iterations of the rollout algorithm gives the solution sequence $\boldsymbol{S} = \langle 5, 6, 1, 2\rangle$, a value of 1.9, and a gap of $2 - 1.9 = 0.1$.

Summarizing this example, Blind-Greedy gives the solution sequence $\boldsymbol{S} = \langle 1, 2, 3\rangle$ with value 1.7. The first iteration of Exhaustive-Rollout gives the solution $\boldsymbol{S} = \langle 5, 1, 2\rangle$ with value 1.8, and the second iteration of Exhaustive-Rollout gives the solution $\boldsymbol{S} = \langle 5, 6, 1, 2\rangle$ with value 1.9[2]. This solution given by the second iteration remains unchanged during additional iterations.

2.4.2 Exhaustive Rollout: Subset Sum Problem Analysis

The proof method for Theorem 2.2 is to visually analyze the solution sequence given by Blind-Greedy on the nonnegative real line, as shown in Figure 2.1. We consider the effect of individually moving each item to the front of the sequence, which will cause the other items to shift to the right. Our strategy is to perform this analysis while conditioning on three parameters: the greedy gap G, the critical item K, and the weight of the last packed item W_{K-1}. We then find the minimum gap given by trying each item at the front of the sequence. Finally, we integrate over conditioned variables to obtain the final bound.

To analyze solutions obtained by using Blind-Greedy after moving a given item to the front of the sequence, we introduce two definitions. The *jth insertion critical*

[2] In this paragraph we are abusing notation so that $\boldsymbol{S}$ is the sequence given by some iteration of Exhaustive-Rollout *and* Blind-Greedy on the remaining items, as opposed to the definition of $\boldsymbol{S}$ given in the algorithm description.

item L_j, defined for $j \geq 2$, is the first item that is infeasible to pack by Blind-Greedy when item j is moved to the front of the sequence. Equivalently, L_j satisfies

$$\begin{cases} W_j + \sum_{i=1}^{L_j-1} W_i \; \mathbb{1}(i \neq j) \leq B < W_j + \sum_{i=1}^{L_j} W_i \; \mathbb{1}(i \neq j), & \text{if } W_j \leq B, \\ L_j = j, & \text{if } W_j > B. \end{cases} \tag{2.28}$$

We then define the corresponding *jth insertion gap* V_j as the gap given by the greedy algorithm when item j is moved to the front of the sequence:

$$V_j := B - \mathbb{1}(W_j \leq B) \left(W_j + \sum_{i=1}^{L_j-1} W_i \; \mathbb{1}(i \neq j) \right), \quad j \geq 2. \tag{2.29}$$

In the following three lemmas, we bound the probability distribution of the insertion gap for packed items ($j \leq K-1$), the critical item ($j = K$), and the remaining items ($j \geq K+1$), while assuming that $K > 1$. Lemma 2.8 then handles the case where $K = 1$. Thereafter, we bound the minimum of these gaps and the greedy gap G, and finally integrate over the conditioned variables to obtain the bound on the expected minimum gap. The key analysis is illustrated in the proof of Lemma 2.6; the related proofs of Lemma 2.7 and Lemma 2.8 are based on similar ideas. The event $\mathcal{C}_j$ indicates that item j is critical, and $\overline{\mathcal{C}_1}$ indicates the event that the first item is not critical. Recall that $\boldsymbol{P_I} = \boldsymbol{W_I}$ for the subset sum problem.

Lemma 2.5. *For $K > 1$ and $j = 2, \ldots, K-1$, the jth insertion gap satisfies*

$$V_j = G \tag{2.30}$$

with probability one.

Proof. This follows trivially since the term $\sum_{i=1}^{K-1} W_i$ in (2.5) does not depend on the order of summation. ■

Lemma 2.6. *For $K > 1$ and $j = K+1, \ldots, n$, the jth insertion gap satisfies $V_j \leq V_j^u$ with probability one, where V_j^u is a deterministic function of (G, W_{K-1}, W_j), and conditioning only on (G, W_{K-1}) gives*

$$\begin{aligned} \mathbb{P}(V_j^u > v | g, w_{K-1}, \overline{\mathcal{C}_1}) &= (g - v)_+ + (w_{K-1} - v)_+ - (g + w_{K-1} - v - 1)_+ \\ &\quad + (1 - g - w_{K-1})_+ \\ &=: \mathbb{P}(V^u > v | g, w_{K-1}, \overline{\mathcal{C}_1}). \end{aligned} \tag{2.31}$$

Proof. Fix $K = k$ for $k > 1$. To simplify notation, make the event $\overline{\mathcal{C}_1}$ implicit throughout the proof. Define the random variable V_j^u so that

$$V_j^u := \begin{cases} V_j, & L_j = k \vee L_j = k-1, \\ 1, & L_j \leq k-2 \vee L_j = j. \end{cases}$$

While V_j may generally depend on $(G, W_j, W_1, \ldots, W_{k-1})$, the variable V_j^u is chosen so that it only depends on (G, W_{k-1}, W_j). In cases where V_j does only depend on (G, W_{k-1}, W_j), we have $V_j^u = V_j$. When V_j depends on more than these three variables, V_j^u assumes a worst-case bound of unit value.

We begin by analyzing the case where $L_j = k \vee L_j = k-1$, so that the insertion gap V_j is equal to V_j^u. For $G = g$ and $\boldsymbol{W_I} = \boldsymbol{w_I}$, a diagram illustrating the insertion gap as determined by g, w_{k-1}, and w_j is shown in Figure 2.4. We will follow the convention of using lowercase letters for random variables shown in figures and when referring to these variables. The knapsack is shown at the top of the figure with items packed sequentially from left to right. The plot at the bottom shows the insertion gap V_j that occurs when item j is inserted at the front of the sequence, causing the remaining packed items to slide to the right. The plot is best understood by visualizing the effect of varying sizes of w_j. If w_j is very small, the items slide to the right and reduce the gap by the amount w_j. Clearly, if $w_j = g$, then $v_j = 0$, as indicated by the function. As soon as w_j is slightly larger than g, it is infeasible to pack item $k-1$, and the gap jumps. Thus for the instance shown, the jth insertion gap is a deterministic function of (g, w_{k-1}, w_j).

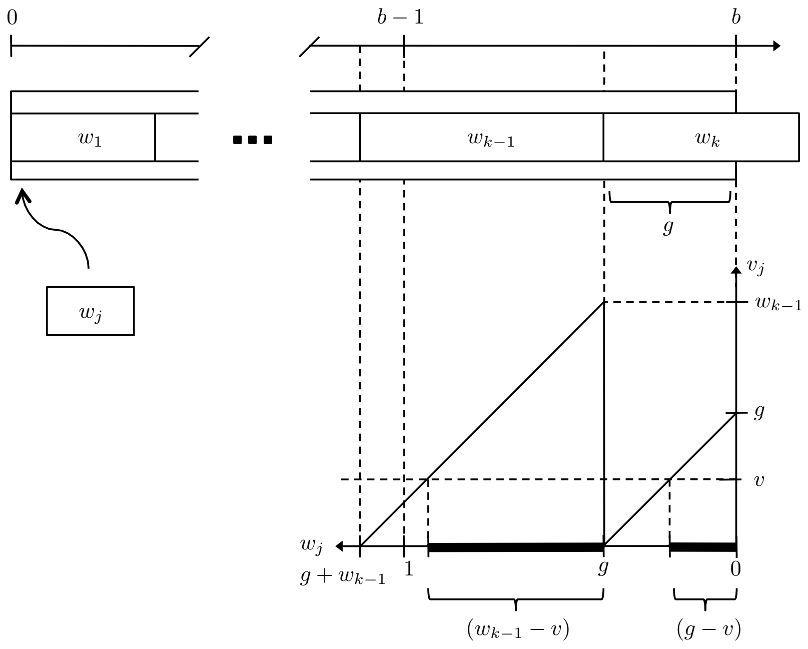


Figure 2.4. Insertion gap v_j as a function of w_j, parameterized by (w_{k-1}, g). The function starts at g and decreases at unit rate, except at $w = g$ where the function jumps to value w_{k-1}. The probability of the event $V_j > v$ conditioned only on w_{k-1} and g is given by the total length of the bold regions, assuming that $v < g$ and $g + w_{k-1} - v \leq 1$. Based on the sizes of g and w_{k-1} shown, only the events $L_j = k$ and $L_j = k-1$ are possible.

Considering the instance in the figure, if we only condition on g and w_{k-1} and allow

W_j to be random, then V_j becomes a random variable whose only source of uncertainty is W_j. Since by Lemma 2.4 W_j has distribution $\mathcal{U}[0,1]$, the probability of the event $V_j > v$ is given by the length of the bold regions on the w_j axis.

We now explicitly describe the length of the bold regions for all cases of w_{k-1} and g; this will include the case $L_j = k-2 \ \vee \ L_j = j$ (not possible for the instance in the figure), so the length of the bold regions will define V_j^u. Starting with the instance shown, we have $\mathbb{P}(V_j^u > v|g, w_{k-1}) = (g-v) + (w_{k-1} - v)$, as given by the lengths of the two bold regions, corresponding to the events $L_j = k$ and $L_j = k-1$, respectively. This requires that $v \le g$ and $v \le w_{k-1}$, so the expression becomes $\mathbb{P}(V_j^u > v|g, w_{k-1}) = (g-v)_+ + (w_{k-1}-v)_+$. We must account for the case where $g + w_{k-1} - v > 1$, requiring that we subtract length $(g + w_{k-1} - v - 1)$, so we revise the expression to $\mathbb{P}(V_j^u > v|g, w_{k-1}) = (g-v)_+ + (w_{k-1}-v)_+ - (g+w_{k-1}-v-1)_+$. Finally, for the case of $g + w_{k-1} < 1$, we must take care of the region where $w_i \in [g + w_{k-1}, 1]$. It is at this point that the event $L_j \le k-2$ or $L_j = j$ becomes possible and the distinction between V_j^u and V_j is made. Here we have by definition $V_j^u = 1$, which trivially satisfies $V_j \le V_j^u$, so for any $0 \le v < 1$ this region contributes $(1 - g - w_{k-1})$ to $\mathbb{P}(V_j^u > v|g, w_{k-1})$. This is handled by adding the term $(1 - g - w_{k-1})_+$ to the expression. We finally arrive at

$$\begin{aligned}\mathbb{P}(V_j^u > v|g, w_{k-1}) \quad &= \quad (g-v)_+ + (w_{k-1}-v)_+ - (g + w_{k-1} - v - 1)_+ \\ &\quad\ + (1 - g - w_{k-1})_+. \end{aligned} \tag{2.32}$$

This holds true for any fixed k as long as $k > 1$, so we may replace w_{k-1} with w_{K-1} and make the event $\overline{\mathcal{C}_1}$ explicit to obtain the statement of the lemma. ■

Lemma 2.7. *For $K > 1$, the Kth insertion gap satisfies $V_K \le V_K^u$ with probability one, where V_K^u is a deterministic function of (G, W_{K-1}, W_K), and conditioning only on (G, W_{K-1}) gives*

$$\begin{aligned}\mathbb{P}(V_K^u > v|g, w_{K-1}, \overline{\mathcal{C}_1}) \quad &= \quad \left(\frac{1}{1-g}\right)((w_{K-1}-v)_+ - (g + w_{K-1} - v - 1)_+ \\ &\quad\ + (1 - g - w_{K-1})_+) \\ &=: \quad \mathbb{P}(\widetilde{V}^u > v|g, w_{K-1}, \overline{\mathcal{C}_1}). \end{aligned} \tag{2.33}$$

Proof. Again fix $K = k$ for $k > 1$, $G = g$, $W_{k-1} = w_{k-1}$. Define the random variable V_k^u so that

$$V_k^u = \begin{cases} V_k, & L_k = k-1, \\ 1, & L_k \le k-2 \vee L_k = k. \end{cases}$$

From Lemma 2.4 we are guaranteed that given $G = g$, W_k follows distribution $\mathcal{U}[g,1]$. Thus, to determine $\mathbb{P}(V_k^u > v|g, w_{k-1}, \overline{\mathcal{C}}_1)$, we can use the same analysis for Lemma 2.6 but restricted to the interval $g \le w_k \le 1$. Taking the expression $\mathbb{P}(V^u > v|g, w_{K-1}, \overline{\mathcal{C}}_1)$

in (2.31), removing the $(g-v)_+$ term, and normalizing by $(1-g)$, we have

$$\mathbb{P}(V_k^u > v|g, w_{k-1}, \overline{\mathcal{C}}_1) = \left(\frac{1}{1-g}\right)((w_{k-1}-v)_+ - (g+w_{k-1}-v-1)_+ + (1-g-w_{k-1})_+) . \tag{2.34}$$

This holds for all $k > 1$, so we replace k with K in the expression. ■

Lemma 2.8. *For $K = 1$ and $j = 2, \ldots, n$, the jth insertion gap is a deterministic function of (W_j, G), and conditioning only on G gives*

$$\mathbb{P}(V_j > v|g, \mathcal{C}_1) = (1-v)\mathbb{1}(v < g). \tag{2.35}$$

Proof. Fix $G = g$. Note that for $K = 1$, the jth insertion gap can never be greater than g. Keeping the analysis for Lemma 2.6 in mind and using Lemma 2.4, we have that for $v < g$,

$$\mathbb{P}(V_j > v|g, \mathcal{C}_1) = (g-v) + (1-g), \tag{2.36}$$

where $(g-v)$ corresponds to the case where $w_j \in [0, g]$ and $(1-g)$ corresponds to $w_j \in (g, 1]$. ■

Recall that $V_*(n)$ is the gap obtained after the first iteration of the rollout algorithm on an instance n items, which we refer to as the *minimum gap*,

$$V_*(n) := \min(G, V_2, \ldots, V_n). \tag{2.37}$$

We will make the dependence on n implicit in what follows so that $V_* = V_*(n)$. We may now prove the final result.

Proof of Theorem 2.2. For $K = k > 1$, we have $V_* \leq V_*^u$ with probability one, where

$$V_*^u := \min(G, V_k^u, V_{k+1}^u, \ldots, V_n^u). \tag{2.38}$$

This follows from Lemmas 2.5 - 2.7, as $V_j = G$ for $j \leq k-1$. From the analysis in Lemmas 2.6 and 2.7, for each $j \geq k$, V_j^u is a deterministic function of (G, W_{k-1}, W_k, W_j). Furthermore, from Lemma 2.4, the item weights W_j for $j \geq k+1$ are independently distributed on $\mathcal{U}[0,1]$, and W_k is independently distributed on $\mathcal{U}[g,1]$. Thus, conditioning only on G and W_{k-1} makes V_j^u independent for $j \geq k$, and by the definition of the minimum function,

$$\begin{aligned}\mathbb{P}(V_*^u > v|g, w_{k-1}, k, \overline{\mathcal{C}_1}) &= \mathbb{P}(G > v|g, w_{k-1}, \overline{\mathcal{C}_1})\mathbb{P}(V_k^u > v|g, w_{k-1}, \overline{\mathcal{C}_1}) \prod_{j=k+1}^{n} \mathbb{P}(V_j^u > v|g, w_{k-1}, \overline{\mathcal{C}_1}) \\ &= \mathbb{P}(G > v|g, w_{k-1}, \overline{\mathcal{C}_1})\mathbb{P}(\widetilde{V}^u > v|g, w_{k-1}, \overline{\mathcal{C}_1}) \left(\mathbb{P}(V^u > v|g, w_{k-1}, \overline{\mathcal{C}_1})\right)^{(n-k)} .\end{aligned} \tag{2.39}$$

Marginalizing over W_{k-1} and G using Lemma 2.4 and Theorem 2.1,

$$\mathbb{P}(V_*^u > v|k, \overline{\mathcal{C}_1}) = \int_0^1 \int_0^1 \mathbb{P}(V_*^u > v|g, w_{k-1}, k, \overline{\mathcal{C}_1}) f_{w_{k-1}}(w_{k-1}) f_G(g) \mathrm{d}w_{k-1} \mathrm{d}g. \tag{2.40}$$

We refer to $\mathbb{P}(V_*^u > v|k, \overline{\mathcal{C}_1})$ as $\mathbb{P}(V_*^u > v|m, \overline{\mathcal{C}_1})$ via the substitution $M := n - K$ to simplify expressions. As shown in Section 2.8, evaluation of the integral gives

$$\mathbb{P}(V_*^u > v|m, \overline{\mathcal{C}_1}) = \begin{cases} \mathbb{P}(V_*^u > v|m, \overline{\mathcal{C}_1})_{\leq \frac{1}{2}}, & v \leq \frac{1}{2}, \\ \mathbb{P}(V_*^u > v|m, \overline{\mathcal{C}_1})_{> \frac{1}{2}}, & v > \frac{1}{2}, \end{cases} \tag{2.41}$$

where

$$\begin{aligned} \mathbb{P}(V_*^u > v|m, \overline{\mathcal{C}_1})_{\leq \frac{1}{2}} = {} & \frac{1}{3(3+m)} \left(2m(1-2v)^m + m(1-v)^m + 9(1-v)^{3+m}\right. \\ & -12m(1-2v)^m v - 3m(1-v)^m v + 24m(1-2v)^m v^2 \\ & \left. +3m(1-v)^m v^2 - 16m(1-2v)^m v^3 - m(1-v)^m v^3\right), \end{aligned} \tag{2.42}$$

$$\mathbb{P}(V_*^u > v|m, \overline{\mathcal{C}_1})_{> \frac{1}{2}} = \frac{1}{3}(1-v)^{3+m} + \frac{2(1-v)^{3+m}}{3+m}. \tag{2.43}$$

Calculating the expected value gives a surprisingly simple expression

$$\mathbb{E}[V_*^u|m, \overline{\mathcal{C}_1}] = \int_0^1 \mathbb{P}(V_*^u > v|m, \overline{\mathcal{C}_1}) \mathrm{d}v = \frac{9+2m}{3(3+m)(4+m)}. \tag{2.44}$$

We now consider the case $\mathcal{C}_1$ where the first item is critical. By Lemma 2.8, each V_j for $j \geq 2$ is a deterministic function of G and W_j. All W_j for $j \geq 2$ are independent by Lemma 2.2, so

$$\mathbb{P}(V_* > v|g, \mathcal{C}_1) = \prod_{j=2}^{n} \mathbb{P}(V_j > v|g, \mathcal{C}_1) = (1-v)^{(n-1)} \mathbb{1}(v < g). \tag{2.45}$$

Integrating over G by Theorem 2.1, we have

$$\begin{aligned} \mathbb{P}(V_* > v|\mathcal{C}_1) &= \int_0^1 \mathbb{P}(V_* > v|g, \mathcal{C}_1) f_G(g) \mathrm{d}g \\ &= (1-v)^{(n-1)} \int_v^1 (2-2g) \mathrm{d}g \\ &= (1-v)^{(n-1)}(1-2v+v^2) \end{aligned} \tag{2.46}$$

and

$$\begin{aligned}\mathbb{E}[V_*|\mathcal{C}_1] &= \int_0^1 \mathbb{P}(V_* > v|\mathcal{C}_1)\mathrm{d}v \\ &= \int_0^1 (1-v)^{(n-1)}(1-2v+v^2)\mathrm{d}v \\ &= \frac{1}{n+2}. \end{aligned} \tag{2.47}$$

Finally, accounting for all cases of K using total expectation and Lemma 2.1,

$$\begin{aligned}\mathbb{E}[V_*] &= \mathbb{E}[V_*|\mathcal{C}_1]\mathbb{P}(\mathcal{C}_1) + \sum_{k=2}^{n}\mathbb{E}[V_*|\mathcal{C}_k]\mathbb{P}(\mathcal{C}_k) \\ &= \frac{1}{n}\mathbb{E}[V_*|\mathcal{C}_1] + \frac{1}{n}\sum_{k=2}^{n}\mathbb{E}[V_*|\mathcal{C}_k] \\ &\leq \frac{1}{n}\mathbb{E}[V_*|\mathcal{C}_1] + \frac{1}{n}\sum_{k=2}^{n}\mathbb{E}[V_u^*|\overline{\mathcal{C}_1}, m] \\ &= \frac{1}{n(2+n)} + \frac{1}{n}\sum_{m=0}^{n-2}\frac{9+2m}{3(3+m)(4+m)}. \end{aligned} \tag{2.48}$$

Throughout all of the analysis in this section, we have implicitly assumed that $\sum_{i=1}^n W_i > B$. Making this condition explicit gives the desired expression. ■

2.4.3 Exhaustive Rollout: 0–1 Knapsack Problem Analysis

We follow the same approach that was used for the subset sum problem and assume that the reader understands this analysis (and thus less detail is included here). We employ the results from Section 2.3, and we use the same definition for the jth insertion item that was used for the subset sum problem. Analogous to the jth insertion gap V_j, we define here the *jth insertion gain* Z_j, where

$$Z_j := \max\left(0,\ \mathbb{1}(W_j \leq B)\left(P_j + \sum_{i=1}^{L_j-1} P_i\mathbb{1}(i \neq j)\right) - \sum_{i=1}^{K-1} P_i\right). \tag{2.49}$$

The jth insertion gain is simply the positive part of the difference between the value of the solution obtained by using Blind-Greedy after moving item j to the front of the sequence, and the value of the solution from using Blind-Greedy on the original input sequence.

We will bound the expected insertion gains while conditioning on (G, W_{K-1}, P_{K-1}). Assuming $K > 1$, this is done in the following three lemmas for packed items, the critical item, and remaining items, just as we did for the subset sum problem. The lemma after these three handles the case where $K = 1$. We assume that $\sum_{i=1}^n W_i > B$ throughout the section.

Lemma 2.9. *For $K > 1$ and $j = 1, \dots, K-1$, the jth insertion gain satisfies*

$$Z_j = 0 \tag{2.50}$$

with probability one.

Proof. By the proof of Lemma 2.5. ■

Lemma 2.10. *For $K > 1$ and $j = K+1, \dots, n$, the jth insertion gain satisfies $Z_j \geq Z_j^l$ with probability one, where Z_j^l is a deterministic function of $(G, W_{K-1}, W_j, P_{K-1}, P_j)$, and conditioning only on (G, W_{K-1}, P_{K-1}) gives*

$$\begin{aligned} \mathbb{P}(Z_j^l \leq z | g, w_{K-1}, p_{K-1}, \overline{\mathcal{C}_1}) &= zg + \min(z + p_{K-1}, 1)\,(w_{K-1} - (g + w_{K-1} - 1)_+) \\ &\quad + (1 - g - w_{K-1})_+ \\ &=: \mathbb{P}(Z^l \leq z | g, w_{K-1}, p_{K-1}, \overline{\mathcal{C}_1}). \end{aligned} \tag{2.51}$$

Proof. Fix $K = k$ for any $k > 1$, and let the event $\overline{\mathcal{C}_1}$ be implicit. We define the lower bound random variable Z_j^l so that

$$Z_j^l = \begin{cases} Z_j, & L_j = k \vee L_j = k-1, \\ 0, & L_j \leq k-2 \vee L_j = j. \end{cases}$$

This means we have an exact characterization of the jth insertion gain when the insertion critical item is either k or $k-1$, and a worst case gain of zero value in other cases. Thus it can be seen that $Z_j \geq Z_j^l$ with probability one, and Z_j^l uniquely depends on the random variables $(G, W_{K-1}, W_j, P_{K-1}, P_j)$. Let $\mathcal{D}_k$, $\mathcal{D}_{k-1}$, and $\mathcal{D}_{(k-2)-}$ indicate the events $L_j = k$, $L_j = k-1$, and $L_j \leq k-2 \vee L_j = j$, respectively. Using an illustration similar to Figure 2.4 under the assumption that $G = g$ and $\boldsymbol{W_S} = \boldsymbol{w_s}$, we have that if we only allow W_j to be random, then by Lemma 2.4,

$$\begin{aligned} \mathbb{P}(\mathcal{D}_k | g, w_{k-1}, p_{k-1}, p_j) &= g, & (2.52) \\ \mathbb{P}(\mathcal{D}_{k-1} | g, w_{k-1}, p_{k-1}, p_j) &= w_{k-1} - (g + w_{k-1} - 1)_+, & (2.53) \\ \mathbb{P}(\mathcal{D}_{(k-2)-} | g, w_{k-1}, p_{k-1}, p_j) &= (1 - g - w_{k-1})_+. & (2.54) \end{aligned}$$

Note that these expressions do not depend on any of the $\boldsymbol{P_S}$ values since item weights and profits are independent. For each of the above cases, we can find the probability distribution for Z_j^l while allowing only P_j to be random, so that

$$\begin{aligned} \mathbb{P}(Z_j^l \leq z | \mathcal{D}_k, g, w_{k-1}, w_j, p_{k-1}) &= \mathbb{P}(P_j \leq z) = z, & (2.55) \\ \mathbb{P}(Z_j^l \leq z | \mathcal{D}_{k-1}, g, w_{k-1}, w_j, p_{k-1}) &= \mathbb{P}(P_j - P_{k-1} \leq z | p_{k-1}) = \min(z + p_{k-1}, 1), & (2.56) \\ \mathbb{P}(Z_j^l \leq z | \mathcal{D}_{(k-2)-}, g, w_{k-1}, w_j, p_{k-1}) &= \mathbb{P}(0 \leq z) = 1. & (2.57) \end{aligned}$$

Again, these expressions do not depend on any of the $\boldsymbol{W_S}$ values by item profit and weight independence. Then, combining terms and noting that the above functions do not depend on all of the conditioned parameters, we have that if we only condition on (G, W_{k-1}, P_{k-1}),

$$\begin{aligned}\mathbb{P}(Z_j^l \leq z | g, w_{k-1}, p_{k-1}) &= \mathbb{P}(Z_j^l \leq z | \mathcal{D}_k, g, w_{k-1}, p_{k-1}) \mathbb{P}(\mathcal{D}_k | g, w_{k-1}, p_{k-1}) \\ &\quad + \mathbb{P}(Z_j^l \leq z | \mathcal{D}_{k-1}, g, w_{k-1}, p_{k-1}) \mathbb{P}(\mathcal{D}_{k-1} | g, w_{k-1}, p_{k-1}) \\ &\quad + \mathbb{P}(Z_j^l \leq z | \mathcal{D}_{(k-2)-}, g, w_{k-1}, p_{k-1}) \mathbb{P}(\mathcal{D}_{(k-2)-} | g, w_{k-1}, p_{k-1}) \\ &= zg + \min(z + p_{k-1}, 1)\,(w_{k-1} - (g + w_{k-1} - 1)_+) \\ &\quad + (1 - g - w_{k-1})_+. \end{aligned} \tag{2.58}$$

The analysis holds for all $k > 1$, so we replace k with K, which yields the expression in the lemma. ■

Lemma 2.11. *For $K > 1$, the Kth insertion gap satisfies $Z_K \geq Z_K^l$ with probability one, where Z_K^l is a deterministic function of $(G, W_{K-1}, W_K, P_{K-1}, P_K)$, and conditioning only on (G, W_{K-1}, P_{K-1}) gives*

$$\begin{aligned}&\mathbb{P}(Z_K^l \leq z | g, w_{K-1}, p_{K-1}, \overline{\mathcal{C}_1}) \\ &\quad = \frac{1}{1-g}\left(\min(z + p_{K-1}, 1)\,(w_{k-1} - (g + w_{K-1} - 1)_+) + (1 - g - w_{K-1})_+\right) \\ &\quad =: \mathbb{P}(\widetilde{Z}^l \leq z | g, w_{K-1}, p_{K-1}, \overline{\mathcal{C}_1}). \end{aligned} \tag{2.59}$$

Proof. Fix $K = k$ for $k > 1$ and make the event $\overline{\mathcal{C}_1}$ implicit. We define the lower bound random variable Z_k^l so that

$$Z_k^l := \begin{cases} Z_k, & L_k = k-1, \\ 0, & L_k \leq k-2 \vee L_k = k. \end{cases}$$

This random variable assumes a worst-case bound of zero gain if item $k-1$ becomes infeasible. By definition, we have $Z_k \geq Z_k^l$ with probability one and that Z_k^l is uniquely determined by $(G, W_{k-1}, W_j, P_{k-1}, P_j)$. Let $\mathcal{D}_{k-1}$ be the event that $L_k = k-1$ and let $\mathcal{D}_{(k-2)-}$ indicate the event $L_k \leq k-2 \vee L_k = k$. By Lemma 2.4, we have that for $G = g$, item k has distribution $\mathcal{U}[g, 1]$. Using the analysis in the previous lemma but restricted to the interval $[g, 1]$, we have

$$\mathbb{P}(\mathcal{D}_{k-1} | g, w_{k-1}, p_{k-1}, p_k) = \frac{1}{1-g}\left(w_{k-1} - (g + w_{k-1} - 1)_+\right), \tag{2.60}$$

$$\mathbb{P}(\mathcal{D}_{(k-2)-} | g, w_{k-1}, p_{k-1}, p_k) = \frac{1}{1-g}(1 - g - w_{k-1})_+. \tag{2.61}$$

By the independence of item weights and profits, the following results carry over from the proof of the previous lemma:

$$\mathbb{P}(Z_k^l \leq z | \mathcal{D}_{k-1}, g, w_{k-1}, w_k, p_{k-1}) = \mathbb{P}(P_k - P_{k-1} \leq z | p_{k-1}) = \min(z + p_{k-1}, 1), \tag{2.62}$$

$$\mathbb{P}(Z_k^l \leq z | \mathcal{D}_{(k-2)-}, g, w_{k-1}, w_k, p_{k-1}) = \mathbb{P}(0 \leq z) = 1. \tag{2.63}$$

We then have that if we only condition on (G, W_{k-1}, P_{k-1}),

$$\begin{aligned}\mathbb{P}(Z_k^l \leq & z|g, w_{k-1}, p_{k-1}) \\ &= \mathbb{P}(Z_k^l \leq z|\mathcal{D}_{k-1}, g, w_{k-1}, p_{k-1})\mathbb{P}(\mathcal{D}_{k-1}|g, w_{k-1}, p_{k-1}) \\ &\quad + \mathbb{P}(Z_k^l \leq z|\mathcal{D}_{(k-2)-}, g, w_{k-1}, p_{k-1})\mathbb{P}(\mathcal{D}_{(k-2)-}|g, w_{k-1}, p_{k-1}) \\ &= \frac{1}{1-g}\left(\min(z+p_{k-1}, 1)\left(w_{k-1} - (g + w_{k-1} - 1)_+\right) + (1 - g - w_{k-1})_+\right).\end{aligned} \tag{2.64}$$

The analysis is valid for all $k > 1$, so we replace k with K to obtain the expression in the lemma. ■

We now define $Z_*(n)$ as the gain given by the first iteration of the rollout algorithm on an instance with n items,

$$Z_*(n) := \max(Z_1, \ldots, Z_n). \tag{2.65}$$

For the rest of the section, we will usually refer to $Z_*(n)$ simply as Z_*.

Proof of Theorem 2.4. We proceed in a fashion nearly identical to the proof of Theorem 2.2. We have that for $K = k > 1$, $Z_* \geq Z_*^l$ with probability one, where

$$Z_*^l := \max(Z_k^l, Z_{k+1}^l, \ldots, Z_n^l). \tag{2.66}$$

This makes use of Lemmas 2.9 - 2.11. By Lemmas 2.10 and 2.11, each Z_j^l for $j \geq k$ is a deterministic function of $(G, W_{k-1}, W_j, P_{k-1}, P_j)$. Lemma 2.4 gives that item weights W_j for $j > k$ independently follow the distribution $\mathcal{U}[0,1]$, and W_k independently follows the distribution $\mathcal{U}[g,1]$. As a result, conditioning on only (G, W_{k-1}, P_{k-1}) makes Z_j^l independent for $j \geq k$, and then by the definition of the maximum,

$$\begin{aligned}\mathbb{P}(Z_*^l \leq z|g, & w_{k-1}, p_{k-1}, k, \overline{\mathcal{C}_1}) \\ &= \mathbb{P}(Z_k^l \leq z|g, w_{k-1}, p_{k-1}, \overline{\mathcal{C}_1}) \prod_{j=k+1}^{n} \mathbb{P}(Z_j^l \leq z|g, w_{k-1}, p_{k-1}, \overline{\mathcal{C}_1}) \\ &= \mathbb{P}(\widetilde{Z}^l \leq z|g, w_{k-1}, p_{k-1}, \overline{\mathcal{C}_1})\left(\mathbb{P}(Z^l \leq z|g, w_{k-1}, p_{k-1}, \overline{\mathcal{C}_1})\right)^{(n-k)}.\end{aligned} \tag{2.67}$$

In the remainder of the proof, we first integrate over the conditioned variables and then consider the case $\mathcal{C}_1$. For the integrals, we adopt some simplified notation to make expressions more manageable. As with the subset sum problem, let $M := n - K$. Also define

$$\begin{aligned}
\pi_+ &:= g, && (2.68)\\
\pi_0 &:= w_{K-1} - (g + w_{k-1} - 1)_+, && (2.69)\\
\pi_- &:= (1 - g - w_{k-1}), && (2.70)\\
\widetilde{\pi}_0 &:= \frac{1}{1-g}\left(w_{k-1} - (g + w_{k-1} - 1)_+\right), && (2.71)\\
\widetilde{\pi}_- &:= \frac{1}{1-g}(1 - g - w_{k-1})_+. && (2.72)
\end{aligned}$$

This allows us to write (2.67) as

$$\begin{aligned}
&\mathbb{P}(Z_*^l \le z | g, w_{k-1}, p_{k-1}, m, \overline{\mathcal{C}_1})\\
&\quad = \left(\min(z + p_{k-1}, 1)\widetilde{\pi}_0 + \widetilde{\pi}_-\right)\left(z\pi_+ + \min(z + p_{k-1}, 1)\pi_0 + \pi_-\right)^m. \qquad (2.73)
\end{aligned}$$

Integrating over p_{k-1}, which follows density $\mathcal{U}[0,1]$,

$$\begin{aligned}
&\mathbb{P}(Z_*^l \le z | g, w_{k-1}, m, \overline{\mathcal{C}_1})\\
&\quad = \int \mathbb{P}(Z_*^l \le z | g, w_{k-1}, p_{k-1}, m, \overline{\mathcal{C}_1}) f_{P_{k-1}}(p_{k-1}) \mathrm{d}p_{k-1}\\
&\quad = \int_0^{1-z} \left((z + p_{k-1})\widetilde{\pi}_0 + \widetilde{\pi}_-\right)\left(z\pi_+ + (z + p_{k-1})\pi_0 + \pi_-\right)^m \mathrm{d}p_{k-1}\\
&\qquad + \int_{1-z}^{1} \left(\widetilde{\pi}_0 + \widetilde{\pi}_-\right)\left(z\pi_+ + \pi_0 + \pi_-\right)^m \mathrm{d}p_{k-1}\\
&\quad = (\widetilde{\pi}_0 + \widetilde{\pi}_-)(\pi_0 + \pi_- + \pi_+ z)^m z + \frac{1}{(m+1)(m+2)\pi_0^2}\cdot\\
&\qquad \big((\pi_0 + P_n + \pi_+ z)^{m+1}(\pi_0\widetilde{\pi}_0(m+1) + \pi_0\widetilde{\pi}_-(m+2) - \widetilde{\pi}_0\pi_- - \widetilde{\pi}_0\pi_+ z)\\
&\qquad - (\pi_- + (\pi_0 + \pi_+)z)^{m+1}((2+m)\pi_0\widetilde{\pi}_- - \pi_-\widetilde{\pi}_0 + \widetilde{\pi}_0(\pi_0 + m\pi_0 - \pi_+)z)\big). \qquad (2.74)
\end{aligned}$$

At this point, it is useful to evaluate separately the cases where $g + w_{k-1} < 1$ and $g + w_{k-1} \ge 1$. Let $\mathcal{E}$ indicate the event that $g + w_{k-1} < 1$ holds, and let $\overline{\mathcal{E}}$ be the complement of this event. Also, define $A := W_{K-1}$. This allows us to define

$$\begin{aligned}
\mathbb{P}(Z_*^l \le z | g, w_{k-1}, m, \overline{\mathcal{C}_1})_{\mathcal{E}} &:= \mathbb{P}(Z_*^l \le z | g, w_{k-1}, m, \overline{\mathcal{C}_1})\mathbb{1}(g + w_{k-1} < 1), && (2.75)\\
\mathbb{P}(Z_*^l \le z | g, w_{k-1}, m, \overline{\mathcal{C}_1})_{\overline{\mathcal{E}}} &:= \mathbb{P}(Z_*^l \le z | g, w_{k-1}, m, \overline{\mathcal{C}_1})\mathbb{1}(g + w_{k-1} \ge 1), && (2.76)
\end{aligned}$$

so that

$$\mathbb{P}(Z_*^l \le z | g, w_{k-1}, m, \overline{\mathcal{C}_1}) = \mathbb{P}(Z_*^l \le z | g, w_{k-1}, m, \overline{\mathcal{C}_1})_{\mathcal{E}} + \mathbb{P}(Z_*^l \le z | g, w_{k-1}, m, \overline{\mathcal{C}_1})_{\overline{\mathcal{E}}}. \qquad (2.77)$$

Starting with the case where $\mathcal{E}$ holds and substituting A for W_{k-1},

$$\begin{aligned}
&\mathbb{P}(Z_*^l \leq z | g, a, m, \overline{\mathcal{C}_1})_{\mathcal{E}} \\
&\quad = z(1-g+gz)^m + \frac{(1-g+gz)^{m+1}(1-g+m-gm-gz)}{(1-g)a(m+1)(m+2)} \\
&\qquad - \frac{(1-g+gz+a(1-z))^{m+1}\left(1-g+m-gm-gz+a(-1-m+z+mz)\right)}{(1-g)a(m+1)(m+2)}.
\end{aligned} \tag{2.78}$$

We now wish to calculate

$$\mathbb{P}(Z_*^l \leq z | m, \overline{\mathcal{C}_1})_{\mathcal{E}} \;:=\; \int_0^1 \int_0^{1-g} \mathbb{P}(Z_*^l \leq z | g, a, m, \overline{\mathcal{C}_1})_{\mathcal{E}} f_A(a) f_G(g) \mathrm{d}a\mathrm{d}g. \tag{2.79}$$

The evaluation of this integral is given in Section 2.8.2, which shows

$$\mathbb{P}(Z_*^l \leq z | m, \overline{\mathcal{C}_1})_{\mathcal{E}} \;=\; \rho_1(m,z) + \sum_{j=1}^{m+1} \rho_{2j}(m,z) + \rho_3(m,z) + \rho_4(m,z), \tag{2.80}$$

where

$$\rho_1(m,z) \;=\; -\frac{2z\left(2+m^2(-1+z)^2+m(-1+z)(-3+5z)-2z\left(3-3z+z^{2+m}\right)\right)}{(1+m)(2+m)(3+m)(-1+z)^3}, \tag{2.81}$$

$$\rho_{2j}(m,z) \;=\; \frac{2z^{3+m}(j+(2+m)(-2+z)-jz)}{j(-3+j-m)(-2+j-m)(1+m)(2+m)(-1+z)^2}, \tag{2.82}$$

$$+\frac{2z^j(-j(1+m)(-1+z)+(2+m)(-1+m(-1+z)+2z))}{j(-3+j-m)(-2+j-m)(1+m)(2+m)(-1+z)^2}, \tag{2.83}$$

$$\rho_3(m,z) \;=\; -\frac{2H(m+1)\left(-1+m(-1+z)+2z+(-2+z)z^{3+m}\right)}{(1+m)(2+m)(3+m)(-1+z)^2}, \tag{2.84}$$

$$\rho_4(m,z) \;=\; -\frac{2}{(2+m)^2(3+m)(-1+z)} - \frac{2z^{2+m}}{(2+m)^2} + \frac{2z^{3+m}}{(2+m)^2(3+m)(-1+z)}. \tag{2.85}$$

Since we are ultimately interested in the expected value of Z_*^l, we wish to evaluate

$$\overline{\mathbb{E}}[Z_*^l | m, \overline{\mathcal{C}_1}]_{\mathcal{E}} := \int_0^1 \mathbb{P}(Z_*^l \leq z | m, \overline{\mathcal{C}_1})_{\mathcal{E}} \mathrm{d}z. \tag{2.86}$$

Recall that $\overline{\mathbb{E}}[\cdot] := 1 - \mathbb{E}[\cdot]$. Using the definition

$$\xi_j(m) := \int_0^1 \rho_j(m,z)\mathrm{d}z, \tag{2.87}$$

we have

$$\overline{\mathbb{E}}[Z_*^l|m,\overline{\mathcal{C}_1}]_{\mathcal{E}} = \xi_1(m) + \sum_{j=1}^{m+1} \xi_{2j}(m) + \xi_3(m) + \xi_4(m), \tag{2.88}$$

where

$$\xi_1(m) = -\frac{2H(m+1)(3+m-H(m+3)(2+m))}{(m+1)(m+2)(m+3)}, \tag{2.89}$$

$$\xi_{2j}(m) = \frac{2\left(-(-3+j-m)(2+m)+\left(j+(2+m)^2\right)H(j)-\left(j+(2+m)^2\right)H(m+3)\right)}{j(-3+j-m)(-2+j-m)(1+m)(2+m)}, \tag{2.90}$$

$$\xi_3(m) = \frac{2(H(m+3)-1)}{(2+m)^2(3+m)}, \tag{2.91}$$

$$\xi_4(m) = -\frac{(2+m)(17+5m)-2(3+m)(4+m)H(m+2)}{(m+1)(m+2)(m+3)}. \tag{2.92}$$

This completes the case for the event $\mathcal{E}$ (i.e. $g + w_{k-1} < 1$). Now when the event $\overline{\mathcal{E}}$ holds,

$$\begin{aligned}\mathbb{P}(Z_*^l \le z|g,a,m,\overline{\mathcal{C}_1})_{\overline{\mathcal{E}}} &= z(1-g+gz)^m - \frac{(1-2g+(1-g)m)z^{2+m}}{(1-g)^2(1+m)(2+m)} \\ &\quad + \frac{((1-g)(1+m)-gz)(1-g+gz)^{1+m}}{(1-g)^2(1+m)(2+m)}.\end{aligned} \tag{2.93}$$

Continuing as we did with the case $\mathcal{E}$,

$$\begin{aligned}\mathbb{P}(Z_*^l \le z|m,\overline{\mathcal{C}_1})_{\overline{\mathcal{E}}} &:= \int_0^1 \int_{1-g}^1 \mathbb{P}(Z_*^l \le z|g,a,m,\overline{\mathcal{C}_1})_{\overline{\mathcal{E}}} f_A(a) f_G(g)\mathrm{d}a\mathrm{d}g \\ &= \int_0^1 g\mathbb{P}(Z_*^l \le z|g,a,m,\overline{\mathcal{C}_1})_{\overline{\mathcal{E}}} f_G(g)\mathrm{d}g,\end{aligned} \tag{2.94}$$

where we have used the fact that the expression $\mathbb{P}(Z_*^l \le z|g,a,m,\overline{\mathcal{C}_1})_{\overline{\mathcal{E}}}$ is not a function of a. Evaluation of this integral is given in Section 2.8.3; the expression is

$$\begin{aligned}
&\mathbb{P}(Z_*^l \leq z|m,\overline{\mathcal{C}_1})_{\overline{\mathcal{E}}} \\
&= -\frac{2z\left(1+m-3z-mz+z^{2+m}(3+m-(1+m)z)\right)}{(1+m)(2+m)(3+m)(-1+z)^3} + \frac{-2z}{(m+1)(m+2)}\sum_{j=1}^{m+1}\frac{z^{m+1-j}}{j} \\
&\quad + \frac{2z}{(m+1)(m+2)^2(1-z)} + \frac{2(1+m+z)}{(m+1)(m+2)^2(m+3)(1-z)^2} - \frac{(6+2m)z^{m+2}}{(m+1)(m+2)} \\
&\quad + \frac{z^{m+2}}{m+1} + \frac{2H(m+1)z^{m+2}}{(m+1)(m+2)} - \frac{2(1+m+2z)z^{m+2}}{(m+1)(m+2)^2(1-z)} \\
&\quad - \frac{2(1+m+z)z^{m+3}}{(m+1)(m+2)^2(m+3)(1-z)^2}.
\end{aligned} \tag{2.95}$$

We again calculate the following term for the expected value

$$\begin{aligned}
\overline{\mathbb{E}}[Z_*^l|m,\overline{\mathcal{C}_1}]_{\overline{\mathcal{E}}} &:= \int_0^1 \mathbb{P}(Z_*^l \leq z|m)\mathrm{d}z \\
&= \frac{20+10m+m^2-2(3+m)H(1+m)}{(2+m)(3+m)^2} \\
&\quad + \sum_{j=1}^{m+1}\frac{2}{j(-3+j-m)(1+m)(2+m)}.
\end{aligned} \tag{2.96}$$

Bringing together both cases $\mathcal{E}$ and $\overline{\mathcal{E}}$, we have

$$\begin{aligned}
&\mathbb{E}[Z_*^l|m,\overline{C}_1] \\
&= \int_0^1 (1-\mathbb{P}(Z_*^l \leq z|m,\overline{\mathcal{C}_1}))\mathrm{d}z \\
&=1 - \int_0^1 \mathbb{P}(Z_*^l \leq z|m,\overline{\mathcal{C}_1})\mathrm{d}z \\
&=1 - \overline{\mathbb{E}}[Z_*^l|m,\overline{\mathcal{C}_1}]_{\mathcal{E}} - \overline{\mathbb{E}}[Z_*^l|m,\overline{\mathcal{C}_1}]_{\overline{\mathcal{E}}} \\
&=1 + \frac{1}{(m+1)(m+2)^3(m+3)^2}\big((186+472m+448m^2+203m^3+45m^4+4m^5) \\
&\quad + (-244-454m-334m^2-124m^3-24m^4-2m^5)H(m+1) \\
&\quad +(-48-88m-60m^2-18m^3-2m^4)(H(m+1))^2\big) \\
&\quad + \sum_{j=1}^{m+1} T(j,m),
\end{aligned} \tag{2.97}$$

where

$$T(j,m) := \frac{2\left(-4+j-4m+jm-m^2-\left(j+(2+m)^2\right)H(j)\right)}{j(-3+j-m)(-2+j-m)(1+m)(2+m)} + \frac{2(j+(2+m)^2)H(3+m)}{j(-3+j-m)(-2+j-m)(1+m)(2+m)}. \tag{2.98}$$

If the first item is critical,

$$\mathbb{P}(Z_* \le z | g, m, \mathcal{C}_1) = (1-g+gz)^m. \tag{2.99}$$

Marginalizing over G and taking the expectation gives

$$\begin{aligned}\mathbb{P}(Z_* \le z | m, \mathcal{C}_1) &= \int_0^1 \mathbb{P}(Z_*^l \le z | g, m, \mathcal{C}_1) f_G(g)\mathrm{d}g \\ &= \int_0^1 (1-g+gz)^m (2-2g)\mathrm{d}g \\ &= \frac{2\left(1+m-2z-mz+z^{2+m}\right)}{(1+m)(2+m)(-1+z)^2}.\end{aligned} \tag{2.100}$$

$$\begin{aligned}\mathbb{E}(Z_* | m, \mathcal{C}_1) &= 1 - \int_0^1 \mathbb{P}(Z_* \le z | m, \mathcal{C}_1)\mathrm{d}z \\ &= 1 + \frac{2}{2+m} - \frac{2H(m+1)}{m+1}.\end{aligned} \tag{2.101}$$

Since the event $\mathcal{C}_1$ indicates $M = n-1$,

$$\mathbb{E}(Z_* | \mathcal{C}_1) = 1 + \frac{2}{n+1} - \frac{2H(n)}{n}. \tag{2.102}$$

Finally, accounting for the distribution of M with Lemma 2.1 gives the expression in the theorem:

$$\begin{aligned}&\mathbb{E}\left[Z_*(n) \middle| \sum_{i=1}^n W_i > B\right] \ge 1 + \frac{2}{n(n+1)} - \frac{2H(n)}{n^2} \\ &+\frac{1}{n}\sum_{m=0}^{n-2}\left(\sum_{j=1}^{m+1} T(j,m) + \left((186+472m+448m^2+203m^3+45m^4+4m^5)\right.\right. \\ &-(244+454m+334m^2+124m^3+24m^4+2m^5)H(m+1) \\ &\left.\left.-(48+88m+60m^2+18m^3+2m^4)(H(m+1))^2\right)\frac{1}{(m+1)(m+2)^3(m+3)^2}\right),\end{aligned} \tag{2.103}$$

where

$$\begin{aligned} T(j,m) &:= \frac{2\left(-4+j-4m+jm-m^2-\left(j+(2+m)^2\right)H(j)\right)}{j(-3+j-m)(-2+j-m)(1+m)(2+m)} \\ &\quad + \frac{2(j+(2+m)^2)H(3+m)}{j(-3+j-m)(-2+j-m)(1+m)(2+m)}. \end{aligned} \tag{2.104}$$

■

We observe that the nested summation term may be omitted without significant loss in the performance bound. This is accomplished by showing that the argument of the sum is always positive.

Lemma 2.12. *For all $m > 0$ and $1 \leq j \leq m+1$,*

$$\frac{\left(-4+j-4m+jm-m^2-\left(j+(2+m)^2\right)H(j)+\left(j+(2+m)^2\right)H(3+m)\right)}{j(-3+j-m)(-2+j-m)(1+m)(2+m)} > 0. \tag{2.105}$$

Proof. The denominator is always positive, so we focus on the numerator. The numerator consists of two parts,

$$N_1(j,m) := (4-j)(m+1)+m^2, \tag{2.106}$$

$$N_2(j,m) := (j+(2+m)^2)\sum_{i=j+1}^{m+3}\frac{1}{i}. \tag{2.107}$$

Our goal is to show that $N_2(j,m) > N_1(j,m)$ always holds. The difference equation for $N_2(j,m)$ with respect to j satisfies

$$\begin{aligned} \Delta(N_2(j,m)) &:= N_2(j+1,m)-N_2(j,m) \\ &= \sum_{i=j+2}^{m+3}\frac{1}{i} + (j+(2+m)^2)\sum_{i=j+2}^{m+3}\frac{1}{i} - (j+(2+m)^2)\sum_{i=j+1}^{m+3}\frac{1}{i} \\ &= \sum_{i=j+2}^{m+3}\frac{1}{i} - \frac{j+(2+m)^2}{j+1} \\ &\leq \frac{m-j+2}{j+2} - \frac{j+(2+m)^2}{j+1} \\ &< \frac{m-j+2}{j+1} - \frac{j+(2+m)^2}{j+1} \\ &= \frac{-2-3m-m^2-2j}{j+1} \\ &\leq \frac{-4-3m-m^2}{m+2}. \end{aligned} \tag{2.108}$$

For the other term, we have

$$\Delta(N_1(j,m)) = -(m+1). \tag{2.109}$$

Both $N_1(j,m)$ and $N_2(j,m)$ are decreasing in j and $N_2(j,m)$ decreases at a greater rate. We approximate $N_2(j,m)$ with the following:

$$H(m+3) - H(j) = \sum_{i=j+1}^{m+3} \frac{1}{i} \geq \int_{j+1}^{m+4} \frac{1}{x}\mathrm{d}x = \log\left(\frac{m+4}{j+1}\right). \tag{2.110}$$

Looking at $j=1$,

$$\begin{aligned} N_1(1,m) &= 3+3m+m^2, && (2.111)\\ N_2(1,m) &\geq (5+4m+m^2)\log\left(\frac{m+4}{2}\right), && (2.112) \end{aligned}$$

guaranteeing $N_2(1,m) > N_1(1,m)$. With consideration of starting points and slopes for the two numerator terms, ensuring that $N_2(m+1,m) > N_1(m+1,m)$ is sufficient for the lemma. We have

$$\begin{aligned} N_1(m+1,m) &= 3+2m, && (2.113)\\ N_2(m+1,m) &= (m+1+(2+m)^2)\left(\frac{1}{m+2}+\frac{1}{m+3}\right) \\ &> (5+5m+m^2)\left(\frac{2}{m+3}\right) \\ &= \frac{10+10m+2m^2}{m+3} \\ &> 3+2m. && (2.114) \end{aligned}$$

■

Proof of Theorem 2.5. We will show that $\lim_{n\to\infty} \mathbb{E}[Z_*|\cdot] = 1$, so we are interested in bounding the rate at which $1-\mathbb{E}[Z_*|\cdot]$ approaches 0. Accordingly, we are only concerned with the negative terms in (2.103). The magnitudes of these terms are

$$\begin{aligned} T_1(n) &= \frac{2H(n)}{n^2}, && (2.115)\\ T_2(n) &= \frac{1}{n}\sum_{m=0}^{n-2} \frac{(244+454m+334m^2+24m^4+2m^5)H(m+1)}{(m+1)(m+2)^3(m+3)^2}, && (2.116)\\ T_3(n) &= \frac{1}{n}\sum_{m=0}^{n-2} \frac{(48+88m+60m^2+18m^3+2m^4)(H(m+1))^2}{(m+1)(m+2)^3(m+3)^2}. && (2.117) \end{aligned}$$

The second and third sum arguments are decreasing in m, so they are bounded by their respective integrals. Using a logarithmic bound on the harmonic numbers, we have

$$\begin{aligned}
T_1(n) &= O\left(\frac{\log n}{n^2}\right), && (2.118)\\
T_2(n) &= \frac{1}{n}\sum_{m=0}^{n-2} O\left(\frac{\log m}{m}\right)\\
&= \frac{1}{n}\int_0^{n-1} O\left(\frac{\log m}{m}\right) \mathrm{d}m\\
&= O\left(\frac{\log^2 n}{n}\right), && (2.119)\\
T_3(n) &= \frac{1}{n}\sum_{m=0}^{n-2} O\left(\frac{\log^2 m}{m^2}\right)\\
&= \frac{1}{n}\int_0^{n-1} O\left(\frac{\log^2 m}{m^2}\right) \mathrm{d}m\\
&= O\left(\frac{\log^2 n}{n^2}\right). && (2.120)
\end{aligned}$$

The largest growth rate is $O(\frac{\log^2 n}{n})$. Also, we have that $\lim_{n\to\infty} \mathbb{E}[Z_*(n)|\cdot] = 1$ since the gain has a natural upper bound of unit value.

2.5 Consecutive Rollout

The Consecutive-Rollout algorithm is shown in Algorithm 3. The algorithm takes as input a sequence of items $\boldsymbol{I}$ and capacity b and makes calls to Blind-Greedy as a subroutine. At iteration i, the algorithm calculates the value (U_+) of adding item i to the solution and using Blind-Greedy on the remaining items, and the value (U_-) of not adding the item to the solution and using Blind-Greedy thereafter. The item is then added to the solution only if the former valuation (U_+) is larger. Again, we assume a consistent tie-breaking method.

We again only focus on the result of the first iteration of the algorithm; bounds from the first iteration are valid for future iterations. A single iteration of Consecutive-Rollout effectively takes the best of two solutions, the solution obtained by Blind-Greedy, and the solution obtained from using Blind-Greedy after removing the first item. Let $V_*(n)$ denote the gap obtained by a single iteration of the rollout algorithm for the subset sum problem with n items under the stochastic model. The results in this section hold for $n \geq 3$ and are tight for $n = 3$, as the proofs only depend on the first three items. Employing this number of items strikes a balance between proof tractability and performance tightness.

```
Input: Item sequence $\boldsymbol{I} = \langle 1, \dots, n \rangle$ with profit sequence $\boldsymbol{p_I}$ and weight sequence $\boldsymbol{w_I}$,
    capacity $b$.
Output: Feasible solution sequence $\boldsymbol{S}$, value $U$.
 1: Initialize $\boldsymbol{S} \leftarrow \langle\rangle$, remaining item sequence $\overline{\boldsymbol{I}} \leftarrow \boldsymbol{I}$, $\bar{b} \leftarrow b$, $U \leftarrow 0$.
 2: for $i = 1$ to $n$ (each item) do
 3:     Estimate the value of adding item $i$, $(\cdot,\ U_+) = \text{Blind-Greedy}(\overline{\boldsymbol{I}}, \bar{b})$.
 4:     Estimate the value of skipping item $i$, $(\cdot,\ U_-) = \text{Blind-Greedy}(\overline{\boldsymbol{I}} \setminus \langle i \rangle, \bar{b})$.
 5:     if $U_+ > U_-$ (estimated value of adding the item is larger) then
 6:         Add item $i$ to solution sequence, $\boldsymbol{S} \leftarrow \boldsymbol{S} : \langle i \rangle$.
 7:         Update remaining capacity, $\bar{b} \leftarrow \bar{b} - w_i$, and value, $U \leftarrow U + p_i$.
 8:     end if
 9:     Remove item $i$ from the remaining item sequence, $\overline{\boldsymbol{I}} \leftarrow \overline{\boldsymbol{I}} \setminus \langle i \rangle$.
10: end for
11: Return $\boldsymbol{S}$, $U$.
```

Algorithm 3. Consecutive-Rollout

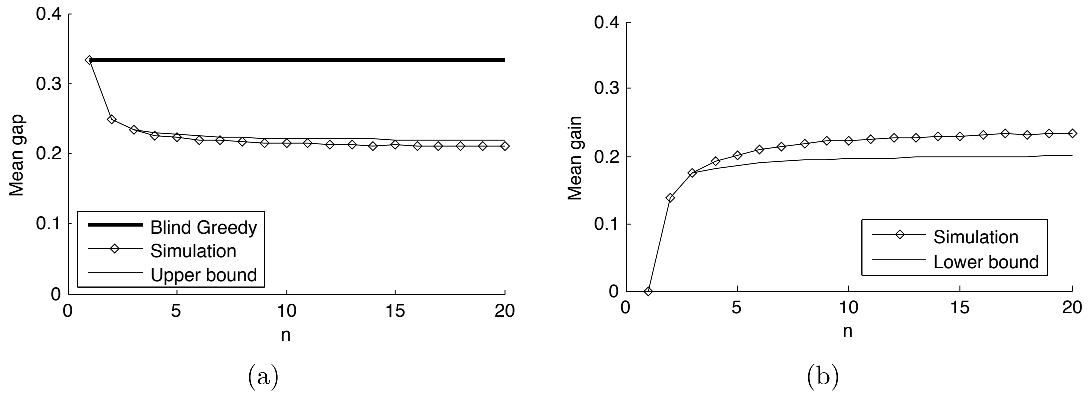


Figure 2.5. Performance bounds and simulated values for (a) expected gap $\mathbb{E}[V_*(n)|\cdot]$ and (b) expected gain $\mathbb{E}[Z_*(n)|\cdot]$ after running a single iteration of the Consecutive-Rollout algorithm on the subset sum problem and knapsack problem, respectively. For each n, the mean values are shown for 10^5 simulations.

Theorem 2.6. *For the subset sum problem with $n \geq 3$, the gap $V_*(n)$, obtained by running a single iteration of* Consecutive-Rollout*, satisfies*

$$\mathbb{E}\left[V_*(n) \middle| \sum_{i=1}^{n} W_i > B\right] \leq \frac{3 + 13n}{60n} \leq \frac{7}{30} \approx 0.233. \tag{2.121}$$

As expected, there is not a strong dependence on n for this algorithm. The bound is tight for $n = 3$, where it evaluates to $\frac{7}{30} \approx 0.233$. It is also clear that $\lim_{n\to\infty} \mathbb{E}[V_*(n)|\cdot] \leq \frac{13}{60} \approx 0.217$. The bounds are shown with simulated performance in Figure 2.5(a). A similar result holds for the 0–1 knapsack problem.

Theorem 2.7. *For the knapsack problem with $n \geq 3$, the gain $Z_*(n)$, obtained by running a single iteration of* Consecutive-Rollout, *satisfies*

$$\mathbb{E}\left[Z_*(n) \middle| \sum_{i=1}^{n} W_i > B\right] \geq \frac{-26+59n}{288n} \geq \frac{151}{864} \approx 0.175. \tag{2.122}$$

The bound is plotted with simulated values in Figure 2.5(b). Again, the bound is tight for $n = 3$ with a gain of $\frac{151}{864} \approx 0.175$. Asymptotically, $\lim_{n\to\infty} \mathbb{E}[Z_*(n)|\cdot] \geq \frac{59}{288} \approx 0.205$. The rest of this section is devoted to an example and the theorem proofs.

2.5.1 Example

We demonstrate the Consecutive-Rollout algorithm using the same subset sum problem that we used for the example in the previous section. Recall that we have $n = 6$, the weight sequence $\boldsymbol{w_I} = \langle 0.3,\ 0.8,\ 0.6,\ 0.4,\ 0.7,\ 0.1\rangle$, and a capacity $b = 2$. The solution given by Blind-Greedy has a value of 1.7, shown below.

$\boldsymbol{I}$	1	2	3	4	5	6	value
$\boldsymbol{w_I}$	0.3	0.8	0.6	0.4	0.7	0.1	
cum.	0.3	1.1	1.7	*2.1*	*2.8*	*2.9*	1.7

Starting with the first iteration ($i = 1$) of Consecutive-Rollout, we initially have $\overline{\boldsymbol{I}} = \boldsymbol{I}$, and we know that the value of including the first item is 1.7. The value of skipping the first item follows from the sequence $\boldsymbol{J} = \overline{\boldsymbol{I}} \setminus \langle 1\rangle$; we will again use $\boldsymbol{J}$ to denote the sequence that the algorithm is considering.

$\boldsymbol{J}$	2	3	4	5	6	value
$\boldsymbol{w_J}$	0.8	0.6	0.4	0.7	0.1	
cum.	0.8	1.4	1.8	*2.5*	*2.6*	1.8

The value of skipping the first item is 1.8. Since this value is greater than the value of including the first item, the first item is removed from the remaining item sequence.

The second iteration starts with solution sequence $\boldsymbol{S} = \langle\rangle$ and remaining item sequence $\overline{\boldsymbol{I}} = \langle 2, 3, 4, 5, 6\rangle$. We already know that the value of including item 2 is 1.8. The value of skipping item 2, via the sequence $\boldsymbol{J} = \overline{\boldsymbol{I}} \setminus \langle 2\rangle$, is also 1.8.

$\boldsymbol{J}$	3	4	5	6	value
$\boldsymbol{w_J}$	0.6	0.4	0.7	0.1	
cum.	0.6	1.0	1.7	1.8	1.8

We break the tie in favor of keeping the second item. Note that if we had chosen otherwise, the value given by additional iterations would have remained at 1.8.

The third iteration starts with solution sequence $\boldsymbol{S} = \langle 2\rangle$ and remaining item sequence $\overline{\boldsymbol{I}} = \langle 3, 4, 5, 6\rangle$. We already know that the value of including item 3 is 1.8. The value of skipping item 3, via the sequence $\boldsymbol{J} = \overline{\boldsymbol{I}} \setminus \langle 3\rangle$, is 2.0.

$\boldsymbol{S}$	2	$\boldsymbol{J}$	4	5	6	value
$\boldsymbol{w_S}$	0.8	$\boldsymbol{w_J}$	0.4	0.7	0.1	
cum.	0.8		1.2	1.9	2.0	2.0

Since the value of 2.0 is optimal, item 3 is not added to the solution set. The fourth iteration starts with the solution sequence $\boldsymbol{S} = \langle 2 \rangle$ and remaining item sequence $\overline{\boldsymbol{I}} = \langle 4, 5, 6 \rangle$. Since after the third iteration a value matching the capacity has been found, there are no improvements to be made, and it is not necessary to demonstrate additional iterations. The algorithm finishes with solution sequence $\boldsymbol{S} = \langle 2, 4, 5, 6 \rangle$.

In summary, Blind-Greedy gives the solution sequence $\boldsymbol{S} = \langle 1, 2, 3 \rangle$ and value 1.7. The first iteration of Consecutive-Rollout gives the solution sequence $\boldsymbol{S} = \langle 2, 3, 4 \rangle$ with value 1.8[3]. The second iteration of Consecutive-Rollout also gives the solution sequence $\boldsymbol{S} = \langle 2, 3, 4 \rangle$ with value 1.8. The third iteration of Consecutive-Rollout gives solution sequence $\boldsymbol{S} = \langle 2, 4, 5, 6 \rangle$ with value 2.0, which is optimal. All following iterations $i = 4, 5, 6$ give the same optimal solution and value.

2.5.2 Consecutive Rollout: Subset Sum Problem Analysis

The proof method for Theorem 2.6 is similar to the approach used for Theorem 2.2. Keeping Figure 2.1 in mind, we look at modifications to the Blind-Greedy solution caused by removing the first item. Removing the first item causes the other items to slide to the left and may make some remaining items feasible to pack. We determine bounds on the gap produced by this procedure while conditioning on the greedy gap G, the critical item K, and the item weights (W_K, W_{K+1}). We then take the minimum of this gap and the greedy gap, and integrate over conditioned variables to obtain the final bound. Our analysis is divided into lemmas based on the critical item K, where a separate lemma is given for the cases $K = 1$, $2 \leq K \leq n-1$, and $K = n$.

To formalize the behavior of Consecutive-Rollout, we introduce the following two definitions. The *drop critical item* L_1 is the index of the item that becomes critical when the first item is removed and thus satisfies

$$\begin{cases} \displaystyle\sum_{i=2}^{L_1-1} W_i \leq B < \sum_{i=2}^{L_1} W_i, & \text{if } \sum_{i=2}^{n} W_i > B, \\ L_1 = n+1, & \text{if } \sum_{i=2}^{n} W_i \leq B, \end{cases}$$

where the latter case signifies that all remaining items can be packed. The *drop gap* V_1 then has definition

$$V_1 := B - \sum_{i=2}^{L_1-1} W_i. \tag{2.123}$$

[3]Here we are again abusing notation so that $\boldsymbol{S}$ is the sequence given by some iteration of Consecutive-Rollout *and* Blind-Greedy on the remaining items, as opposed to the definition of $\boldsymbol{S}$ given in the algorithm description.

We are ultimately interested in the minimum of the drop gap and the greedy gap, which we refer to as the *minimum gap*. The minimum gap is the value obtained by the first iteration of the rollout algorithm:

$$V_*(n) := \min(G, V_1). \tag{2.124}$$

We will often write $V_*(n)$ simply as V_*. We will also use $\mathcal{C}_i$ to denote the event that item i is critical and $\overline{\mathcal{C}_{1n}}$ for the event that $2 \leq K \leq n-1$. Furthermore, recall that we have $\boldsymbol{P_I} = \boldsymbol{W_I}$ for the subset sum problem.

Lemma 2.13. *For $2 \leq K \leq n-1$, the expected minimum gap satisfies*

$$\mathbb{E}[V_*(n)|2 \leq K \leq n-1] \leq \frac{13}{60}. \tag{2.125}$$

Proof. Fix $K = k$ for $2 \leq k \leq n-1$. The drop gap, in general, may be a function of the weights of all remaining items. To make things more tractable, we define the random variable V_1^u that satisfies $V_1 \leq V_1^u$ with probability one and, as we will show, is a deterministic function of only (G, W_1, W_k, W_{k+1}). The variable V_1^u is specifically defined as

$$V_1^u := \begin{cases} V_1, & L_1 = k \vee L_1 = k+1, \\ B - \sum_{i=2}^{k+1} W_i, & L_1 \geq k+2. \end{cases} \tag{2.126}$$

In effect, V_1^u does not account for the additional reduction in the gap given if any of the items $i \geq k+2$ become feasible, so clearly, $V_1^u \geq V_1$.

To determine the distribution of V_1^u, we start by considering scenarios where $L_1 \geq k+2$ is not possible, and thus $V_1^u = V_1$. For $G = g$ and $\boldsymbol{W_I} = \boldsymbol{w_I}$, an illustration of the drop gap, as determined by (g, w_1, w_k, w_{k+1}), is shown in Figure 2.6. The knapsack is shown at the top of the figure with items packed from left to right, and at the bottom the drop gap v_1 is shown as a function of w_1. The shape of the function is justified by considering different sizes of w_1. As long as w_1 is smaller than $w_k - g$, the gap given by removing the first item increases at unit rate. As soon as $w_1 = w_k - g$, item k becomes feasible, and the gap jumps to zero. The gap then increases at unit rate, and another jump occurs when w_1 reaches $w_k - g + w_{k-1}$. The case shown in the figure satisfies $w_k - g + w_{k+1} + w_{k+2} > 1$. It can be seen that this is a sufficient condition for the event $L_1 \geq k+2$ to be impossible, since even if $w_1 = 1$, item $k+2$ cannot become feasible. It is for this reason that v_1 is uniquely determined by (g, w_1, w_k, w_{k+1}) here.

Continuing with the case shown in the figure, if we only condition on (g, w_k, w_{k+1}), we have by Lemma 2.4 that W_1 follows distribution $\mathcal{U}[0,1]$, meaning that the event $V_1 > v$ is given by the length of the bold regions on the w_1 axis. We explicitly describe the size of these regions. Assuming that $L_1 \leq k+1$, we derive the following expression:

$$\begin{aligned} \mathbb{P}(V_1 > v|g, w_k, w_{k+1}, &\overline{\mathcal{C}_{1n}}, L_1 \leq k+1) \\ &= (w_k - g) + (w_{k+1} - v)_+ + (1 - w_k + g - w_{k+1} - v)_+ \\ &\quad - (w_k - g + w_{k+1} - 1)_+, \quad v < g. \end{aligned} \tag{2.127}$$

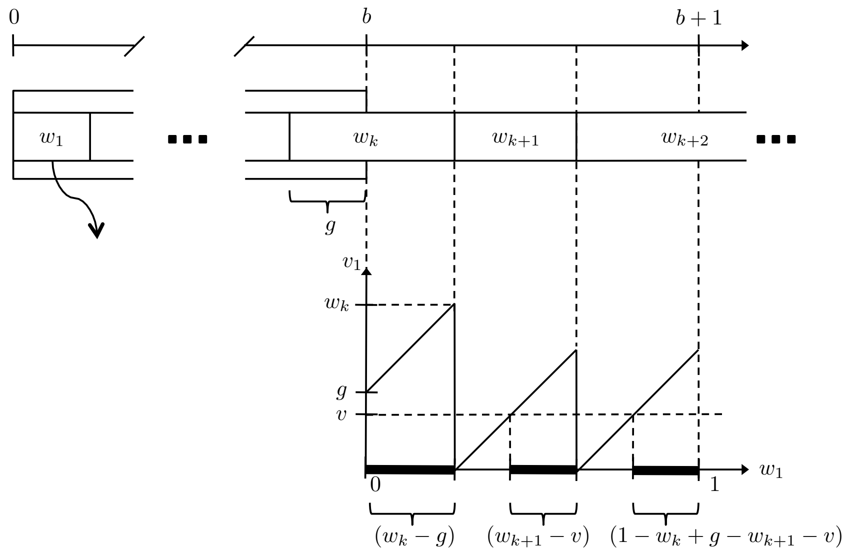


Figure 2.6. Gap v_1 as a function of w_1, parameterized by (g, w_k, w_{k+1}), resulting from the removal of the first item and assuming that $K = k$ with $2 \leq k \leq n-1$. The function starts at g and increases at unit rate, except at $w_1 = w_k - g$ and $w_1 = w_k - g + w_{k+1}$, where the function drops to zero. If we only condition on (g, w_k, w_{k+1}), the probability of the event $V_1 > v$ is given by the total length of the bold regions for $v < g$. Note that in the figure, $w_k - g + w_{k+1} < 1$, and the second two bold segments have positive length; these properties do not hold in general.

The first three terms in the expression come from the three bold regions shown in Figure 2.6. We have specified that $v < g$, so the length of the first segment is always $w_k - g$. For the second term, it is possible that $v > w_{k+1}$, so we only take the positive portion of $w_{k+1} - v$. In the third term, we take the positive portion to account for the cases where item $k+1$ does not become feasible, meaning $w_k - g + w_{k+1} > 1$, and if it is feasible, where v is greater than the height of the third peak, meaning $v > 1 - w_k + g - w_{k+1}$.

The last term is required for the case where item $k+1$ does not become feasible, as we must subtract the length of the bold region that potentially extends beyond $w_1 = 1$. Note that we always subtract one in this expression since it is not possible for the w_1 value, where $v_1 = v$ on the second peak, to be greater than one. To see this, assume the contrary, so that $v + w_k - g > 1$. This inequality is obtained since, on the second peak, we have $v_1 = g - w_k + w_1$ and the w_1 value that satisfies $v_1 = v$ is equal to $v + w_k - g$. The statement $v + w_k - g > 1$, however, violates our previously stated assumption that $g < v$.

We now argue that we, in fact, have $V_1 \leq V_1^u$ with probability one, where

$$\begin{aligned}\mathbb{P}(V_1^u > v | g, w_k, w_{k+1}, \overline{\mathcal{C}_{1n}}) &= (w_k - g) + (w_{k+1} - v)_+ + (1 - w_k + g - w_{k+1} - v)_+ \\ &\quad -(w_k - g + w_{k+1} - 1)_+, \quad v < g. \end{aligned} \tag{2.128}$$

We have simply replaced V_1 with V_1^u in (2.127) and removed the condition $L_1 \leq k+1$.

We already know that the expression is true for $L_1 \leq k+1$. For $L_1 \geq k+2$, we refer to Figure 2.6 and visualize the effect of a much smaller w_{k+2}, so that $w_k - g + w_{k+1} + w_{k+2} < 1$. This would yield four (or more) peaks in the v_1 function. To determine the probability of the event $V_1 > v$ while W_1 is random, we would have to evaluate the sizes of these extra peaks, which would be a function of w_{k+2}, w_{k+3}, etc. However, our definition of V_1^u does not account for the additional reductions in the gap given by items beyond $k+1$. We have already shown that $V_1 \leq V_1^u$, and now, clearly, V_1^u is a deterministic function of (G, W_1, W_k, W_{k+1}), and (2.128) is justified.

We now evaluate the minimum of V_1^u and G and integrate over the conditioned variables. To begin, note that conditioning on the gap G makes V_1^u and G independent, so,

$$\mathbb{P}(V_1^u > v, G > v | \overline{\mathcal{C}_{1n}}, g, w_k, w_{k+1}) = \mathbb{P}(V_1^u > v | \overline{\mathcal{C}_{1n}}, g, w_k, w_{k+1}) \mathbb{1}(v < g). \tag{2.129}$$

Marginalizing over W_{k+1}, which has uniform density according to Lemma 2.4, gives

$$\begin{aligned}
&\mathbb{P}(V_1^u > v, G > v | \overline{\mathcal{C}_{1n}}, g, w_k) \\
&\quad = \int_0^1 \mathbb{P}(V_1^u > v, g > v | \overline{\mathcal{C}_{1n}}, g, w_k, w_{k+1}) f_{W_{k+1}}(w_{k+1}) \mathrm{d}w_{k+1} \\
&\quad = \Bigg((w_k - g) + \int_v^1 (w_{k+1} - v) \mathrm{d}w_{k+1} - \int_{1+g-w_k}^1 (w_k - g + w_{k+1} - 1) \mathrm{d}w_{k+1} \\
&\qquad + \int_0^{1-w_k+g+v} (1 - w_k + g - w_{k+1} - v)_+ \mathrm{d}w_{k+1} \Bigg) \mathbb{1}(v < g) \\
&\quad = \Bigg((w_k - g) + \frac{1}{2}(1-v)^2 - \frac{1}{2}(w_k - g)^2 \\
&\qquad + \frac{1}{2}(1 - w_k + g - v)_+^2 \Bigg) \mathbb{1}(v < g).
\end{aligned} \tag{2.130}$$

Using Lemma 2.3, we have

$$\begin{aligned}
\mathbb{P}(V_1^u > v, G > v | \overline{\mathcal{C}_{1n}}, w_k) &= \int_0^{w_k} \mathbb{P}(V_1^u > v, G > v | \overline{\mathcal{C}_{1n}}, g, w_k) f_{G|\overline{\mathcal{C}_{1n}}, W_k}(g | \overline{\mathcal{C}_{1n}}, w_k) \mathrm{d}g \\
&= \int_v^{w_k} \Bigg((w_k - g) + \frac{1}{2}(1-v)^2 - \frac{1}{2}(w_k - g)^2 \\
&\qquad + \frac{1}{2}(1 - w_k + g - v)_+^2 \Bigg) \frac{1}{w_k} \mathrm{d}g \\
&= 1 - 2v - \frac{v}{w_k} + \frac{2v^2}{w_k} - \frac{v^3}{2w_k} + \frac{v w_k}{2}.
\end{aligned} \tag{2.131}$$

Finally, we integrate over W_k according to Lemma 2.2

$$\begin{aligned}\mathbb{P}(V_1^u > v, G > v|\overline{\mathcal{C}_{1n}}) &\leq \int_v^1 \mathbb{P}(V_1^u > v, G > v|\overline{\mathcal{C}_{1n}}, w_k) f_{W_k}(w_k)\mathrm{d}w_k \\ &= \int_v^1 \left(1 - 2v - \frac{v}{w_k} + \frac{2v^2}{w_k} - \frac{v^3}{2w_k} + \frac{vw_k}{2}\right) 2w_k \mathrm{d}w_k \\ &= 1 - \frac{11v}{3} + 5v^2 - 3v^3 + \frac{2v^4}{3}. \qquad (2.132)\end{aligned}$$

This term is sufficient for calculating the expected value bound. ■

Lemma 2.14. *For $K = n$, the expected minimum gap satisfies*

$$\mathbb{E}[V^*(n)|K = n] = \frac{1}{4}. \qquad (2.133)$$

Proof. We follow the same approach that we used for Lemma 2.13. Figure 2.7 shows the drop gap V_1 as a function of w_1, given w_n and g. The figure is justified using the same arguments that are in the proof of Lemma 2.13, but since no other items can become feasible, we can derive an exact expression for the probability of the event $V_1 > v$ when only conditioning on (g, w_n). Since W_1 has distribution $\mathcal{U}[0,1]$ via Lemma 2.4, we can simply take the total length of the bold regions to find $\mathbb{P}(V_1 > v|\mathcal{C}_n, w_n, g)$. Thus,

$$\mathbb{P}(V_1 > v|\mathcal{C}_n, w_n, g) = (w_n - g) + (1 - w_n + g - v) = (1 - v), \qquad v < g, \qquad (2.134)$$

where we have that $1 - w_n + g - v$ is nonnegative since $v < g$ and $w_n \leq 1$. To find

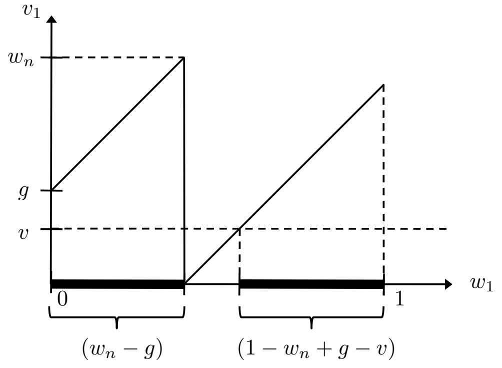


Figure 2.7. Drop gap v_1 as a function of w_1, parameterized by (w_n, g), resulting from the removal of the first item and assuming that the last item is critical ($K = n$). The function starts at g and increases at unit rate until $w_1 = w_n - g$, where it drops to zero, and then continues to increase at unit rate. If we only condition on $(w_n,\ g)$, the probability of the event $V_1 > v$ is given by the total length of the bold regions for $v < g$.

the probability of the event $V_* > v$, we note that the events $V > v$ and $G > v$ are conditionally independent given $G = g$, so

$$\mathbb{P}(V > v, G > v|\mathcal{C}_n, w_n, g) \quad = \quad (1 - v)\mathbb{1}(v < g). \tag{2.135}$$

Marginalizing over G using Lemma 2.3 gives

$$\begin{aligned}\mathbb{P}(V > v, G > v|\mathcal{C}_n, w_n) &= \int_0^1 \mathbb{P}(V > v, G > v|\mathcal{C}_n, w_n, g) f_{G|\mathcal{C}_n, W_n}(g|\mathcal{C}_n, w_n)\mathrm{d}g \\ &= \int_v^{w_n} (1 - v)\frac{1}{w_n}\mathrm{d}g \\ &= \frac{(w_n - v)(1 - v)}{w_n}. \end{aligned} \tag{2.136}$$

Noting the distribution of the critical item from Lemma 2.2,

$$\begin{aligned}\mathbb{P}(V > v, G > v|\mathcal{C}_n) &= \int_0^1 \mathbb{P}(V > v, G > v|\mathcal{C}_n, w_n) f_{W_n|\mathcal{C}_n}(w_n|\mathcal{C}_n)\mathrm{d}w_n \\ &= \int_0^1 \frac{(w_n - v)(1 - v)}{w_n} 2w_n \mathrm{d}w_n \\ &= 1 - 3v + 3v^2 - v^3 = \mathbb{P}(V^* > v|\mathcal{C}_n). \end{aligned} \tag{2.137}$$

Finally, using the fact that V^* is nonnegative,

$$\mathbb{E}[V^*|C_n] = \int_0^1 \mathbb{P}(V^* > v|C_n)\mathrm{d}v = \int_0^1 (1 - 3v + 3v^2 - v^3)\mathrm{d}v = \frac{1}{4}. \tag{2.138}$$

■

Lemma 2.15. *For $K = 1$, the expected minimum gap satisfies*

$$\mathbb{E}[V^*(n)|K = 1] \leq \frac{7}{30}. \tag{2.139}$$

Proof. We use a more direct approach when the first item is critical, since W_1 no longer has a uniform distribution (from Lemma 2.2). However, the analysis here is similar to the proof of Lemma 2.13 in how we bound the drop gap. Note that we have $B = G$ for this case. Additionally, the gap given by the minimum gap will always be equal to the drop gap since BLIND-GREEDY does not pack any items. We define a variable V_1^u that satisfies $V_1 \leq V_1^u$ with probability one, where

$$V_1^u := \begin{cases} V_1, & L_1 = 2 \vee L_1 = 3, \\ G - W_2 - W_3, & L_1 \geq 4. \end{cases} \tag{2.140}$$

We let the event $L_1 \geq 4$ also account for the case where $n = 3$ and the two remaining items are feasible. If, in fact, $n \geq 4$ and $L_1 \geq 4$, then V_1^u does not account for the

Table 2.1. Drop gap bound values when the first item is critical ($\mathcal{C}_1$).

Case	Defining inequalities	Minimum gap bound
$\mathcal{D}_2$	$W_2 > G$	$V_1^u = G$
$\mathcal{D}_3$	$W_2 \leq G, W_2 + W_3 > G$	$V_1^u = G - W_2$
$\mathcal{D}_{4+}$	$W_2 + W_3 \leq G$	$V_1^u = G - W_2 - W_3$

additional reductions in the gap caused by more items becoming feasible. Thus, we see that V_1^u is a deterministic function of (G, W_2, W_3).

To further simplify our expressions, we define $\mathcal{D}_2$, $\mathcal{D}_3$, $\mathcal{D}_{4+}$ to be the events $L_1 = 2$, $L_1 = 3$, and $L_1 \geq 4$, respectively. Based on these cases, the drop gap bound V_1^u is given by the values shown in Table 2.1.

We begin by finding some necessary distributions for the cases. For case $\mathcal{D}_3$, the posterior distribution of W_2 is needed. We have

$$f_{W_2|\mathcal{C}_1,\mathcal{D}_3,G}(w_2|\mathcal{C}_1,\mathcal{D}_3,g) = \frac{\mathbb{P}(W_2 \leq G, W_2 + W_3 > G|\mathcal{C}_1, g, w_2) f_{W_2}(w_2)}{\mathbb{P}(W_2 \leq G, W_2 + W_3 > G|\mathcal{C}_1, g)}, \tag{2.141}$$

where we have used Bayes' theorem and that $f_{W_2|\mathcal{C}_1,G}(w_2, \mathcal{C}_1, g) = f_{W_2}(w_2) = \mathcal{U}[0,1]$ by Lemma 2.4. For the numerator, we have

$$\mathbb{P}(W_2 \leq G, W_2 + W_3 > G|\mathcal{C}_1, g, w_2) = (1 - g + w_2)\mathbb{1}(w_2 \leq g), \tag{2.142}$$

which follows using Lemma 2.4 for the distribution of W_3. Integrating over W_2 gives

$$\begin{aligned}\mathbb{P}(\mathcal{D}_3|\mathcal{C}_1, g) &= \int_0^1 \mathbb{P}(W_2 \leq g, W_2 + W_3 > G|\mathcal{C}_1, w_2) f_{W_2}(w_2)\mathrm{d}w_2 \\ &= \int_0^g (1 - g + w_2)\mathrm{d}w_2 \\ &= g - \frac{g^2}{2}. \end{aligned}\tag{2.143}$$

Returning to the posterior distribution of W_2,

$$f_{W_2|\mathcal{C}_1,\mathcal{D}_3,G}(w_2|\mathcal{C}_1,\mathcal{D}_3,g) = \frac{2(1-g+w_2)}{(2-g)g}, \quad 0 \leq w_2 \leq g, \tag{2.144}$$

$$\mathbb{P}(W_2 \leq w_2|\mathcal{C}_1,\mathcal{D}_3,g) = \frac{(2-2g+w_2)w_2}{(2-g)g}, \quad 0 \leq w_2 \leq g. \tag{2.145}$$

Moving to the case $\mathcal{D}_{4+}$, define $W' := W_2 + W_3$;

$$\mathbb{P}(\mathcal{D}_{4+}|\mathcal{C}_1, g) = \mathbb{P}(W' \leq g|\mathcal{C}_1, g) = \frac{g^2}{2}, \tag{2.146}$$

where we have used that the distribution of W', conditioned on the first item being critical, is the distribution for the sum of two independent uniform random variables (via Lemma 2.4). Finally, for the posterior distribution of $W_2 + W_3$, we have

$$\mathbb{P}(W' \leq w' | \mathcal{C}_1, \mathcal{D}_{4+}, g) = \frac{w'^2}{g^2}, \quad 0 \leq w' \leq g. \tag{2.147}$$

We can now find distributions for V_1^u conditioned on all cases for the drop critical item. For case $\mathcal{D}_2$, it is clear that $V_1^u = G$, and

$$\mathbb{P}(\mathcal{D}_2 | \mathcal{C}_1, g) = \mathbb{P}(W_2 > g) = 1 - g. \tag{2.148}$$

For $\mathcal{D}_3$, we have

$$\begin{aligned}\mathbb{P}(V_1^u > v | \mathcal{C}_1, \mathcal{D}_3, g) &= \mathbb{P}(G - W_2 > v | C_1, D_3, g) \\ &= \mathbb{P}(W_2 < G - v | C_1, D_3, g) \\ &= \frac{(2 - 2g + (g - v))(g - v)}{(2 - g)g} \\ &= \frac{(2 - g - v)(g - v)}{(2 - g)g}, \quad 0 \leq v < g. \end{aligned}\tag{2.149}$$

Then for $\mathcal{D}_{4+}$,

$$\begin{aligned}\mathbb{P}(V_1^u > v | \mathcal{C}_1, \mathcal{D}_{4+}, g) &= \mathbb{P}(W' < G - v | \mathcal{C}_1, \mathcal{D}_{4+}, g) \\ &= \frac{(g - v)^2}{g^2}, \quad 0 \leq v < g. \end{aligned}\tag{2.150}$$

Considering all three cases, we have

$$\begin{aligned}\mathbb{P}(V_1^u > v | \mathcal{C}_1, g) &= \mathbb{P}(V_1^u > v | \mathcal{C}_1, \mathcal{D}_2, g)\mathbb{P}(\mathcal{D}_2 | \mathcal{C}_1, g) + \mathbb{P}(V_1^u > v | \mathcal{C}_1, \mathcal{D}_3, g)\mathbb{P}(\mathcal{D}_3 | \mathcal{C}_1, g) \\ &\quad + \mathbb{P}(V_1^u > v | \mathcal{C}_1, \mathcal{D}_{4+}, g)\mathbb{P}(\mathcal{D}_{4+} | \mathcal{C}_1, g) \\ &= \left((1 - g) + \frac{(2 - g - v)(g - v)}{(2 - g)g}\left(g - \frac{g^2}{2}\right)\right. \\ &\quad \left. + \frac{(g - v)^2}{g^2}\left(\frac{g^2}{2}\right)\right)\mathbb{1}(v < g) \\ &= (1 - v - gv + v^2)\mathbb{1}(v < g). \end{aligned}\tag{2.151}$$

This gives the expected value bound

$$\mathbb{E}[V_* | \mathcal{C}_1, g] \leq \mathbb{E}[V_1^u | \mathcal{C}_1, g] = \int_0^g (1 - v - gv + v^2) \mathrm{d}v = g - \frac{g^2}{2} - \frac{g^3}{6}. \tag{2.152}$$

Finally, integrating over G using Theorem 2.1,

$$\mathbb{E}[V_* | \mathcal{C}_1] \leq \int_0^1 \mathbb{E}[V_* | \mathcal{C}_1, g] f_G(g) \mathrm{d}g = \int_0^1 \left(g - \frac{g^2}{2} - \frac{g^3}{6}\right)(2 - 2g)\mathrm{d}g = \frac{7}{30}. \tag{2.153}$$

■

The final result for the subset sum problem follows easily from the stated lemmas.

Proof of Theorem 2.6 Using the above lemmas and noting that the events $\mathcal{C}_1$, $\overline{\mathcal{C}_{1n}}$, and C_n form a partition of the event $\sum_{i=1}^{n} W_i > B$ gives

$$\begin{aligned}\mathbb{E}\left[V_*(n)\middle|\sum_{i=1}^{n} W_i > B\right] &= \mathbb{E}[V_*(n)|\mathcal{C}_1]\mathbb{P}(\mathcal{C}_1) + \mathbb{E}[V_*(n)|\overline{\mathcal{C}_{1n}}]\mathbb{P}(\overline{\mathcal{C}_{1n}}) + \mathbb{E}[V_*(n)|\mathcal{C}_n]\mathbb{P}(\mathcal{C}_n) \\ &\leq \frac{7}{30}\left(\frac{1}{n}\right) + \frac{13}{60}\left(\frac{n-2}{n}\right) + \frac{1}{4}\left(\frac{1}{n}\right) = \frac{3+13n}{60n}. \qquad (2.154)\end{aligned}$$

■

2.5.3 Consecutive Rollout: 0–1 Knapsack Problem Analysis

The analysis of the CONSECUTIVE-ROLLOUT algorithm for the 0–1 knapsack problem follows the same structure as the analysis for the subset sum problem and makes use of the properties described in Section 2.3. The development here assumes that the reader is familiar with the subset sum analysis, so less detail is presented.

We use the same definition of the drop critical item L_1 that was used on the subset sum problem. From the algorithm description of CONSECUTIVE-ROLLOUT and the gain definition in (2.7), we have that the gain $Z_*(n)$ satisfies

$$Z_*(n) = \max\left(0, \sum_{i=2}^{L_1-1} P_i - \sum_{i=1}^{K-1} P_i\right). \qquad (2.155)$$

We will sometimes write $Z_*(n)$ simply as Z_*. The following three lemmas bound the gain $Z_*(n)$ for different cases of the critical item, assuming $n \geq 3$. Theorem 2.7 then follows easily. We implicitly assume that $\sum_{i=1}^{n} W_i > B$ holds for the section.

Lemma 2.16. *For $K = n$, the expected gain satisfies*

$$\mathbb{E}[Z_*(n)|K = n] = \frac{1}{9}. \qquad (2.156)$$

Proof. A positive gain can only obtained in the case where the last item becomes feasible when removing the first. Consistent with our subset sum notation, let $\mathcal{D}_{n+1}$ denote the event where item n becomes feasible when the first item is removed. Using Lemma 2.4 and the perspective of Figure 2.1, this probability is given by

$$\mathbb{P}(\mathcal{D}_{n+1}|g, w_n, \mathcal{C}_n) = \mathbb{P}(W_1 \geq W_n - G|w_n, g, \mathcal{C}_n) = (1 - w_n + g). \qquad (2.157)$$

Integrating over G using Lemma 2.3 and W_n using Lemma 2.4 gives

$$\mathbb{P}(\mathcal{D}_{n+1}|\mathcal{C}_n, w_n) = \int_0^{w_n} (1 - w_n + g)\frac{1}{w_n}\mathrm{d}g = 1 - \frac{w_n}{2}, \qquad (2.158)$$

$$\mathbb{P}(\mathcal{D}_{n+1}|\mathcal{C}_n) = \int_0^1 \left(1 - \frac{w_n}{2}\right) 2w_n \mathrm{d}w_n = \frac{2}{3}. \tag{2.159}$$

Now assuming that item n becomes feasible, we are interested in the case where it provides a larger value. This is simply given by the probability

$$\mathbb{P}(P_n \geq P_1) = \frac{1}{2}, \tag{2.160}$$

following from the symmetry of the distributions of P_1 and P_n. Conditioned on the event $P_n \geq P_1$, we are interested in the distribution of the gain, which is equal to $P_n - P_1$. We have

$$\mathbb{P}(P_n - P_1 \leq q | P_n \geq P_1) = \frac{\mathbb{P}(0 \leq P_n - P_1 \leq q)}{\mathbb{P}(P_n \geq P_1)}. \tag{2.161}$$

For the numerator,

$$\begin{aligned}\mathbb{P}(0 \leq P_n - P_1 \leq q) &= \int_0^{1-q}\int_{p_1}^{p_1+q} \mathrm{d}p_n \mathrm{d}p_1 + \int_{1-q}^1 \int_{p_1}^1 \mathrm{d}p_n \mathrm{d}p_1 \\ &= q - \frac{q^2}{2},\end{aligned} \tag{2.162}$$

which gives

$$\mathbb{P}(P_n - P_1 \leq q | P_n \geq P_1) = 2q - q^2, \tag{2.163}$$

$$\mathbb{E}[P_n - P_1 | P_n \geq P_1] = \frac{1}{3}. \tag{2.164}$$

Finally, by the independence of item weight and profit, we have

$$\mathbb{E}[Z_*(n)|\mathcal{C}_n] = \mathbb{E}[P_n - P_1|P_n \geq P_1]\mathbb{P}(P_n \geq P_1)\mathbb{P}(\mathcal{D}_{n+1}|\mathcal{C}_n) = \frac{1}{3}\cdot\frac{1}{2}\cdot\frac{2}{3} = \frac{1}{9}. \tag{2.165}$$

■

Lemma 2.17. *For $2 \leq K \leq n-1$, the expected gain satisfies*

$$\mathbb{E}[Z_*(n)|2 \leq K \leq n] \geq \frac{59}{288} \approx 0.205. \tag{2.166}$$

Proof. We again let $\overline{\mathcal{C}_{1n}}$ be the event that $2 \leq K \leq n-1$. We fix $K = k$, and the proof holds for all valid values of k. In the case of event $\overline{\mathcal{C}_{1n}}$, it is possible that removing the first item allows for the critical item to become feasible as well as additional items (i.e. $L_1 \geq k+2$). However, we are only guaranteed the existence of one item beyond the critical item since it is possible that $k = n-1$. Let $\mathcal{D}_{k+1}$ indicate the event $L_1 = k+1$, and let $\mathcal{D}_{(k+2)+}$ indicate the event $L_1 \geq k+2$. If item $k+2$ does not exist (i.e. $k = n-1$), then this event means that all remaining items are packed.

For the probability of the event $\mathcal{D}_{k+1}$, we have from Lemma 2.4 that W_1 has distribution $\mathcal{U}[0,1]$. Then,

$$\mathbb{P}(\mathcal{D}_{k+1}|g, w_k, w_{k+1}, \overline{\mathcal{C}_{1n}}) = w_{k+1} - (w_k - g + w_{k+1} - 1)_+. \tag{2.167}$$

This can be argued using an illustration similar to Figure 2.6, where the second term mitigates that case where w_{k+1} extends beyond $b+1$ for $B = b$. Likewise, for the event $\mathcal{D}_{(k+2)+}$, we have

$$\mathbb{P}(\mathcal{D}_{(k+2)+}|g, w_k, w_{k+1}, \overline{\mathcal{C}_{1n}}) = (1 - w_k + g - w_{k+1})_+. \tag{2.168}$$

Starting with event $\mathcal{D}_{k+1}$, we integrate over W_{k+1}, which has uniform density by Lemma 2.4.

$$\begin{aligned}
\mathbb{P}(\mathcal{D}_{k+1}|g, w_k, \overline{\mathcal{C}_{1n}}) &= \int_0^1 \mathbb{P}(\mathcal{D}_{k+1}|g, w_k, w_{k+1}, \overline{\mathcal{C}_{1n}}) f_{W_{k+1}}(w_{k+1}) \mathrm{d}w_{k+1} \\
&= \int_0^1 w_{k+1} \mathrm{d}w_{k+1} - \int_{1+g-w_k}^1 (w_k - g + w_{k+1} - 1) \mathrm{d}w_{k+1} \\
&= \frac{1}{2} - \frac{g^2}{2} + g w_k - \frac{w_k^2}{2}.
\end{aligned} \tag{2.169}$$

Marginalizing over G with Lemma 2.3 gives

$$\begin{aligned}
\mathbb{P}(\mathcal{D}_{k+1}|w_k, \overline{\mathcal{C}_{1n}}) &= \int_0^1 \mathbb{P}(\mathcal{D}_{k+1}|g, w_k, \overline{\mathcal{C}_{1n}}) f_{G|W_k}(g|w_k) \mathrm{d}g \\
&= \int_0^{w_k} \left(\frac{1}{2} - \frac{g^2}{2} + g w_k - \frac{w_k^2}{2} \right) \frac{1}{w_k} \mathrm{d}g \\
&= \frac{1}{2} - \frac{w_k^2}{6}.
\end{aligned} \tag{2.170}$$

Finally, by Lemma 2.2,

$$\begin{aligned}
\mathbb{P}(\mathcal{D}_{k+1}|\overline{\mathcal{C}_{1n}}) &= \int_0^1 \mathbb{P}(\mathcal{D}_{k+1}|w_k, \overline{\mathcal{C}_{1n}}) f_{W_k}(w_k) \mathrm{d}w_k \\
&= \int_0^1 \left(\frac{1}{2} - \frac{w_k^2}{6} \right) 2 w_k \mathrm{d}w_k = \frac{5}{12}.
\end{aligned} \tag{2.171}$$

Now for the event $\mathcal{D}_{(k+2)+}$, we integrate in the same order, using the same lemmas.

$$\begin{aligned}
\mathbb{P}(\mathcal{D}_{(k+2)+}|g, w_k, \overline{\mathcal{C}_{1n}}) &= \int_0^1 \mathbb{P}(\mathcal{D}_{(k+2)+}|g, w_k, w_{k+1}, \overline{\mathcal{C}_{1n}}) f_{W_{k+1}}(w_{k+1}) \mathrm{d}w_{k+1} \\
&= \int_0^{1-w_k+g} (1 - w_k + g - w_{k+1}) \mathrm{d}w_{k+1} \\
&= \frac{1}{2} + g + \frac{g^2}{2} - w_k - g w_k + \frac{w_k^2}{2}.
\end{aligned} \tag{2.172}$$

$$\begin{aligned}\mathbb{P}(\mathcal{D}_{(k+2)+}|w_k,\overline{\mathcal{C}_{1n}}) &= \int_0^1 \mathbb{P}(\mathcal{D}_{(k+2)+}|g,w_k,\overline{\mathcal{C}_{1n}}) f_{G|W_k}(g|w_k)\mathrm{d}g \\ &= \int_0^{w_k}\left(\frac{1}{2}+g+\frac{g^2}{2}-w_k-gw_k+\frac{w_k^2}{2}\right)\frac{1}{w_k}\mathrm{d}g \\ &= \frac{1}{2}-\frac{w_k}{2}+\frac{w_k^2}{6}. \end{aligned} \tag{2.173}$$

$$\begin{aligned}\mathbb{P}(\mathcal{D}_{(k+2)+}|\overline{\mathcal{C}_{1n}}) &= \int_0^1 \mathbb{P}(\mathcal{D}_{(k+2)+}|\overline{\mathcal{C}_{1n}},w_k) f_{W_k}(w_k)\mathrm{d}w_k \\ &= \int_0^1\left(\frac{1}{2}-\frac{w_k}{2}+\frac{w_k^2}{6}\right)2w_k\mathrm{d}w_k = \frac{1}{4}. \end{aligned} \tag{2.174}$$

Equipped with these probabilities, we now consider the gain from the rollout for the different drop critical item cases. For the case where only one item becomes feasible ($\mathcal{D}_{k+1}$), the analysis in the previous lemma holds, so we have

$$\mathbb{E}[P_n-P_1|\overline{\mathcal{C}_{1n}},\mathcal{D}_{k+1}] = \mathbb{E}[P_n-P_1|P_n>P_1]\mathbb{P}(P_n>P_1)\mathbb{P}(\mathcal{D}_{k+1}|\overline{\mathcal{C}_{1n}}) = \frac{1}{3}\cdot\frac{1}{2}\cdot\frac{5}{12}=\frac{5}{72}. \tag{2.175}$$

If two or more items become feasible ($\mathcal{D}_{(k+1)+}$), we only consider the gain resulting from adding two items, and this serves as a lower bound for the case of more items becoming feasible. Accordingly, define

$$P' := P_k + P_{k+1}. \tag{2.176}$$

The probability that the profits of the two items are greater than P_1 is given by

$$\mathbb{P}(P'\geq P_1) = 1-\mathbb{P}(P'<P_1) = 1-\int_0^1\int_0^{p_1} p'\mathrm{d}p'\mathrm{d}p_1 = 1-\int_0^1\frac{p_1^2}{2}\mathrm{d}p_1 = \frac{5}{6}. \tag{2.177}$$

The gain conditioned on the event $P' > P_1$ is given by

$$\mathbb{P}(P'-P_1\leq q|P'\geq P_1) = \frac{\mathbb{P}(0\leq P'-P_1\leq q)}{\mathbb{P}(P'\geq P_1)}. \tag{2.178}$$

Proceeding with the numerator and assuming $0 \leq q \leq 1$,

$$\begin{aligned}\mathbb{P}(0\leq P'-P_1\leq q) &= \int_0^{1-q}\int_{p_1}^{p_1+q} p'\mathrm{d}p'\mathrm{d}p_1 + \int_{1-q}^1\int_{p_1}^1 p'\mathrm{d}p'\mathrm{d}p_1 \\ &\quad + \int_{1-q}^1\int_1^{p_1+q}(2-p')\mathrm{d}p'\mathrm{d}p_1 \\ &= \int_0^{1-q}\left(p_1q+\frac{q^2}{2}\right)\mathrm{d}p_1 + \int_{1-q}^1\left(\frac{1}{2}-\frac{p_1^2}{2}\right)\mathrm{d}p_1 \\ &\quad + \int_{1-q}^1\left(-\frac{3}{2}+2p_1-\frac{p_1^2}{2}+2q-p_1q-\frac{q^2}{2}\right)\mathrm{d}p_1 \\ &= \frac{q}{2}+\frac{q^2}{2}-\frac{q^3}{3},\ 0\leq q\leq 1. \end{aligned} \tag{2.179}$$

Now for $1 < q \leq 2$,

$$\begin{aligned}
\mathbb{P}(0 \leq P' - P_1 \leq q) &= \int_0^1 \int_{p_1}^1 p' \mathrm{d}p' \mathrm{d}p_1 + \int_0^{2-q} \int_1^{p_1+q} (2-p') \mathrm{d}p' \mathrm{d}p_1 \\
&\quad + \int_{2-q}^1 \int_1^2 (2-p') \mathrm{d}p' \mathrm{d}p_1 \\
&= \int_0^1 \left(\frac{1}{2} - \frac{p_1^2}{2} \right) \mathrm{d}p_1 \\
&\quad + \int_0^{2-q} \left(-\frac{3}{2} + 2p_1 - \frac{p_1^2}{2} + 2q - p_1 q - \frac{q^2}{2} \right) \mathrm{d}p_1 + \frac{1}{2} \int_{2-q}^1 \mathrm{d}p_1 \\
&= -\frac{1}{2} + 2q - q^2 + \frac{q^3}{6}, \ 1 < q \leq 2. \qquad (2.180)
\end{aligned}$$

The distribution for the gain is thus given by

$$\mathbb{P}(P' - P_1 \leq q | P' - P_1 \geq 0) = \begin{cases} \frac{3}{5}q + \frac{3}{5}q^2 - \frac{2}{5}q^3, & 0 \leq q \leq 1, \\ -\frac{3}{5} + \frac{12}{5}q - \frac{6}{5}q^2 + \frac{1}{5}q^3, & 1 < q \leq 2. \end{cases} \qquad (2.181)$$

The expected value is

$$\mathbb{E}[P' - P_1 | P' - P_1 \geq 0] = \int_0^1 q \left(\frac{3}{5} + \frac{6}{5}q - \frac{6}{5}q^2 \right) \mathrm{d}q + \int_1^2 q \left(\frac{12}{5} - \frac{12}{5}q + \frac{3}{5}q^2 \right) \mathrm{d}q = \frac{13}{20}. \qquad (2.182)$$

Recalling that it is possible for more than two items to be added in the case $\mathcal{D}_{(k+2)+}$, let P'' be the total value of items added for the case. We may bound the expected gain as follows, where the term $\mathbb{P}(\mathcal{D}_k | \overline{\mathcal{C}_{1n}})$ is omitted since it provides zero gain. We are implicitly using the fact that item weights and profits are independent.

$$\begin{aligned}
\mathbb{E}[Z_*(n) | \overline{\mathcal{C}_{1n}}] &= \mathbb{E}[P'' - P_1 | P'' > P_1] \mathbb{P}(P'' \geq P_1) \mathbb{P}(\mathcal{D}_{(k+2)+} | \overline{\mathcal{C}_{1n}}) \\
&\quad + \mathbb{E}[P_n - P_1 | P_n \geq P_1] \mathbb{P}(P_n \geq P_1) \mathbb{P}(\mathcal{D}_{k+1} | \overline{\mathcal{C}_{1n}}) \\
&\geq \mathbb{E}[P' - P_1 | P' > P_1] \mathbb{P}(P' \geq P_1) \mathbb{P}(\mathcal{D}_{(k+2)+} | \overline{\mathcal{C}_{1n}}) \\
&\quad + \mathbb{E}[P_n - P_1 | P_n \geq P_1] \mathbb{P}(P_n \geq P_1) \mathbb{P}(\mathcal{D}_{k+1} | \overline{\mathcal{C}_{1n}}) \\
&= \frac{13}{20} \cdot \frac{5}{6} \cdot \frac{1}{4} + \frac{1}{3} \cdot \frac{1}{2} \cdot \frac{5}{12} = \frac{59}{288}. \qquad (2.183)
\end{aligned}$$

■

Lemma 2.18. *For $K = 1$, the expected gain satisfies*

$$\mathbb{E}[Z_*(n) | K = 1] \geq \frac{5}{24} \approx 0.208. \qquad (2.184)$$

Proof. We use the drop events $\mathcal{D}_2$, $\mathcal{D}_3$, and $\mathcal{D}_{4+}$ just as we did for the subset sum problem. The event probabilities given $G = g$ are the same as those for the subset sum problem. Accordingly,

$$\mathbb{P}(\mathcal{D}_2|\mathcal{C}_1) = \int_0^1 \mathbb{P}(\mathcal{D}_2|\mathcal{C}_1, g) f_G(g)\mathrm{d}g = \int_0^1 (1-g)(2-2g)\mathrm{d}g = \frac{2}{3}, \tag{2.185}$$

$$\mathbb{P}(\mathcal{D}_3|\mathcal{C}_1) = \int_0^1 \mathbb{P}(\mathcal{D}_3|\mathcal{C}_1, g) f_G(g)\mathrm{d}g = \int_0^1 \left(g - \frac{g^2}{2}\right)(2-2g)\mathrm{d}g = \frac{1}{4}, \tag{2.186}$$

$$\mathbb{P}(\mathcal{D}_{4+}|\mathcal{C}_1) = \frac{1}{12}. \tag{2.187}$$

The greedy solution gives zero value, so the expected gain is easily determined using independence of item weights and profits,

$$\mathbb{E}[Z_*|\mathcal{C}_1, \mathcal{D}_2] = 0, \tag{2.188}$$

$$\mathbb{E}[Z_*|\mathcal{C}_1, \mathcal{D}_3] = \mathbb{E}[P_2] = \frac{1}{2}, \tag{2.189}$$

$$\mathbb{E}[Z_*|\mathcal{C}_1, \mathcal{D}_{4+}] \geq \mathbb{E}[P_2 + P_3] = 1. \tag{2.190}$$

Combining all cases for the drop critical item,

$$\begin{aligned} \mathbb{E}[Z_*|\mathcal{C}_1] &= \mathbb{E}[Z_*|\mathcal{C}_1, \mathcal{D}_3]\mathbb{P}(\mathcal{D}_3|\mathcal{C}_1) + \mathbb{E}[Z_*|\mathcal{C}_1, \mathcal{D}_{4+}]\mathbb{P}(\mathcal{D}_{4+}|\mathcal{C}_1) \\ &\geq \frac{1}{2} \cdot \frac{1}{4} + 1 \cdot \frac{1}{12} = \frac{5}{24}. \end{aligned} \tag{2.191}$$

■

The result for the knapsack problem then follows.

Proof of Theorem 2.7. The events $\mathcal{C}_1$, $\overline{\mathcal{C}_{1n}}$, and $\mathcal{C}_n$ form a partition of the event $\sum_{i=1}^n W_i > B$, so using Lemma 2.1 gives

$$\begin{aligned} \mathbb{E}\left[Z_*(n) \middle| \sum_{i=1}^n W_i > B\right] &= \mathbb{E}[Z_*(n)|\mathcal{C}_1]\mathbb{P}(\mathcal{C}_1) + \mathbb{E}[Z_*(n)|\overline{\mathcal{C}_{1n}}]\mathbb{P}(\overline{\mathcal{C}_{1n}}) + \mathbb{E}[Z_*(n)|\mathcal{C}_n]\mathbb{P}(\mathcal{C}_n) \\ &\geq \frac{5}{24}\left(\frac{1}{n}\right) + \frac{59}{288}\left(\frac{n-2}{n}\right) + \frac{1}{9}\left(\frac{1}{n}\right) = \frac{-26+59n}{288n}. \end{aligned} \tag{2.192}$$

■

2.6 Rollout on Binary Decision Trees

In this section, we analyze a rollout algorithm on a completely different problem, that of finding the shortest path from a root node to a leaf node in a binary decision tree. This allows us to make some interesting observations about average-case and worst-case performance of rollout algorithms that may apply in general. The binary tree model

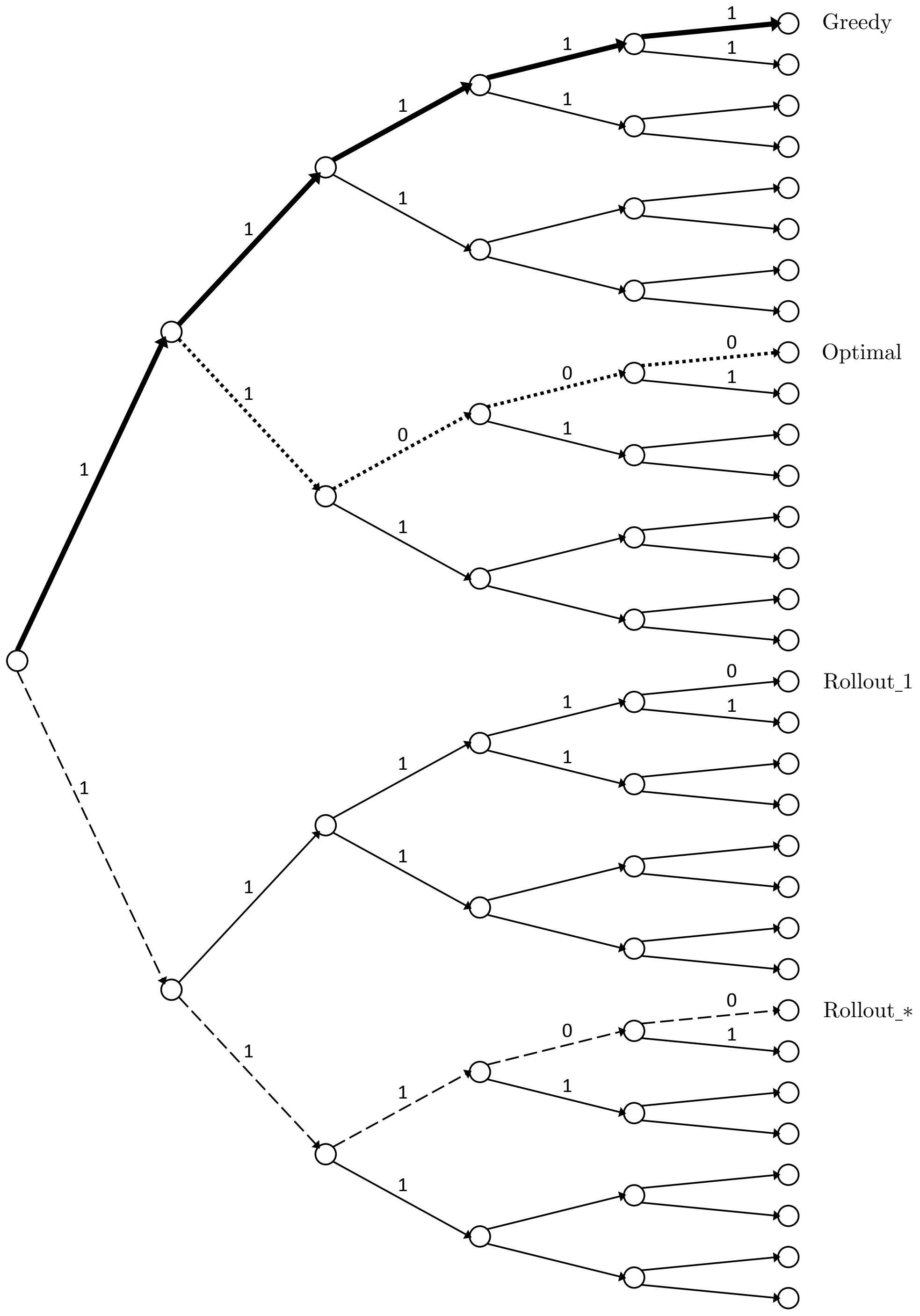


Figure 2.8. Binary decision tree of height $H = 5$. The goal of the problem is to find the shortest path from the root node to any leaf node. The greedy algorithm chooses the path shown in bold and obtains a cost equal to 5. The rollout algorithm, run at every iteration, chooses the dashed path, which has a cost of 3. Running only the first iteration of the rollout algorithm gives the solution with the leaf node labeled Rollout_1. The optimal solution is shown as dashed and has a cost equal to 2. Irrelevant edge costs are not shown.

was studied by Karp and Pearl [81] and it inspired the result on the breakthrough problem of Bertsekas [27].

Consider a binary decision tree represented by a directed graph, where each node has two outgoing edges, except for the leaf nodes. We assume that the tree is *perfect*, so that all leaf nodes have the same distance (in terms of number of edges) to the root node. The *height* of the tree is the distance between the root node and leaf nodes. Each edge e has a cost $x_e \in \{0,1\}$. Starting at the root node, the goal of the problem is to find a path to any leaf node with the minimum cost. We visualize the problem as moving from left to right to traverse the tree and choosing children nodes by moving up or down, as shown in Figure 2.8.

A simple greedy algorithm for this problem does the following: at each node, choose the outgoing edge with the smaller value and, in case of ties, always choose the upper edge. The rollout algorithm combines lookahead with the greedy algorithm. At each node, it looks ahead to both the upper and lower children nodes, estimates the future cost of these nodes using the greedy algorithm, and then chooses the edge with the lowest total cost. In case of ties, again, the rollout algorithm always chooses the upper node.

For the instance shown in Figure 2.8, the greedy algorithm encounters a series of ties at each level and takes the uppermost path shown in bold, yielding a cost equal to 5. Running the first iteration of the rollout algorithm leads to the leaf node labeled Rollout_1, which has an improved cost of 4. Running any additional iterations of the rollout algorithm leads to the node labeled Rollout_∗, shown by the dashed path, giving a total cost of 3. The rollout approach does not give the optimal solution in this instance; the optimal path is the dotted path, which has a cost of 2.

We determine the average-case behavior of both the greedy and the rollout algorithm on this problem when edge costs are independently equal to 1 with probability p and 0 otherwise. That is,

$$\mathbb{P}(x_e = 1) = p, \quad \forall e \in E. \tag{2.193}$$

We can immediately characterize the performance of the greedy algorithm using independence. Let $z_{\mathrm{G}}(H)$ denote the cost of the greedy algorithm on a binary decision tree of height H.

Theorem 2.8. *The cost of the greedy algorithm for the average-case model on a binary decision tree of height H satisfies*

$$\mathbb{E}[z_{\mathrm{G}}(H)] = Hp^2. \tag{2.194}$$

Proof. The greedy algorithm is forced to take an edge with cost 1 only if both outgoing edges have unit cost. This happens with probability p^2. ■

There is not a simple closed-form expression for the expected performance of the rollout algorithm (that we know of), but we find a recursion that allows us to compute the expected performance. We derive this recursion by constructing a tree of height h via joining two trees of height $h-1$, which we refer to as the upper and lower trees. Let the values obtained by the greedy and rollout algorithms from the root of the lower tree be denoted by g_1 and r_1, respectively. Likewise, let the values for the upper tree be denoted by g_2 and r_2. Let x_1 and x_2 denote the edge costs joining the tree rooted at height h to the lower tree and upper tree, respectively. Finally, let g' and r' denote the values of the greedy algorithm and rollout algorithm for the tree at height h. Based on the description of the two algorithms and the tie-breaking rules, we have

$$g' = \begin{cases} g_1 + x_1, & x_1 < x_2, \\ g_2 + x_2, & x_1 \geq x_2, \end{cases} \tag{2.195}$$

$$r' = \begin{cases} r_1 + x_1, & g_1 + x_1 < g_2 + x_2, \\ r_2 + x_2, & g_1 + x_1 \geq g_2 + x_2. \end{cases} \tag{2.196}$$

This allows us to calculate the joint probability distribution for the rollout and greedy costs on a tree of height h, given the joint distribution of costs for two trees at height $h-1$.

For a tree of height h, let G_h be the random variable for the greedy cost and let R_h be the random variable for the rollout cost. We use the notation

$$P^h(g,r) := \mathbb{P}(G_h = g, R_h = r), \tag{2.197}$$

which is nonzero for $g = 0,\ldots,h$ and $r = 0,\ldots,g$, as the rollout cost can never be greater than the greedy cost. For a node that we are interested in, we will use the random variables X_1 and X_2 to denote the lower and upper outgoing edge costs, respectively. In the following four lemmas, we evaluate $\mathbb{P}(G_h = g, R_h = r | X_1 = x_1, X_2 = x_2)$ for the joint cases of (x_1, x_2). To simplify notation, we use

$$P^h(g,r|x_1,x_2) := \mathbb{P}(G_h = g, R_h = r | X_1 = x_1, X_2 = x_2). \tag{2.198}$$

With further use of this notation, we observe by independence of two trees at height $h-1$ that

$$P^h(g,r,g_1,r_1,g_2,r_2|x_1,x_2) = \mathbb{P}(g,r|g_1,r_1,g_2,r_2,x_1,x_2)P^{h-1}(g_1,r_1)P^{h-1}(g_2,r_2) \tag{2.199}$$

and

$$P^h(g,r|x_1,x_2) = \sum_{g_1,r_1,g_2,r_2} \mathbb{P}(g,r|g_1,r_1,g_2,r_2,x_1,x_2)P^{h-1}(g_1,r_1)P^{h-1}(g_2,r_2). \tag{2.200}$$

In the following lemmas, we use the fact that $\mathbb{P}(g,r|g_1,r_1,g_2,r_2,x_1,x_2)$ is a deterministic function based on (2.195) and (2.196).

Lemma 2.19. *Given the joint distribution $P^{h-1}(\cdot,\cdot)$ for a tree of height $h-1$, the joint distribution $P^h(\cdot,\cdot)$ at height h conditioned on the edges $X_1 = 0$ and $X_2 = 0$ is*

$$P^h(g,r|0,0) \quad = \quad \sum_{r_2=0}^{g} P^{h-1}(g,r_2) \sum_{g_1=r}^{g-1} P^{h-1}(g_1,r) + P^{h-1}(g,r) \sum_{g_1=g}^{h-1} \sum_{r_1=0}^{g_1} P^{h-1}(g_1,r_1). \tag{2.201}$$

Proof. We have $G' = G_2$ and

$$R' = \begin{cases} R_1, & G_1 < G_2, \\ R_2, & G_1 \geq G_2. \end{cases} \tag{2.202}$$

Next,

$$\begin{aligned} P^h(g,r|0,0) \quad = \quad & \sum_{g_1,g_2:g_1<g_2} \sum_{r_1,r_2} P(g,r|g_1,r_1,g_2,r_2,0,0)P^{h-1}(g_1,r_1)P^{h-1}(g_2,r_2) \\ & + \sum_{g_1,g_2:g_1\geq g_2} \sum_{r_1,r_2} P(g,r|g_1,r_1,g_2,r_2,0,0)P^{h-1}(g_1,r_1)P^{h-1}(g_2,r_2). \end{aligned} \tag{2.203}$$

For the first term,

$$\begin{aligned} & \sum_{g_1,g_2:g_1<g_2} \sum_{r_1,r_2} P(g,r|g_1,r_1,g_2,r_2,0,0)P^{h-1}(g_1,r_1)P^{h-1}(g_2,r_2) \\ = \quad & \sum_{g_1,g_2:g_1<g_2} \sum_{r_1,r_2} \mathbb{1}(G_2 = g, R_1 = r)P^{h-1}(g_1,r_1)P^{h-1}(g_2,r_2) \\ = \quad & \sum_{g_1:g_1<g} \sum_{r_2} P^{h-1}(g_1,r)P^{h-1}(g,r_2) \\ = \quad & \sum_{r_2=0}^{g} P^{h-1}(g,r_2) \sum_{g_1=r}^{g-1} P^{h-1}(g_1,r). \end{aligned} \tag{2.204}$$

For the second term,

$$\begin{aligned} & \sum_{g_1,g_2:g_1\geq g_2} \sum_{r_1,r_2} P(g,r|g_1,r_1,g_2,r_2,0,0)P^{h-1}(g_1,r_1)P^{h-1}(g_2,r_2) \\ = \quad & \sum_{g_1,g_2:g_1\geq g_2} \sum_{r_1,r_2} \mathbb{1}(G_2 = g, R_2 = r)P^{h-1}(g_1,r_1)P^{h-1}(g_2,r_2) \\ = \quad & \sum_{g_1:g_1\geq g} \sum_{r_1} P^{h-1}(g_1,r_1)P^{h-1}(g,r) \\ = \quad & P^{h-1}(g,r) \sum_{g_1=g}^{h-1} \sum_{r_1=0}^{g_1} P^{h-1}(g_1,r_1). \end{aligned} \tag{2.205}$$

■

Lemma 2.20. *Given the joint distribution $P^{h-1}(\cdot,\cdot)$ for a tree of height $h-1$, the joint distribution $P^h(\cdot,\cdot)$ at height h conditioned on the edges $X_1 = 0$ and $X_2 = 1$ is*

$$\begin{aligned} P^h(g,r|0,1) &= \sum_{r_1=0}^{g} P^{h-1}(g,r_1) \sum_{g_2=r-1}^{g-1} P^{h-1}(g_2, r-1) \\ &\quad + P^{h-1}(g,r) \sum_{g_2=g}^{h-1} \sum_{r_2=0}^{g_2} P^{h-1}(g_2,r_2). \end{aligned} \tag{2.206}$$

Proof. We have $G' = G_1$ and

$$R' = \begin{cases} R_1, & G_1 < G_2 + 1, \\ R_2 + 1, & G_1 \geq G_2 + 1. \end{cases} \tag{2.207}$$

$$\begin{aligned} P^h(g,r|0,1) &= \sum_{g_1,g_2:g_1<g_2+1} \sum_{r_1,r_2} P(g,r|g_1,r_1,g_2,r_2,0,1)P^{h-1}(g_1,r_1)P^{h-1}(g_2,r_2) \\ &+ \sum_{g_1,g_2:g_1\geq g_2+1} \sum_{r_1,r_2} P(g,r|g_1,r_1,g_2,r_2,0,1)P^{h-1}(g_1,r_1)P^{h-1}(g_2,r_2). \end{aligned} \tag{2.208}$$

For the first term,

$$\begin{aligned} & \sum_{g_1,g_2:g_1<g_2+1} \sum_{r_1,r_2} P(g,r|g_1,r_1,g_2,r_2,0,1)P^{h-1}(g_1,r_1)P^{h-1}(g_2,r_2) \\ = & \sum_{g_1,g_2:g_1<g_2+1} \sum_{r_1,r_2} \mathbb{1}(G_1 = g, R_1 = r)P^{h-1}(g_1,r_1)P^{h-1}(g_2,r_2) \\ = & \sum_{g_2:g<g_2+1} \sum_{r_2} P^{h-1}(g,r)P^{h-1}(g_2,r_2) \\ = & \; P^{h-1}(g,r) \sum_{g_2=g}^{h-1} \sum_{r_2=0}^{g_2} P^{h-1}(g_2,r_2). \end{aligned} \tag{2.209}$$

For the second term,

$$\begin{aligned} & \sum_{g_1,g_2:g_1\geq g_2+1} \sum_{r_1,r_2} P(g,r|g_1,r_1,g_2,r_2,0,1)P^{h-1}(g_1,r_1)P^{h-1}(g_2,r_2) \\ = & \sum_{g_1,g_2:g_1\geq g_2+1} \sum_{r_1,r_2} \mathbb{1}(G_1 = g, R_2 = r-1)P^{h-1}(g_1,r_1)P^{h-1}(g_2,r_2) \\ = & \sum_{g_2:g\geq g_2+1} \sum_{r_1} P^{h-1}(g,r_1)P^{h-1}(g_2,r-1) \\ = & \sum_{r_1=0}^{g} P^{h-1}(g,r_1) \sum_{g_2=r-1}^{g-1} P^{h-1}(g_2,r-1). \end{aligned} \tag{2.210}$$

■

Lemma 2.21. *Given the joint distribution $P^{h-1}(\cdot,\cdot)$ for a tree of height $h-1$, the joint distribution $P^h(\cdot,\cdot)$ at height h conditioned on the edges $X_1 = 1$ and $X_2 = 0$ is*

$$\begin{aligned} P^h(g,r|1,0) &= \sum_{r_2=0}^{g} P^{h-1}(g,r_2) \sum_{g_1=r-1}^{g-2} P^{h-1}(g_1,r-1) \\ &\quad + P^{h-1}(g,r) \sum_{g_1=g-1}^{h-1} \sum_{r_1=0}^{g_1} P^{h-1}(g_1,r_1). \end{aligned} \tag{2.211}$$

Proof. We have $G' = G_2$ and

$$R' = \begin{cases} R_1 + 1, & G_1 + 1 < G_2, \\ R_2, & G_1 + 1 \geq G_2. \end{cases} \tag{2.212}$$

$$\begin{aligned} P^h(g,r|1,0) &= \sum_{g_1,g_2:g_1+1<g_2} \sum_{r_1,r_2} P(g,r|g_1,r_1,g_2,r_2,1,0)P^{h-1}(g_1,r_1)P^{h-1}(g_2,r_2) \\ &\quad + \sum_{g_1,g_2:g_1+1\geq g_2} \sum_{r_1,r_2} P(g,r|g_1,r_1,g_2,r_2,1,0)P^{h-1}(g_1,r_1)P^{h-1}(g_2,r_2). \end{aligned} \tag{2.213}$$

For the first term,

$$\begin{aligned} & \sum_{g_1,g_2:g_1+1<g_2} \sum_{r_1,r_2} P(g,r|g_1,r_1,g_2,r_2,1,0)P^{h-1}(g_1,r_1)P^{h-1}(g_2,r_2) \\ = & \sum_{g_1,g_2:g_1+1<g_2} \sum_{r_1,r_2} \mathbb{1}(R_1 = r-1, G_2 = g)P^{h-1}(g_1,r_1)P^{h-1}(g_2,r_2) \\ = & \sum_{g_1:g_1+1<g} \sum_{r_2} P^{h-1}(g_1,r-1)P^{h-1}(g,r_2) \\ = & \sum_{r_2=0}^{g} P^{h-1}(g,r_2) \sum_{g_1=r-1}^{g-2} P^{h-1}(g_1,r-1). \end{aligned} \tag{2.214}$$

For the second term,

$$\begin{aligned} & \sum_{g_1,g_2:g_1+1\geq g_2} \sum_{r_1,r_2} P(g,r|g_1,r_1,g_2,r_2,1,0)P^{h-1}(g_1,r_1)P^{h-1}(g_2,r_2) \\ = & \sum_{g_1,g_2:g_1+1\geq g_2} \sum_{r_1,r_2} \mathbb{1}(G_2 = g, R_2 = r)P^{h-1}(g_1,r_1)P^{h-1}(g_2,r_2) \\ = & \sum_{g_1:g_1+1\geq g} \sum_{r_1} P^{h-1}(g_1,r_1)P^{h-1}(g,r) \\ = & P^{h-1}(g,r) \sum_{g_1=g-1}^{h-1} \sum_{r_1=0}^{g_1} P^{h-1}(g_1,r_1). \end{aligned} \tag{2.215}$$

■

Lemma 2.22. *Given the joint distribution $P^{h-1}(\cdot,\cdot)$ for a tree of height $h-1$, the joint distribution $P^h(\cdot,\cdot)$ at height h conditioned on the edges $X_1 = 1$ and $X_2 = 1$ is*

$$\begin{aligned} P^h(g,r|1,1) &= \sum_{r_2=0}^{g-1} P^{h-1}(g-1,r_2) \sum_{g_1=r-1}^{g-2} P^{h-1}(g_1,r-1) \\ &\quad + P^{h-1}(g-1,r-1) \sum_{g_1=g-1}^{h-1} \sum_{r_1=0}^{g_1} P^{h-1}(g_1,r_1). \end{aligned} \tag{2.216}$$

Proof. We have $G' = G_2 + 1$ and

$$R' = \begin{cases} R_1+1, & G_1 < G_2, \\ R_2+1, & G_1 \geq G_2. \end{cases} \tag{2.217}$$

$$\begin{aligned} P^h(g,r|1,1) &= \sum_{g_1,g_2:g_1<g_2} \sum_{r_1,r_2} P(g,r|g_1,r_1,g_2,r_2,1,1) P^{h-1}(g_1,r_1) P^{h-1}(g_2,r_2) \\ &\quad + \sum_{g_1,g_2:g_1\geq g_2} \sum_{r_1,r_2} P(g,r|g_1,r_1,g_2,r_2,1,1) P^{h-1}(g_1,r_1) P^{h-1}(g_2,r_2). \end{aligned} \tag{2.218}$$

For the first term,

$$\begin{aligned} &\sum_{g_1,g_2:g_1<g_2} \sum_{r_1,r_2} P(g,r|g_1,r_1,g_2,r_2,1,1) P^{h-1}(g_1,r_1) P^{h-1}(g_2,r_2) \\ = &\sum_{g_1,g_2:g_1<g_2} \sum_{r_1,r_2} \mathbb{1}(G_2 = g-1, R_1 = r-1) P^{h-1}(g_1,r_1) P^{h-1}(g_2,r_2) \\ = &\sum_{g_1:g_1<g-1} \sum_{r_2} P^{h-1}(g_1,r-1) P^{h-1}(g-1,r_2) \\ = &\sum_{r_2=0}^{g-1} P^{h-1}(g-1,r_2) \sum_{g_1=r-1}^{g-2} P^{h-1}(g_1,r-1). \end{aligned} \tag{2.219}$$

For the second term,

$$\begin{aligned} &\sum_{g_1,g_2:g_1\geq g_2} \sum_{r_1,r_2} P(g,r|g_1,r_1,g_2,r_2,1,1) P^{h-1}(g_1,r_1) P^{h-1}(g_2,r_2) \\ = &\sum_{g_1,g_2:g_1\geq g_2} \sum_{r_1,r_2} \mathbb{1}(G_2 = g-1, R_2 = r-1) P^{h-1}(g_1,r_1) P^{h-1}(g_2,r_2) \\ = &\sum_{g_1:g_1\geq g-1} \sum_{r_1} P^{h-1}(g_1,r_1) P^{h-1}(g-1,r-1) \\ = &\; P^{h-1}(g-1,r-1) \sum_{g_1=g-1}^{h-1} \sum_{r_1=0}^{g_1} P^{h-1}(g_1,r_1). \end{aligned} \tag{2.220}$$

■

We can now state the full recursion.

Theorem 2.9. *Given the joint distribution $P^{h-1}(\cdot,\cdot)$ for a tree of height $h-1$, the joint distribution $P^h(\cdot,\cdot)$ at height h satisfies*

$$\begin{aligned}
P^h(g,r) &= (1-p)^2 \left(\sum_{r_2=0}^{g} P^{h-1}(g,r_2) \sum_{g_1=r}^{g-1} P^{h-1}(g_1,r) + P^{h-1}(g,r) \sum_{g_1=g}^{h-1} \sum_{r_1=0}^{g_1} P^{h-1}(g_1,r_1) \right) \\
&\quad + p(1-p) \left(\sum_{r_1=0}^{g} P^{h-1}(g,r_1) \sum_{g_2=r-1}^{g-1} P^{h-1}(g_2,r-1) + P^{h-1}(g,r) \sum_{g_2=g}^{h-1} \sum_{r_2=0}^{g_2} P^{h-1}(g_2,r_2) \right) \\
&\quad + p(1-p) \left(\sum_{r_2=0}^{g} P^{h-1}(g,r_2) \sum_{g_1=r-1}^{g-2} P^{h-1}(g_1,r-1) + P^{h-1}(g,r) \sum_{g_1=g-1}^{h-1} \sum_{r_1=0}^{g_1} P^{h-1}(g_1,r_1) \right) \\
&\quad + p^2 \left(\sum_{r_2=0}^{g-1} P^{h-1}(g-1,r_2) \sum_{g_1=r-1}^{g-2} P^{h-1}(g_1,r-1) + P^{h-1}(g-1,r-1) \sum_{g_1=g-1}^{h-1} \sum_{r_1=0}^{g_1} P^{h-1}(g_1,r_1) \right),
\end{aligned} \tag{2.221}$$

for $h \geq 2$.

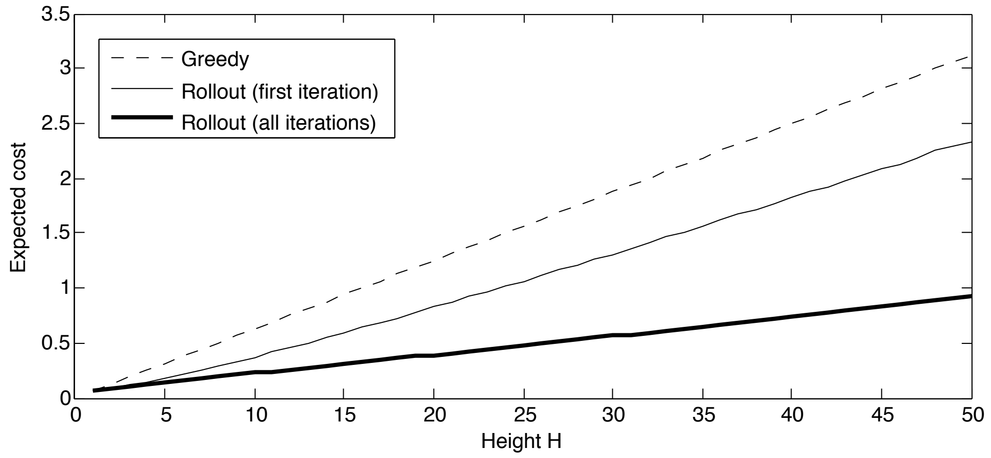
Proof. This follows using the total probability theorem with the previous four lemmas. ■

We evaluate the above recursion computationally to determine the expected performance of the rollout algorithm. Note that we start our recursion at height $h = 1$ with

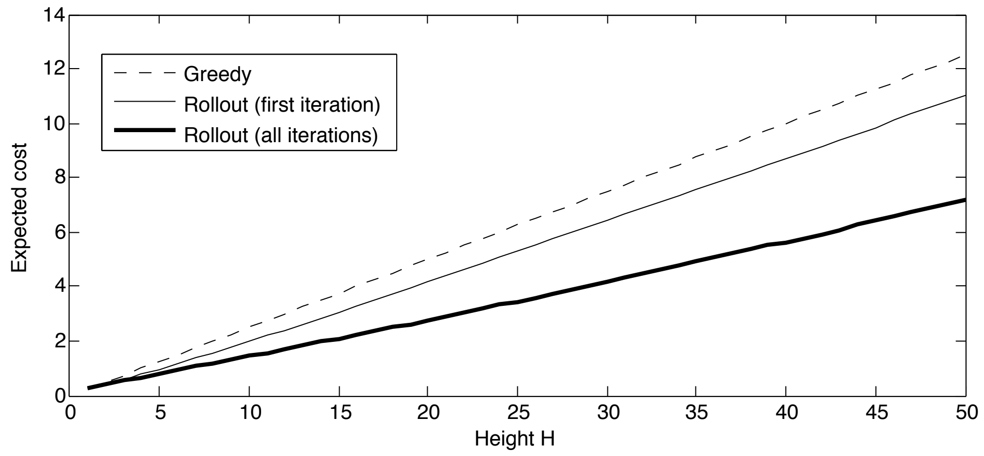
$$P^1(g,r) = \begin{cases} 1-p^2, & g=0 \text{ and } r=0, \\ 0, & g=1 \text{ and } r=0, \\ p^2, & g=1 \text{ and } r=1. \end{cases} \tag{2.222}$$

Results of the recursion for three values of p and tree heights ranging up to $H = 50$ are shown in Figure 2.9. The plots show the expected performance of the greedy algorithm, given by Theorem 2.8, the rollout algorithm when run at every iteration, calculated using Theorem 2.9, and the result from the first iteration of the rollout algorithm (and using greedy thereafter). The first iteration data correspond to mean values over 10^5 simulations.

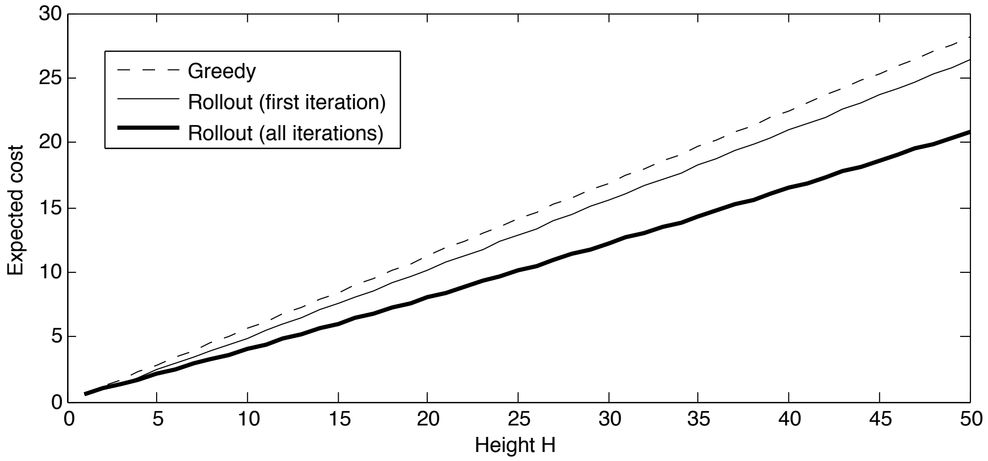
These average-case results show that while a significant performance gain is achieved with only the first iteration of the rollout algorithm, there is a clear benefit to running additional iterations. For $p = 0.25$ and $H = 50$, for instance, the first iteration of the rollout algorithm gives an expected cost equal to 75% of the expected greedy cost, while running every iteration gives an expected cost equal to 30% of the expected greedy cost. Interestingly, this same behavior is not seen from a worst-case perspective. Figure 2.10 shows an instance where a small improvement in performance is made from the first iteration, but no improvements can be made with following iterations. This is a result of the fact that the first iteration of the rollout algorithm mistakenly chooses the lower tree, which does not contain an optimal path. We formalize this with the following theorem.



(a) $p = 0.25$



(b) $p = 0.5$



(c) $p = 0.75$

Figure 2.9. Average-case results of rollout algorithms on binary decision tree for (a) $p = 0.25$, (b) $p = 0.5$, and (c) $p = 0.75$. Expected costs are shown for the greedy algorithm, the first iteration of the rollout algorithm, and the rollout algorithm run at every iteration (corresponding to H iterations total). The first iteration results are the mean values from 10^5 simulations, and the results for all iterations are are calculated using Theorem 2.9.

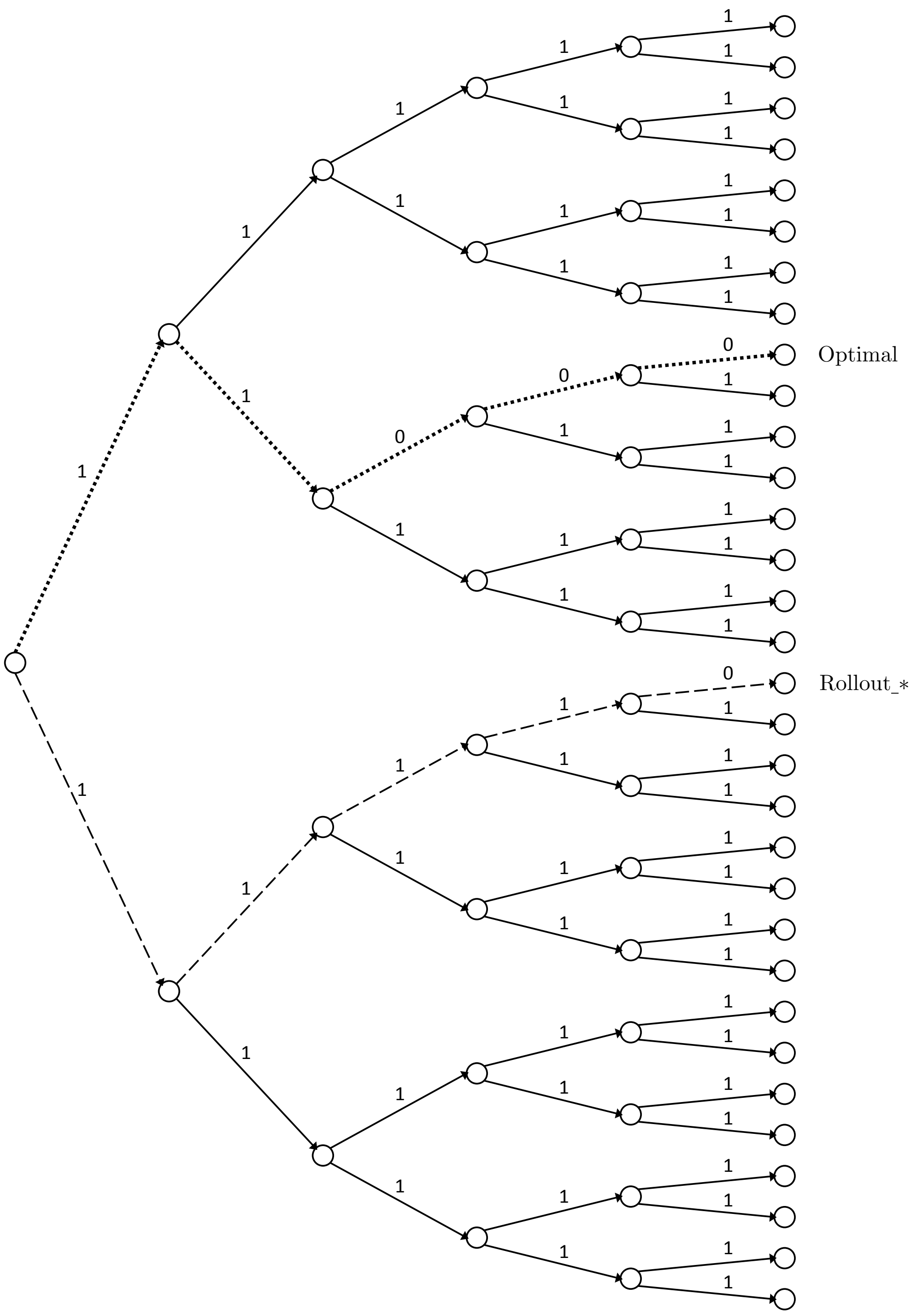


Figure 2.10. Example demonstrating poor worst-case performance of rollout procedure even if the rollout algorithm is run at every iteration. During the first iteration, the rollout algorithm chooses the lower edge. Thereafter, it takes the upper edge at each node. The final path chosen by the rollout algorithm is shown as dashed and has a cost of 4. The optimal path is shown as dotted and has a cost of 2. This example can be generalized so that for any tree of height $H \geq 4$, the cost obtained by the rollout algorithm is $H - 1$, while the optimal cost is equal to 2.

Theorem 2.10. *For every height $H \geq 4$, there exists a binary decision tree for which the performance of the rollout algorithm, run at every iteration, obtains a cost $z_{\mathrm{R}}(H) = H - 1$, while the optimal cost is $z^*(H) = 2$.*

Proof. The example for $H = 5$ is shown in Figure 2.10. The problem instance can be easily generalized to larger values of H. ■

The strong average-case performance and the poor worst-case performance may be characteristic of the rollout approach for many problems, including the knapsack problem. As we noted at the beginning of this chapter for the knapsack problem, using the decreasing density greedy algorithm gives a worst-case performance guarantee of 1/2, and the first iteration of the rollout algorithm improves the guarantee to 2/3 [26]. Yet, it is not possible to guarantee benefits from additional iterations in the worst case. As shown in Figure 2.2 for the subset sum problem, however, performance gains are observable (using the Exhaustive-Rollout algorithm) for iterations beyond the first iteration in the average case. It may be possible to prove that these performance improvements hold; this is a topic for future research.

2.7 Discussion

We have shown strong performance bounds for both the Consecutive-Rollout and Exhaustive-Rollout algorithms on the subset sum problem and 0–1 knapsack problem. These results hold after only a single iteration and provide bounds for additional iterations. Simulations indicate that these bounds are very close in comparison with realized performance of a single iteration. We have also characterized the asymptotic behavior (asymptotic with respect to the total number of items) of the expected performance of both rollout techniques for the two problems.

The results for Exhaustive-Rollout are particularly interesting. It is remarkable how simple the expression is for the bound on the Exhaustive-Rollout algorithm (Theorem 2.2), despite the extensive analysis necessary to prove it. This is evidence that there may be a simpler proof. A commonly cited drawback of average-case analysis is that it does not lead to simple, intuitive bounds for small problem instances [47, 133]. Theorem 2.2 seems to be an exception to this rule. Nonetheless, a slight modification to stochastic subset sum model, namely the stochastic knapsack model that we have used, leads to an unwieldy expression (Theorem 2.4). Apart from the asymptotic behavior that this expression gives, it is difficult to argue that this result is useful. In some sense, this is a negative result, suggesting that average-case bounds for rollout algorithms on more elaborate problems are likely to become analytically intricate.

In the previous section, we looked at rollout algorithms on binary trees and made an important observation: running the rollout algorithm beyond the first iteration proved beneficial in the average case but not in the worst case. This appears to also be true for knapsack problems. There are provable gains from the first iteration in both the worst case (due to Bertazzi [26]) and the average case (from this chapter). In the worst case, Bertazzi showed an example where it is not possible to guarantee additional improvements with more iterations. This is not true in the average case. For the stochastic models used in this chapter, it can be verified via simulation (as we did for Figure 2.2) that additional performance gains are obtained with more iterations. Note also that the example problem that we presented shows gains after the first iteration for both rollout algorithms.

It may be possible to make theoretical statements about this behavior. That is, it may be possible to prove average-case performance bounds resulting from additional iterations on the stochastic models. However, this is likely to become cumbersome. Such an analysis likely requires working with the full distribution of the gap after the first iteration, rather than just the expected value. More importantly, the useful property that non-critical item weights remain independently distributed on $\mathcal{U}[0,1]$ (specifically Lemma 2.2) does not seem to hold for the remaining items after the first iteration of the rollout has occurred.

Apart from obtaining bounds from additional iterations, there is certainly an opportunity in finding tighter versions of our results in future research, though it is not obvious how this should be done. For both the consecutive and the exhaustive rollout techniques, considering the contribution of each additional packed item to the final minimum gap adds a dimension to the space of variables that must be integrated over, and these integrals are already complex. In general, finding the expected value of the minimum of many random variables can be difficult. A different topic is to still consider only the first iteration of the rollout algorithm, but with a larger lookahead length (e.g. trying all pairs of items for the exhaustive rollout, rather than just each item individually). From the worst-case perspective, it seems easier to analyze this case than the effect of additional iterations (see, e.g. results on partial enumeration [89, 125]). Average-case analysis here might be manageable.

The ideal result for rollout algorithms in general would be a combination of a computationally difficult problem, a corresponding stochastic model, and a simple expression describing the expected performance gain as a function of the number of iterations run. While this is a lofty goal, such a problem/result with some of these characteristics may be found in the field of random graphs. A proof technique that accounts for multiple iterations from an average-case perspective will have to handle the stochastic evolution of the rollout solution. On the binary decision tree, for example, we can think of the state of the system at each level as the remaining cost of the best currently selected path. This changes in a random fashion, of course, but it may be concentrated around a deterministic path. A common approach for such a scenario is the differential equation method of Kurtz [97] and Wormald [148]. In fact, we will see in Chapter 3 that the differential equation method can be used to analyze a simple rollout algorithm on random 2-regular bipartite graphs, where every iteration is accounted for. The problem of matching on 2-regular bipartite graphs is trivial, though, and the algorithm that we use there as a base policy is a bit contrived. The difficulty in using the differential equation method on the binary decision tree, or on random structures in general, is that sums of independent random variables usually lead to normal distributions, which are less tractable analytically.

While daunting, it is desirable to have theoretical results for more complex problems, both in the worst case and average case. Studying problems with multidimensional state space is appealing since these are the types of problems where rollout techniques are often used and perform well in practice. In this direction, it would be useful to consider problems such as the bin packing problem, the multiple knapsack problem, and the multidimensional knapsack problem. Finally, it would be interesting to see how effective the rollout approach is in a game-theoretic setting. In a dynamic two-person game, for instance, a base policy would have to state a strategy for a player as well as an opponent. To our knowledge, this topic has not been considered, although there has been some work on lookahead search in game playing [109].

2.8 Evaluations of Integrals

The following lemma is used in integral evaluations described in this section.

Lemma 2.23. *For constant values $\kappa_1,\ \kappa_2$ and nonnegative integer θ,*

$$\int \frac{(\kappa_1+\kappa_2 x)^\theta}{x}\mathrm{d}x = \kappa_1^\theta \log(x) + \sum_{j=1}^{\theta} \frac{\kappa_1^{\theta-j}(\kappa_1+\kappa_2 x)^j}{j}. \tag{2.223}$$

Proof. We begin by noting that

$$\begin{aligned}\int \frac{(\kappa_1+\kappa_2 x)^\theta}{x}\mathrm{d}x &= \int \kappa_2(\kappa_1+\kappa_2 x)^{\theta-1}\mathrm{d}x + \int \frac{\kappa_1(\kappa_1+\kappa_2 x)^{\theta-1}}{x}\mathrm{d}x \\ &= \frac{(\kappa_1+\kappa_2 x)^\theta}{\theta} + \kappa_1 \int \frac{(\kappa_1+\kappa_2 x)^{\theta-1}}{x}\mathrm{d}x. \end{aligned} \tag{2.224}$$

The statement of the lemma clearly holds for $\theta = 0$. Assuming that it holds for $\theta = t$, we have for $\theta = t+1$,

$$\begin{aligned}\int \frac{(\kappa_1+\kappa_2 x)^{t+1}}{x}\mathrm{d}x &= \frac{(\kappa_1+\kappa_2 x)^{t+1}}{t+1} + \kappa_1\left(\kappa_1^t \log(x) + \sum_{j=1}^{t} \frac{\kappa_1^{t-j}(\kappa_1+\kappa_2 x)^j}{j}\right) \\ &= \kappa_1^{t+1}\log(x) + \sum_{j=1}^{t+1} \frac{\kappa_1^{t+1-j}(\kappa_1+\kappa_2 x)^j}{j}. \end{aligned} \tag{2.225}$$

The property then holds for all θ by induction.

■

2.8.1 Evaluation of Integral (2.40)

To simplify expressions, we use $A := W_{K-1}$. Also recall that $M := n-K$. The integral is

$$\begin{aligned}&\mathbb{P}(V_*^u > v|m, \overline{\mathcal{C}}_1) \\ &= \int_0^1\int_0^1 \mathbb{P}(G > v|a,g,\overline{\mathcal{C}}_1)\mathbb{P}(\widetilde{V}^u > v|a,g,\overline{\mathcal{C}}_1)\left(\mathbb{P}(V^u > v|a,g,\overline{\mathcal{C}}_1)\right)^m f_A(a)f_G(g)\mathrm{d}a\mathrm{d}g.\end{aligned} \tag{2.226}$$

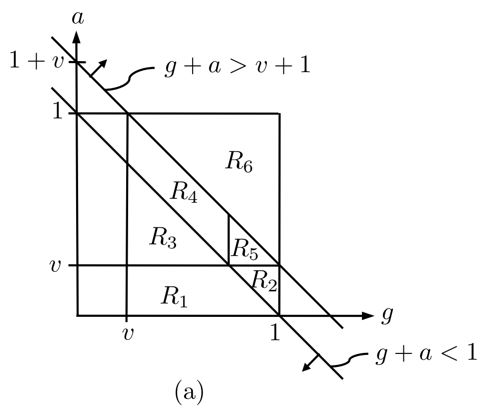
This may be evaluated by considering regions where the arguments have simple analytical descriptions as a function of a and g. We begin by noting that $\mathbb{P}(G > v|a,g,\overline{\mathcal{C}}_1) = \mathbb{1}(v < g)$, so we may restrict our analysis to regions where $v < g$. For the integral evaluation of $(a, g) \in R_j$, we use the notation

$$\rho_j(v, m) = \iint_{R_j} \mathbb{P}(G > v | a, g, \overline{\mathcal{C}}_1) \mathbb{P}(\widetilde{V}^u > v | a, g, \overline{\mathcal{C}}_1) \left(\mathbb{P}(V^u > v | a, g, \overline{\mathcal{C}}_1)\right)^m f_A(a) f_G(g) \mathrm{d}a\mathrm{d}g. \tag{2.227}$$

The relevant regions are shown in Figure 2.11, where different enumerations are necessary for $v \le \frac{1}{2}$ and $v > \frac{1}{2}$. The values of $(\mathbb{P}(V^u > v | a, g, \overline{\mathcal{C}}_1))^m$ and $\mathbb{P}(\widetilde{V}^u > v | a, g, \overline{\mathcal{C}}_1)$ are shown in Table 2.2. Note that in many cases, the $\frac{1}{1-g}$ factor from $\mathbb{P}(G > v | a, g, \overline{\mathcal{C}}_1)$ cancels with the $(1 - g)$ factor from $f_G(g)$, which simplifies the expression.

Table 2.2. Arguments of (2.40) for regions shown in Figure 2.11.

Region	$(\mathbb{P}(V^u > v \vert a, g, \overline{\mathcal{C}}_1))^m$	$\mathbb{P}(\widetilde{V}^u > v \vert a, g, \overline{\mathcal{C}}_1)$
R_1	$(1 - v - a)^m$	$(1 - g - a)/(1 - g)$
R_2	$(g - v)^m$	0
R_3	$(1 - 2v)^m$	$(1 - g - v)/(1 - g)$
R_4	$(a + g - 2v)^m$	$(a - v)/(1 - g)$
R_5	$(a + g - 2v)^m$	$(a - v)/(1 - g)$
R_6	$(1 - v)^m$	1
R_7	$(1 - v - a)^m$	$(1 - g - a)/(1 - g)$
R_8	$(g - v)^m$	0
R_9	$(a + g - 2v)^m$	$(a - v)/(1 - g)$
R_{10}	$(1 - v)^m$	1



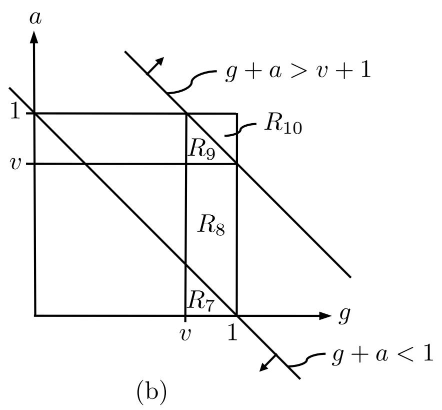


Figure 2.11. Integration regions for (a) $v \le \frac{1}{2}$ and (b) $v > \frac{1}{2}$.

Regions 1-6 correspond to the case where $v \le \frac{1}{2}$.

$$\begin{aligned}\rho_1(v,m) &= \int_0^v \int_v^{1-a} 2(1-v-a)^m(1-g-a)\mathrm{d}g\mathrm{d}a = \int_0^v (1-a-v)^{2+m}\mathrm{d}a \\ &= \frac{-(1-2v)^{3+m}+(1-v)^{3+m}}{3+m}. \qquad (2.228)\end{aligned}$$

$$\rho_2(v,m) = 0. \qquad (2.229)$$

$$\begin{aligned}\rho_3(v,m) &= \int_v^{1-v}\int_v^{1-a} 2(1-2v)^m(1-g-v)\mathrm{d}g\mathrm{d}a \\ &= \int_v^{1-v} (3v-a-1)(1-2v)^m(v+a-1)\mathrm{d}a \\ &= \frac{2}{3}(1-2v)^{3+m}. \qquad (2.230)\end{aligned}$$

$$\begin{aligned}&\rho_4(v,m) \\ &= \int_v^{1-v}\int_{1-g}^{v+1-g} 2(a+g-2v)^m(a-v)\mathrm{d}a\mathrm{d}g \\ &= \frac{1}{(1+m)(2+m)}\int_v^{1-v}\big(-2(1-2v)^{1+m}+4g(1-2v)^{1+m}-2m(1-2v)^{1+m} \\ &+2gm(1-2v)^{1+m}+2(1-v)^{1+m}-4g(1-v)^{1+m}+2m(1-v)^{1+m}-2gm(1-v)^{1+m} \\ &+2m(1-2v)^{1+m}v+2(1-v)^{1+m}v\big)\,\mathrm{d}g \\ &= \frac{1}{(1+m)(2+m)}\big(m(1-2v)^{3+m}+m(1-v)^m+2(1-v)^m v-3m(1-v)^m v \\ &-6(1-v)^m v^2+2m(1-v)^m v^2+4(1-v)^m v^3\big). \qquad (2.231)\end{aligned}$$

$$\begin{aligned}&\rho_5(v,m) \\ &= \int_{1-v}^{1}\int_v^{1+v-g} 2(a+g-2v)^m(a-v)\mathrm{d}a\mathrm{d}g \\ &= \frac{1}{(1+m)(2+m)}\int_{1-v}^{1}\big(2(1-v)^{1+m}-4g(1-v)^{1+m}+2m(1-v)^{1+m} \\ &-2gm(1-v)^{1+m}+2(g-v)^{2+m}+2(1-v)^{1+m}v\big)\,\mathrm{d}g \\ &= \frac{1}{(1+m)(2+m)(3+m)}\big(-2(1-2v)^{3+m}+2(1-v)^{1+m}-10(1-v)^{1+m}v \\ &-2m(1-v)^{1+m}v+14(1-v)^{1+m}v^2+7m(1-v)^{1+m}v^2+m^2(1-v)^{1+m}v^2\big). \qquad (2.232)\end{aligned}$$

$$\begin{aligned}\rho_6(v,m) &= \int_v^1 \int_{1+v-g}^1 (1-v)^m(2-2g)\mathrm{d}a\mathrm{d}g \\ &= \int_v^1 (2-2g)(1-v)^m(g-v)\mathrm{d}g = \frac{1}{3}(1-v)^{3+m}. \end{aligned} \tag{2.233}$$

Summing all terms of $\mathbb{P}(V_*^u > v|m, \overline{\mathcal{C}}_1)$ for $v \le \frac{1}{2}$ gives

$$\begin{aligned}&\mathbb{P}(V_*^u > v|m, \overline{\mathcal{C}}_1)_{\le\frac{1}{2}} \\ &\quad:= \rho_1(v,m)+\rho_2(v,m)+\rho_3(v,m)+\rho_4(v,m)+\rho_5(v,m)+\rho_6(v,m) \\ &\quad= \frac{1}{3(3+m)}\big(2m(1-2v)^m + m(1-v)^m + 9(1-v)^{3+m} - 12m(1-2v)^m v \\ &\qquad - 3m(1-v)^m v + 24m(1-2v)^m v^2 + 3m(1-v)^m v^2 - 16m(1-2v)^m v^3 \\ &\qquad - m(1-v)^m v^3\big). \end{aligned} \tag{2.234}$$

Regions 7-10 are for the case $v > \frac{1}{2}$.

$$\begin{aligned}\rho_7(v,m) &= \int_v^1 \int_0^{1-g} 2(1-v-a)^m(1-g-a)\mathrm{d}a\mathrm{d}g \\ &= \frac{1}{(1+m)(2+m)}\int_v^1 \big(2(1-v)^{1+m} - 4g(1-v)^{1+m} + 2m(1-v)^{1+m} \\ &\quad -2gm(1-v)^{1+m} + 2(g-v)^{2+m} + 2(1-v)^{1+m}v\big)\,\mathrm{d}g = \frac{(1-v)^{3+m}}{3+m}. \end{aligned} \tag{2.235}$$

$$\rho_8(v,m) = 0. \tag{2.236}$$

$$\begin{aligned}\rho_9(v,m) &= \int_v^1 \int_v^{1+v-g} 2(a+g-2v)^m(a-v)\mathrm{d}a\mathrm{d}g \\ &= \frac{1}{(1+m)(2+m)}\int_v^1 \big(2(1-v)^{1+m} - 4g(1-v)^{1+m} + 2m(1-v)^{1+m} \\ &\quad -2gm(1-v)^{1+m} + 2(g-v)^{2+m} + 2(1-v)^{1+m}v\big)\,\mathrm{d}g = \frac{(1-v)^{3+m}}{3+m}. \end{aligned} \tag{2.237}$$

$$\begin{aligned}\rho_{10}(v,m) &= \int_v^1 \int_{1+v-g}^1 (1-v)^m(2-2g)\mathrm{d}a\mathrm{d}g \\ &= \int_v^1 (2-2g)(1-v)^m(g-v)\mathrm{d}g = \frac{1}{3}(1-v)^{3+m}. \end{aligned} \tag{2.238}$$

Summing these terms yields for $v > \frac{1}{2}$,

$$\begin{aligned}\mathbb{P}(V_*^u > v|m, \overline{\mathcal{C}}_1)_{>\frac{1}{2}} &:= \rho_7(v,m)+\rho_8(v,m)+\rho_9(v,m)+\rho_{10}(v,m) \\ &= \frac{1}{3}(1-v)^{3+m} + \frac{2(1-v)^{3+m}}{3+m}. \end{aligned} \tag{2.239}$$

In summary, we have

$$\mathbb{P}(V_*^u > v|m,\overline{\mathcal{C}}_1) = \begin{cases} \mathbb{P}(V_*^u > v|m,\overline{\mathcal{C}}_1)_{\le\frac{1}{2}}, & v \le \frac{1}{2}, \\ \mathbb{P}(V_*^u > v|m,\overline{\mathcal{C}}_1)_{>\frac{1}{2}}, & v > \frac{1}{2}. \end{cases} \tag{2.240}$$

2.8.2 Evaluation of Integral (2.79)

We wish to evaluate

$$\mathbb{P}(Z_*^l \le z|m,\overline{\mathcal{C}_1})_{\mathcal{E}} \quad := \quad \int_0^1 \int_0^{1-g} \mathbb{P}(Z_*^l \le z|g,a,m,\overline{\mathcal{C}_1})_{\mathcal{E}} f_A(a) f_G(g) \mathrm{d}a\mathrm{d}g, \tag{2.241}$$

where

$$\begin{aligned}\mathbb{P}(Z_*^l \le & z|g,a,m,\overline{\mathcal{C}_1})_{\mathcal{E}} \\ = \; & z(1-g+gz)^m + \frac{(1-g+gz)^{m+1}(1-g+m-gm-gz)}{(1-g)a(m+1)(m+2)} \\ & - \frac{(1-g+gz+a(1-z))^{m+1}\,(1-g+m-gm-gz+a(-1-m+z+mz))}{(1-g)a(m+1)(m+2)}.\end{aligned} \tag{2.242}$$

We first determine the following using the fact that A follows distribution $\mathcal{U}[0,1]$

$$\int \mathbb{P}(Z_*^l \le z|g,a,m,\overline{\mathcal{C}_1})_{\mathcal{E}} f_A(a)\mathrm{d}a = \int \mathbb{P}(Z_*^l \le z|g,a,m,\overline{\mathcal{C}_1})_{\mathcal{E}}\mathrm{d}a. \tag{2.243}$$

The following constants simplify the expression:

$$\lambda_1 \quad := \quad 1-g+gz, \tag{2.244}$$

$$\lambda_2 \quad := \quad z-1, \tag{2.245}$$

$$\lambda_3 \quad := \quad 1-g+m-gm-gz, \tag{2.246}$$

$$\lambda_4 \quad := \quad -1-m+z+mz, \tag{2.247}$$

$$\lambda_5 \quad := \quad \frac{-1}{(1-g)(m+1)(m+2)}. \tag{2.248}$$

This gives

$$\begin{aligned}
&\int \mathbb{P}(Z_*^l \leq z | g, a, m, \overline{\mathcal{C}_1})_{\mathcal{E}} \mathrm{d}a \\
&\quad = \int \left(z\lambda_1^m - \frac{\lambda_5\lambda_3\lambda_1^{m+1}}{a} + \lambda_5\lambda_3 \frac{(\lambda_1 + a\lambda_2)^{m+1}}{a} + \lambda_5\lambda_4(\lambda_1 + a\lambda_2)^{m+1} \right) \mathrm{d}a \\
&\quad = az\lambda_1^m - \lambda_5\lambda_3\lambda_1^{m+1}\log(a) + \lambda_5\lambda_3 \left(\lambda_1^{m+1}\log(a) + \sum_{j=1}^{m+1} \frac{\lambda_1^{m+1-j}(\lambda_1 + \lambda_2 a)^j}{j} \right) \\
&\qquad + \frac{\lambda_5\lambda_4}{\lambda_2(m+2)}(\lambda_1 + \lambda_2 a)^{m+2} \\
&\quad = az\lambda_1^m + \lambda_5\lambda_3 \sum_{j=1}^{m+1} \frac{\lambda_1^{m+1-j}(\lambda_1 + \lambda_2 a)^j}{j} + \frac{\lambda_5\lambda_4}{\lambda_2(m+2)}(\lambda_1 + \lambda_2 a)^{m+2}, \qquad (2.249)
\end{aligned}$$

where we have made use of the integral identity from Lemma 2.23. Evaluating over the domain of integration gives

$$\begin{aligned}
&\int_0^{1-g} \mathbb{P}(Z_*^l \leq z | g, a, m, \overline{\mathcal{C}_1})_{\mathcal{E}} \mathrm{d}a \\
&\quad = (1-g)z\lambda_1^m + \lambda_5\lambda_3 \left(\sum_{j=1}^{m+1} \frac{\lambda_1^{m+1-j} z^j}{j} - \lambda_1^{m+1} H(m+1) \right) \\
&\qquad + \frac{\lambda_5\lambda_4(z^{m+2} - \lambda_1^{m+2})}{\lambda_2(m+2)}. \qquad (2.250)
\end{aligned}$$

Next, we calculate

$$\begin{aligned}
&\int_0^1 \int_0^{1-g} \mathbb{P}(Z_*^l \leq z | g, a, m, \overline{\mathcal{C}_1})_{\mathcal{E}} f_A(a) f_G(g) \mathrm{d}a \mathrm{d}g \\
&\quad = \int_0^1 \left(\int_0^{1-g} \mathbb{P}(Z_*^l \leq z | g, a, m, \overline{\mathcal{C}_1})_{\mathcal{E}} \mathrm{d}a \right) (2-2g) \mathrm{d}g. \qquad (2.251)
\end{aligned}$$

We integrate each additive term separately:

$$\begin{aligned}
\rho_1(m, z) &= \int_0^1 (1-g)z\lambda_1^m (2-2g) \mathrm{d}g \\
&= \int_0^1 2(1-g)^2 z(1-g+gz)^m \mathrm{d}g \\
&= -\frac{2z\left(2 + m^2(-1+z)^2 + m(-1+z)(-3+5z) - 2z\left(3 - 3z + z^{2+m}\right)\right)}{(1+m)(2+m)(3+m)(-1+z)^3}. \qquad (2.252)
\end{aligned}$$

$$\begin{aligned}
\rho_{2j}(m,z) &= \int_0^1 \lambda_5\lambda_3\frac{\lambda_1^{m+1-j}z^j}{j}(2-2g)\mathrm{d}g \\
&= \int_0^1 -\frac{2(1-g+m-gm-gz)(1-g+gz)^{m+1-j}z^j}{(m+1)(m+2)j}\mathrm{d}g \\
&= \frac{2z^{3+m}(j+(2+m)(-2+z)-jz)}{j(-3+j-m)(-2+j-m)(1+m)(2+m)(-1+z)^2} \\
&\quad +\frac{2z^j(-j(1+m)(-1+z)+(2+m)(-1+m(-1+z)+2z))}{j(-3+j-m)(-2+j-m)(1+m)(2+m)(-1+z)^2}.
\end{aligned} \tag{2.253}$$

$$\begin{aligned}
\rho_3(m,z) &= \int_0^1 -\lambda_5\lambda_3\lambda_1^{m+1}H(m+1)(2-2g)\mathrm{d}g \\
&= \int_0^1 \frac{2H(m+1)(1-g+m-gm-gz)(1-g+gz)^{m+1}}{(m+1)(m+2)}\mathrm{d}g \\
&= -\frac{2H(m+1)\left(-1+m(-1+z)+2z+(-2+z)z^{3+m}\right)}{(1+m)(2+m)(3+m)(-1+z)^2}.
\end{aligned} \tag{2.254}$$

$$\begin{aligned}
\rho_4(m,z) &= \int_0^1 \frac{\lambda_5\lambda_4(z^{m+2}-\lambda_1^{m+2})}{\lambda_2(m+2)}(2-2g)\mathrm{d}g \\
&= -\frac{2(-1-m+z+mz)(z^{m+2}-(1-g+gz)^{m+2})}{(m+1)(m+2)^2(-1+z)}\mathrm{d}g \\
&= -\frac{2}{(2+m)^2(3+m)(-1+z)}-\frac{2z^{2+m}}{(2+m)^2}+\frac{2z^{3+m}}{(2+m)^2(3+m)(-1+z)}.
\end{aligned} \tag{2.255}$$

With these terms, we have

$$\mathbb{P}(Z_*^l \le z|m,\mathcal{E},\overline{\mathcal{C}_1}) = \rho_1(m,z)+\sum_{j=1}^{m+1}\rho_{2j}(m,z)+\rho_3(m,z)+\rho_4(m,z). \tag{2.256}$$

2.8.3 Evaluation of Integral (2.94)

The integral is

$$\mathbb{P}(Z_*^l \le z|m,\overline{\mathcal{C}_1})_{\overline{\mathcal{E}}} = \int_0^1 g\mathbb{P}(Z_*^l \le z|g,a,m,\overline{\mathcal{C}_1})_{\overline{\mathcal{E}}}f_G(g)\mathrm{d}g, \tag{2.257}$$

where

$$\begin{aligned}
\mathbb{P}(Z_*^l \le z|g,a,m,\overline{\mathcal{C}_1})_{\overline{\mathcal{E}}} &= z(1-g+gz)^m-\frac{(1-2g+(1-g)m)z^{2+m}}{(1-g)^2(1+m)(2+m)} \\
&\quad +\frac{((1-g)(1+m)-gz)(1-g+gz)^{1+m}}{(1-g)^2(1+m)(2+m)}.
\end{aligned} \tag{2.258}$$

For the first term in $\mathbb{P}(Z_*^l \leq z|m,\overline{\mathcal{C}_1})_{\overline{\mathcal{E}}}$,

$$\int_0^1 gz(1-g+gz)^m(2-2g)\mathrm{d}g = -\frac{2z\left(1+m-3z-mz+z^{2+m}(3+m-(1+m)z)\right)}{(1+m)(2+m)(3+m)(-1+z)^3}. \tag{2.259}$$

To find the indefinite integral of the second term in $\mathbb{P}(Z_*^l \leq z|m,\overline{\mathcal{C}_1})_{\overline{\mathcal{E}}}$, we use the substitution $g = 1-e$.

$$\begin{aligned}
&\int -\frac{2g(1-2g+(1-g)m)z^{2+m}}{(1-g)(1+m)(2+m)}\mathrm{d}g \\
= &\int \frac{2(1-e)(1-2(1-e)+em)z^{m+2}}{e(m+1)(m+2)}\mathrm{d}e \\
= &\int \frac{(-2+2e(m+3)-2e^2(m+2))z^{m+2}}{e(m+1)(m+2)}\mathrm{d}e \\
= &\int \frac{-2z^{m+2}}{e(m+1)(m+2)}\mathrm{d}e + \int \frac{2(m+3)z^{m+2}}{(m+1)(m+2)}\mathrm{d}e + \int \frac{-2ez^{m+2}}{(m+1)}\mathrm{d}e \\
= &\frac{-2z^{m+2}}{(m+1)(m+2)}\log(e) + \frac{2e(3+m)z^{m+2}}{(m+1)(m+2)} - \frac{e^2z^{m+2}}{(m+1)}.
\end{aligned} \tag{2.260}$$

For the indefinite integral of the final term in $\mathbb{P}(Z_*^l \leq z|m,\overline{\mathcal{C}_1})_{\overline{\mathcal{E}}}$, we again use the substitution $g = 1-e$.

$$\begin{aligned}
&\int \frac{2g((1-g)(1+m)-gz)(1-g+gz)^{1+m}}{(1-g)(1+m)(2+m)}\mathrm{d}g \\
= &\int \frac{-2(1-e)(e(m+1)-(1-e)z)(1+(1-e)(z-1))^{m+1}}{e(m+1)(m+2)}\mathrm{d}e \\
= &\int \frac{(2z-2e(1+m+2z)-2e^2(-1-m-z))(z+e(1-z))^{m+1}}{e(m+1)(m+2)}\mathrm{d}e \\
= &\int \frac{2z(z+e(1-z))^{m+1}}{e(m+1)(m+2)}\mathrm{d}e + \int \frac{-2(1+m+2z)(z+e(1-z))^{m+1}}{(m+1)(m+2)}\mathrm{d}e \\
&+\int \frac{-2e(-1-m-z)(z+e(1-z))^{m+1}}{(m+1)(m+2)}\mathrm{d}e \\
= &\frac{2z}{(m+1)(m+2)}\left(z^{m+1}\log(e) + \sum_{j=1}^{m+1}\frac{z^{m+1-j}(z+e(1-z))^j}{j}\right) \\
&-\frac{2(1+m+2z)(z+e(1-z))^{m+2}}{(m+1)(m+2)^2(1-z)} - \frac{2e(-1-m-z)(z+e(1-z))^{m+2}}{(m+1)(m+2)^2(1-z)} \\
&+\frac{2(-1-m-z)(z+e(1-z))^{m+3}}{(m+1)(m+2)^2(m+3)(1-z)^2}.
\end{aligned} \tag{2.261}$$

Note that we have used the integral identity from Lemma 2.23. For the second and third terms, we have

$$\begin{aligned}
&\int_0^1 \left(\frac{((1-g)(1+m)-gz)(1-g+gz)^{1+m}}{(1-g)^2(1+m)(2+m)} - \frac{(1-2g+(1-g)m)z^{2+m}}{(1-g)^2(1+m)(2+m)} \right) \mathrm{d}g \\
= \;& \frac{2z}{(m+1)(m+2)} \sum_{j=1}^{m+1} \frac{z^{m+1-j}(z+e(1-z))^j}{j} - \frac{2(1+m+2z)(z+e(1-z))^{m+2}}{(m+1)(m+2)^2(1-z)} \\
&-\frac{2e(-1-m-z)(z+e(1-z))^{m+2}}{(m+1)(m+2)^2(1-z)} + \frac{2(-1-m-z)(z+e(1-z))^{m+3}}{(m+1)(m+2)^2(m+3)(1-z)^2} \\
&+\frac{2e(3+m)z^{m+2}}{(m+1)(m+2)} - \frac{e^2z^{m+2}}{(m+1)}\Bigg|_{e=1}^{e=0} \\
= \;& \frac{-2z}{(m+1)(m+2)} \sum_{j=1}^{m+1} \frac{z^{m+1-j}}{j} + \frac{2z}{(m+1)(m+2)^2(1-z)} \\
&+\frac{2(1+m+z)}{(m+1)(m+2)^2(m+3)(1-z)^2} - \frac{(6+2m)z^{m+2}}{(m+1)(m+2)} + \frac{z^{m+2}}{m+1} + \frac{2H(m+1)z^{m+2}}{(m+1)(m+2)} \\
&-\frac{2(1+m+2z)z^{m+2}}{(m+1)(m+2)^2(1-z)} - \frac{2(1+m+z)z^{m+3}}{(m+1)(m+2)^2(m+3)(1-z)^2}. \qquad (2.262)
\end{aligned}$$

Altogether,

$$\begin{aligned}
&\mathbb{P}(Z_*^l \le z | m, \overline{\mathcal{C}_1})_{\overline{\mathcal{E}}} \\
= \;& -\frac{2z\left(1+m-3z-mz+z^{2+m}(3+m-(1+m)z)\right)}{(1+m)(2+m)(3+m)(-1+z)^3} + \frac{-2z}{(m+1)(m+2)} \sum_{j=1}^{m+1} \frac{z^{m+1-j}}{j} \\
&+\frac{2z}{(m+1)(m+2)^2(1-z)} + \frac{2(1+m+z)}{(m+1)(m+2)^2(m+3)(1-z)^2} - \frac{(6+2m)z^{m+2}}{(m+1)(m+2)} \\
&+\frac{z^{m+2}}{m+1} + \frac{2H(m+1)z^{m+2}}{(m+1)(m+2)} - \frac{2(1+m+2z)z^{m+2}}{(m+1)(m+2)^2(1-z)} \\
&-\frac{2(1+m+z)z^{m+3}}{(m+1)(m+2)^2(m+3)(1-z)^2}. \qquad (2.263)
\end{aligned}$$

Chapter 3

Greedy Online Matching on Random Graphs

IN the online bipartite matching problem, we are given a bipartite graph $G = (I, J, E)$ where I is a set of n bins and J is a set of n balls. Balls arrive online; when a ball $j \in J$ arrives, its edges are revealed and it must immediately be either matched with an unmatched neighboring bin or dropped (left unmatched). Each bin may be matched to at most one ball and decisions are irrevocable. The goal is to maximize the number of matched balls.

The problem has received significant attention due to applications in Internet advertising as well as under streaming models of computation, which place limits on memory utilization for processing large datasets [61, 107, 124]. From a worst-case perspective, it is well known that the GREEDY algorithm, which matches each ball to a random unmatched neighboring bin (if possible), always achieves a matching size that is at least 1/2 the size of the maximum matching. The RANKING algorithm of Karp, Vazirani, and Vazirani [83] picks a random permutation of bins up front and matches each ball to the unmatched neighboring bin that is ranked highest in the permutation. The RANKING algorithm guarantees a matching size at least $1 - 1/e \approx 0.632$ of the size of the maximum matching in expectation, which is the best possible worst-case guarantee.

Surprisingly, the average-case performance of these algorithms has been largely overlooked. In this chapter, we study the performance of greedy-type matching algorithms on random graph models. We focus on Erdős-Rényi binomial graphs and random regular graphs. The binomial bipartite graph $\mathcal{G}(n, n, p)$ is the random graph with n vertices in each partition and where each potential edge $(i, j) \in I \times J$ occurs independently with probability $p = p(n)$. We study this model for essentially *all* functions $p(n)$. The random r-regular bipartite graph $\mathcal{G}(n, n, r)$ is defined by uniform distribution over all r-regular bipartite graphs. We analyze this model for $r = 2$ since we conjecture that this gives a lower bound on performance for $r \geq 2$.

We start by analyzing the OBLIVIOUS algorithm, which has knowledge of a ball's edges when it arrives but does not know which neighboring bins are occupied. The algorithm picks a random neighboring bin; if the bin is unmatched, then the ball is matched, otherwise it is dropped. This rule models load balancing scenarios where a central server knows which machines can process a given job, but is not aware of the machine availability. Next, we analyze Greedy and Ranking, showing that under the $\mathcal{G}(n, n, p)$ model, the performance of the two algorithms is equivalent. We extend our analysis to the vertex-weighted matching problem under the $\mathcal{G}(n, n, p)$ model, where each bin has a weight and the greedy algorithm matches each ball to the available neighboring bin with largest weight.

For 2-regular random bipartite graphs, we analyze Oblivious and Greedy, as well as a variation on Greedy that we refer to as Degree-Greedy. Suited specifically for online matching on random regular graphs, the Degree-Greedy algorithm favors matching to bins with more edges that have been revealed at the time of a ball arrival, as there will be fewer future opportunities to match such bins. We also consider a rollout technique for *offline* matching on 2-regular bipartite graphs, which we refer to as the Oblivious-Rollout algorithm. While this algorithm is awkward in a practical setting, we study it with the intent of better understanding rollout algorithms.

The online matching problem is motivated primarily for bipartite graphs, but we also define a generalization of the online matching for non-bipartite graphs. For this problem, all vertices in the graph arrive sequentially; when a vertex arrives, its edges connecting to *already existing vertices* are revealed. Each vertex is only allowed to be matched when it arrives or when one of its neighbors arrives, and matches are again irrevocable. We determine matching sizes obtained by a non-bipartite version of Greedy on the binomial graph $\mathcal{G}(n, p)$ and on the random regular graph $\mathcal{G}(n, r)$ for $r = 2$. We also examine a Degree-Greedy algorithm for non-bipartite graphs.

Our primary tool for analysis in this chapter is the differential equation method of Kurtz [97] and Wormald [148]. We specifically use Wormald's general theorem, which is tailored to random graph processes. In our proofs, we calculate the expected change in the number of matched vertices upon the arrival of each vertex as a function of the number of matched vertices. Normalizing these equations and taking the limit as $n \to \infty$ gives deterministic differential equations and solutions. The stochastic nature of the algorithms are shown to be concentrated around the deterministic solutions via Wormald's theorem. We believe that our proofs contain some of the simplest applications of Wormald's theorem. Our proofs for Greedy and Ranking on $\mathcal{G}(n, n, p)$, for instance, are nearly as simple as the worst-case proof for Ranking of Devanur, Jain, and Kleinberg [52].

One of the fascinating features of the differential equation method for random graphs is that, even in cases requiring the use of a complicated system of differential equations, it is often possible to obtain analytically tractable solutions. We will see that this is true for nearly all of our results.

In the next section we discuss related work. Section 3.2 presents some existing results that we will use and presents Wormald's theorem. Section 3.3 is dedicated to the analysis of bipartite graphs, first under the $\mathcal{G}(n, n, p)$ model and then under the regular model $\mathcal{G}(n, n, r)$ for $r = 2$. Section 3.4 handles non-bipartite graphs, likewise starting with $\mathcal{G}(n, p)$ and then moving to non-bipartite $\mathcal{G}(n, r)$ for $r = 2$. A discussion is given in Section 3.5, and Section 3.6 shows a proof of Wormald's theorem.

3.1 Related Work

The online matching problem was originally analyzed by Karp, Vazirani and Vazirani [83]; they introduced the RANKING algorithm and showed that it obtains a competitive ratio of $1 - 1/e$. Simpler proofs for RANKING algorithm have since been found [38, 52]. A $1 - 1/e$ competitive algorithm is known for vertex-weighted online bipartite matching, which was given by Aggarwal, Goel, Karande, and Mehta [2]. Outside of online matching, there are a variety of problems that have been studied in an online setting; the book by Borodin and El-Yaniv [42] gives a good introduction.

The use of deterministic differential equations to model random processes was first studied by Kurtz, who gave a general purpose theorem for continuous-time jump Markov processes [97]. A discrete-time theorem tailored for random graphs was given by Wormald, which we use in this chapter [147, 148]. The differential equation method has been used to study a variety of graph properties including k-cores, independent sets, and greedy packing on hypergraphs [120, 147, 148]. It was also used to analyze a load balancing scenario similar to ours by Mitzenmacher [110].

Early studies of matchings on random graphs focused on determining the existence of perfect matchings; Erdős and Rényi showed that the threshold for the existence of a perfect matching occurs when the graph is likely to have a minimum degree of one [57, 58]. One of the first studies of greedy matchings on random graphs, and one of the first to employ the differential equation method (via Kurtz's theorem), was the work of Karp and Sipser [82, 97]. They considered non-bipartite sparse graphs, specifically the $\mathcal{G}(n, p)$ model with $p = c/n$. The Karp-Sipser algorithm first matches all vertices with degree one until there are no such vertices remaining, and then obtains matches by selecting random edges. This results in a matching size that is within a factor $1 - o(1)$ of the maximum matching[1]. The algorithm was studied more in depth by Aronson, Frieze, and Pittel [13], who gave a precise bound on the $o(1)$ term.

Simpler greedy matching algorithms have been studied on a variety of non-bipartite graphs. Two general algorithms have been used. The first algorithm, which Dyer et al. [56] refer to as GREEDY (or RANDOMIZED-GREEDY), simply picks random edges to add to the matching until there are no edges remaining. The other algorithm, which is referred to as MODIFIED-GREEDY (or MODIFIED-RANDOMIZED-GREEDY), picks a vertex at random and then randomly chooses one of its neighboring edges to add to the matching (if no such edges exist, the vertex is simply removed).

Considering results that are valid for all graphs, Dyer and Frieze [55] showed that there exist graphs for which the GREEDY algorithm gives an average matching size nearly equal to the worst-case bound. Aronson, Dyer, and Frieze [12] showed that MODIFIED-GREEDY gives an expected matching size greater than $1/2 + \epsilon$ of the maxi-

[1]We write $f(n) = o(g(n))$ if and only if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$ [130].

mum matching size for some $\epsilon > 1/400,000$; remarkably, this holds for all graphs. The lower bound was improved to $1 + 1/256$ in the work of Poloczek and Szegedy [121].

In application to random graphs, Dyer, Frieze, and Pittel [56] analyzed the two algorithms on the Erdős-Rényi random graph $\mathcal{G}(n, M)$ where $M = \lfloor cn/2 \rfloor$. They showed that as $n \to \infty$, the matching sizes obtained by the algorithms follow a normal distribution with mean and variance determined by c. As we will show, the expected fraction of matched vertices for their MODIFIED-GREEDY algorithm is identical to the fraction matched by our GREEDY algorithm, both for bipartite and non-bipartite graphs $\mathcal{G}(n, n, p)$ and $\mathcal{G}(n, p)$ where $p = c/n$. Their results were obtained using differential equations based on moment generating functions – this approach is much more elaborate than ours and gives statements about the distribution of matching sizes rather than just expected sizes.

Greedy matching on random regular graphs was considered by Frieze, Radcliffe, and Shen [66]. They focused on random 3-regular graphs, which are referred to as random *cubic graphs*, and they employed the MINGREEDY algorithm. The MINGREEDY algorithm is similar to MODIFIED-GREEDY, except it always picks a starting vertex among those with minimum degree in the graph. They showed that this algorithm leaves $O(n^{1/5} \log n)$ unmatched vertices on random cubic graphs.

3.2 Background

In this section, we introduce graphs and random graph models, state a few key tools that will be used in our analysis, and provide some definitions for characterizing our results. Throughout the chapter, we use the term *asymptotically almost surely* (a.a.s.) to mean occurring with probability $1 - o(1)$.

An undirected graph $G = (V, E)$ is defined by a set V of *vertices* and a set E of *edges*. Each edge is an unordered pair (u, v) of vertices $u, v \in V$. The edge (u, v) is said to be a *neighbor* of u and of v; we also say a vertex is a neighbor of another vertex if they share an edge. A graph is *bipartite* if the vertex set V can be partitioned into two sets I and J such that every edge (u, v) has one vertex $u \in I$ and one vertex $v \in J$; such a graph is usually denoted by $G = (I, J, E)$. To provide intuition in the online setting, we sometimes refer to the set I as being a set of *bins* and the set J as being a set of *balls*. The *degree* of a vertex is the number of edges incident to it. An *r-regular* graph is a graph where all vertices have degree r.

A *matching* in a graph is a vertex-disjoint set of edges (i.e. a set of edges where no two edges share a vertex). A vertex is said to be *matched* if one of its edges is part of the matching. The *size* of a matching refers to the number of edges in the set, which is equal to half of the number of matched vertices. A *maximum matching* is a matching with maximum size[2]. A *perfect matching* is a matching where all vertices are matched.

Moving to random graph models, we start by defining the *binomial* random graph

[2]A *maximal matching*, on the other hand, is a matching that cannot accommodate any additional edges.

$\mathcal{G}(n,p)$, which is one of the two classical Erdős-Rényi models [39, 79]. The $\mathcal{G}(n,p)$ graph is the non-bipartite graph with n vertices, where each of the $\binom{n}{2}$ potential edges is present independently with probability p. The total number of edges occurring thus follows a binomial distribution with parameters $\binom{n}{2}$ and p. The other Erdős-Rényi graph model $\mathcal{G}(n,M)$, referred to as the *uniform* model, is specified by the uniform distribution over all n-vertex graphs with exactly M edges. We only consider $\mathcal{G}(n,p)$ for ease of analysis, but as discussed in both [39] and [79], properties of the two graphs are often similar for the appropriate choices of M and p.

The random regular graph $\mathcal{G}(n,r)$ is defined by the uniform distribution over all n-vertex r-regular graphs. We will always assume that rn is even. As noted in [79], despite the similar notation $\mathcal{G}(n,p)$ and $\mathcal{G}(n,r)$, it is usually clear from context which model is being referred to. In our analysis, we will need to construct random regular graphs, which takes some care. We will use the *configuration model*, which generates a random graph serving as an approximation to a random r-regular graph. In the configuration model for a given r, each vertex has r *slots*, giving a total of rn slots. All slots are paired uniformly at random, allowing for potential multiple edges and self loops. Note that it is not valid to generate $\mathcal{G}(n,r)$ by revealing each pair sequentially and resampling edges that result in multiple edges or self loops. This strategy can lead to a state where the only remaining slot pairing options give multiple edges or self loops. We use the notation $\mathcal{P}(n,r)$ to denote a random graph generated by the configuration model. If this process happens to generate a graph without multiple edges and self loops – such a graph is referred to as *simple* – the graph is a valid uniform sample from $\mathcal{G}(n,r)$.

The bipartite versions of binomial random graphs and random regular graphs are defined analogously over n by n bipartite graphs (meaning there are a total of $2n$ vertices with n vertices in each partition). The binomial random graph $\mathcal{G}(n,n,p)$ is the n by n bipartite graph with each edge between the two partitions occurring independently with probability p. Similarly, $\mathcal{G}(n,n,r)$ is defined by the uniform distribution over all n by n r-regular bipartite graphs. The bipartite configuration model is denoted by $\mathcal{P}(n,n,r)$. This model may generate graphs with multiple edges, but self loops cannot occur since all pairings are bipartite. We will prove many of our results using the configuration model via the following theorem (see, e.g. [79]).

Theorem 3.1. *Any property that holds a.a.s. for $\mathcal{P}(n,r)$ also holds a.a.s. for $\mathcal{G}(n,r)$. Likewise, any property that holds a.a.s. for $\mathcal{P}(n,n,r)$ also holds a.a.s. for $\mathcal{G}(n,n,r)$.*

For some algorithm A, let $\mu_{\mathrm{A}}(n,n,p)$ denote the matching size (equal to one half the total number of matched vertices) obtained by algorithm A on $\mathcal{G}(n,n,p)$, and let $\mu_{\mathrm{A}}(n,n,r)$ denote the matching size obtained by algorithm A on $\mathcal{G}(n,n,r)$. For non-bipartite graphs, let $\mu_{\mathrm{A}}(n,p)$ denote the matching size given by algorithm A on $\mathcal{G}(n,p)$ and let $\mu_{\mathrm{A}}(n,r)$ denote the matching size given by A on $\mathcal{G}(n,r)$.

For the binomial graphs $\mathcal{G}(n,n,p)$ and $\mathcal{G}(n,p)$, we will be interested in algorithm performance where p is a function $p=p(n)$. Define a *valid* function $p(n)$ as a function that for all $n>0$, is monotonic and satisfies $0<p(n)<1$ (thus implying that $0 \leq$

$\lim_{n\to\infty} p(n) \leq 1$). Our results will be stated by giving the asymptotic matching sizes obtained by algorithms that hold asymptotically almost surely. We will also state results normalized by the asymptotic expected maximum matching size. Denote the maximum matching size on $\mathcal{G}(n,n,p)$ by $\mu^*(n,n,p)$; note that $\mu^*(n,n,p)$ is a random variable. Consider an algorithm A and a function $p = p(n)$. We define the *performance ratio*[3] for the bipartite graph $\mathcal{G}(n,n,p)$ as

$$\mathcal{R}_{\mathrm{A}}(p(n)) := \lim_{n\to\infty} \frac{\mathbb{E}[\mu_{\mathrm{A}}(n,n,p(n))]}{\mathbb{E}[\mu^*(n,n,p(n))]}. \tag{3.1}$$

This definition also applies to the non-bipartite graph $\mathcal{G}(n,p)$, where $\mu^*(n,p)$ denotes the maximum matching size;

$$\mathcal{R}_{\mathrm{A}}(p(n)) := \lim_{n\to\infty} \frac{\mathbb{E}[\mu_{\mathrm{A}}(n,p(n))]}{\mathbb{E}[\mu^*(n,p(n))]}. \tag{3.2}$$

Extending the definition to random r-regular graphs, we note that all regular bipartite graphs have a perfect matching, so the denominator in the performance ratio is equal to n. Let $\mu_{\mathrm{A}}(n,n,r)$ denote the matching size obtained by algorithm A on $\mathcal{G}(n,n,r)$. The performance ratio for the bipartite graph $\mathcal{G}(n,n,r)$ is

$$\mathcal{R}_{\mathrm{A}}(r) := \lim_{n\to\infty} \frac{\mathbb{E}[\mu_{\mathrm{A}}(n,n,r)]}{n}. \tag{3.3}$$

Non-bipartite regular graphs are not guaranteed to have a perfect matching. In particular, it holds a.a.s. that the graph $\mathcal{G}(n,2)$ contains an odd cycle and thus cannot have a perfect matching [149]. On the other hand, $\mathcal{G}(n,r)$ for $r \geq 3$ a.a.s. has a perfect matching for nr even [39, 148]. Let $\mu_{\mathrm{A}}(n,r)$ denote the matching size obtained by algorithm A on $\mathcal{G}(n,r)$ and let $\mu^*(n,r)$ denote the maximum matching size. The performance ratio for the non-bipartite graph $\mathcal{G}(n,r)$ is

$$\mathcal{R}_{\mathrm{A}}(r) := \lim_{n\to\infty} \frac{\mathbb{E}[\mu_{\mathrm{A}}(n,r)]}{\mathbb{E}[\mu^*(n,r)]}. \tag{3.4}$$

While this definition includes $\mathbb{E}[\mu^*(n,r)]$ in the denominator instead of simply $n/2$, we will bound the performance ratio in our results using the fact that $\mu^*(n,r) \leq n/2$. Altogether, the performance ratio is essentially the analog of the competitive ratio from online algorithms adapted for average-case analysis.

Binomial random graphs with an edge probability function $p(n) = c/n$ for a constant $c > 0$ are referred to as *sparse* random graphs. The following results indicate the maximum matching sizes on $\mathcal{G}(n,n,p)$ and $\mathcal{G}(n,p)$ for the sparse regime, which we will use in the denominator of the performance ratios. For bounding the maximum

[3]A more rigorous metric is the expectation of the ratio of matching sizes, as used in [67], rather than the ratio of expectations. However, concentration results can often be used to establish relations between the two metrics; for example, see [100].

matching size of $\mathcal{G}(n,n,p)$, we use a result from Bollobás and Brightwell [40]. Their result (specifically Theorem 14 in [40]) is stated in terms of the size of the largest independent set for a bipartite graph, which by König's theorem bounds the maximum matching size [91]. Here and throughout the rest of the chapter, we define $\gamma_* = \gamma_*(c)$ as the smallest root of the equation $x = c\exp(-ce^{-x})$, and $\gamma^* = \gamma^*(c) = ce^{-\gamma_*}$.

Theorem 3.2 (Bollobás and Brightwell [40]). *Let $\mu^*(n,n,c/n)$ denote the size of the maximum matching on the graph $\mathcal{G}(n,n,p)$ where $p = c/n$. Then a.a.s.,*

$$\frac{\mu^*(n,n,c/n)}{n} \le 2 - \frac{\gamma^* + \gamma_* + \gamma^*\gamma_*}{c} + o(1), \tag{3.5}$$

where γ_* is the smallest root of the equation $x = c\exp(-ce^{-x})$ and $\gamma^* = ce^{-\gamma_*}$.

The above bound is known to be tight for $c \le e$. The asymptotic maximum matching size for the non-bipartite graph $\mathcal{G}(n,p)$ was determined in numerous papers [82, 119], and most recently by Aronson et al. [13]. Remarkably, this result is equivalent to the above result for bipartite graphs (differing only by a factor of two since the bipartite graph has twice as many vertices).

Theorem 3.3 (Aronson et al. [13]). *Let $\mu^*(n,c/n)$ denote the size of the maximum matching on the graph $\mathcal{G}(n,p)$ where $p = c/n$. Then a.a.s.,*

$$\frac{\mu^*(n,c/n)}{n} = 1 - \frac{\gamma^* + \gamma_* + \gamma^*\gamma_*}{2c} + o(1), \tag{3.6}$$

where γ_* is the smallest root of the equation $x = c\exp(-ce^{-x})$ and $\gamma^* = ce^{-\gamma_*}$.

We now discuss the two important theoretical tools that we will use in our proofs. The first and simpler is the celebrated Azuma-Hoeffding inequality, stated as follows [148]. We will use the inequality in conjunction with Doob martingales to show concentration of properties around their expected values.

Lemma 3.1 (Azuma-Hoeffding [17, 75]). *Let $X_0, X_1, \dots, X_t$ be a martingale such that $|X_i - X_{i-1}| \le c_i$, $1 \le i \le t$, for constants c_i. Then for any $\alpha > 0$,*

$$\mathbb{P}(|X_t - X_0| \ge \alpha) \le 2\exp\left(-\frac{\alpha^2}{2\sum_{i=1}^t c_i^2}\right). \tag{3.7}$$

Our second and more general tool is Wormald's theorem, which is used to show concentration of random processes around deterministic differential equations. We introduce the theorem using notation consistent with [148].

Consider a discrete-time random process defined for times $t = 0, 1, \dots$ and let S be a set of states. (The set S can be thought of as a set of deterministic graphs, for example.) At each time t, the process is in a unique state $q_t \in S$. Let Q_t denote the random variable for the state at time t. We define the probability space Ω to have elements corresponding to all possible sequences $(q_0, q_1, \ldots)$. Let $h_t = (q_0, \ldots, q_t)$ be the *history* of the process up to time t, and denote its random variable by $H_t = (Q_0, \ldots, Q_T)$.

We define a sequence of random processes indexed by $n = 1, 2, \ldots$ so that each process has a set of states $S^{(n)}$, and for each time t, the nth process is in a state $q_t^{(n)} \in S^{(n)}$. To simplify notation, we often omit n and write q_t to mean $q_t^{(n)}$ and S to mean $S^{(n)}$. Let $S^{(n)+}$ denote the set of all histories $h_t = (q_0, \ldots, q_t)$ where each $q_k \in S^{(n)}$ for $t = 0, 1, \ldots$.

Wormald's general theorem is stated as follows [148]. Note that in part (iii), a Lipschitz condition is satisfied for a function $f(u)$ on the domain D if there exists a constant $L > 0$ satisfying

$$|f(u) - f(v)| \leq L||u - v||_\infty \tag{3.8}$$

for all $u, v \in D$. The stopping time T_D is needed for situations where the Lipschitz condition fails, which often happens near the end of some graph processes.

Theorem 3.4 (Wormald [148]). *For $1 \leq l \leq a$, where a is fixed, let $y^{(l)} : S^{(n)+} \to \mathbb{R}$ and $f_l : \mathbb{R}^{(a+1)} \to \mathbb{R}$, such that for some constant C_0 and all l, $|y^{(l)}(h_t)| < C_0 n$ for all $h_t \in S^{(n)+}$ for all n. Let $Y_l(t)$ denote the random variable induced by $y_l(H_t)$. Assume the following three conditions hold, where D is some bounded connected open set containing the closure of*

$$\{(0, z_1, \ldots, z_a) : \mathbb{P}(Y_l(0) = z_l n, 1 \leq l \leq a) \neq 0 \text{ for some } n\}, \tag{3.9}$$

and T_D is a stopping time for the minimum t such that $(t/n, Y_1(t)/n, \ldots, Y_a(t)/n) \notin D$.

(i) (Boundedness hypothesis.) For some function $\beta = \beta(n) \geq 1$

$$\max_{1 \leq l \leq a} |Y_l(t+1) - Y_l(t)| \leq \beta, \tag{3.10}$$

for $t < T_D$.

(ii) (Trend hypothesis.) For some function $\lambda_1 = \lambda_1(n) = o(1)$, for all $1 \leq l \leq a$

$$|\mathbb{E}[Y_l(t+1) - Y_l(t)|H_t] - f_l(t/n, Y_1(t)/n, \ldots, Y_a(t)/n)| \leq \lambda_1 \tag{3.11}$$

for $t < T_D$.

(iii) (Lipschitz hypothesis.) Each function f_l is continuous, and satisfies a Lipschitz condition, on

$$D \cap \{(t, z_1, \ldots, z_a) : t \geq 0\}, \tag{3.12}$$

with the same Lipschitz constant for each l.

Then the following are true.

(a) For $(0, \widehat{z}_1, \ldots, \widehat{z}_a) \in D$ *the system of differential equations*

$$\frac{\partial z_l}{\partial x} = f_l(x, z_1, \ldots, z_a), \quad 1 \leq l \leq a \tag{3.13}$$

has a unique solution in D *for* $z_l : \mathbb{R} \to \mathbb{R}$ *passing through*

$$z_l(0) = \widehat{z}_l, \quad 1 \leq l \leq a \tag{3.14}$$

and which extends to points arbitrarily close to the boundary of D*;*

(b) Let $\lambda > \lambda_1$ *with* $\lambda = o(1)$*. For a sufficiently large constant* C*, with probability* $1 - O(\frac{\beta}{\lambda} \exp(-\frac{n\lambda^3}{\beta^3}))$*,*

$$Y_t^{(l)} = nz_l(t/n) + O(\lambda n) \tag{3.15}$$

uniformly for $0 \leq t \leq \sigma n$ *and for each* l*, where* $z_l(x)$ *is the solution in (a) with* $\widehat{z}_l = \frac{1}{n} Y_l(0)$*, and* $\sigma = \sigma(n)$ *is the supremum of those* x *to which the solution can be extended before reaching within* l^∞*-distance* $C\lambda$ *of the boundary of* D*.*

While this version of the theorem is most appropriate for our work, there are more general forms that allow the boundedness hypothesis to hold only with a certain probability and permit a to grow as a function of n [148].

3.3 Bipartite Graphs

As described in the chapter introduction, the online bipartite matching problem is given by a set of n *bin* vertices, which are given up front, and n sequentially arriving *ball* vertices. Recall that edges neighboring a ball only become known when the ball arrives, each ball must be matched before the next ball arrives, and matches are irrevocable. This implies that the simple greedy procedure of randomly assigning each ball to an unmatched neighboring bin is not necessarily optimal.

An example problem instance and progression of the greedy algorithm are shown in Figure 3.1. In this particular instance, the greedy algorithm makes two mistakes, leading to a suboptimal matching size. In Figure 3.1 (b), the first ball is matched to the third bin, which blocks the third ball from being matched in part (d), as the third ball can only be matched to the third bin. The same problem occurs in part (e) and part (f), where the fourth ball is matched to the fourth bin and the fifth ball is left unmatched.

The first subsection here is devoted the the Erdős-Rényi random graph $\mathcal{G}(n, n, p)$ and the second subsection considers the random regular graph $\mathcal{G}(n, n, r)$. Throughout our analysis, we will use both t and j to index arriving balls.

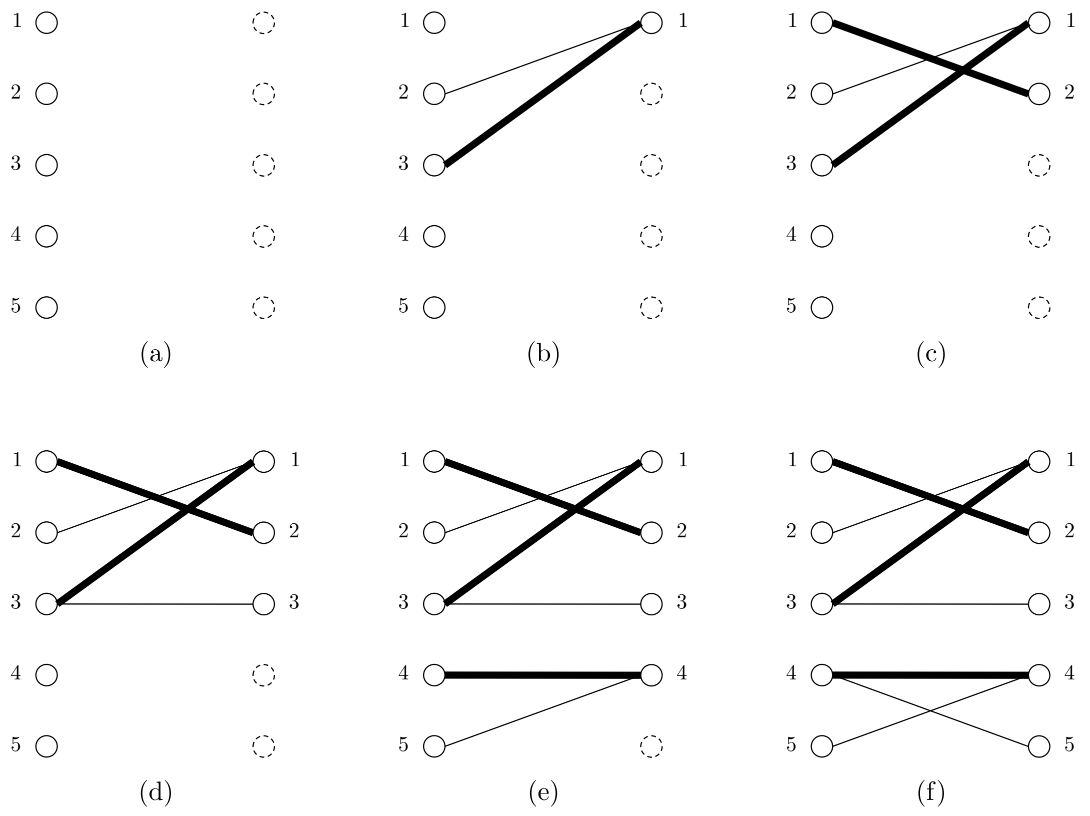


Figure 3.1. Instance and progression of online bipartite matching problem for $n = 5$. Bins vertices are shown on the left and arriving ball vertices are shown on the right. Bold edges indicate matches. When each ball arrives, all of its neighboring edges are revealed and it must be matched or dropped. As shown in part (d), the third ball must be dropped since its only neighbor is the third bin, which is occupied via the first ball. Similarly, the fifth ball must be dropped in part (f) since its only neighboring bin is occupied. It is easy to see that the maximum matching size for the offline problem is equal to 5.

3.3.1 Binomial Graphs

We introduce the Oblivious and Greedy online matching algorithms here, and analyze them on $\mathcal{G}(n, n, p)$ for all growth rates of valid functions $p = p(n)$. We then consider the extension to vertex weighted matching on sparse graphs, where $p = c/n$.

The following lemma will be used to satisfy the trend hypothesis of Wormald's theorem.

Lemma 3.2. *For $n > 0$, $c \leq n/2$, and $0 \leq w \leq 1$,*

$$0 \leq e^{-cw} - \left(1 - \frac{c}{n}\right)^{nw} \leq \frac{c}{ne}. \tag{3.16}$$

Proof. Using the inequalities $1 - x \geq e^{-x-x^2}$ for $x \leq 1/2$ and $1 - x \leq e^{-x}$ for $x \geq 0$, we obtain $e^{-cw}\left(1 - \frac{c^2 w}{n}\right) \leq \left(1 - \frac{c}{n}\right)^{nw} \leq e^{-cw}$. The result follows by rearranging terms and using $cwe^{-cw} \leq 1/e$. ■

Oblivious Algorithm

The OBLIVIOUS algorithm performs a one-shot trial for each ball j, where it attempts to match j to a random neighbor. The algorithm is unaware of which bins are already matched, so an attempted match to an already occupied bin means that ball j is dropped. This is shown in Algorithm 4. We use $N(j)$ to denote the set of neighboring bins of ball j.

```
1: for $j = 1$ to $n$ (each ball) do
2:     When ball $j$ arrives, let $N(j)$ denote the set of neighboring bins
3:     if $|N(j)| > 0$ then
4:         Select a random neighboring bin $i' \in N(j)$
5:         if bin $i'$ is unmatched then
6:             Match $j$ to $i'$
7:         else
8:             $j$ is dropped
9:         end if
10:    end if
11: end for
```

Algorithm 4. OBLIVIOUS

Intuitively, the algorithm is expected to do well on extremely sparse graphs, specifically those where balls are likely to have at most one neighbor. On the other hand, the performance on dense graphs should be suboptimal since each ball has a variety of neighboring bins that are not utilized by the algorithm. The following results confirm this behavior. For the sparse regime where $p = c/n$, Theorem 3.5 gives the asymptotic matching size obtained by OBLIVIOUS as a function of c. Theorem 3.6 then characterizes the performance ratio of OBLIVIOUS for all valid functions $p(n)$. For $p(n) = o(1/n)$, the performance ratio is equal to 1, whereas for $p(n) = \omega(1/n)$, the performance ratio is at least $1 - 1/e$. The phase transition of the lower bound thus occurs where $p(n) = c/n$, and is plotted in Figure 3.2. As Corollary 3.1 states, the global lower bound for the performance ratio is $1 - 1/e$.

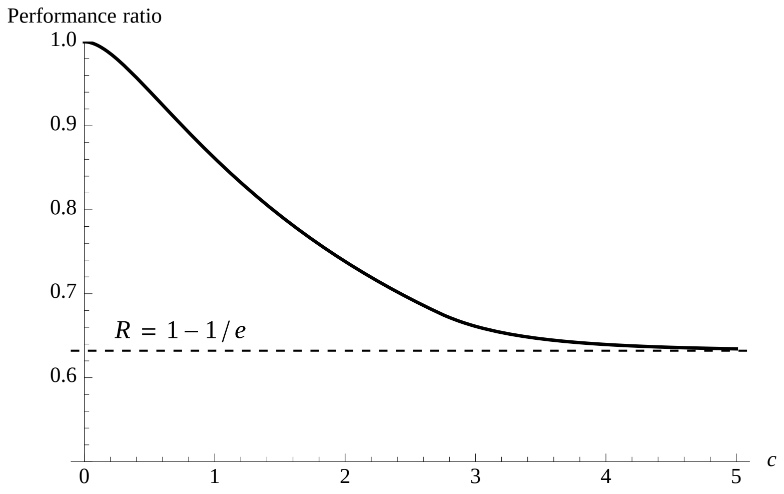


Figure 3.2. Performance ratio lower bound for OBLIVIOUS algorithm on $\mathcal{G}(n,n,c/n)$, according to Theorem 3.6. The global lower bound of $1-1/e$ is shown as dashed.

Theorem 3.5. *Let $\mu_{\mathrm{O}}(n,n,c/n)$ denote the matching size obtained by the* OBLIVIOUS *algorithm on the graph $\mathcal{G}(n,n,p)$, where $p=c/n$ and $c>0$ is a constant. Then a.a.s.,*

$$\frac{\mu_{\mathrm{O}}(n,n,c/n)}{n} = 1 - e^{(e^{-c}-1)} + o(1). \tag{3.17}$$

Theorem 3.6. *The performance ratio $\mathcal{R}_{\mathrm{O}}(p(n))$ of* OBLIVIOUS *on $\mathcal{G}(n,n,p)$, for all valid functions $p=p(n)$, satisfies*

$$\begin{aligned}
\mathcal{R}_{\mathrm{O}}(p(n)) &= 1, & p(n) &= o(1/n),\\
\mathcal{R}_{\mathrm{O}}(p(n)) &\geq \frac{c - ce^{(e^{-c}-1)}}{2c-(\gamma^*+\gamma_*+\gamma^*\gamma_*)}, & p(n) &= c/n,\\
\mathcal{R}_{\mathrm{O}}(p(n)) &\geq 1-\tfrac{1}{e}, & p(n) &= \omega(1/n).
\end{aligned} \tag{3.18}$$

Corollary 3.1. *The performance ratio of* OBLIVIOUS *on $\mathcal{G}(n,n,p)$ for all valid functions $p=p(n)$ satisfies*

$$\mathcal{R}_{\mathrm{O}}(p(n)) \geq 1 - \frac{1}{e}. \tag{3.19}$$

We use simple arguments to prove the $p(n)=o(1/n)$ and $p(n)=\omega(1/n)$ parts of Theorem 3.6, corresponding to Lemma 3.3 and Lemma 3.4 below. We then present two proofs of Theorem 3.5, one based on a linearity of expectation argument, and another using Wormald's theorem. This allows us to prove the $p=c/n$ part of Theorem 3.6, with the help of Theorem 3.2.

Lemma 3.3. *For all valid functions $p(n)$ where $p(n) = o(1/n)$, the performance of* Oblivous *on $\mathcal{G}(n, n, p)$ with $p = p(n)$ satisfies $\mathcal{R}_{\mathrm{G}}(p(n)) = 1$.*

Proof. We consider the number of isolated edges in $\mathcal{G}(n, n, p)$, where an edge (i, j) is isolated if $N(i) = \{j\}$ and $N(j) = \{i\}$. For any bin and ball pair (i, j), the probability of an isolated edge occurring between them is

$$\mathbb{P}((i, j) \text{ is isolated edge}) = p(1-p)^{(2n-2)}. \tag{3.20}$$

Every isolated edge is matched by Oblivious, so

$$\mathbb{E}[\mu_{\mathrm{O}}] \geq n^2 p(1-p)^{(2n-2)}, \tag{3.21}$$

where μ_{O} is the size of the Oblivious matching. The maximum matching size, denoted by μ^*, is upper bounded in expectation by the expected number of edges, $n^2 p$, so

$$\frac{\mathbb{E}[\mu_{\mathrm{O}}]}{\mathbb{E}[\mu^*]} \geq (1-p)^{(2n-2)} \geq (e^{-p-p^2})^{(2n-2)} = (e^{-o(\frac{1}{n})})^{(2n-2)} = e^{-o(1)} \to 1. \tag{3.22}$$

■

Lemma 3.4. *For all valid functions $p(n)$ where $p(n) = \omega(1/n)$, the performance of* Oblivous *on $\mathcal{G}(n, n, p)$ with $p = p(n)$ satisfies $\mathcal{R}_{\mathrm{G}}(p(n)) \geq 1 - 1/e$.*

Proof. Fix a bin i. For each ball j, there is an attempted assignment of j to i with probability $1/n$ if j has at least one neighbor. Thus,

$$\mathbb{P}(\text{no attempted match of } j \text{ to } i) = 1 - \left(\frac{1-(1-p)^n}{n}\right). \tag{3.23}$$

Considering all balls and using $1 - x \leq e^{-x}$,

$$\begin{aligned} \mathbb{P}(\text{no attempted match to } i) &\leq \left(1 - \left(\frac{1-e^{-pn}}{n}\right)\right)^n \\ &= \left(1 - \left(\frac{1-e^{-\omega(1)}}{n}\right)\right)^n \to \left(1 - \frac{1}{n}\right)^n \to e^{-1}. \end{aligned} \tag{3.24}$$

Accordingly, the expected number of bins matched by Oblivious is at least $n(1-1/e)$. The maximum matching size is at most n, so the result follows. The bound is shown to be tight by simply setting $p(n) = 1 - \epsilon$ where $\epsilon > 0$ and letting $\epsilon \to 0$. ■

First proof of Theorem 3.5. As in the proof directly above, we fix a bin i and note that for ball j,

$$\begin{aligned} \mathbb{P}(\text{no attempted match of } j \text{ to } i) &= 1 - \left(\frac{1-(1-p)^n}{n}\right) \\ &= 1 - \left(\frac{1-(1-\frac{c}{n})^n}{n}\right) \\ &\to 1 - \left(\frac{1-e^{-c}}{n}\right). \end{aligned} \tag{3.25}$$

where on the second line we have substituted $p = c/n$ and on the third line we have used the limiting definition of the natural exponent function. Considering all balls,

$$\begin{aligned} \mathbb{P}(\text{no attempted match to } i) &= \left(1 - \left(\frac{1-(1-\frac{c}{n})^n}{n}\right)\right)^n \\ &\to \left(1 - \left(\frac{1-e^{-c}}{n}\right)\right)^n \\ &\to e^{\left(e^{-c}-1\right)}, \end{aligned} \tag{3.26}$$

where on the last line we have again used the definition of the natural exponent function. We then have the probability that bin i is matched is

$$\mathbb{P}(\text{bin } i \text{ is matched}) = 1 - e^{\left(e^{-c}-1\right)}. \tag{3.27}$$

This gives that

$$\mathbb{E}[\mu_{\mathrm{O}}] = n(1 - e^{\left(e^{-c}-1\right)}). \tag{3.28}$$

To see the concentration around the expected value, we use a Doob martingale indexed by the arriving balls. Using Lemma 3.1 with $c_i = 1$ for $i = 1, \dots, n$ and $\alpha = n^{3/4}$, we have with probability $1 - e^{-\sqrt{n}/2}$,

$$\mu_{\mathrm{O}} = n(1 - e^{\left(e^{-c}-1\right)}) + n^{3/4}. \tag{3.29}$$

■

Second proof of Theorem 3.5. Let $Y(t)$ denote the number of occupied bins immediately before the t^{th} ball arrives, where $Y(1) = 0$. Slightly abusing notation, we use $Y(t)$ to denote both the random variable and instances of the random variable. Given $Y(t)$, the t^{th} ball is dropped if it is isolated or if its selected neighbor is already matched. Thus,

$$\begin{aligned} \mathbb{P}(\text{ball } t \text{ is matched} | Y(t)) &= (1-(1-p)^n)\left(1 - \frac{Y(t)}{n}\right) \\ &= \mathbb{E}[Y(t+1) - Y(t) | Y(t)]. \end{aligned} \tag{3.30}$$

Define the normalized random variable

$$Z(x) = \frac{Y(nx)}{n}, \quad 0 \le x \le 1, \tag{3.31}$$

so that $Z(x)$ indicates the fraction of occupied bins after nx of the arrivals have occurred. We have

$$\begin{aligned}\frac{\mathbb{E}[Z(x+1/n) - Z(x)|Z(x)]}{1/n} &= (1-(1-p)^n)(1-Z(x)) \\ &= \left(1-\left(1-\frac{c}{n}\right)^n\right)(1-Z(x)) \\ &= (1-e^{-c})(1-Z(x)) + o(1). \end{aligned} \tag{3.32}$$

As $n \to \infty$, we arrive at the differential equation

$$\frac{dz(x)}{dx} = (1-e^{-c})(1-z(x)), \tag{3.33}$$

where $z(\cdot)$ is deterministic and replaces $Z(\cdot)$. Integrating and using $z(0) = 0$ gives

$$z(x) = 1 - e^{(e^{-c}-1)x}, \quad 0 \le x \le 1. \tag{3.34}$$

Applying Theorem 3.4, we choose the domain D defined by $-\epsilon < x < 1+\epsilon$ and $-\epsilon < z(x) < 1+\epsilon$, for $\epsilon > 0$. Clearly we have $C_0 = 1$ and $\beta = 1$ by the nature of the matching process. Let $\lambda_1 = c/(en)$ for the trend hypothesis, which is satisfied according to Lemma 3.2. The Lipschitz hypothesis is satisfied with a Lipschitz constant $L = (1+\epsilon)(1-e^{-c})$. Setting $x = 1$ and choosing $\lambda = cn^{-1/4}$, Theorem 3.4 (b) gives that with probability $1 - O(n^{1/4}e^{-c^3 n^{1/4}})$,

$$\mu_{\mathrm{O}}(n,n,p) = n(1-e^{(e^{-c}-1)}) + O(n^{3/4}). \tag{3.35}$$

■

Proof of Theorem 3.6. The cases of $p(n) = o(1/n)$ and $p(n) = \omega(1/n)$ are given by Lemma 3.3 and Lemma 3.4. For the regime $p = c/n$, define the normalized random variables

$$\widetilde{\mu}_{\mathrm{O}}(n,n,c/n) := \frac{\mu_{\mathrm{O}}(n,n,c/n)}{n}, \tag{3.36}$$

$$\widetilde{\mu}^*(n,n,c/n) := \frac{\mu^*(n,n,c/n)}{n}. \tag{3.37}$$

By Theorem 1, $\widetilde{\mu}_{\mathrm{O}}$ converges in probability. Additionally, $\widetilde{\mu}_{\mathrm{O}}$ is bounded and thus uniformly integrable, so convergence in probability implies convergence in mean [72];

$$\lim_{n\to\infty} \mathbb{E}[\widetilde{\mu}_{\mathrm{O}}(n,n,c/n)] = 1 - e^{(e^{-c}-1)}. \tag{3.38}$$

Similarly, $\widetilde{\mu}^*$ must satisfy

$$\lim_{n\to\infty} \mathbb{E}[\widetilde{\mu}^*(n, n, c/n)] \geq 2 - \frac{\gamma^* + \gamma_* + \gamma^*\gamma_*}{c}. \tag{3.39}$$

■

Proof of Corollary 3.1. The expected maximum matching size is no greater than the expected number of non-isolated vertices on one side of the graph, so $\mathcal{R}_{\mathrm{O}}(c/n) \geq \frac{1-e^{(e^{-c}-1)}}{1-e^{-c}} \geq 1 - 1/e$. ■

Greedy Algorithm

The Greedy algorithm is shown in Algorithm 5. Upon the arrival of each ball, Greedy assigns the ball to a randomly selected neighboring bin that is unmatched. If no such bin exists, the ball is dropped. Recall that the Ranking algorithm instead picks an initial ranking of bins and always assigns each ball to the neighboring unmatched bin with highest rank. The Ranking algorithm is shown in Algorithm 6, where $\rho : I \to \{1, \ldots, n\}$ is an injective mapping, or *ranking* of bins, chosen uniformly at random. We will see that Ranking and Greedy perform equivalently on $\mathcal{G}(n, n, p)$.

```
1: for $j = 1$ to $n$ (each ball) do
2:     When ball $j$ arrives, let $U(j)$ denote the set of unmatched neighboring bins
3:     if $|U(j)| > 0$ then
4:         Match $j$ to a random bin $i' \in U(j)$
5:     else
6:         $j$ is dropped
7:     end if
8: end for
```

Algorithm 5. Greedy (bipartite)

```
1: Choose a random permutation of bins $\rho$
2: for $j = 1$ to $n$ (each ball) do
3:     When ball $j$ arrives, let $U(j)$ denote the set of unmatched neighboring bins
4:     if $|U(j)| > 0$ then
5:         Match $j$ to the bin $i' = \operatorname{argmin}_{i \in U(j)} \rho(i)$
6:     else
7:         $j$ is dropped
8:     end if
9: end for
```

Algorithm 6. Ranking

Just as with Oblivious, we consider the performance of Greedy in three regimes. It is clear that on extremely sparse graphs, Greedy should perform near optimally, as

OBLIVIOUS does. In contrast to OBLIVIOUS, however, the GREEDY algorithm should also perform well on very dense graphs because many bins are available to each ball. Indeed, this is the case; the following results show that GREEDY is only suboptimal in the regime $p(n) = c/n$. For this regime, the asymptotic matching size is given by Theorem 3.7. Theorem 3.8 gives the lower bound on the performance ratio across all three regimes, stating that the performance ratio for $p(n) = o(1/n)$ and $p(n) = \omega(1/n)$ is equal to one. Figure 3.3 shows that for $p(n) = c/n$, the lower bound on the GREEDY performance ratio passes through its global minimum of 0.837. This minimum is formalized with Corollary 3.2. Finally, Theorem 3.9 states that GREEDY performs equivalently to any online matching algorithm ALG. Since RANKING is a valid instance of ALG, the equivalence of RANKING and GREEDY follows from this theorem, as stated by Corollary 3.3. When we say that two algorithms perform *equivalently*, we mean that they generate the same probability distribution of matching sizes.

Theorem 3.7. *Let $\mu_G(n, n, c/n)$ denote the matching size obtained by the* GREEDY *algorithm on the graph $\mathcal{G}(n, n, p)$, where $p = c/n$ and $c > 0$ is a constant. Then a.a.s.,*

$$\frac{\mu_G(n, n, c/n)}{n} = 1 - \frac{\log(2 - e^{-c})}{c} + o(1). \tag{3.40}$$

Theorem 3.8. *The performance ratio $\mathcal{R}_G(p(n))$ of* GREEDY *on $\mathcal{G}(n, n, p)$, for all valid functions $p = p(n)$, satisfies*

$$\begin{aligned} \mathcal{R}_G(p(n)) &= 1, & p(n) &= o(1/n), \\ \mathcal{R}_G(p(n)) &\geq \frac{c - \log(2 - e^{-c})}{2c - (\gamma^* + \gamma_* + \gamma^*\gamma_*)}, & p(n) &= c/n, \\ \mathcal{R}_G(p(n)) &= 1, & p(n) &= \omega(1/n). \end{aligned} \tag{3.41}$$

Corollary 3.2. *The performance ratio of* GREEDY *on $\mathcal{G}(n, n, p)$ for all valid functions $p = p(n)$ satisfies*

$$\mathcal{R}_G(p(n)) \geq 0.837. \tag{3.42}$$

Theorem 3.9. *Consider any online bipartite matching algorithm* ALG *that always matches each ball to an unmatched neighboring bin, if possible. Then* ALG *and* GREEDY *perform equivalently on $\mathcal{G}(n, n, p)$.*

Corollary 3.3. *The* GREEDY *and* RANKING *algorithms perform equivalently on $\mathcal{G}(n, n, p)$.*

We begin our analysis by proving Theorem 3.9, which implies the equivalence of GREEDY and RANKING. We address the very sparse ($p(n) = o(1/n)$) and dense ($p(n) = \omega(1/n)$) regions for Theorem 3.8 with Lemma 3.5 and 3.6, respectively.

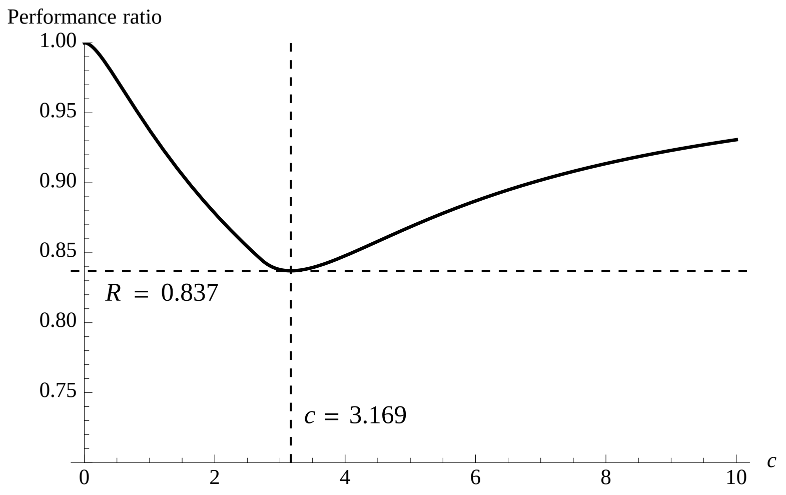


Figure 3.3. Performance ratio lower bound for GREEDY algorithm on $\mathcal{G}(n,n,c/n)$, according to Theorem 3.8. The global minimum of 0.837 at $c = 3.169$ is shown at the intersection of the dashed lines.

Then, we prove Theorem 3.7 and the sparse ($p(n) = c/n$) component of Theorem 3.8 using Wormald's theorem. The proof of Corollary 3.2 follows easily afterwards.

Proof of Theorem 3.9. We define a Markov chain for the simultaneous evolution of the decisions made by ALG and the construction of $\mathcal{G}(n,n,p)$. The Markov chain has a finite state space defined by variables $(t, Y(t))$. Specifically, $t = 1, \ldots, n+1$ indexes arriving balls and $Y(t)$ denotes the number of matched bins immediately before ball t arrives. The extra index $t = n+1$ indicates the completion of the process and $Y(n+1)$ is the final number of matched bins. Referring to all states with a fixed t as a *stage* in the process, each stage has t states corresponding to the possible number of matched bins $Y(t) = 0, \ldots, t-1$.

By definition of $\mathcal{G}(n,n,p)$, the neighboring edges for each ball are drawn independently; in particular, they are drawn independent of the bins that have been matched by ALG. Note also that since ALG is an online algorithm, it is not possible for ALG to know which edges are present for future balls. Thus, the probability of a transition from state (t, Y) to state $(t+1, Y)$ (where a match is not possible) is $(1-p)^{(n-Y(t))}$, while the probability of a transition to state $(t+1, Y+1)$ (where a match occurs) is $1-(1-p)^{(n-Y(t))}$.

The distribution of the number of bins matched by ALG is given by the distribution of states for stage $t = n+1$ (i.e. the distribution of $Y(n+1)$). The Markov chain is valid for any algorithm ALG, again as long as it is an online algorithm that matches a ball to an unmatched neighboring bin, given the opportunity. However, for a particular graph instance of $\mathcal{G}(n,n,p)$, two algorithms may obviously generate different matching sizes.

Since the Greedy algorithm is a valid instance of ALG, the proof is complete. ∎

Lemma 3.5. *For all valid functions $p(n)$ where $p(n) = o(1/n)$, the performance of* Greedy *on $\mathcal{G}(n, n, p)$ with $p = p(n)$ satisfies $\mathcal{R}_{\mathrm{G}}(p(n)) = 1$.*

Proof. The proof for Lemma 3.3 holds. ∎

Lemma 3.6. *For all valid functions $p(n)$ where $p(n) = \omega(1/n)$, the performance of* Greedy *on $\mathcal{G}(n, n, p)$ with $p = p(n)$ satisfies $\mathcal{R}_{\mathrm{G}}(p(n)) = 1$.*

Proof. We use a crude lower bound for the probability that each ball is matched. When ball t arrives, at most $t-1$ bins are occupied, so

$$\mathbb{P}(\text{ball } t \text{ is matched}) \geq 1 - (1-p)^{(n-t+1)}. \tag{3.43}$$

Let μ_{G} denote the matching size obtained by Greedy. Then

$$\mathbb{E}[\mu_{\mathrm{G}}] \geq n - \sum_{t=1}^{n} (1-p)^{(n-t+1)} = n - \frac{(1-(1-p)^n)(1-p)}{p} \geq n - \frac{1}{p}. \tag{3.44}$$

The maximum matching size is at most n, so

$$\frac{\mathbb{E}[\mu_{\mathrm{G}}]}{\mathbb{E}[\mu^*]} \geq 1 - \frac{1}{np(n)} = 1 - \frac{1}{\omega(1)} \to 1. \tag{3.45}$$

∎

Proof of Theorem 3.7. Again let $Y(t)$ denote the number of occupied bins immediately before the t^{th} arrival. Conditioning on $Y(t)$, the t^{th} ball cannot be assigned only if edges connecting the ball to the $n - Y(t)$ neighboring bins are not present. The probability of a match is then

$$\begin{aligned} \mathbb{P}(\text{ball } t \text{ is matched} | Y(t)) &= 1 - (1-p)^{n\left(1 - \frac{Y(t)}{n}\right)} \\ &= \mathbb{E}[Y(t+1) - Y(t) | Y(t)]. \end{aligned} \tag{3.46}$$

Normalizing as before, we obtain

$$\frac{\mathbb{E}[Z(x+1/n) - Z(x)|Z(x)]}{1/n} = 1 - \left(1 - \frac{c}{n}\right)^{n(1-Z(x))} \tag{3.47}$$

$$= 1 - e^{-c(1-Z(x))} + o(1). \tag{3.48}$$

The corresponding differential equation for $n \to \infty$ is

$$\frac{dz(x)}{dx} = 1 - e^{-c(1-z(x))}. \tag{3.49}$$

By integration and the initial value for $z(x)$,

$$z(x) = 1 - \frac{\log\left(1 + e^{-cx}(e^c - 1)\right)}{c}, \quad 0 \le x \le 1. \tag{3.50}$$

The application of Wormald's theorem is the same as the proof for Oblivious but with Lipschitz constant $L = ce^{c\epsilon}$. ■

Proof of Theorem 3.8. The proof follows the same approach as the proof of Theorem 3.6. ■

Proof of Corollary 3.2. The approach for Corollary 3.1 can be used to show that the property holds for small c values (e.g. $c < 1/2$) and large c values (e.g. $c > 5$). For the remaining region, the exact expression for the performance ratio can be minimized numerically; the minimum is obtained at $c^* = 3.1685009$ and $\mathcal{R}_{\mathrm{G}}(c^*) = 0.8370875$. ■

Vertex-Weighted Matching

We now modify the online bipartite matching problem so that each bin i has a *rank*, $r_i \in \{1, 2, \ldots, m\}$ for some constant m, and our goal is to match balls to lower ranked bins. This is better understood by allowing each bin to have one of m distinct weights $w_i \in \mathbb{R}$, where a larger weighted bin has a lower rank, and the goal of the matching problem is to maximize the weighted sum of matched bins. The appropriate greedy algorithm for this problem is the Vertex-Weighted-Greedy algorithm, which matches each ball to its neighboring bin with smallest rank, as shown in Algorithm 7.

```
1: for $j = 1$ to $n$ (each ball) do
2:     When ball $j$ arrives, let $U(j)$ denote the set of unmatched neighboring bins
3:     if $|U(j)| > 0$ then
4:         Match $j$ to a neighboring bin $i' \in \operatorname{argmin}_{i \in U(j)} r_i$
5:     else
6:         $j$ is dropped
7:     end if
8: end for
```

Algorithm 7. Vertex-Weighted-Greedy

We will only consider the performance of this algorithm in the sparse regime, where $p(n) = c/n$. Interestingly, the differential equation method is valid and tractable for this setting. Let n_r denote the total number of bins with rank r, where $\sum_r n_r = n$, and let $g_r := n_r/n$. The result of this section is as follows.

Theorem 3.10. *Let $\mu_{(\mathrm{W},r)}(n, n, c/n)$ denote the number of bins with rank r that are matched by* Vertex-Weighted-Greedy *on the graph $\mathcal{G}(n, n, p)$, where $p = c/n$ and*

$c > 0$ *is a constant. Then a.a.s.,*

$$\frac{\mu_{(\mathrm{W},r)}(n,n,c/n)}{n} = g_r - \frac{1}{c}\log\left(\frac{1+e^{-cx}(e^{c\sum_{k=1}^{r} g_k}-1)}{1+e^{-cx}(e^{c\sum_{k=1}^{r-1} g_k}-1)}\right) + o(1), \quad 1 \le r \le m. \tag{3.51}$$

Proof. We use the notation $Y_r(t)$, $1 \le r \le m$, to denote the number of rank r bins that are occupied immediately prior to the arrival of ball t. Consider the probability that a given ball t is matched to a rank two bin. This occurs if edges connecting to the $n_1 - Y_1(t)$ unmatched rank one bins are not present and there is at least one unmatched neighboring bin with rank two, so

$$\mathbb{E}[Y_2(t+1) - Y_2(t)|\mathbf{Y}(t)] = \left((1-p)^{n\left(g_1 - \frac{Y_1(t)}{n}\right)}\right)\left(1-(1-p)^{n\left(g_2 - \frac{Y_2(t)}{n}\right)}\right), \tag{3.52}$$

where $\mathbf{Y}(t) = (Y_1(t), Y_2(t), \ldots Y_a(t))$. Generalizing, for $1 \le r \le m$,

$$\mathbb{E}[Y_r(t+1) - Y_r(t)|\mathbf{Y}(t)] \;=\; \left(1-(1-p)^{n\left(g_r - \frac{Y_r(t)}{n}\right)}\right)\prod_{k=1}^{r-1}\left((1-p)^{n\left(g_k - \frac{Y_k(t)}{n}\right)}\right). \tag{3.53}$$

After normalizing and substituting $p = c/n$,

$$\frac{\mathbb{E}[Z_r(x+1/n) - Z_r(x)|\mathbf{Z}(x)]}{1/n} \;=\; \left(1-e^{-c(g_r - Z_r(x))}\right)e^{-c\sum_{k=1}^{r-1}(g_k - Z_k(x))} + o(1). \tag{3.54}$$

We arrive at the following system of differential equations for $n \to \infty$:

$$\frac{dz_r(x)}{dx} = \left(1-e^{-c(g_r - z_r(x))}\right)e^{-c\sum_{k=1}^{r-1}(g_k - z_k(x))}, \quad 1 \le r \le m. \tag{3.55}$$

It can be verified that the solution to the system of differential equations with initial conditions $z_r(0) = g_r$, $1 \le r \le m$, is

$$z_r(x) = g_r - \frac{1}{c}\log\left(\frac{1+e^{-cx}(e^{c\sum_{k=1}^{r} g_k}-1)}{1+e^{-cx}(e^{c\sum_{k=1}^{r-1} g_k}-1)}\right), \quad 0 \le x \le 1, \tag{3.56}$$

for $1 \le r \le m$. The application of Wormald's theorem is the same as the application of Oblivious and Greedy, but with Lipschitz constant $L = (a-1)(1-e^{-c(1+\epsilon)})ce^{(a-1)c\epsilon} + ce^{ac\epsilon}$. ■

3.3.2 Regular Graphs

We now move to the random regular graph $\mathcal{G}(n,n,r)$. Ideally, we would state general results for matching sizes obtained as a function of r, but the complexity of the analysis for regular graphs restricts us to the case where $r = 2$ for most of our work.

While offline matching on 2-regular graphs is obviously a trivial problem, there is no simple algorithm for the online problem on $\mathcal{G}(n,n,r)$ for $r = 2$ that guarantees a perfect matching. Note that for $r = 1$, even the online problem is trivial and essentially any algorithm with knowledge of arriving edges is optimal. For $r \geq 2$, the Greedy algorithm is not optimal for the online problem, but we expect the asymptotic matching size obtained by Greedy on r-regular graphs to be monotonically increasing in r for $r \geq 2$.

We will see that random 2-regular graphs are still amenable to analysis via Wormald's theorem, albeit with a bit more work than for Erdős-Rényi graphs. We will have to designate cases for various bins based on how many edges have been revealed, and model the density of these cases with the differential equations. Nevertheless, we can still obtain simple expressions for matching sizes.

This subsection is structured as follows. We begin with an analysis of Greedy on 2-regular graphs. We then introduce the Degree-Greedy algorithm, which is suited for regular graphs and performs slightly better than Greedy. Finally, we step away from the online setting to study a rollout algorithm for the offline matching problem on 2-regular graphs. This algorithm uses the Oblivious algorithm as a base policy and is accordingly called the Oblivious-Rollout algorithm. Offline matching on 2-regular graphs, as we have stated, is a trivial problem. Our goal in studying Oblivious-Rollout is to better understand the general behavior of rollout algorithms.

Oblivious and Greedy

The Oblivious algorithm is easy enough to analyze that we can state its performance on $\mathcal{G}(n,n,r)$ as a general function of r, and we can use the uniform model directly instead of the configuration model. The following theorem follows using linearity of expectation and the Azuma-Hoeffding inequality.

Theorem 3.11. *Let $\mu_{\mathrm{O}}(n,n,r)$ denote the matching size obtained by the* Oblivious *algorithm on the random regular graph $\mathcal{G}(n,n,r)$ for $r = 1, 2, \ldots$. Then a.a.s.,*

$$\frac{\mu_{\mathrm{O}}(n,n,r)}{n} = 1 - \left(1 - \frac{1}{r}\right)^r + o(1). \tag{3.57}$$

Proof. The expected value follows using a simple linearity of expectation argument, similar to the first proof of Theorem 3.5. Then, using Lemma 3.1 with $c_i = 1$ for $i = 1, \ldots, n$ and $\alpha = n^{3/4}$, we have with probability $1 - e^{-\sqrt{n}/2}$,

$$\mu_{\mathrm{O}}(n,n,r) = n\left(1 - \left(1 - \frac{1}{r}\right)^r\right) + n^{3/4}. \tag{3.58}$$

■

To analyze the Greedy algorithm on $\mathcal{G}(n,n,2)$, we must use the configuration model $\mathcal{P}(n,n,r)$. This means that we will allow multiple edges to occur, though this will happen with negligible probability. When each ball arrives, two edges are drawn to determine neighboring bins. Prior to drawing the edges, there are three possibilities for the type of each bin. These possibilities are labeled as

$$\begin{aligned} A:&\quad \text{the bin has no edges,}\\ B:&\quad \text{the bin has one edge and is matched,}\\ C:&\quad \text{the bin has one edge and is unmatched.}\end{aligned}$$

Ball arrivals are indexed by t. Due to Lipschitz continuity requirements in Wormald's theorem, we make the restriction $t < n(1-\epsilon)$ for some $\epsilon > 0$. Immediately before ball t arrives, let $Y_A(t), Y_B(t), Y_C(t)$ be the number of bins with labels A, B, and C, respectively, and define the vector $\mathbf{Y}(t) = (Y_A(t), Y_B(t), Y_C(t))$. Let $Y_\mu(t)$ indicate the number of matched bins. Throughout the process, we always have

$$2Y_A(t) + Y_B(t) + Y_C(t) = 2(n-t+1), \tag{3.59}$$

which follows from the number of remaining edge slots, where each A bin contributes two slots.

Let $\ell_1(t)$ and $\ell_2(t)$ indicate the bin labels for the first and second edges of ball t. The probability that both neighbors satisfy case A, for example, is

$$\begin{aligned}\mathbb{P}(\ell_1(t) = A, \ell_2(t) = A) &= \frac{4Y_A(t)(Y_A(t)-1)}{(2Y_A(t)+Y_B(t)+Y_C(t))(2Y_A(t)+Y_B(t)+Y_C(t)-1)}\\ &= \frac{Y_A(t)^2}{(n-t+1)^2} + o(1),\end{aligned} \tag{3.60}$$

conditioned on the vector $\mathbf{Y}(t)$. The GREEDY algorithm matches the ball to one of the two bin, so one case A bin is lost, one case C bin is gained, and one case B bin is gained. Let $\chi\eta$ denote the event $(\ell_1(t) = \chi, \ell_2(t) = \eta) \vee (\ell_1(t) = \eta, \ell_2(t) = \chi)$. Hence for the event AA we have

$$\begin{aligned}\mathbb{E}[Y_A(t+1) - Y_A(t)|\mathbf{Y}(t), AA] &= -2,\\ \mathbb{E}[Y_B(t+1) - Y_B(t)|\mathbf{Y}(t), AA] &= 1,\\ \mathbb{E}[Y_C(t+1) - Y_C(t)|\mathbf{Y}(t), AA] &= 1,\\ \mathbb{E}[Y_\mu(t+1) - Y_\mu(t)|\mathbf{Y}(t), AA] &= 1.\end{aligned} \tag{3.61}$$

Table 3.1 enumerates all possible events for the joint neighbor configuration of each ball with probabilities and resulting changes in variables. Figure 3.4 also shows an example evolution of bin labels. In the table, $\Delta Y_\ell(t) = Y_\ell(t+1) - Y_\ell(t)$ and the $o(1)$ terms assume $t < n(1-\epsilon)$. Note that all cases result in a deterministic change of bin counts except for case AC. In this case, the ball is matched with the A bin or the C bin each with probability $1/2$. There is one omission from the table: let $\mathcal{E}_m$ indicate the event that both edges of ball t connect to the same bin (this must be an A bin),

Table 3.1. Variable changes for bipartite GREEDY according to cases of neighboring bins.

Case	Probability	$\mathbb{E}[\Delta Y_A(t)\mid\cdot]$	$\mathbb{E}[\Delta Y_B(t)\mid\cdot]$	$\mathbb{E}[\Delta Y_C(t)\mid\cdot]$	$\mathbb{E}[\Delta Y_\mu(t)\mid\cdot]$
AA	$\frac{Y_A(t)^2}{(n-t+1)^2}+o(1)$	-2	1	1	1
AB	$\frac{Y_A(t)Y_B(t)}{(n-t+1)^2}+o(1)$	-1	0	0	1
AC	$\frac{Y_A(t)Y_C(t)}{(n-t+1)^2}+o(1)$	-1	$1/2$	$-1/2$	1
BB	$\frac{Y_B(t)^2}{4(n-t+1)^2}+o(1)$	0	-2	0	0
BC	$\frac{Y_B(t)Y_C(t)}{2(n-t+1)^2}+o(1)$	0	-1	-1	1
CC	$\frac{Y_C(t)^2}{4(n-t+1)^2}+o(1)$	0	0	-2	1

which can occur under the $\mathcal{P}(n,n,r)$ model and results in a multiple edge. This occurs with probability

$$\mathbb{P}(\mathcal{E}_m|\mathbf{Y}(t)) = \frac{Y_A(t)}{2(n-t+1)(n-t+1/2)}. \tag{3.62}$$

We will see shortly that this probability is $o(1)$, so we omit it in our analysis for the time being. Using the table, we can determine the expected changes in all variables conditioned on $\mathbf{Y}(t)$ as follows.

$$\mathbb{E}[\Delta Y_A(t)|\mathbf{Y}(t)] = \frac{-2Y_A(t)^2 - Y_A(t)Y_B(t) - Y_A(t)Y_C(t)}{(n-t+1)^2} + o(1), \tag{3.63}$$

$$\mathbb{E}[\Delta Y_B(t)|\mathbf{Y}(t)] = \frac{Y_A(t)^2 + \frac{1}{2}Y_A(t)Y_C(t) - \frac{1}{2}Y_B(t)^2 - \frac{1}{2}Y_B(t)Y_C(t)}{(n-t+1)^2} + o(1), \tag{3.64}$$

$$\mathbb{E}[\Delta Y_C(t)|\mathbf{Y}(t)] = \frac{Y_A(t)^2 - \frac{1}{2}Y_A(t)Y_C(t) - \frac{1}{2}Y_B(t)Y_C(t) - \frac{1}{2}Y_C(t)^2}{(n-t+1)^2} + o(1), \tag{3.65}$$

$$\mathbb{E}[\Delta Y_\mu(t)|\mathbf{Y}(t)] = 1 - \frac{Y_B(t)^2}{4(n-t+1)^2} + o(1). \tag{3.66}$$

For any label ℓ, let $z_\ell(x)$ be the normalized version of $Y_\ell(t)$ so that $z_\ell(x) = \frac{Y_\ell(nx)}{n}$ and thus $0 \le z_\ell(x) \le 1$ for $0 \le x \le 1$. As $n \to \infty$, the expected changes above yield the

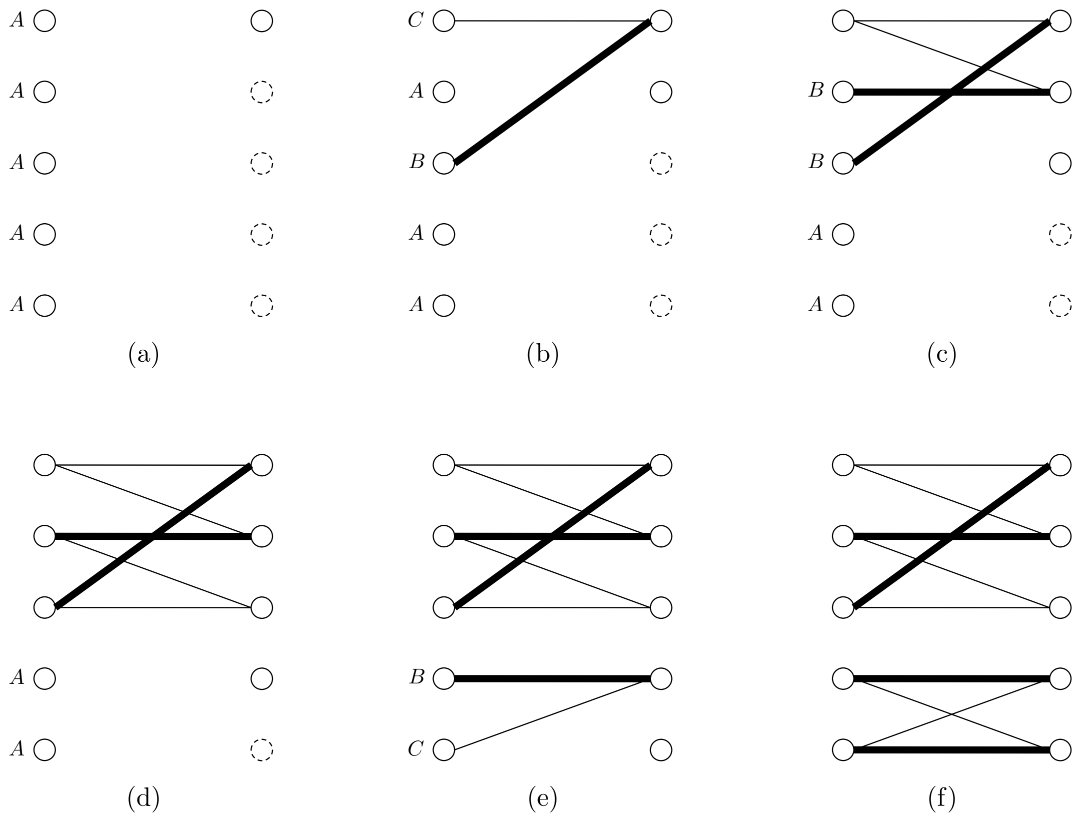


Figure 3.4. Example evolution of bin labels for a 2-regular bipartite graph where $n = 5$ and GREEDY is used. Bold edges indicate matched balls and bins. Part (a) shows the bin labels at $t = 1$ prior to edges for the first ball being revealed; part (b) shows bin labels at $t = 2$ prior to edges for the second ball being revealed, and indicates that the first ball was matched to the third bin; etc. Notice in part (d) that the third ball is unable to be matched because both of its neighboring bins are occupied. The final matching of size 4 is shown in part (f).

following deterministic system of differential equations

$$\frac{dz_A(x)}{dx} = \frac{-2z_A(x)^2 - z_A(x)z_B(x) - z_A(x)z_C(x)}{(1-x)^2}, \tag{3.67}$$

$$\frac{dz_B(x)}{dx} = \frac{z_A(x)^2 + \frac{1}{2}z_A(x)z_C(x) - \frac{1}{2}z_B(x)^2 - \frac{1}{2}z_B(x)z_C(x)}{(1-x)^2}, \tag{3.68}$$

$$\frac{dz_C(x)}{dx} = \frac{z_A(x)^2 - \frac{1}{2}z_A(x)z_C(x) - \frac{1}{2}z_B(x)z_C(x) - \frac{1}{2}z_C(x)^2}{(1-x)^2}, \tag{3.69}$$

$$\frac{dz_\mu(x)}{dx} = 1 - \frac{z_B(x)^2}{4(1-x)^2}. \tag{3.70}$$

Lemma 3.7. *The solution to the system of differential equations (3.67-3.70) with the initial conditions $z_A(0) = 1,\ z_B(0) = 0,\ z_C(0) = 0,\ z_\mu(0) = 0$ is*

$$\begin{aligned} z_A(x) &= (1-x)^2, \\ z_B(x) &= 2(1-x)(e^{x/2}-1), \\ z_C(x) &= 2(1-x)(1+x-e^{x/2}), \\ z_\mu(x) &= 4e^{x/2} - e^x - 3. \end{aligned} \tag{3.71}$$

Proof. We can simplify the normalized version of (3.67) by noting that

$$\begin{aligned} \frac{dz_A(x)}{dx} &= \frac{-2z_A(x)^2 - z_A(x)z_B(x) - z_A(x)z_C(x)}{(1-x)^2} \\ &= (2z_A(x) + z_B(x) + z_C(x))\left(\frac{-z_A(x)}{(1-x)^2}\right) \\ &= \frac{-z_A(x)}{(1-x)}. \end{aligned} \tag{3.72}$$

We can then put the equation in separable form

$$\frac{dz_A(x)}{z_A(x)} = \frac{-dx}{(1-x)}. \tag{3.73}$$

Integrating both sides and using the initial condition gives the expression for $z_A(x)$. Moving to (3.68), the derivative of $z_B(x)$, we can substitute $z_B(x) = 2(1-x) - 2z_A(x) - z_C(x)$ and $z_A(x) = (1-x)^2$ to obtain

$$\frac{dz_B(x)}{dx} = 1 - x + \frac{z_B(x)}{2}. \tag{3.74}$$

This can be solved with a conventional method (e.g. integrating factor) to give the expression for $z_B(x)$. The expression for the derivative of $z_C(x)$ simplifies to

$$\frac{dz_C(x)}{dx} = (1-x)^2 + \frac{(x+1)z_C(x)}{2(x-1)}, \tag{3.75}$$

which does not need to be solved since $z_C(x)$ follows easily from $z_A(x)$ and $z_B(x)$. Finally, the derivative of $z_\mu(x)$ simplifies to

$$\frac{dz_\mu(x)}{dx} = 2e^{x/2} - e^x. \tag{3.76}$$

This is easily integrated to give the result. ∎

The lemma suggests that the expected fraction of balls matched by the greedy algorithm is $4e^{1/2} - e^1 - 3 \approx 0.877$. We will use Wormald's theorem to formally justify this result. First we need the following lemma to satisfy the trend hypothesis.

Lemma 3.8. *For $x = t/n$, $z_\ell(x) = Y_\ell(t)/n$, and $t/n < 1 - \epsilon$, the expected changes (3.63-3.66) and differential equations (3.67-3.70) satisfy*

$$\mathbb{E}[\Delta Y_\ell(t)|\mathbf{Y}(t)] - \frac{dz_\ell(x)}{dx} = O(1/n). \tag{3.77}$$

for $\ell = A, B, C, \mu$.

Proof. We show that the event $\mathcal{E}_m$, where both edges of a ball connect to the same bin, occurs with probability $O(1/n)$. Let $t' = n - t$ so that $t' > n\epsilon$. Then,

$$\begin{aligned}\mathbb{P}(\mathcal{E}_m|\mathbf{Y}(t)) &= \frac{Y_A(t)}{2(n-t+1)(n-t+1/2)} \\ &\leq \frac{1}{2(t'+1/2)} \\ &< \frac{1}{2n\epsilon} = O(1/n),\end{aligned} \tag{3.78}$$

where we have used that $Y_A(t) \leq (n - t + 1)$ from (3.59).

For the remaining events, it is sufficient to bound the difference between the true and estimated probabilities for the event cases shown in Table 3.3. For the event AA, we have the following exact and estimated probabilities,

$$\begin{aligned}\mathbb{P}(AA) &= \frac{Y_A(t)(Y_A(t)-1)}{(n-t+1)(n-t+1/2)}, \\ \widehat{\mathbb{P}}(AA) &= \frac{(Y_A(t)/n)^2}{(1-t/n)^2},\end{aligned} \tag{3.79}$$

where we have made the conditioning on $\mathbf{Y}(t)$ implicit. Using $t' = n - t$, the difference is

$$\begin{aligned}\widehat{\mathbb{P}}(AA) - \mathbb{P}(AA) &= \frac{Y_A(t)^2}{(t')^2} - \frac{Y_A(t)^2}{(t'+1)(t'+1/2)} + \frac{Y_A(t)}{(t'+1)(t'+1/2)} \\ &< \frac{Y_A(t)^2}{(t')^2} - \frac{Y_A(t)^2}{(t'+1)^2} + \frac{Y_A(t)}{(t')^2} \\ &= \frac{(1+2t')Y_A(t)^2}{(t')^2(1+t')^2} + \frac{Y_A(t)}{(t')^2} \\ &< \frac{(1+2t')}{(1+t')^2} + \frac{1}{t'} \\ &< \frac{3}{t'} + \frac{1}{(t')^2} \\ &= O(1/t') = O(1/n).\end{aligned} \tag{3.80}$$

where we have used $Y_A(t) \leq t'$ and $1/t' < 1/(n\epsilon)$. Similar analysis of the other cases shows that the differences are $O(1/n)$. ■

Theorem 3.12. *Let $\mu_{\mathrm{G}}(n,n,2)$ denote the matching size obtained by the* Greedy *algorithm on the regular graph $\mathcal{G}(n,n,2)$. Then a.a.s.,*

$$\frac{\mu_{\mathrm{G}}(n,n,2)}{n} = 4e^{1/2} - e - 3 + o(1), \tag{3.81}$$

where $4e^{1/2} - e - 3 \approx 0.877$.

Proof. Using Wormald's theorem, we set $C_0 = 1$ and $\beta = 2$. To satisfy the Lipschitz condition, we choose the domain defined by $-\epsilon < z_\ell(x) < 1 + \epsilon$ for $\ell = A, B, C, \mu$ and $-\epsilon < x < 1 - \epsilon$ where $\epsilon > 0$. By Lemma 3.8, the trend hypothesis is satisfied with $\lambda_1 = O(1/n)$. Choosing $\lambda = n^{-1/4}$, for any $\epsilon > 0$ we have that $\lambda > \lambda_1$ for a sufficiently large n. We can make ϵ arbitrarily small so that the differential equations are valid arbitrarily close to the end of the process. The solution to the system of differential equations is given by Lemma 3.7. Using Theorem 3.1, we have with probability $1 - O(n^{1/4}e^{-n^{1/4}/8})$,

$$\mu_{\mathrm{G}} = n(4e^{1/2} - e - 3) + O(n^{3/4}). \tag{3.82}$$

■

Corollary 3.4. *The performance ratio of* Greedy *on $\mathcal{G}(n,n,r)$ for $r = 2$ satisfies*

$$\mathcal{R}_{\mathrm{G}}(2) = 4e^{1/2} - e - 3 \approx 0.877. \tag{3.83}$$

Degree-Greedy

When performing online matching in the presence of a regular graph, it is evident that the Greedy algorithm does not make optimal online choices. In particular, it does not favor assignments to bins that already have many edges exposed. This is clear with 2-regular graphs as we have just seen; recall that for Greedy, if a ball arrives and it has both an A neighbor and a C neighbor, the ball is assigned to each bin with probability 1/2. In these cases it is actually preferable to assign the ball to the C neighbor since this is the last time that the C neighbor can be matched, while there will be another opportunity for the A neighbor later. The instance in Figure 3.4 shows an example, where in part (b), matching the ball to the C bin would have allowed for a matching size of 5 instead of 4.

This motivates a variation of the Greedy algorithm that always chooses the unmatched neighboring bin with the largest degree. We refer to this as the Degree-Greedy algorithm. It is important to note that this algorithm makes sense for regular graphs, but for other graphs, it may perform worse than Greedy. In many realistic scenarios, certain bins are likely to be more popular than others, meaning they will have more edges. In these cases it is actually preferable to assign balls to bins that have a smaller degree, given the opportunity, because these bins are less likely to have future matching opportunities.

```
1: for $j = 1$ to $n$ (each ball) do
2:     When ball $j$ arrives, let $U(j)$ denote the set of unmatched neighboring bins
3:     if $|U(j)| > 0$ then
4:         Let $d(i)$ denote the current degree of bin $i$
5:         Match $j$ to a neighboring bin $i' \in \operatorname{argmax}_{i \in U(j)} d(i)$
6:     end if
7: end for
```

Algorithm 8. DEGREE-GREEDY (bipartite)

We analyze the DEGREE-GREEDY algorithm for 2-regular bipartite graphs in this section. For general r-regular bipartite graphs with $r \geq 2$, the analysis becomes significantly more difficult, but we expect the algorithm performance to be lower bounded by the 2-regular case. The DEGREE-GREEDY algorithm is shown in Algorithm 8. We use the same definitions and approach that we used to analyze GREEDY. To begin, we list the expected variable changes for the algorithm in Table 3.2; the table only differs from Table 3.1 for the AC case.

Table 3.2. Variable changes for bipartite DEGREE-GREEDY according to cases of neighboring bins.

Case	Probability	$\mathbb{E}[\Delta Y_A(t)\|\cdot]$	$\mathbb{E}[\Delta Y_B(t)\|\cdot]$	$\mathbb{E}[\Delta Y_C(t)\|\cdot]$	$\mathbb{E}[\Delta Y_\mu(t)\|\cdot]$
AA	$\frac{Y_A(t)^2}{(n-t+1)^2} + o(1)$	-2	1	1	1
AB	$\frac{Y_A(t)Y_B(t)}{(n-t+1)^2} + o(1)$	-1	0	0	1
AC	$\frac{Y_A(t)Y_C(t)}{(n-t+1)^2} + o(1)$	-1	0	0	1
BB	$\frac{Y_B(t)^2}{4(n-t+1)^2} + o(1)$	0	-2	0	0
BC	$\frac{Y_B(t)Y_C(t)}{2(n-t+1)^2} + o(1)$	0	-1	-1	1
CC	$\frac{Y_C(t)^2}{4(n-t+1)^2} + o(1)$	0	0	-2	1

From Table 3.1, we have the following expected changes.

$$\mathbb{E}[\Delta Y_A(t)|\mathbf{Y}(t)] = \frac{-2Y_A(t)^2 - Y_A(t)Y_B(t) - Y_A(t)Y_C(t)}{(n-t+1)^2} + o(1), \tag{3.84}$$

$$\mathbb{E}[\Delta Y_B(t)|\mathbf{Y}(t)] = \frac{Y_A(t)^2 - \frac{1}{2}Y_B(t)^2 - \frac{1}{2}Y_B(t)Y_C(t)}{(n-t+1)^2} + o(1), \tag{3.85}$$

$$\mathbb{E}[\Delta Y_C(t)|\mathbf{Y}(t)] = \frac{Y_A(t)^2 - \frac{1}{2}Y_B(t)Y_C(t) - \frac{1}{2}Y_C(t)^2}{(n-t+1)^2} + o(1), \tag{3.86}$$

$$\mathbb{E}[\Delta Y_\mu(t)|\mathbf{Y}(t)] = 1 - \frac{Y_B(t)^2}{4(n-t+1)^2} + o(1). \tag{3.87}$$

The corresponding differential equations are

$$\frac{dz_A(x)}{dx} = \frac{-2z_A(x)^2 - z_A(x)z_B(x) - z_A(x)z_C(x)}{(1-x)^2}, \tag{3.88}$$

$$\frac{dz_B(x)}{dx} = \frac{z_A(x)^2 - \frac{1}{2}z_B(x)^2 - \frac{1}{2}z_B(x)z_C(x)}{(1-x)^2}, \tag{3.89}$$

$$\frac{dz_C(x)}{dx} = \frac{z_A(x)^2 - \frac{1}{2}z_B(x)z_C(x) - \frac{1}{2}z_C(x)^2}{(1-x)^2}, \tag{3.90}$$

$$\frac{dz_\mu(x)}{dx} = 1 - \frac{z_B(x)^2}{4(1-x)^2}. \tag{3.91}$$

The solution to the equations are given by the following lemma; we omit its proof as it is similar to the proof of Lemma 3.7. The result of the lemma indicates that the fraction of matched vertices is $\frac{11}{12} \approx 0.917$, which is indeed larger than the fraction of 0.877 for the Greedy algorithm. The use of Wormald's theorem is the same as the application for Greedy and the result is stated below.

Lemma 3.9. *The solution to the system of differential equations (3.88-3.112) with the initial conditions* $z_A(0) = 1,\ z_B(0) = 0,\ z_C(0) = 0,\ z_\mu(0) = 0$ *is*

$$\begin{aligned} z_A(x) &= (1-x)^2, \\ z_B(x) &= x(1-x), \\ z_C(x) &= x(1-x), \\ z_\mu(x) &= x - \frac{x^3}{12}. \end{aligned} \tag{3.92}$$

Theorem 3.13. *Let* $\mu_{\mathrm{DG}}(n,n,2)$ *denote the matching size obtained by the* Degree-Greedy *algorithm on the regular graph* $\mathcal{G}(n,n,2)$. *Then a.a.s.,*

$$\frac{\mu_{\mathrm{DG}}(n,n,2)}{n} = \frac{11}{12} + o(1). \tag{3.93}$$

Proof. The proof of Theorem 3.12 applies, making use of Lemma 3.8 for the trend hypothesis, but with the solutions given in Lemma 3.9. ■

Corollary 3.5. *The performance ratio of* Degree-Greedy *on* $\mathcal{G}(n,n,r)$ *for* $r = 2$ *satisfies*

$$\mathcal{R}_{\mathrm{DG}}(2) = \frac{11}{12}. \tag{3.94}$$

We have specified that the Degree-Greedy algorithm is appropriate for random regular graphs, but how does it perform on the Erdős-Rényi random graph $\mathcal{G}(n,n,p)$? Observe that Degree-Greedy is a valid instance of an online algorithm ALG stated in Theorem 3.9, so it follows that its performance is characterized by Greedy. Note that the arguments in the proof of Theorem 3.9 are not valid for random regular graphs because the edges for each arriving ball are not drawn independently; they are a function of the existing graph. We have the following.

Corollary 3.6. *The* Degree-Greedy *and* Greedy *algorithms perform equivalently on* $\mathcal{G}(n,n,p)$.

Simulations and intuition suggest that the expected matching size of Greedy on $\mathcal{G}(n,n,r)$ is monotonically increasing in r for $r \geq 2$. This is difficult to prove immediately because the expected matching size is not monotonically increasing in the edge set E. That is, for two edge sets E_1 and E_2 such that $E_1 \subset E_2$, it is not necessarily true that the expected matching size given by Greedy on E_2 is greater than the expected size for E_1. This is obvious for 1-regular graphs, since Greedy returns a perfect matching on these graphs but has an expected matching size on random 2-regular graphs that is strictly less than n (as we have shown). However, it also holds for graphs with a minimum degree of 2, as demonstrated by the example in Figure 3.5.

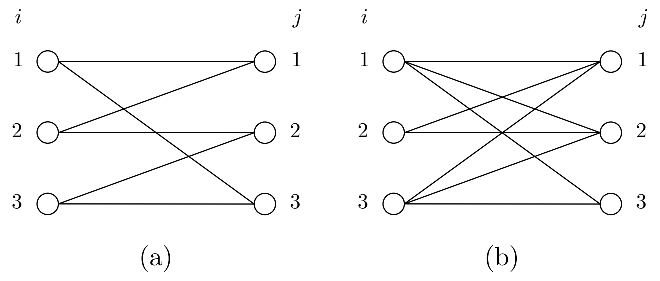


Figure 3.5. Example showing that the expected matching size of Greedy is not monotonically increasing in edge sets, even for graphs with a minimum degree of 2. The expected matching size for the graph in (a) is $11/4 = 2.75$. The graph in (b) is a superset of the graph in (a) and has a lower expected matching size of $8/3 \approx 2.67$.

The graph in Figure 3.5 (a) is a subgraph of the graph in Figure 3.5 (b). Note that the latter graph is not quite a 3-regular graph because of the missing edge $(2,3)$.

Both graphs are small enough that we can enumerate all possible progressions of the Greedy algorithm. For the graph in part (a), there are two progressions that occur with probability 1/4 – one giving a matching size of 3 and another giving matching size of 2 – and one progression that occurs with probability 1/2 that gives a matching of size 3. For the graph in part (b), there are six unique progressions of Greedy, each occurring with equal probability. Four of the progressions give a matching size of 3 and two give a matching size of 2, so the expected matching size for part (a) is 2.67. Thus the expected matching size for part (a) is 2.75, larger than that of part (b) despite having fewer edges. We have the following theorem.

Theorem 3.14. *The expected matching size given by* Greedy *for online bipartite matching is not monotonically increasing in the edge set E, even for graphs with a minimum degree of 2.*

Rollout

We now step away from the online scenario to make a connection with matching and rollout algorithms; we study the *offline* matching problem on $\mathcal{G}(n, n, 2)$. Offline matching on 2-regular bipartite graphs is trivial since the graph consists just of even cycles. However, we are interested in a scenario where it is possible to make theoretical statements about rollout algorithms. We consider a rollout algorithm that uses the Oblivious algorithm as a base policy. The algorithm is best interpreted as a *fortified rollout* algorithm because it picks an initial global solution and seeks to improve this solution locally at each step [27].

The algorithm makes three sequential passes over all balls. During the first pass, each ball proposes a match to a random neighboring bin. During the second pass, the proposed bin for each ball is changed if the bin has more than one proposal. The third pass matches each ball to its proposed bin, as long as the proposed bin is not matched. We present and analyze the algorithm on 2-regular random graphs, but the algorithm can be generalized to other bipartite graphs fairly easily.

This algorithm is referred to as Oblivious-Rollout and is shown in Algorithm 9. Phase 1, phase 2, and phase 3 correspond to the first, second, and third passes, respectively. The algorithm reduces to Oblivious if phase 2 is omitted. In the description, $K(i)$ indicates the proposal count for each bin i, which must be between 0 and 2, $N(j)$ indicates the set of neighbors for ball j, and the function $P(\cdot)$ is a mapping of balls to bins indicating proposals.

We still assume the configuration model in the analysis, but with a different order of edge exposure. Each ball reveals one neighboring edge during phase 1, and the other edge during phase 2 (as opposed to revealing both edges immediately, for example). The proposal during phase 1 is given exactly by first revealed edge. During phase 2, the second edge of each ball is revealed and it is determined whether or not the ball is colliding with another ball. A *colliding* ball means that its proposed bin has two proposals (the proposals remaining from the first phase). If a ball is colliding and its newly revealed edge connects to an unproposed bin, the proposal is switched to the new bin.

```
1: Initialize the proposal count for all bins: $K(i) \leftarrow 0,\ \forall i$.
2: for $j = 1$ to $n$ (phase 1) do
3:     Choose a random neighboring bin $i' \in N(j)$.
4:     Set the proposed bin for ball $j$ equal to $i'$: $P(j) \leftarrow i'$.
5:     Increase the proposal count for bin $i'$: $K(i') \leftarrow K(i') + 1$.
6: end for
7: for $j = 1$ to $n$ (phase 2) do
8:     Let $i'$ denote the current proposed bin for ball $j$: $i' \leftarrow P(j)$.
9:     Let $i''$ denote the remaining neighboring bin for ball $j$: $i'' \leftarrow N(j) \setminus \{i'\}$.
10:    if $K(i'') = 0$ and $K(i') = 2$ then
11:        Switch proposal for ball $j$ to bin $i''$: $P(j) \leftarrow i''$.
12:        Update proposal counts: $K(i') \leftarrow K(i') - 1, \quad K(i'') \leftarrow K(i'') + 1$
13:    end if
14: end for
15: for $j = 1$ to $n$ (phase 3) do
16:    Let $i$ denote the current proposed bin for ball $j$: $i \leftarrow P(j)$.
17:    if bin $i$ is unmatched then
18:        Match ball $j$ to bin $i$.
19:    end if
20: end for
```

Algorithm 9. Oblivious-Rollout

Since phase 1 represents the Oblivious algorithm, we know that with simple arguments (i.e. those used to prove Theorem 3.11), bins can be classified into three types: bins with zero proposal, bins with one proposal, and bins with two proposals. The expected number of bins with these descriptions is $n/4$, $n/2$, and $n/4$, respectively. With this in mind, we introduce some definitions to to analyze phase 2.

At step j during phase 2, the jth ball will have an existing proposed edge from phase 1 that connects to some bin. We enumerate three possibilities for this bin, where an *encountered* neighbor of a bin i refers to a ball $j' \in N(i)$ with index less than the current ball j; that is $j' < j$. The bin labels are

$R2:$	Two proposals such that neither of its neighbors have been encountered,
$S2:$	Two proposals such that exactly one of its neighbors has been encountered,
$R1:$	One proposal whose neighbor has not been encountered.

The second edge that is drawn during phase 2 must connect to some bin, and this bin must have one of the following three labels, prior to the new edge being drawn,

$M0:$	No edges,
$M1:$	One unproposed edge,
$Q1:$	One proposed edge.

Finally, we must keep track of the number of bins with two unproposed edges throughout the process; for this we use the label

$$M2: \quad \text{Two unproposed edges.}$$

An example of phase 2 of the algorithm with evolution of the bin labels is shown in Figure 3.6.

Let $Y_\ell(t)$ denote the number of bins at time t during phase 2 that have the label ℓ. Also define the vector $\mathbf{Y}(t) = (Y_{R2}(t), Y_{S2}(t), Y_{R1}(t), Y_{M1}(t), Y_{M0}(t), Y_{Q1}(t))$. Defining $t = 0$ as the end of phase 1 and the beginning of phase 2, we now make a formal statement about the end of phase 1.

Lemma 3.10. *With probability* $1 - 2e^{-\sqrt{n}/2}$, *at the end of phase 1 we have*

$$\begin{aligned} Y_{R2}(0) = Y_{M0}(0) &= \frac{n}{4} + n^{3/4}, && (3.95)\\ Y_{R1}(0) = Y_{Q1}(0) &= \frac{n}{2} + n^{3/4}. && (3.96)\end{aligned}$$

Proof. The mean values follow simply by linearity of expectation. For example, the probability that a bin has two proposals ($R2$) is the probability that its two neighbors both revealed their edges in phase 1, which is equal to $(1/2)^2$. The concentration follows by using Lemma 3.1 with $\alpha = n^{3/4}$ via the edge exposure (Doob) martingale. ■

Moving to phase 2, we study the expected change in the variables conditioned on $\mathbf{Y}(t)$. We again assume $t < (1-\epsilon)n$ for some $\epsilon > 0$ with regards for Lipschitz continuity. For each ball, two items are sampled. From phase 1, the ball has a neighboring bin with label $\ell_1 \in \{R2, S2, R1\}$. When drawing the second edge for the ball, a bin label $\ell_2 \in \{M0, M1, Q1\}$ is sampled. It is during phase 2 that multiple edges can occur under the configuration model; the probability of multiple edges is negligible and we address this in Lemma 3.12. The total number of existing edges to encounter and new edges to draw decreases by one for each ball, so we have for all t,

$$\begin{aligned} 2Y_{R2}(t) + Y_{R1}(t) + Y_{S2}(t) &= n - t + 1, && (3.97)\\ 2Y_{M0}(t) + Y_{M1}(t) + Y_{Q1}(t) &= n - t + 1. && (3.98)\end{aligned}$$

To calculate the expected change in the variables at each step, we begin by noting that in all cases where a $R2$ label is encountered, the number of $R2$ labeled bins decreases by one. If an $R2$ label is not encountered, the number of such bins remains constant. This gives

$$\mathbb{E}[\Delta Y_{R2}(t)|\mathbf{Y}(t)] = \frac{-2Y_{R2}(t)}{2Y_{R2}(t) + Y_{R1}(t) + Y_{S2}(t)} = \frac{-2Y_{R2}(t)}{n - t + 1}. \tag{3.99}$$

Now consider the label $S2$. To gain an $S2$ bin, we must have a case where $\ell_1(t) = R2$, meaning there is a colliding ball, and it must not be possible to switch the proposal to the alternative bin. This only occurs if $\ell_2(t) = Q1$. On the other hand, an $S2$ bin is lost whenever one is encountered. Together,

$$\begin{aligned}\mathbb{E}[\Delta Y_{S2}(t)|\mathbf{Y}(t)] &= \frac{2Y_{R2}(t)Y_{Q1}(t)}{(2Y_{R2}(t)+Y_{R1}(t)+Y_{S2}(t))(2Y_{M0}(t)+Y_{M1}(t)+Y_{Q1}(t))} \\ &\quad -\frac{Y_{S2}(t)}{2Y_{R2}(t)+Y_{R1}(t)+Y_{S2}(t)} \\ &= \frac{2Y_{R2}(t)Y_{Q1}(t)}{(n-t+1)^2} - \frac{Y_{S2}(t)}{n-t+1}. \end{aligned} \tag{3.100}$$

The label count for $R1$ is partly the complement of the $S2$ case. An additional $R2$ bin is generated whenever a collision is encountered and it is possible to switch the proposal to the newly drawn bin. This requires $\ell_1(t) = R2(t)$ and we must have $\ell_2(t) = M1$ or $\ell_2(t) = M2$. An $R1$ bin is lost whenever it is encountered.

$$\mathbb{E}[\Delta Y_{R1}(t)|\mathbf{Y}(t)] = \frac{2Y_{R2}(t)(2Y_{M0}(t)+Y_{M1}(t))}{(n-t+1)^2} - \frac{Y_{R1}(t)}{n-t+1}. \tag{3.101}$$

An $M1$ bin is gained whenever $\ell_1(t) = R1$ and $\ell_2(t) = M0$. One is lost whenever it is encountered, so

$$\mathbb{E}[\Delta Y_{M1}(t)|\mathbf{Y}(t)] = \frac{2Y_{R1}(t)Y_{M0}(t)}{(n-t+1)^2} - \frac{Y_{M1}(t)}{n-t+1}. \tag{3.102}$$

The number of $M0$ bins only decreases, and does so by one whenever one is encountered.

$$\mathbb{E}[\Delta Y_{M0}(t)|\mathbf{Y}(t)] = \frac{-2Y_{M0}(t)}{n-t+1}. \tag{3.103}$$

A $Q1$ bin is gained whenever $\ell_2(t) = M0$ and the t^{th} ball is colliding, and lost whenever one is encountered.

$$\mathbb{E}[\Delta Y_{Q1}(t)|\mathbf{Y}(t)] = \frac{(2Y_{R2}(t)+Y_{S2}(t))(2Y_{M0}(t))}{(n-t+1)^2} - \frac{Y_{Q1}(t)}{n-t+1}. \tag{3.104}$$

Finally, it is easy to see that

$$\mathbb{E}[\Delta Y_{M2}(t)|\mathbf{Y}(t)] = \frac{Y_{R1}(t)Y_{M1}(t)}{(n-t+1)^2}. \tag{3.105}$$

Using the same normalization procedure as before, the system of differential equations is

$$\frac{dz_{R2}(x)}{dx} = \frac{-2z_{R2}(x)}{1-x}, \tag{3.106}$$
$$\frac{dz_{S2}(x)}{dx} = \frac{2z_{R2}(x)z_{Q1}(x)}{(1-x)^2} - \frac{z_{S2}(x)}{1-x}, \tag{3.107}$$
$$\frac{dz_{R1}(x)}{dx} = \frac{2z_{R2}(x)(2z_{M0}(x)+z_{M1}(x))}{(1-x)^2} - \frac{z_{R1}(x)}{1-x}, \tag{3.108}$$
$$\frac{dz_{M0}(x)}{dx} = \frac{-2z_{M0}(x)}{1-x}, \tag{3.109}$$
$$\frac{dz_{M1}(x)}{dx} = \frac{2z_{R1}(x)z_{M0}(x)}{(1-x)^2} - \frac{z_{M1}(x)}{1-x}, \tag{3.110}$$
$$\frac{dz_{Q1}(x)}{dx} = \frac{(2z_{R2}(x)+z_{S2}(x))(2z_{M0}(x))}{(1-x)^2} - \frac{z_{Q1}(x)}{1-x}, \tag{3.111}$$
$$\frac{dz_{M2}(x)}{dx} = \frac{z_{R1}(x)z_{M1}(x)}{(1-x)^2}. \tag{3.112}$$

Lemma 3.11. *The solution to the system of differential equations (3.106-3.112) with the initial conditions $z_{R2}(0) = 1/4$, $z_{S2}(0) = 0$, $z_{R1}(0) = 1/2$, $z_{M1}(0) = 0$, $z_{M0}(0) = 1/4$, $z_{Q1}(0) = 1/2$, $z_{M2}(0) = 0$ is*

$$z_{R2}(x) = \frac{(1-x)^2}{4}, \tag{3.113}$$
$$z_{S2}(x) = \frac{1}{2}(e^{-x/2}+x-1)(1-x), \tag{3.114}$$
$$z_{R1}(x) = \left(1-\frac{e^{-x/2}}{2}\right)(1-x), \tag{3.115}$$
$$z_{M1}(x) = \frac{1}{2}(e^{-x/2}+x-1)(1-x), \tag{3.116}$$
$$z_{M0}(x) = \frac{(1-x)^2}{4}, \tag{3.117}$$
$$z_{Q1}(x) = \left(1-\frac{e^{-x/2}}{2}\right)(1-x), \tag{3.118}$$
$$z_{M2}(x) = \frac{e^{-x}}{4}\left(e^{x/2}x - e^{x/2}+1\right)^2. \tag{3.119}$$

Proof. Based on the similarities of the differential equations and the starting conditions, we guess the following identities:

$$z_{R2}(x) = z_{M0}(x), \tag{3.120}$$
$$z_{S2}(x) = z_{M1}(x), \tag{3.121}$$
$$z_{R1}(x) = z_{Q1}(x). \tag{3.122}$$

We can immediately verify the first identity since the differential equation for $z_{R2}(x)$ is a function only of itself and x; the same can be said for $z_{M0}(x)$. Integration gives

$$z_{R2}(x) = z_{M0}(x) = \frac{(1-x)^2}{4}. \tag{3.123}$$

Next we substitute the guessed identities into the differential equation for $z_{R1}(x)$:

$$\begin{aligned}
\frac{dz_{R1}(x)}{dx} &= \frac{2z_{R2}(x)(2z_{M0}(x)+z_{M1}(x))}{(1-x)^2} - \frac{z_{R1}(x)}{1-x} \\
&= \frac{2z_{R2}(x)(1-x-z_{Q1}(x))}{(1-x)^2} - \frac{z_{R1}(x)}{1-x} \\
&= \frac{2z_{R2}(x)(1-x-z_{R1}(x))}{(1-x)^2} - \frac{z_{R1}(x)}{1-x} \\
&= \frac{1}{2}(1-x) - z_{R1}(x)\left(\frac{1}{2} + \frac{1}{1-x}\right), \qquad (3.124)
\end{aligned}$$

where we have used $2z_{M0}(x) + z_{M1}(x) + z_{Q1}(x) = 1 - x$ in the second equality and the solution for $z_{R2}(x)$ in the last equality. Thus

$$z_{R1}(x) = \left(1 - \frac{e^{-x/2}}{2}\right)(1-x). \tag{3.125}$$

Now moving to the differential equation for $z_{S2}(x)$,

$$\begin{aligned}
\frac{dz_{S2}(x)}{dx} &= \frac{2z_{R2}(x)z_{Q1}(x)}{(1-x)^2} - \frac{z_{S2}(x)}{1-x} \qquad (3.126) \\
&= \frac{2z_{R2}(x)z_{R1}(x)}{(1-x)^2} - \frac{z_{S2}(x)}{1-x} \qquad (3.127) \\
&= \left(1 - \frac{e^{-x/2}}{2}\right)\frac{(1-x)}{2} - \frac{z_{S2}(x)}{1-x}, \qquad (3.128)
\end{aligned}$$

where the first inequality follows from the guess $z_{R1}(x) = z_{Q1}(x)$. The solution is

$$z_{S2}(x) = \frac{1}{2}(e^{-x/2} + x - 1)(1-x). \tag{3.129}$$

Now to verify that the guessed identities are correct for $z_{Q1}(x)$, we have

$$\frac{d}{dx}\left(\left(1 - \frac{e^{-x/2}}{2}\right)(1-x)\right) = \frac{(3-x)e^{-x/2}}{4} - 1. \tag{3.130}$$

Substituting the derived solutions in (3.111),

$$\begin{aligned}\frac{(2z_{R2}(x)+z_{S2}(x))(2z_{M0}(x))}{(1-x)^2}-\frac{z_{Q1}(x)}{1-x} &= \frac{(1-x)^2}{4}+\left(e^{-x/2}+x-1\right)\frac{(1-x)}{4}\\ &\quad -\left(1-\frac{e^{-x/2}}{2}\right)\\ &= \frac{(3-x)e^{-x/2}}{4}-1. \end{aligned} \tag{3.131}$$

We apply the same reasoning for $z_{M1}(x)$,

$$\frac{d}{dx}\left(\left(e^{-x/2}+x-1\right)\frac{(1-x)}{2}\right)=\frac{(x-3)e^{-x/2}}{4}+(1-x). \tag{3.132}$$

Substituting in (3.110),

$$\begin{aligned}\frac{2z_{R1}(x)z_{M0}(x)}{(1-x)^2}-\frac{z_{M1}(x)}{1-x} &= \left(1-\frac{e^{-x/2}}{2}\right)\frac{(1-x)}{2}-\frac{(e^{-x/2}+x-1)}{2}\\ &= \frac{(x-3)e^{-x/2}}{4}+(1-x). \end{aligned} \tag{3.133}$$

This validates all of the solutions except for $z_{M2}(x)$. We reduce (3.112) to

$$\frac{dz_{M2}(x)}{dx} = \frac{e^{-x}}{4}\left(2e^{x/2}-1\right)\left(1-e^{x/2}(1-x)\right). \tag{3.134}$$

Direct integration then gives

$$z_{M2}(x)=\frac{e^{-x}}{4}\left(e^{x/2}x-e^{x/2}+1\right)^2. \tag{3.135}$$

■

To determine the final fraction of matched bins suggested by the differential equations, we simply calculate $1-z_{M2}(1)=1/(4e)\approx 0.908$. We state one more lemma for the trend hypothesis in Wormald's theorem, and then formally justify this result.

Lemma 3.12. *For $x=t/n$, $z_\ell(x)=Y_\ell(t)/n$, and $t/n<1-\epsilon$, the expected changes (3.99-3.105) and differential equations (3.106-3.112) satisfy*

$$\mathbb{E}[\Delta Y_\ell(t)|\mathbf{Y}(t)]-\frac{dz_\ell(x)}{dx}=O(1/n), \tag{3.136}$$

for $\ell=R2,S2,R1,M1,M0,Q1,M2$.

Proof. Let $\mathcal{E}_m$ denote the event that a multiple edge is encountered under the configuration model. At any point during phase 2 we have, using $n - t > n\epsilon$,

$$\mathbb{P}(\mathcal{E}_m | \mathbf{Y}(t)) = \frac{1}{n-t+1} < \frac{1}{n\epsilon} = O(1/n). \tag{3.137}$$

Now for the labels; for the $R2$ label, we have

$$\begin{aligned}\mathbb{E}[\Delta Y_{R2}(t) | \mathbf{Y}(t)] - \frac{dz_{R2}(x)}{dx} &= \frac{-2Y_{R2}(t)}{n-t+1} + \frac{2Y_{R2}(t)/n}{1-t/n} \\ &= \frac{2Y_{R2}(t)}{(n-t+1)(n-t)} \\ &\leq \frac{1}{n-t} \\ &< \frac{1}{n\epsilon},\end{aligned} \tag{3.138}$$

where in the first inequality, we have used $2Y_{R2}(t) \leq n - t + 1$, and in the second inequality, $t/n < 1 - \epsilon$. The analysis for the other labels follows similarly. ■

Theorem 3.15. *Let $\mu_{\mathrm{OR}}(n, n, 2)$ denote the matching size obtained by the* Oblivious-Rollout *algorithm on the regular graph $\mathcal{G}(n, n, 2)$. Then a.a.s.,*

$$\frac{\mu_{\mathrm{OR}}(n, n, 2)}{n} = 1 - \frac{1}{4e} + o(1), \tag{3.139}$$

where $1 - \frac{1}{4e} \approx 0.908$.

Proof. We employ Wormald's theorem using $C_0 = 1$ and $\beta = 1$. The Lipschitz condition is satisfied with the domain defined by $-\epsilon < z_\ell(x) < 1 + \epsilon$ for $\ell = R2, S2, \ldots, M2$ and $-\epsilon < x < 1 - \epsilon$ where $\epsilon > 0$. From Lemma 3.10, we have $z_{R2}(0) = z_{M2}(0) = 1/4 + o(1)$ and $z_{R1}(0) = z_{Q1}(0) = 1/2 + o(1)$. The trend hypothesis is satisfied with $\lambda_1 = O(1/n)$ by Lemma 3.12. Choosing $\lambda = n^{-1/4}$, for any $\epsilon > 0$ we have that $\lambda > \lambda_1$ for a sufficiently large n. We can make ϵ arbitrarily small so that the differential equations are valid arbitrarily close to the end of the process. With the solution to the differential equations given by Lemma 3.11, we have with probability $1 - O(n^{1/4} e^{-n^{1/4}})$,

$$\mu_{\mathrm{OR}} = n\left(1 - \frac{1}{4e}\right) + O(n^{3/4}). \tag{3.140}$$

■

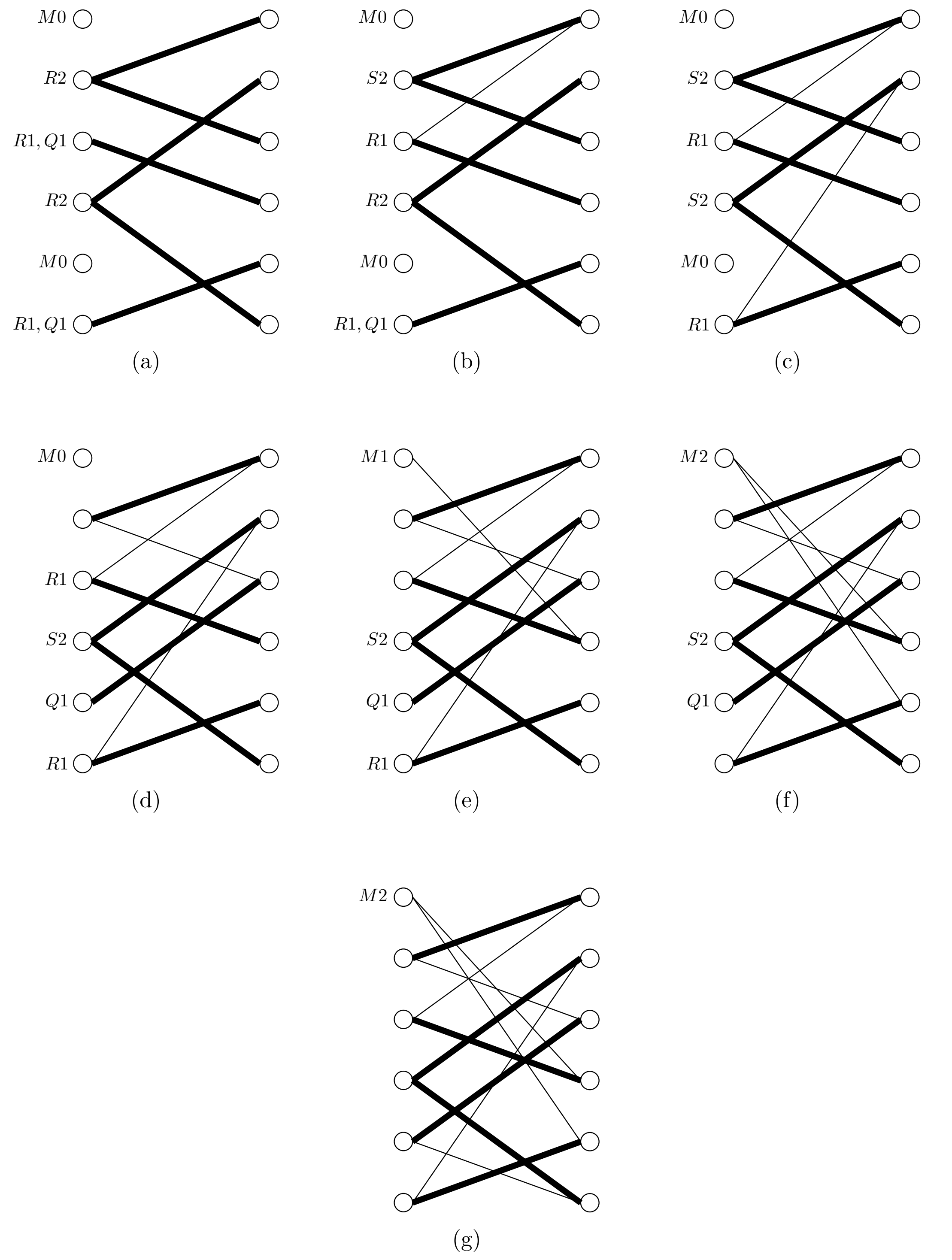


Figure 3.6. Example evolution of bin labels during phase 2 of the Oblivious-Rollout algorithm. Proposals from phase 1 are shown in part (a). In parts (b) - (g), the bin labels are shown as updated after the new edge for each ball has been exposed. The only proposal change occurs in (d), where the new edge and proposal change are shown in a single step. Phase 2 thus improves the matching size from 4 to 5.

3.4 Non-bipartite Graphs

In this section, we generalize the online greedy matching problem to non-bipartite graphs. This model starts with an empty graph and all vertices arrive sequentially. When each vertex arrives, only its neighboring edges connecting to existing vertices are revealed. We allow each vertex to be irrevocably matched when either it arrives or when it is matched to one of its arriving neighboring vertices. This is in contrast with the bipartite model, where vertices in one partition are given up front and vertices in the other partition arrive sequentially.

The intuitive greedy algorithm for this problem is essentially identical to the Greedy algorithm for the bipartite case: it matches each arriving vertex to a random unmatched neighbor if possible. An example progression of Greedy on a small graph is shown in Figure 3.7. Part (a) shows the full graph, including with a maximum matching of size 5 shown in bold. Parts (b)-(l) show the progression of the online algorithm, which obtains a matching of size 3.

The section is organized similar to the previous section. We start with the binomial random graph $\mathcal{G}(n, p)$ and then move to the random regular graph $\mathcal{G}(n, r)$.

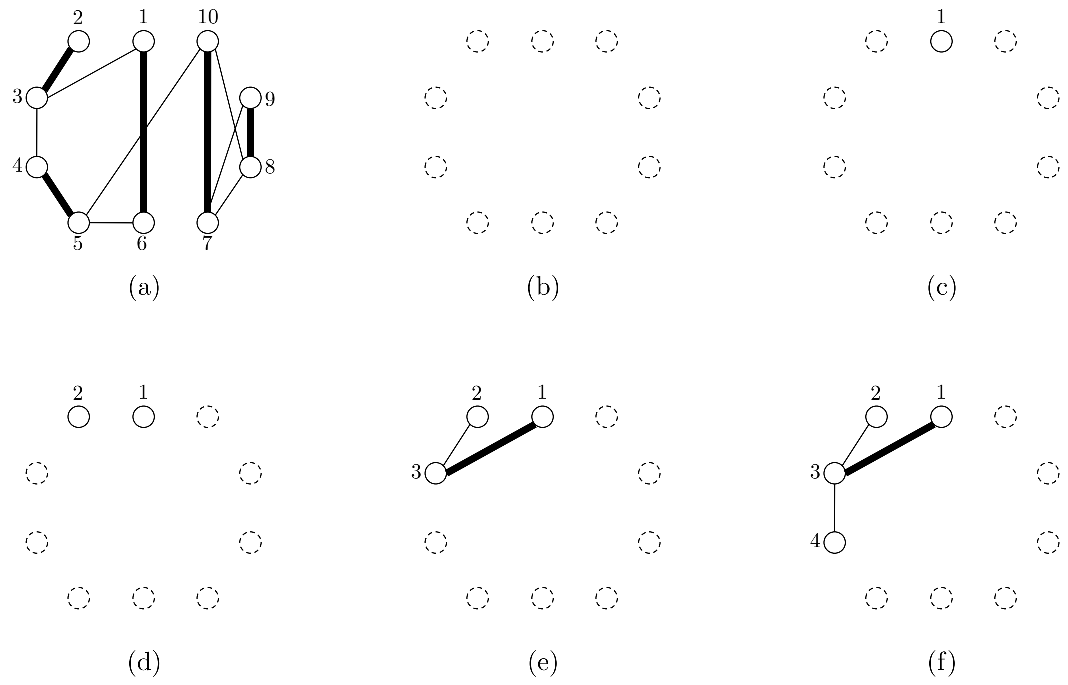


Figure 3.7. Instance and progression of online non-bipartite matching problem for $n = 10$ (continued on following page). The graph with a maximum matching shown in bold is shown in (a). The behavior of the online algorithm is shown in (b) - (l); the size of the matching obtained by the online algorithm is 3.

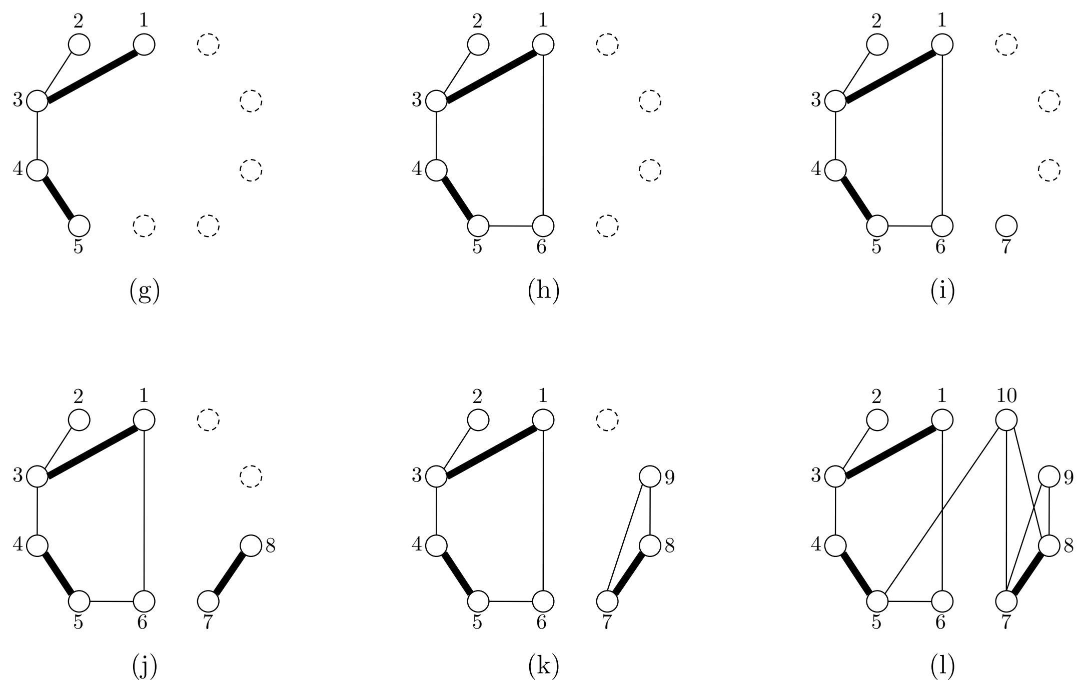


Figure 3.7 (continued)**.** Instance and progression of online non-bipartite matching problem for $n = 10$. The graph with a maximum matching shown in bold is shown in (a). The behavior of the online algorithm is shown in (b) - (l); the size of the matching obtained by the online algorithm is 3.

3.4.1 Binomial Graphs

We state here the Greedy algorithm for non-bipartite graphs and determine its performance on $\mathcal{G}(n, p)$ for valid functions $p = p(n)$.

The Greedy algorithm for non-bipartite graphs is shown in Algorithm 10; it simply matches each arriving vertex to a randomly selected unmatched neighboring vertex. As with the bipartite case, Wormald's theorem is only needed for the sparse regime $p = c/n$.

```
for $j = 1$ to $n$ (each vertex) do
    When vertex $j$ arrives, let $U(j)$ be the set of unmatched neighboring vertices
    if $|U(j)| > 0$ then
        Match $j$ to a random vertex $i' \in U(j)$
    end if
end for
```

Algorithm 10. Greedy (non-bipartite)

Theorem 3.16. *Let $\mu_{\mathrm{G}}(n, c/n)$ denote the matching size obtained by the* Greedy *algorithm on the graph $\mathcal{G}(n,p)$, where $p = c/n$ and $c > 0$ is a constant. Then a.a.s.,*

$$\frac{\mu_{\mathrm{G}}(n,c/n)}{n} = \frac{1}{2} - \frac{\log\left(2 - e^{-c}\right)}{2c} + o(1). \tag{3.141}$$

Proof. Let $Y(t)$ denote the number of matched vertices immediately before the t^{th} arrival. Conditioning on $Y(t)$, the t^{th} vertex cannot be matched only if edges connecting the vertex to the $t-1-Y(t)$ neighboring vertices are not present. The probability of a match is then

$$\begin{aligned} \mathbb{P}(\text{vertex } t \text{ is matched}|Y_t) &= 1-(1-p)^{t-1-Y(t)} \\ &= \frac{1}{2}\,\mathbb{E}[Y(t+1)-Y(t)|Y(t)]. \end{aligned} \tag{3.142}$$

Normalizing gives

$$\begin{aligned} \frac{\mathbb{E}[Z(x+1/n)-Z(x)|Z(x)]}{1/n} &= 2-2\left(1-\frac{c}{n}\right)^{n(x-1/n-Z(x))} && (3.143)\\ &= 2-2e^{-c(x-Z(x))}+o(1). && (3.144) \end{aligned}$$

The corresponding differential equation for $n\to\infty$ is

$$\frac{dz(x)}{dx} = 2-2e^{-c(x-z(x))}, \tag{3.145}$$

which has solution

$$z(x) = 2x - \frac{\log\left(-1+2e^{cx}\right)}{c}, \quad 0 \le x \le 1. \tag{3.146}$$

Note that since $Y(t)$ denotes the number of matched vertices, the matching size at time t is equal to $Y(t)/2$. To apply Wormald's theorem, let the domain D be defined by $-\epsilon < x < 1+\epsilon$ and $-\epsilon < z(x) < 1+\epsilon$, for $\epsilon > 0$. We have $C_0 = 1$ and $\beta = 2$. Let $\lambda_1 = O(1/n)$. The Lipschitz hypothesis is satisfied with a Lipschitz constant $L = 4ce^{c(1+2\epsilon)}$. Setting $\lambda = cn^{-1/4}$, we have that with probability $1-O(n^{1/4}e^{-c^3 n^{1/4}})$,

$$\mu_{\mathrm{G}}(n,c/n) = n\left(\frac{1}{2} - \frac{\log\left(2-e^{-c}\right)}{2c}\right) + O(n^{3/4}). \tag{3.147}$$

■

The first interesting observation about the expression in Theorem 3.16 is that it is exactly equal to the expression in Theorem 3.7 for bipartite graphs, differing by a factor of two since the bipartite graph has twice as many vertices. However, the differential equations for the two processes, specifically (3.50) and (3.146), are clearly not the same. Both processes thus arrive at the same matching size despite taking different paths. Even more remarkable, the matching size (3.141) given by GREEDY for non-bipartite graphs is equal to the matching size obtained by using MODIFIED-GREEDY on non-bipartite graphs, which was studied by Dyer et al. [56]. Recall that MODIFIED-GREEDY solves the *offline* matching problem, where at each step, it selects a random vertex and then chooses a random neighboring edge, and removes the corresponding matched vertices. Accordingly, MODIFIED-GREEDY performs the same procedure as GREEDY, but starts with and operates on the entire graph, and still obtains the same matching size that GREEDY does for the online problem!

Equipped with the above theorem and using some of the proof techniques that we used for bipartite graphs, the performance ratio for all valid functions $p = p(n)$ is stated as follows.

Theorem 3.17. *The performance ratio $\mathcal{R}_{\mathrm{G}}(p(n))$ of* GREEDY *on $\mathcal{G}(n,p)$, for all valid functions $p = p(n)$, satisfies*

$$\begin{aligned} \mathcal{R}_{\mathrm{G}}(p(n)) &= 1, & p(n) &= o(1/n), \\ \mathcal{R}_{\mathrm{G}}(p(n)) &= \frac{c - \log(2 - e^{-c})}{2c - (\gamma^* + \gamma_* + \gamma^*\gamma_*)}, & p(n) &= c/n, \\ \mathcal{R}_{\mathrm{G}}(p(n)) &= 1, & p(n) &= \omega(1/n). \end{aligned} \tag{3.148}$$

Proof. The performance ratio for $p(n) = o(1/n)$ follows from arguments in the proof of Lemma 3.3, and for $p(n) = \omega(1/n)$ from the arguments in the proof of Lemma 3.6. The sparse regime result follows from Theorem 3.16 and Theorem 3.3. ■

Corollary 3.7. *The performance ratio of* GREEDY *on $\mathcal{G}(n,p)$ for all valid functions $p = p(n)$ satisfies*

$$\mathcal{R}_{\mathrm{G}}(p(n)) \geq 0.837. \tag{3.149}$$

Proof. The proof is the same as the proof of Corollary 3.2. ■

3.4.2 Regular Graphs

We determine the matching sizes produced by both DEGREE-GREEDY and GREEDY for random 2-regular non-bipartite graphs. We only show the analysis for DEGREE-GREEDY, however, since the analysis for GREEDY is nearly identical. The DEGREE-GREEDY algorithm for regular graphs is shown in Algorithm 11.

We perform the analysis via the configuration model $\mathcal{P}(n, 2)$. We use $\mathcal{E}_m$ again to denote the occurrence of a multiple edge, and since we are now working with an non-bipartite graph, it is possible for self loops to occur. We use $\mathcal{E}_s$ to denote the occurrence of a self loop. Both of these events will happen with negligible probability, as we show in Lemma 3.13.

We use bin labels defined similarly to the bipartite case. Upon the arrival of a new vertex, there are two possibilities for each one of its edges. The edge either connects to an existing vertex or a vertex that will arrive later. In the latter case, we say that the

```
1: for $j = 1$ to $n$ do
2:     When vertex $j$ arrives, let $U(j)$ be the set of unmatched neighboring vertices
3:     if $|U(j)| > 0$ then
4:         For each bin $i$, let $d(i)$ denote its current degree
5:         Match $j$ to a neighboring bin $i' \in \operatorname{argmax}_{i \in U(j)} d(i)$
6:     end if
7: end for
```

Algorithm 11. Degree-Greedy (non-bipartite)

Table 3.3. Variable changes for non-bipartite Degree-Greedy according to cases of neighboring bins.

Case	$\mathbb{E}[\Delta Y_A(t)\|\cdot]$	$\mathbb{E}[\Delta Y_B(t)\|\cdot]$	$\mathbb{E}[\Delta Y_C(t)\|\cdot]$	$\mathbb{E}[\Delta Y_\mu(t)\|\cdot]$
DD	1	0	0	0
AD	-1	2	0	1
BD	0	-1	1	0
CD	0	1	-1	1
AA	-2	1	1	1
AB	-1	0	0	1
AC	-1	0	0	1
BB	0	-2	0	0
BC	0	-1	-1	1
CC	0	0	-2	1

edge is *deferred*.. There are three cases for vertices having at least one deferred edge. We label these vertices as

A :	the vertex has two deferred edges,
B :	the vertex has one deferred edge and is matched,
C :	the vertex has one deferred edge and is unmatched.

Altogether, each edge of an arriving vertex can connect to a vertex of type A, B, C, or it can be deferred. We denote a deferred edge by D.

As before, we define the vector $\mathbf{Y}(t) = (Y_A(t), Y_B(t), Y_C(t))$. We use t to index arriving vertices. For Lipschitz continuity, we make the restriction $t < n(1 - \epsilon)$ for some $\epsilon > 0$. Let $Y_\mu(t)$ denote the matching size at time t. The changes in variables conditioned on $\mathbf{Y}(t)$ are shown in Table 3.3. Note that the first vertex (i.e. $t = 1$) will always result in the DD case, which generates a type A vertex. Figure 3.8 shows an example progression of bin labels.

Letting $Y_D(t)$ denote the number of deferred edges prior to the arrival of the vertex

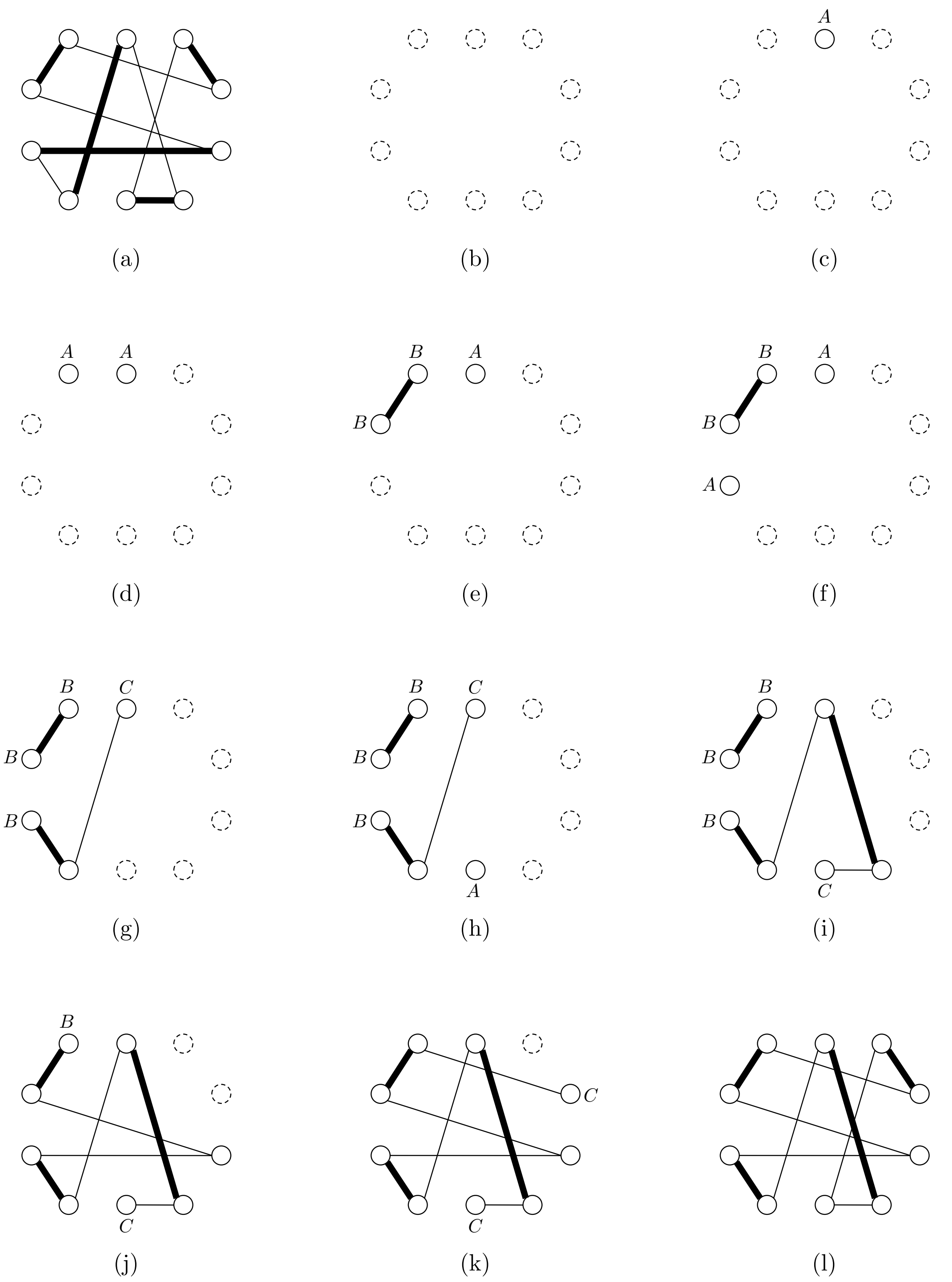


Figure 3.8. Example evolution of bin labels for a 2-regular non-bipartite graph where $n = 10$ and Degree-Greedy is used. The graph with a maximum matching shown in bold is shown in (a). The behavior of the online algorithm is shown in (b) - (l); it obtains a matching size of 4.

at time t, we have

$$2Y_A(t) + Y_B(t) + Y_C(t) + Y_D(t) = 2(n - t + 1). \tag{3.150}$$

The probabilities of the events in Table 3.3 are

$$\begin{aligned}
\mathbb{P}(DD|\mathbf{Y}(t)) &= \left(1 - \frac{(2Y_A(t) + Y_B(t) + Y_C(t))}{2(n-t+1)}\right)^2 + o(1), & (3.151)\\
\mathbb{P}(AD|\mathbf{Y}(t)) &= \frac{2Y_A(t)}{(n-t+1)}\left(1 - \frac{(2Y_A(t) + Y_B(t) + Y_C(t))}{2(n-t+1)}\right) + o(1), & (3.152)\\
\mathbb{P}(BD|\mathbf{Y}(t)) &= \frac{Y_B(t)}{(n-t+1)}\left(1 - \frac{(2Y_A(t) + Y_B(t) + Y_C(t))}{2(n-t+1)}\right) + o(1), & (3.153)\\
\mathbb{P}(CD|\mathbf{Y}(t)) &= \frac{Y_C(t)}{(n-t+1)}\left(1 - \frac{(2Y_A(t) + Y_B(t) + Y_C(t))}{2(n-t+1)}\right) + o(1), & (3.154)
\end{aligned}$$

$$\begin{aligned}
\mathbb{P}(AA|\mathbf{Y}(t)) &= \frac{Y_A(t)^2}{(n-t+1)^2} + o(1), & (3.155)\\
\mathbb{P}(AB|\mathbf{Y}(t)) &= \frac{Y_A(t)Y_B(t)}{(n-t+1)^2} + o(1), & (3.156)\\
\mathbb{P}(AC|\mathbf{Y}(t)) &= \frac{Y_A(t)Y_C(t)}{(n-t+1)^2} + o(1), & (3.157)\\
\mathbb{P}(BB|\mathbf{Y}(t)) &= \frac{Y_B(t)^2}{4(n-t+1)^2} + o(1), & (3.158)\\
\mathbb{P}(BC|\mathbf{Y}(t)) &= \frac{Y_B(t)Y_C(t)}{2(n-t+1)^2} + o(1), & (3.159)\\
\mathbb{P}(CC|\mathbf{Y}(t)) &= \frac{Y_C(t)^2}{4(n-t+1)^2} + o(1). & (3.160)
\end{aligned}$$

Combining the variable changes with the probabilities gives the following expected changes and differential equations.

$$\begin{aligned}
\mathbb{E}[\Delta Y_A(t)|\mathbf{Y}(t)] &= \frac{4Y_A(t)^2 + (Y_B(t)+Y_C(t)-2(n-t+1))^2}{4(n-t+1)^2} \\
&\quad + \frac{4Y_A(t)\,(Y_B(t)+Y_C(t)-4(n-t+1))}{4(n-t+1)^2} + o(1), \qquad (3.161) \\
\mathbb{E}[\Delta Y_B(t)|\mathbf{Y}(t)] &= -\frac{6Y_A(t)^2 + Y_B(t)(Y_C(t)+2(n-t+1))}{2(n-t+1)^2} \\
&\quad + \frac{Y_C(t)(Y_C(t)-2(n-t+1))}{2(n-t+1)^2} \\
&\quad - \frac{2Y_A(t)(Y_B(t)+3Y_C(t)-4(n-t+1))}{2(n-t+1)^2} + o(1), \qquad (3.162) \\
\mathbb{E}[\Delta Y_C(t)|\mathbf{Y}(t)] &= -\frac{-2Y_A(t)^2 + Y_B(t)^2 + 2Y_A(t)(Y_B(t)-Y_C(t))}{2(n-t+1)^2} \\
&\quad + \frac{2Y_C(t)(n-t+1)}{2(n-t+1)^2} \\
&\quad - \frac{Y_B(t)(Y_C(t)-2(n-t+1))}{2(n-t+1)^2} + o(1), \qquad (3.163) \\
\mathbb{E}[\Delta Y_\mu(t)|\mathbf{Y}(t)] &= -\frac{(2Y_A(t)+Y_C(t))(2Y_A(t)+Y_C(t)-4(n-t+1))}{4(n-t+1)^2} + o(1). \qquad (3.164)
\end{aligned}$$

$$\begin{aligned}
\frac{dz_A(x)}{dx} &= \frac{4z_A(x)^2 + (z_B(x)+z_C(x)-2(1-x))^2}{4(1-x)^2} \\
&\quad + \frac{4z_A(x)\,(z_B(x)+z_C(x)-4(1-x))}{4(1-x)^2}, \qquad (3.165) \\
\frac{dz_B(x)}{dx} &= -\frac{6z_A(x)^2 + z_B(x)(z_C(x)+2(1-x)) + z_C(x)(z_C(x)-2(1-x))}{2(1-x)^2} \\
&\quad - \frac{2z_A(x)(z_B(x)+3z_C(x)-4(1-x))}{2(1-x)^2}, \qquad (3.166) \\
\frac{dz_C(x)}{dx} &= -\frac{-2z_A(x)^2 + z_B(x)^2 + 2z_A(x)(z_B(x)-z_C(x)) + 2z_C(x)(1-x)}{2(1-x)^2} \\
&\quad - \frac{z_B(x)(z_C(x)-2(1-x))}{2(1-x)^2}, \qquad (3.167) \\
\frac{dz_\mu(x)}{dx} &= -\frac{(2z_A(x)+z_C(x))(2z_A(x)+z_C(x)-4(1-x))}{4(1-x)^2}. \qquad (3.168)
\end{aligned}$$

Lemma 3.13. *For $x = t/n$, $z_\ell(x) = Y_\ell(t)/n$, and $t/n < 1-\epsilon$, the expected changes (3.161 - 3.164) and differential equations (3.165 - 3.168) satisfy*

$$\mathbb{E}[\Delta Y_\ell(t)|\mathbf{Y}(t)] - \frac{dz_\ell(x)}{dx} = O(1/n), \tag{3.169}$$

for $\ell = A, B, C, \mu$.

Proof. We show that the events $\mathcal{E}_m$ of a multiple edge and $\mathcal{E}_s$ of a self loop occurring both have probabilities $O(1/n)$. For the former,

$$\mathbb{P}(\mathcal{E}_m|\mathbf{Y}(t)) = \frac{2Y_A(t)}{(2(n-t+1))^2} < \frac{1}{2(n-t)} = O(1/n), \tag{3.170}$$

which follows using $2Y_A(t) \leq 2(n-t+1)$ and $1/(n-t) < 1/(n\epsilon)$. The probability of a self loop at any time t is simply

$$\mathbb{P}(\mathcal{E}_s) = \frac{1}{2(n-t+1)} = O(1/n). \tag{3.171}$$

By bounding the differences between the true and estimated probabilities listed in equations (3.151 - 3.160), the remainder of the proof follows the same approach as the proof of Lemma 3.8. ■

Lemma 3.14. *The solution to the system of differential equations (3.165-3.168) with the initial conditions $z_A(0) = 1,\ z_B(0) = 0,\ z_C(0) = 0,\ z_\mu(0) = 0$ is*

$$\begin{aligned}
z_A(x) &= x(1-x)^2, \\
z_B(x) &= (1-x)\left(2x + x^2 - \frac{2(1-Q(1)-Q(x-1))}{\phi(x-1)}\right), \\
z_C(x) &= (1-x)\left(-2x + x^2 + \frac{2(1-Q(1)-Q(x-1))}{\phi(x-1)}\right), \\
z_\mu(x) &= \int_0^x -\frac{1}{4}\left(u^2 - \frac{2(1-Q(1)-Q(u-1))}{\phi(u-1)}\right) \\
&\quad \cdot \left(4 + u^2 - \frac{2(1-Q(1)-Q(u-1))}{\phi(u-1)}\right) du,
\end{aligned} \tag{3.172}$$

where

$$\phi(x) := \frac{e^{-\frac{1}{2}x^2}}{\sqrt{2\pi}}, \tag{3.173}$$

$$Q(x) := \frac{1}{\sqrt{2\pi}}\int_x^\infty e^{-\frac{u^2}{2}}\, du. \tag{3.174}$$

Proof. We define

$$\theta(x) := 2z_A(x) + z_B(x) + z_C(x). \tag{3.175}$$

We then have

$$\begin{aligned}
\frac{d\theta(x)}{dx} &= 2\frac{dz_A(x)}{dx} + \frac{dz_B(x)}{dx} + \frac{dz_C(x)}{dx} && (3.176)\\
&= \frac{2(1 - x - 2z_A(x) - z_B(x) - z_C(x))}{(1-x)} \\
&= \frac{2(1 - x - \theta(x))}{(1-x)}, && (3.177)
\end{aligned}$$

where in the second equality we have substituted (3.165 - 3.167). Then using $\theta(0) = 0$, we have the solution

$$\theta(x) = 2x(1-x). \tag{3.178}$$

The next step is to make the substitution

$$z_B(x) + z_C(x) = 2x(1-x) - 2z_A(x) \tag{3.179}$$

in (3.165) to obtain

$$\frac{dz_A(x)}{dx} = \frac{-2z_A(x) + (1-x)^3}{(1-x)}. \tag{3.180}$$

With $z_A(0) = 0$ we have

$$z_A(x) = x(1-x)^2. \tag{3.181}$$

Moving to $z_B(x)$, we may now make the substitution

$$z_C(x) = 2x(1-x) - 2x(1-x)^2 - z_B(x) \tag{3.182}$$

in (3.166), giving

$$\frac{dz_B(x)}{dx} = \frac{-z_B(x)\left(2(1-x) + x^2\right) - (1-x)^2 x(-4 + x + x^2)}{(1-x)}. \tag{3.183}$$

Thus,

$$z_B(x) = (1-x)\left(2x + x^2 - \frac{2(1 - Q(1) - Q(x-1))}{\phi(x-1)}\right). \tag{3.184}$$

This immediately gives via (3.182),

$$z_C(x) = (1-x)\left(-2x + x^2 + \frac{2(1 - Q(1) - Q(x-1))}{\phi(x-1)}\right). \tag{3.185}$$

We finally arrive at

$$\begin{aligned}\frac{dz_\mu(x)}{dx} &= -\frac{1}{4}\left(x^2 - \frac{2(1-Q(1)-Q(x-1))}{\phi(x-1)}\right) \\ &\quad \cdot \left(4 + x^2 - \frac{2(1-Q(1)-Q(x-1))}{\phi(x-1)}\right), \end{aligned} \tag{3.186}$$

$$\begin{aligned} z_\mu(x) &= \int_0^x -\frac{1}{4}\left(u^2 - \frac{2(1-Q(1)-Q(u-1))}{\phi(u-1)}\right) \\ &\quad \cdot \left(4 + u^2 - \frac{2(1-Q(1)-Q(u-1))}{\phi(u-1)}\right) du. \end{aligned} \tag{3.187}$$

We leave this in integral form since it does not have an analytical solution. ■

We now prove the result for Degree-Greedy on non-bipartite graphs. We also state the result for Greedy without proof since the analysis is similar.

Theorem 3.18. *Let $\mu_{\mathrm{DG}}(n,2)$ denote the matching size obtained by the* Degree-Greedy *algorithm on the regular graph $\mathcal{G}(n,2)$. Then a.a.s.,*

$$\frac{\mu_{\mathrm{DG}}(n,2)}{n} = 0.442121 + o(1). \tag{3.188}$$

Proof. We use Wormald's theorem with $C_0 = 1$ and $\beta = 2$. The Lipschitz condition is satisfied by the domain $-\epsilon < z_\ell(x) < 1+\epsilon$ for $\ell = A, B, C, \mu$ and $-\epsilon < x < 1-\epsilon$ where $\epsilon > 0$. By Lemma 3.13, the trend hypothesis is satisfied with $\lambda_1 = O(1/n)$. Choosing $\lambda = n^{-1/4}$, for any $\epsilon > 0$ we have that $\lambda > \lambda_1$ for a sufficiently large n. Making ϵ small makes the differential equations valid arbitrarily close to the end of the process. Thus by the numerical solution from Lemma 3.14, with probability $1 - O(n^{1/4}e^{-n^{1/4}/8})$,

$$\mu_{\mathrm{DG}} = n(0.442121) + O(n^{3/4}). \tag{3.189}$$

■

Corollary 3.8. *The performance ratio of* Degree-Greedy *on $\mathcal{G}(n,r)$ for $r = 2$ satisfies*

$$\mathcal{R}_{\mathrm{DG}}(2) \geq 0.884242. \tag{3.190}$$

Theorem 3.19. *Let $\mu_{\mathrm{G}}(n,2)$ denote the matching size obtained by the* Greedy *algorithm on the random graph $\mathcal{G}(n,2)$. Then a.a.s.,*

$$\frac{\mu_{\mathrm{G}}(n,2)}{n} = 0.434431 + o(1). \tag{3.191}$$

Corollary 3.9. *The performance ratio of* Greedy *on $\mathcal{G}(n,r)$ for $r = 2$ satisfies*

$$\mathcal{R}_{\mathrm{G}}(2) \geq 0.868862. \tag{3.192}$$

Finally, by the same observation we made in the bipartite case, we have the following.

Lemma 3.15. *The* Degree-Greedy *and* Greedy *algorithms perform equivalently on $\mathcal{G}(n,p)$.*

3.5 Discussion

Altogether, we have seen that greedy online matching algorithms perform well on random graphs. Our important results are summarized in Table 3.4. Performance improvements in comparison to worst-case bounds are expected, but the magnitudes of these improvements are surprising. Consider our results on the Erdős-Rényi graph $\mathcal{G}(n,n,p)$. The OBLIVIOUS algorithm, which is clearly inferior to RANKING, has a minimum performance ratio $1-1/e \approx 0.632$ that is equal to the best possible worst-case competitive ratio of any online matching algorithm (i.e. that of RANKING). Likewise, the lower bound of 0.837 on the performance ratio of GREEDY and RANKING is very high.

It is clear that the sparse regime $p(n) \sim 1/n$ is where a *phase change* occurs in matching performance for the graphs $\mathcal{G}(n,n,p)$ and $\mathcal{G}(n,p)$. This is where the performance ratio of the OBLIVIOUS algorithm transitions from 1 to $1-1/e$, and where the performance ratio of GREEDY passes through its global minimum of 0.837 (both for bipartite and non-bipartite cases). The sparse regime is in this sense the rich setting to study online matching on Erdős-Rényi graphs.

Table 3.4. Summary of lower bounds on performance ratios for random graph models and algorithms.

	$\mathcal{G}(n,n,p)$	$\mathcal{G}(n,n,2)$	$\mathcal{G}(n,p)$	$\mathcal{G}(n,2)$
OBLIVIOUS	$1-1/e \approx 0.632$	$3/4 = 0.75$		
GREEDY	0.837	$4e^{1/2} - e - 3 \approx 0.877$	0.837	0.869
RANKING	0.837			
DEGREE-GREEDY	0.837	$11/12 \approx 0.917$	0.837	0.884
OBLIVIOUS-ROLLOUT		$1-1/(4e) \approx 0.908$		

There are surprising similarities in matching properties for Erdős-Rényi graphs. Note again that the bounds/expressions for the asymptotic maximum matching sizes on $\mathcal{G}(n,n,c/n)$ and $\mathcal{G}(n,c/n)$ are equivalent (technically differing only by a factor of two since the former graph is twice as large). This is not necessarily expected since non-bipartite graphs contain odd cycles and bipartite graphs do not. Similar observations were made by Frieze [64], as the differential equations for part of the Karp-Sipser algorithm (specifically phase 2) on bipartite and non-bipartite graphs are equivalent.

Let us also recall the multiple settings that give rise to the expression in Theorem 3.7. As the theorem claims, this is the (asymptotic) matching size obtained by GREEDY on $\mathcal{G}(n,n,c/n)$. According to Theorem 3.9, this is also the matching size given by RANKING on $\mathcal{G}(n,n,c/n)$. For our online matching model, this is the expected matching size given by the GREEDY algorithm on $\mathcal{G}(n,c/n)$. Finally, this is the asymptotic matching size given by MODIFIED-GREEDY for the offline matching problem on $\mathcal{G}(n,c/n)$ [56]. That any of these results should be equal is remarkable, let alone all of them.

On random regular graphs, the behavior of GREEDY is even more optimistic than on binomial random graphs. The performance ratio for GREEDY on $\mathcal{G}(n, n, 2)$ is at least 0.877 and on $\mathcal{G}(n, 2)$ it is at least 0.869. Here the differences between bipartite and non-bipartite graphs are apparent – since odd cycles are present in the latter, the slightly smaller expected matching size makes sense. We conjecture that the performance ratios $\mathcal{R}_{\mathrm{G}}(r)$ are increasing in r for $r \geq 2$ in both the bipartite and non-bipartite cases, but it is an open problem to prove this.

The result for the OBLIVIOUS-ROLLOUT algorithm is important for rollout algorithms. While the analysis in Chapter 2 only considered the first iteration of rollout algorithms, the result for OBLIVIOUS-ROLLOUT characterizes the performance after running *every* iteration. The algorithm is not practical, but we have demonstrated that the differential equation method is useful for one rollout application, and in future research it may be effective for other problems.

Rollout algorithms aside, it is interesting to compare the performance of OBLIVIOUS-ROLLOUT with GREEDY. We have shown that taking a crude solution, as given by the OBLIVIOUS algorithm (with a performance ratio of 0.75), and refining it, which is what the OBLIVIOUS-ROLLOUT algorithm does (for a ratio of 0.908), gives stronger performance than simply using GREEDY (0.877). However, the DEGREE-GREEDY algorithm (0.917) is still the superior greedy algorithm.

A variety of possibilities exist for further research. Most obviously, there are missing entries in Table 3.4. We have not analyzed the OBLIVIOUS algorithm on non-bipartite graphs, but we expect it to be similar to the bipartite case. The same can be said about the RANKING algorithm on non-bipartite graphs. We have also not considered RANKING on random regular graphs. While it may be straightforward to study OBLIVIOUS-ROLLOUT on $\mathcal{G}(n, 2)$, its analysis for Erdős-Rényi graphs is likely to be difficult.

It would be helpful to know if the bound on the maximum matching size on $\mathcal{G}(n, n, p)$ is in fact tight for all $c > 0$ [40]. Recall that this is the bound given by Bollobás and Brightwell (Theorem 3.2), and it is known to be tight for $c \leq e$ [40]. We conjecture that it is in fact tight for all $c > 0$. A tight bound on this result would imply that the performance ratio bound in Theorem 3.8 is tight, including the lower bound of 0.837. In the unlikely event that it is not tight, the best lower bound on the performance ratio cannot be any greater than 0.845, where the denominator bound is tight for $c = e$. The most straightforward way to show tightness of this bound is to carry out the analysis of the Karp-Sipser algorithm on bipartite graphs. In fact, some of this analysis has already been completed in [64].

We have allowed the existence of isolated vertices in our consideration of Erdős-Rényi graphs, which is unlikely to be realistic for many applications. This could be resolved by imposing a restriction on the minimum degree of vertices, as was done in [65], for example. Unbalanced bipartite graphs (i.e. with more vertices on one side) are likely to be encountered in practice – our approach can be used in this situation, but less is known about expected maximum matching size here.

More generally, a valid criticism of random graph research in general is that $\mathcal{G}(n, n, p)$ and $\mathcal{G}(n, p)$ are not realistic models of naturally occurring graphs. The same can be argued for random regular graphs. For internet applications, there is extensive evidence that real graphs follow a power law distribution [10, 18, 19, 90, 94, 95]. It would be interesting to understand the behavior of greedy algorithms in these more realistic models, particularly *preferential attachment models*, where the graph is built such that edges connect to vertices with probabilities proportional to the degree of the vertices [115, 142].

3.6 Proof of Wormald's Theorem

We use the following one-sided version of the Azuma-Hoeffding inequality in the proof [17, 75, 148].

Lemma 3.16. *Let $X_0, X_1, \dots, X_t$ be a supermartingale where $X_0 = 0$ and $|X_i - X_{i-1}| \leq c_i$, $1 \leq i \leq t$, for constants c_i. Then for any $\alpha > 0$,*

$$\mathbb{P}(X_t \geq \alpha) \leq \exp\left(-\frac{\alpha^2}{2\sum_{i=1}^t c_i^2}\right). \tag{3.193}$$

Proof of Theorem 3.4 (Wormald's theorem). Our development is nearly identical to the proof in [148], but we include a few additional details. We will only prove part (b) as part (a) is a standard result. We prove the theorem in only one dimension (i.e. $a = 1$) and explain the generalization to more dimensions afterwards. Let

$$w := \lceil n\lambda/\beta \rceil. \tag{3.194}$$

We will assume that $\beta/\lambda \leq n^{1/3}$ (and thus $w \geq n^{2/3}$) so that the probability bound in the theorem conclusion is nontrivial. Recall that β is a constant, so w is a slowly growing function $w = O(n\lambda)$ that is restricted to integer values.

Fix some $t \geq 0$ and consider the sequence of random variables X_k^- for $k = 0, 1, \dots, w$, where

$$X_k^- := Y(t+k) - Y(t) - kf(t/n, Y(t)/n) - kg(n). \tag{3.195}$$

Here $g(n)$ is some deterministic function to be specified. Similarly define

$$X_k^+ := Y(t+k) - Y(t) - kf(t/n, Y(t)/n) + kg(n). \tag{3.196}$$

We will show that for an appropriately chosen $g(n)$, X_k^- is a supermartingale and X_k^+ is a submartingale. That is, for a correctly chosen $g(n)$, we will have

$$\begin{aligned} \mathbb{E}[X_{k+1}^- | H_{t+k}] &\leq X_k^-, \quad k = 0, 1, \dots, w, \\ \mathbb{E}[X_{k+1}^+ | H_{t+k}] &\geq X_k^+, \quad k = 0, 1, \dots, w. \end{aligned} \tag{3.197}$$

In the following analysis, we assume that the scaled variables $(t/n, Y(t)/n)$ are at a distance (in terms of (3.8)) at least $C\lambda$ from the boundary of D; this will be justified later. Conditioning on H_{t+k} for the sequence X_k^-, we have

$$\begin{aligned}
\mathbb{E}[X_{k+1}^- - X_k^- | H_{t+k}] &= \mathbb{E}[Y(t+k+1) - Y(t) - (k+1)f(t/n, Y(t)/n) \\
&\quad -(k+1)g(n) - Y(t+k) + Y(t) \\
&\quad +kf(t/n, Y(t)/n) + kg(n)|H_{t+k}] \\
&= \mathbb{E}[Y(t+k+1) - Y(t+k)|H_{t+k}] - f(t/n, Y(t)/n) - g(n) \\
&= \mathbb{E}[Y(t+k+1) - Y(t+k)|H_{t+k}] \\
&\quad -f((t+k)/n, Y(t+k)/n) + O(\beta k/n) - g(n) \\
&= O(\lambda_1) + O(\beta k/n) - g(n) \\
&= O(\lambda) - g(n). \qquad (3.198)
\end{aligned}$$

The first equality above follows by definition of X_k^- and the second equality follows from canceling terms and noting that the functions $f(t/n, Y(t)/n)$ and $g(n)$ are deterministic. The third equality uses both the boundedness hypothesis (so that $|Y(t+k) - Y(t)| \leq \beta k$) and the Lipschitz hypothesis. The fourth equality follows from the trend hypothesis and the fifth equality is satisfied since $\lambda_1 = O(\lambda)$ and $\beta w/n = O(\lambda)$.

Using the same analysis, it is easy to show

$$\mathbb{E}[X_{k+1}^+ - X_k^+ | H_{t+k}] \quad = \quad O(\lambda) + g(n). \qquad (3.199)$$

Thus, there exists some function $g(n) = O(\lambda)$ such that X_k^- is a supermartingale and X_k^+ is a submartingale. To bound the differences of the supermartingale we observe

$$\begin{aligned}
|X_{k+1}^- - X_k^-| &\leq |Y(t+k+1) - Y(t+k)| + |f(t/n, Y(t)/n)| + |g(n)| \\
&\leq \beta + O(1) \\
&\leq \kappa\beta, \qquad (3.200)
\end{aligned}$$

where $\kappa > 0$ is some constant. The second inequality uses the boundedness hypothesis, the Lipschitz hypothesis, and the fact that $\lambda = o(1)$; the third inequality follows since $\beta \geq 1$. This analysis again holds for the submartingale.

Now using Lemma 3.16 with the supermartigale X_k^-, we have

$$\mathbb{P}(Y(t+w) - Y(t) - wf(t/n, Y(t)/n) - wg(n) \geq \alpha' | H_t) \leq \exp\left(-\frac{(\alpha')^2}{2w\kappa^2\beta^2}\right), \qquad (3.201)$$

for any constant $\alpha' > 0$, or equivalently,

$$\mathbb{P}(Y(t+w) - Y(t) - wf(t/n, Y(t)/n) \geq wg(n) + \kappa\beta\sqrt{2w\alpha} | H_t) \leq e^{-\alpha}, \qquad (3.202)$$

for any $\alpha > 0$. Using the fact that the sequence $-X_k^+$ is a supermartingale as well, application of Lemma 3.16 gives

$$\mathbb{P}(-Y(t+w) + Y(t) + wf(t/n, Y(t)n) \geq wg(n) + \kappa\beta\sqrt{2w\alpha} | H_t) \leq e^{-\alpha}. \qquad (3.203)$$

Using a union bound, we have

$$\mathbb{P}(|Y(t+w) - Y(t) - wf(t/n, Y(t)/n)| \geq wg(n) + \kappa\beta\sqrt{2w\alpha}|H_t) \leq 2e^{-\alpha}. \quad (3.204)$$

We set

$$\alpha = \frac{n\lambda^3}{\beta^3}, \quad (3.205)$$

so that

$$\mathbb{P}(|Y(t+w) - Y(t) - wf(t/n, Y(t)/n)| \geq O(\lambda w)|H_t) \leq 2e^{-\alpha}, \quad (3.206)$$

where we have used $w = \lceil n\lambda/\beta \rceil$ and $g(n) = O(\lambda)$.

Next we define

$$k_i := iw, \quad i = 0, 1, \ldots, \lfloor \sigma n/w \rfloor. \quad (3.207)$$

We will show that for each i,

$$\mathbb{P}(|Y(k_j) - z(k_j/n)n| \geq B_j \text{ for some } j \leq i) = O(ie^{-\alpha}), \quad (3.208)$$

where

$$B_j := Bw\left(\lambda + \frac{w}{n}\right)\left(\left(1 + \frac{Bw}{n}\right)^j - 1\right)\frac{n}{Bw}, \quad (3.209)$$

and B is a constant that will be specified shortly. We will prove this using induction on i. To begin, the statement clearly holds for $z(0) = Y(0)/n$. For the inductive step, we use the decomposition

$$Y(k_{i+1}) - z(k_{i+1}/n)n = A_1 + A_2 + A_3 + A_4, \quad (3.210)$$

where

$$\begin{aligned} A_1 &= Y(k_i) - z(k_i/n)n, \\ A_2 &= Y(k_{i+1}) - Y(k_i) - wf(k_i/n, Y(k_i)/n), \\ A_3 &= wz'(k_i/n) + z(k_i/n)n - z(k_{i+1}/n)n, \\ A_4 &= wf(k_i/n, Y(k_i)/n) - wz'(k_i/n). \end{aligned} \quad (3.211)$$

From the inductive hypothesis, we have that with probability $1 - O(ie^{-\alpha})$,

$$|A_1| < B_i. \quad (3.212)$$

It follows from (3.206) that for a large enough constant B',

$$|A_2| < B'\lambda w \quad (3.213)$$

with probability $1 - O(e^{-\alpha})$. We will see that $B_i = O(\lambda n)$, so the inductive hypothesis indicates that the scaled variables are at least a distance $C\lambda$ from the boundary of the domain D; this justifies the assumption mentioned previously. Using the Lipschitz hypothesis and the fact that $z(\cdot)$ satisfies (a) in the theorem conclusion, we have

$$z(k_{i+1}/n) - z(k_i/n) \quad \leq \quad (w/n)(z'(k_i/n) + O(w/n)), \tag{3.214}$$

so that for a large enough constant B'',

$$|A_3| \leq \frac{B'' w^2}{n}. \tag{3.215}$$

Finally, from (3.212), and again using the Lipschitz hypothesis and the conclusion of part (a),

$$|A_4| \leq \frac{B''' B_i w}{n} \tag{3.216}$$

for a large enough constant B''' with probability $1 - O(ie^{-\alpha})$. Letting $B = \max(B', B'', B''')$, we have

$$\begin{aligned} |Y(k_{i+1} - z(k_{i+1}/n)n| &\leq |A_1| + |A_2| + |A_3| + |A_4| \\ &\leq B_i\left(1 + \frac{Bw}{n}\right) + B\left(w\lambda + \frac{w^2}{n}\right) \end{aligned} \tag{3.217}$$

with probability $1 - O((i+1)e^{-\alpha})$. To verify that (3.209) holds for $j = i+1$, we substitute B_i in the above expression:

$$\begin{aligned} & B_i\left(1 + \frac{Bw}{n}\right) + B\left(w\lambda + \frac{w^2}{n}\right) \\ = \ & Bw\left(\lambda + \frac{w}{n}\right)\left(\left(1 + \frac{Bw}{n}\right)^i - 1\right)\frac{n}{Bw}\left(1 + \frac{Bw}{n}\right) + B\left(w\lambda + \frac{w^2}{n}\right) \\ = \ & n\left(\lambda + \frac{w}{n}\right)\left(\left(1 + \frac{Bw}{n}\right)^{i+1} - \left(1 + \frac{Bw}{n}\right)\right) + Bw\left(\lambda + \frac{w}{n}\right) \\ = \ & \left(\lambda + \frac{w}{n}\right)\left(n\left(1 + \frac{Bw}{n}\right)^{i+1} - n - Bw + Bw\right) \\ = \ & n\left(\lambda + \frac{w}{n}\right)\left(\left(1 + \frac{Bw}{n}\right)^{i+1} - 1\right). \end{aligned} \tag{3.218}$$

Note that $(1 + Bw/n)^{i+1}$ is bounded by $e^B + O(1)$, so $B_i = O(\lambda n)$. Generalizing to the continuous time process, for any $t \leq \sigma n$, we set $i = \lfloor t/w \rfloor$. The difference in the variables $Y(\cdot)$ and $nz(\cdot)$ from time k_i to t is bounded by $O(\lambda n)$. This gives that with probability $1 - O((n/w)e^{-\alpha})$,

$$|Y(t) - z(t/n)n| = O(\lambda n). \tag{3.219}$$

This concludes the theorem for $a = 1$.

To generalize the proof for $a > 1$, the inductive statement (3.208) becomes

$$\mathbb{P}(|Y_l(k_j) - z_l(k_j/n)n| \geq B_j \text{ for some } j \leq i) = O(aie^{-\alpha}), \quad 1 \leq l \leq a. \tag{3.220}$$

This inductive statement at step i for each l depends on all variables $1 \leq l \leq a$ being well behaved at step $i-1$. The inclusion of the a term in the right-hand side of (3.220) follows from a union bound over the events that each variable is not well behaved at step $i - 1$. ■

Chapter 4

Uncertain Transition Probabilities in Markov Decision Processes

With the widespread use of Markov decision processes (MDPs), it is not difficult to find situations where model parameters are subject to uncertainty. This is especially true of transition probabilities. In operations research applications, historical demands must be used to predict future demands in inventory and resource allocation problems. More generally, estimated distributions are often derived via a limited number of samples of an exact distribution. Transition probabilities may even need to be estimated by an expert.

A variety of algorithms have been developed to optimize over transition probability uncertainty in a robust fashion. These approaches often use a maxmin criteria under various uncertainty descriptions [51, 62, 96, 117, 126, 132]. This has led to useful frameworks, such as the Markov decision process with imprecise probabilities (MDPIP), where transition probabilities are described by a set of linear inequalities, and the bounded-parameter Markov decision process (BMDP), where intervals are given for transition probabilities and rewards [71, 78]. However, scenarios where distribution estimates are used directly in conventional dynamic programming, rather than a robust algorithm, have received less attention. This chapter addresses such scenarios.

We provide a general loss bound for situations where an MDP policy is determined using estimated transition probabilities, but the system evolves according to different, true transition probabilities. The policy is computed using exact dynamic programming with estimated transition probabilities and stored in the form of a lookup table [122]. During the online phase of the algorithm, the MDP evolves according to its true underlying transition probabilities, and decisions are made using the lookup table. This decision process is referred to as the *approximate policy.* The optimal policy, on the other hand, uses knowledge of a lookup table calculated with true transition probabilities. The loss is defined as the difference between the expected total reward obtained by the optimal policy and the approximate policy.

The loss bound applies to finite horizon undiscounted, finite horizon discounted, and infinite horizon discounted scenarios. We show a tight example for the finite horizon undiscounted case that can be generalized to the other cases. We do not assume stationarity, so the transition probabilities, rewards, and states may vary among stages. Our proof approach is to analyze the growth of errors incurred by stepping backward in time while computing value functions. This requires bounding a multilinear program [54].

The organization of the chapter is as follows. Section 4.1 discusses related work and Section 4.2 provides background on Markov decision processes and dynamic programming. Before studying uncertain transition probabilities, we review some known results on uncertain value functions and approximate backward induction in Section 4.3 and Section 4.4. The goal of these sections is twofold. First, we are interested in building intuition for understanding the analysis of uncertain transition probabilities. Second, we wish to discuss uncertain value functions and approximate backward induction in a finite-horizon undiscounted setting since this is useful for general approximate dynamic programming applications. Section 4.5 proves the loss bound for uncertain transition probabilities and gives a tight example for the undiscounted finite horizon case. Note that throughout Sections 4.3-4.5, the definition of the approximate policy will change. A discussion is given in Section 4.6.

4.1 Related Work

The topic of uncertainty in MDP transition probabilities was initially addressed by Silver [132], who characterized uncertainty both by considering potential sets of transition probabilities and treating transition probabilities as random variables. Satia and Lave [126] employed game-theoretic and Bayesian formulations of uncertainty and created a variant of policy-iteration for maxmin and maxmax objectives. They also gave performance bounds under the Bayesian formulation for both objectives. White and Eldeib [78] looked at discounted, infinite horizon problems where transition probabilities for each state and action are described by a finite set of linear inequalities. This model is referred to as a Markov decision process with imprecise probabilities (MDPIP). They considered an adversarial mechanism that selects transition probabilities after an action is selected at each stage, and they used successive approximations to form a maxmin strategy.

Given et al. [71] introduced the bounded-parameter Markov decision process (BPMD), where intervals are given for various parameters of the MDP (e.g. transition probabilities, rewards). They defined analogues of traditional MDP features, such as interval value functions and optimal interval policies, and developed algorithms for using these features. They noted that their framework can be used to perform sensitivity analysis via computations. Nilim and Ghaoui [117] considered uncertainties in transition matrices that can be described via non-convex sets. They showed how such problems can be solved with a robust dynamic programming framework and demonstrated the effectiveness of using likelihood regions and entropy bounds to represent uncertainty. They also proved optimality of the robust control problems. Filho et al. [62] worked with the linear programming approach to dynamic programming and used multilinear and integer reformulation strategies to handle uncertainties in transition probabilities, where potential transition probabilities are given by credal sets. They looked at criteria of maxmin, maxmax, and E-admissibility for their algorithms. A variety of other approaches to optimizing under uncertain transition probabilities has been considered [3, 51, 96].

There has been some analysis of parameter sensitivity in dynamic programming. Hopp [77] analyzed the sensitivity of optimal policies under perturbations of problem parameters. Müller [113] studied variations in value functions resulting from transition probabilities that satisfy various stochastic order relations. There has also been recent work in applying sensitivity analysis for uncertain rewards in dynamic programming [138, 139].

Loss bounds for uncertain value functions in MDPs have been relatively well explored. Singh and Yee [134] proved an upper bound on losses incurred from a bounded error in value functions for the infinite-horizon discounted case. Similar bounds have been found for finite-horizon undiscounted problems [74, 88, 108]. Loss bounds in approximate policy iteration and approximate value iteration scenarios have been considered in [29, 60, 114].

4.2 Markov Decision Processes and Dynamic Programming

We give a brief introduction to Markov decision processes and dynamic programming. Thorough introductions to these topics can be found in [20, 27, 122, 123].

We define a T-stage non-stationary Markov decision process as follows[1]. We are given a set of *stages*, indexed

$$t = 0, \dots, T, \tag{4.1}$$

where $t = 0$ is the starting stage and $t = T$ is the terminal stage. For each stage, we are given a finite set of states

$$\mathcal{S}_t, \quad t = 0, \dots, T. \tag{4.2}$$

At stage t, we say that the process is in some *state* $S_t \in \mathcal{S}_t$. We make the restriction that the starting stage has a singleton set $\mathcal{S}_0 = \{S_0\}$, where S_0 is the *unique starting state*. For each state at non-terminal stages, there exists a finite set of *feasible decisions*

$$\mathcal{X}_t(S_t), \quad S_t \in \mathcal{S}_t, \quad t = 0, \dots, T-1. \tag{4.3}$$

When we are in some non-terminal state S_t, we must choose an *action* or *decision* $x_t \in \mathcal{X}_t(S_t)$. The action results in a reward as determined by the *reward function*

$$R_t : \mathcal{S}_t \times \mathcal{X}_t(\mathcal{S}_t) \to \mathbb{R}, \quad t = 0, \dots, T-1. \tag{4.4}$$

[1]There are actually $T+1$ stages in our definition since we include $t = 0$ as a stage, but the process consists of a total of T decisions.

We have slightly abused notation here to write $\mathcal{X}_t(\mathcal{S}_t) := \cup_{S_t \in \mathcal{S}_t} \mathcal{X}_t(S_t)$[2]. At the terminal stage $t = T$, we also have the terminal reward function

$$V_T : \mathcal{S}_T \to \mathbb{R}. \tag{4.5}$$

In addition to a reward, a selected action results in a random transition to a state at the following stage governed by a *transition probability function*

$$\mathbb{P}_t : \mathcal{S}_{t+1} \times \mathcal{S}_t \times \mathcal{X}_t(\mathcal{S}_t) \to [0,1], \quad t = 0, \dots, T-1. \tag{4.6}$$

We write $\mathbb{P}_t(S_{t+1}|S_t, x_t)$ to indicate the probability that we transition to state S_{t+1} given that we are in state S_t and select action x_t. We have

$$\sum_{S_{t+1} \in \mathcal{S}_{t+1}} \mathbb{P}_t(S_{t+1}|S_t, x_t) = 1, \quad x_t \in \mathcal{X}_t(S_t), \quad S_t \in \mathcal{S}_t, \quad t = 0, \dots, T-1. \tag{4.7}$$

In summary, the Markov decision process is defined by the collection $(T, (\mathcal{S}_t, \mathcal{X}_t(\mathcal{S}_t), R_t, \mathbb{P}_t)_{t=0}^{T-1}, V_T)$. If the set of states, feasible decisions, reward functions, and transition probabilities are all identical for all stages, the MDP is referred to as *stationary*. Otherwise, it is referred to as *non-stationary*.

An example Markov decision process for $T = 3$ is shown in Figure 4.1 with a directed graph structure. The white nodes indicate states, the black nodes indicate terminal states, and the gray nodes indicate decisions. (The gray nodes can also be interpreted as post-decision states [122].) Arcs exiting states point to feasible decisions for each state with arc weights indicating rewards. Arcs exiting decision nodes point to possible state transitions with arc weights indicating transition probabilities. The example has a total of ten states $\{W_0, W_1, \dots, W_9\}$ where

$$\begin{aligned} \mathcal{S}_0 &= \{W_0\}, \\ \mathcal{S}_1 &= \{W_1, W_2, W_3\}, \\ \mathcal{S}_2 &= \{W_4, W_5, W_6\}, \\ \mathcal{S}_3 &= \{W_7, W_8, W_9\}. \end{aligned} \tag{4.8}$$

There are a total of twelve decisions $\{A_1, A_2, \dots, A_{12}\}$ where

$$\begin{aligned} \mathcal{X}_0(W_0) &= \{A_1, A_2, A_3\}, \\ \mathcal{X}_1(W_1) &= \{A_4, A_5\}, \\ &\vdots \\ \mathcal{X}_2(W_6) &= \{A_{12}\}. \end{aligned} \tag{4.9}$$

Of course, only three decisions are made during any evolution of the process, and some states (such as W_6) only have one feasible decision to select.

[2]We will only consider arguments for the function R_t of the form (S_t, x_t) that are well defined, meaning $x_t \in \mathcal{X}_t(S_t)$.

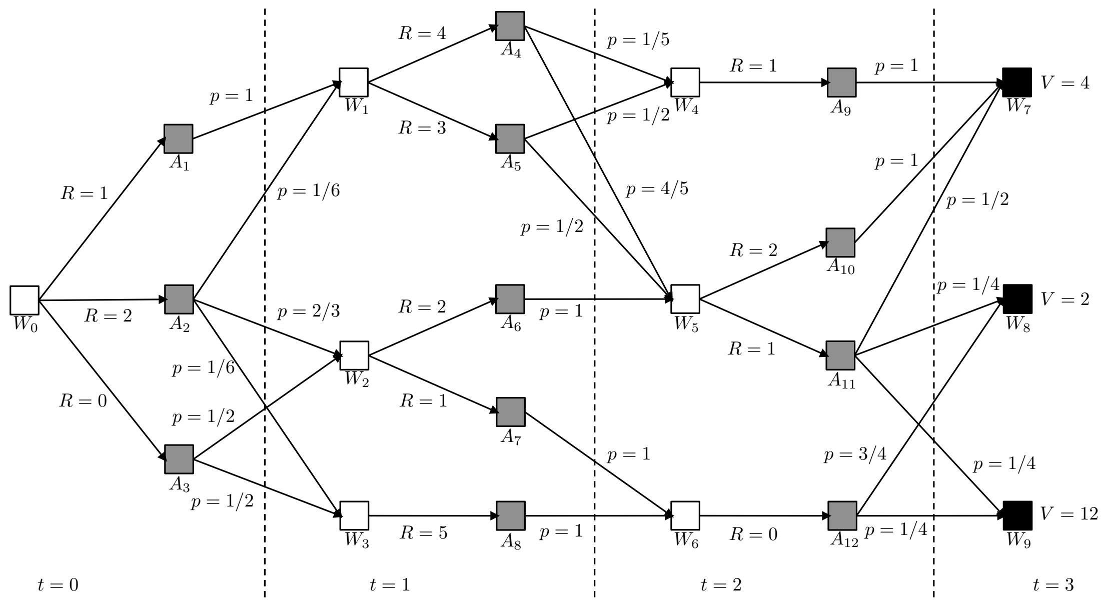


Figure 4.1. A T-stage Markov decision process for $T = 3$. White nodes are states, gray nodes are decisions, and black nodes are terminal states. Weights for arcs exiting states indicate rewards, and weights for arcs exiting decisions indicate transition probabilities. Terminal reward values are labeled for the three terminal states.

The goal in a Markov decision process is to determine a policy that maximizes some objective, usually total expected reward. We define a *decision rule* as a mapping of states to feasible actions at a given stage,

$$X_t : S_t \to \mathcal{X}_t(S_t), \quad t = 0, \dots, T-1. \tag{4.10}$$

A *policy* π is then defined as a series of decision rules

$$\pi = (X_0, X_1, \dots, X_{T-1}). \tag{4.11}$$

We use the notation $X_t^\pi(\cdot)$ to indicate the decision rule at stage t under policy π. In general, an action taken under a policy π is denoted by $x_t^\pi := X_t^\pi(S_t)$; however, we sometimes write the action simply as $x_t^\pi(S_t)$ or just x_t^π. Rewards are time discounted with a *discount factor* α satisfying $0 \le \alpha \le 1$, so that a reward R_t at stage t is worth $\alpha^t R_t$. The goal that we are interested in is maximizing total expected reward,

$$\max_{\pi \in \Pi} \mathbb{E}\left[\sum_{t=0}^{T-1} \alpha^t R_t(S_t, X_t^\pi(S_t)) + \alpha^T V_T(S_T)\right], \tag{4.12}$$

where Π is the set of all policies. The expectation is taken with respect to random transitions in the process as well as decisions made by the policy; for a detailed explanation, see [123]. The policy maximizing the above expression is referred to as the *optimal policy*.

The optimal policy can be found using dynamic programming, which we explain without proof. Dynamic programming uses *optimal value functions* for each stage,

$$V_t : \mathcal{S}_t \to \mathbb{R}, \quad t = 0, \ldots, T-1. \tag{4.13}$$

The value $V_t(S_t)$ indicates the expected value of a state assuming that optimal decisions are made in the future. Optimal value functions are determined recursively, starting at the last stage and moving backward, via the *backward induction equation*

$$\begin{aligned} V_t(S_t) \quad := \quad & \max_{x_t \in \mathcal{X}_t(S_t)} \left(R_t(S_t, x_t) + \alpha \sum_{\mathcal{S}_{t+1}} \mathbb{P}_t(S_{t+1}|S_t, x_t) V_{t+1}(S_{t+1}) \right), \\ & t = 0, \ldots, T-1. \end{aligned} \tag{4.14}$$

The notation $\sum_{\mathcal{S}_{t+1}}(\cdot)$ indicates $\sum_{S_{t+1} \in \mathcal{S}_{t+1}}(\cdot)$. To further simplify notation, we write $\mathbb{E}[V_{t+1}(S_{t+1})|S_t, x_t]$ for $\sum_{\mathcal{S}_{t+1}} \mathbb{P}_t(S_{t+1}|S_t, x_t) V_{t+1}(S_{t+1})$, giving

$$V_t(S_t) = \max_{x_t \in \mathcal{X}_t(S_t)} \left(R_t(S_t, x_t) + \alpha \mathbb{E}[V_{t+1}(S_{t+1})|S_t, x_t] \right). \tag{4.15}$$

We also sometimes omit $x_t \in \mathcal{X}_t(S_t)$ in the argument for the maximum function and write x_t.

The procedure of stepping backward in time and using (4.14) to determine values of all states is referred to *dynamic programming*, *value iteration*, and/or *backward induction*. This procedure is shown explicitly in Algorithm 12; we refer to it as the BACKWARD-INDUCTION algorithm. In general, the value functions are computed in advance before the actual evolution of the MDP.

```
Input: Markov decision process $(T, (\mathcal{S}_t, \mathcal{X}_t(\mathcal{S}_t), R_t, \mathbb{P}_t)_{t=0}^{T-1}, V_T)$, discount factor $\alpha$.
Output: Optimal value functions $(V_t)_{t=0}^{T-1}$.
1: for $t = T-1$ to 0 (each stage) do
2:     for $S_t \in \mathcal{S}_t$ (each state at stage $t$) do
3:         $V_t(S_t) \leftarrow \max_{x_t \in \mathcal{X}_t(S_t)} \left( R_t(S_t, x_t) + \alpha \sum_{\mathcal{S}_{t+1}} \mathbb{P}_t(S_{t+1}|S_t, x_t) V_{t+1}(S_{t+1}) \right)$
4:     end for
5: end for
```

Algorithm 12. BACKWARD-INDUCTION

For the example in Figure 4.1, assuming $\alpha = 1$, the BACKWARD-INDUCTION algorithm gives the state values

$$\begin{aligned} V_0(W_0) &= 11.2, & V_1(W_1) &= 10.2, & V_2(W_4) &= 5, \\ & & V_1(W_2) &= 8.5, & V_2(W_5) &= 6.5, \\ & & V_1(W_3) &= 9.5, & V_2(W_6) &= 4.5. \end{aligned} \tag{4.16}$$

The value of the unique starting state indicates the entire value of the MPD, which in this case is 11.2.

Using the optimal value functions determined by Algorithm 12, the *optimal policy* at each state S_t makes decisions $x_t^*(S_t)$ according to

$$\begin{aligned} x_t^*(S_t) \quad := \quad & \underset{x_t \in \mathcal{X}_t(S_t)}{\operatorname{argmax}} \left(R_t(S_t, x_t) + \alpha \sum_{\mathcal{S}_{t+1}} \mathbb{P}_t(S_{t+1}|S_t, x_t) V_{t+1}(S_{t+1}) \right), \\ & t = 0, \ldots, T-1. \end{aligned} \tag{4.17}$$

The optimal policy can be determined by evaluating this equation for all states. More practically, it can be determined online by evaluating (4.17) only for each state that is observed during the process. We can thus view the online optimal policy in algorithmic form, shown in Algorithm 13. Note that the sequence of decisions given by this algorithm is only optimal for the series of states encountered in a particular realization; this series occurs randomly.

```
Input: Markov decision process $(T, (\mathcal{S}_t, \mathcal{X}_t(\mathcal{S}_t), R_t, \mathbb{P}_t)_{t=0}^{T-1}, V_T)$, discount factor $\alpha$, value
    functions $(V_t)_{t=1}^{T-1}$.
Output: Sequence of optimal decisions $(x_t^*)_{t=0}^{T-1}$ for realized state sequence $(S_t)_{t=0}^{T-1}$.
1: for $t = 0$ to $T-1$ (each stage) do
2:     When state $S_t$ is revealed, choose action
           $x_t^*(S_t) \leftarrow \underset{x_t \in \mathcal{X}_t(S_t)}{\operatorname{argmax}} \left( R_t(S_t, x_t) + \alpha \sum_{\mathcal{S}_{t+1}} \mathbb{P}_t(S_{t+1}|S_t, x_t) V_{t+1}(S_{t+1}) \right).$
3: end for
```

Algorithm 13. OPTIMAL-POLICY

For the example from Figure 4.1, an example evolution under the optimal policy is shown in Figure 4.2 (again assuming $\alpha = 1$). The realized path is shown in bold; it is determined by decisions made under the optimal policy and random outcomes for selected actions. The alternating sequence of states and actions for the evolution shown is $(W_0, A_1, W_1, A_4, W_5, A_{11}, W_8)$, which gives a total value of 8.

We now consider any general policy π that makes decisions $x_t^\pi(\cdot)$. The *policy value* $V_t^\pi(S_t)$ of a state S_t is the expected value given by starting in state S_t and following the policy π through stage T. It is determined by the recursive equation

$$\begin{aligned} V_t^\pi(S_t) \quad := \quad & R_t(S_t, x_t^\pi(S_t)) + \alpha \mathbb{E}[V_{t+1}^\pi(S_{t+1})|S_t, x_t^\pi(S_t)], \\ & t = 0, \ldots, T-1. \end{aligned} \tag{4.18}$$

For all policies at the terminal stage,

$$V_T^\pi(S_T) = V_T(S_T), \quad S_T \in \mathcal{S}_T. \tag{4.19}$$

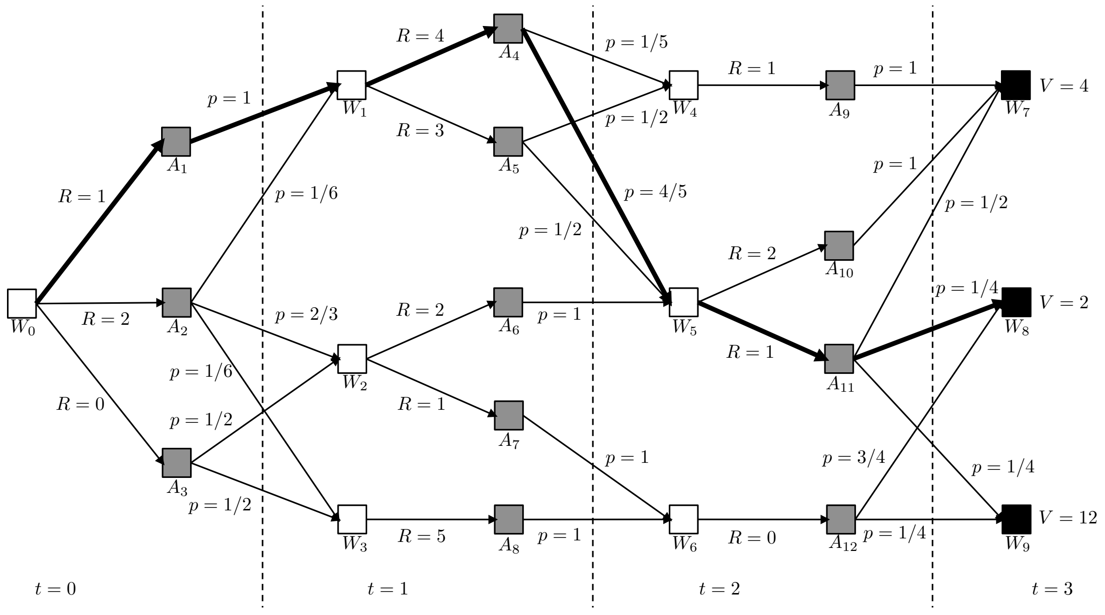


Figure 4.2. Example evolution under the optimal policy for the MDP from Figure 4.1. The evolution is indicated by the path shown in bold. Bold arcs exiting states (white nodes) are due to decisions made by the optimal policy, and bold arcs exiting decision nodes (in gray) are due to realizations of the random state transitions.

Once again, to simplify notation, we use x_t^π in place of $x_t^\pi(S_t)$ for various policies and write (4.18) as

$$V_t^\pi(S_t) \quad = \quad R_t(S_t, x_t^\pi) + \alpha\mathbb{E}[V_{t+1}^\pi(S_{t+1})|S_t, x_t^\pi], \tag{4.20}$$

where the state of interest should be clear from context. An example policy for the MDP in Figure 4.1 is to always choose the decision with the higher index (e.g, among decisions A_1, A_2, A_3, choose A_3). This corresponds to always picking the lower decision arc in the figure. It can be verified that for this policy, the policy values of all states are

$$\begin{aligned} V_0^{\widehat{\pi}}(W_0) &= 7.5, & V_1^{\widehat{\pi}}(W_1) &= 8.75, & V_2^{\widehat{\pi}}(W_4) &= 5, \\ & & V_1^{\widehat{\pi}}(W_2) &= 5.5, & V_2^{\widehat{\pi}}(W_5) &= 6.5, \\ & & V_1^{\widehat{\pi}}(W_3) &= 9.5, & V_2^{\widehat{\pi}}(W_6) &= 4.5. \end{aligned} \tag{4.21}$$

For a general MDP, the value of a state under the optimal policy is denoted by $V_t^*(\cdot)$ and is given by

$$\begin{aligned} V_t^*(S_t) \quad := \quad & R_t(S_t, x_t^*) + \alpha\mathbb{E}[V_{t+1}^*(S_{t+1})|S_t, x_t^*], \\ & t = 0, \dots, T-1. \end{aligned} \tag{4.22}$$

The optimal policy value $V_t^*(S_t)$ defined in (4.22) is equal to $V_t(S_t)$ defined in (4.14).

4.3 Uncertain Value Functions

We begin by considering losses resulting from uncertain value functions. This is of high interest in approximate dynamic programming, where the true/exact value function is computationally difficult to calculate, so an approximate value function is used instead. If the value function approximation has bounded error, one is interested in bounding the resulting loss from the approximate policy, where the loss is defined as the difference between the values of the optimal policy and the approximate policy. The approximate policy in this section is simply the policy that uses the approximate value function. The analysis here draws from the proof of Singh and Lee [134]. Unlike their work, however, our analysis is valid for a non-stationary finite horizon; we also allow for the undiscounted case ($\alpha = 1$) and present a tight example.

Mathematically, we assume the existence of *approximate value functions*

$$\widehat{V}_t : \mathcal{S}_t \to \mathbb{R}, \quad t = 1, \ldots, T. \tag{4.23}$$

During the evolution of the MDP, the *approximate policy* is defined as the policy that makes decisions $\widehat{x}_t(\cdot)$, where

$$\widehat{x}_t(S_t) := \operatorname*{argmax}_{x_t \in \mathcal{X}_t(S_t)} \Big(R_t(S_t, x_t) + \alpha \mathbb{E}[\widehat{V}_{t+1}(S_{t+1}) | S_t, x_t] \Big). \tag{4.24}$$

The value of a state under the approximate policy, which we refer to simply as the *approximate policy value*, is denoted by $V_t^{\widehat{\pi}}(S_t)$ and is given by

$$V_t^{\widehat{\pi}}(S_t) := R_t(S_t, \widehat{x}_t(S_t)) + \alpha \mathbb{E}[V_{t+1}^{\widehat{\pi}}(S_{t+1}) | S_t, \widehat{x}_t(S_t)], \quad t = 0, \ldots, T-1. \tag{4.25}$$

Using $\widehat{x}_t$ in place of $\widehat{x}_t(S_t)$, this equation becomes

$$V_t^{\widehat{\pi}}(S_t) = R_t(S_t, \widehat{x}_t) + \alpha \mathbb{E}[V_{t+1}^{\widehat{\pi}}(S_{t+1}) | S_t, \widehat{x}_t]. \tag{4.26}$$

We assume that the approximate value function at each stage t has error at most δ_t; that is,

$$|V_t(S_t) - \widehat{V}_t(S_t)| \leq \delta_t, \quad \forall S_t \in \mathcal{S}_t, \quad t = 1, \ldots, T. \tag{4.27}$$

The *loss* of the approximate policy for a given state is defined as

$$L_t(S_t) := V_t(S_t) - V_t^{\widehat{\pi}}(S_t), \quad t = 0, \ldots, T. \tag{4.28}$$

The *total loss* $\mathcal{L}$ of the approximate policy is given by the loss of the unique starting state,

$$\mathcal{L} := L_0(S_0) = V_0(S_0) - V_0^{\widehat{\pi}}(S_0). \tag{4.29}$$

The main result for this section is given below, followed by the proof. Note that the infinite horizon case corresponds to $T \to \infty$.

Theorem 4.1. *(Singh and Lee [134]) For a T-stage Markov decision process, if for all stages $t = 1, \ldots, T$ and states $S_t \in \mathcal{S}_t$ the approximate value function satisfies $|V_t(S_t) - \widehat{V}_t(S_t)| \leq \delta_t$, then the total loss of the approximate policy satisfies*

$$\mathcal{L} \leq 2 \sum_{t=1}^{T} \alpha^t \delta_t. \tag{4.30}$$

Corollary 4.1. *For an infinite horizon discounted ($\alpha < 1$) Markov decision process, if for all stages $t = 1, \ldots, T$ and states $S_t \in \mathcal{S}_t$ the approximate value function satisfies $|V_t(S_t) - \widehat{V}_t(S_t)| \leq \delta$, then the total loss of the approximate policy satisfies*

$$\mathcal{L} \leq \frac{2\alpha\delta}{1-\alpha}. \tag{4.31}$$

Corollary 4.2. *For a T-stage undiscounted ($\alpha = 1$) Markov decision process, if for all stages $t = 1, \ldots, T$ and states $S_t \in \mathcal{S}_t$ the approximate value function satisfies $|V_t(S_t) - \widehat{V}_t(S_t)| \leq \delta$, then the total loss of the approximate policy satisfies*

$$\mathcal{L} \leq 2\delta T. \tag{4.32}$$

Proof of Theorem 4.1. The proof is a generalization of the proof used in [134]. Define l_t as the maximum loss over all states at stage t,

$$l_t := \max_{S_t \in \mathcal{S}_t} |L_t(S_t)|, \quad t = 0, \ldots, T. \tag{4.33}$$

From the definition of the approximate policy (4.24), the approximate decision $\widehat{x}_t$ appears as good as the optimal decision x_t^*, so

$$R_t(S_t, x_t^*) + \alpha \mathbb{E}[\widehat{V}_{t+1}(S_{t+1})|S_t, x_t^*] \leq R_t(S_t, \widehat{x}_t) + \alpha \mathbb{E}[\widehat{V}_{t+1}(S_{t+1})|S_t, \widehat{x}_t]. \tag{4.34}$$

We plug in the identities $\widehat{V}_{t+1}(S_{t+1}) \geq V_{t+1}(S_{t+1}) - \delta_{t+1}$ and $\widehat{V}_{t+1}(S_{t+1}) \leq V_{t+1}(S_{t+1}) + \delta_{t+1}$ from (4.27) to obtain

$$\begin{aligned} R_t(S_t, x_t^*) &+ \alpha \mathbb{E}[V_{t+1}(S_{t+1})|S_t, x_t^*] - \alpha\delta_{t+1} \\ &\leq R_t(S_t, \widehat{x}_t) + \alpha \mathbb{E}[V_{t+1}(S_{t+1})|S_t, \widehat{x}_t] + \alpha\delta_{t+1}. \end{aligned} \tag{4.35}$$

Rearranging,

$$R_t(S_t, x_t^*) - R_t(S_t, \widehat{x}_t) \leq 2\alpha\delta_{t+1} + \alpha \mathbb{E}[V_{t+1}(S_{t+1})|S_t, \widehat{x}_t] - \alpha \mathbb{E}[V_{t+1}(S_{t+1})|S_t, x_t^*]. \tag{4.36}$$

Starting with the definition of the loss for some state S_t and using the inequality above,

$$\begin{aligned}
L_t(S_t) &= R_t(S_t, x_t^*) - R_t(S_t, \widehat{x}_t) + \alpha\mathbb{E}[V_{t+1}(S_{t+1})|S_t, x_t^*] - \alpha\mathbb{E}[V_{t+1}^{\widehat{\pi}}(S_{t+1})|S_t, \widehat{x}_t] \\
&\leq 2\alpha\delta_{t+1} + \alpha\mathbb{E}[V_{t+1}(S_{t+1})|S_t, \widehat{x}_t] - \alpha\mathbb{E}[V_{t+1}(S_{t+1})|S_t, x_t^*] \\
&\quad +\alpha\mathbb{E}[V_{t+1}(S_{t+1})|S_t, x_t^*] - \alpha\mathbb{E}[V_{t+1}^{\widehat{\pi}}(S_{t+1})|S_t, \widehat{x}_t] \\
&= 2\alpha\delta_{t+1} + \alpha\mathbb{E}[V_{t+1}(S_{t+1})|S_t, \widehat{x}_t] - \alpha\mathbb{E}[V_{t+1}^{\widehat{\pi}}(S_{t+1})|S_t, \widehat{x}_t] \\
&\leq 2\alpha\delta_{t+1} + \alpha l_{t+1}. \qquad (4.37)
\end{aligned}$$

This gives the recurrence

$$l_t \leq 2\alpha\delta_{t+1} + \alpha l_{t+1}. \qquad (4.38)$$

We form the inductive hypothesis

$$l_t \leq 2\sum_{u=t+1}^{T} \alpha^{u-t}\delta_u. \qquad (4.39)$$

Since terminal states give the same value for every policy, $l_T = 0$. The inductive hypothesis gives

$$l_{T-1} \leq 2\alpha\delta_T, \qquad (4.40)$$

which is correct. Now assuming that (4.39) holds for $t = \tau$, we show that it holds for $t = \tau - 1$.

$$\begin{aligned}
l_{\tau-1} &\leq 2\alpha\delta_\tau + \alpha 2\sum_{u=\tau+1}^{T} \alpha^{u-\tau}\delta_u \\
&= 2\alpha\delta_\tau + 2\sum_{u=\tau+1}^{T} \alpha^{u-\tau+1}\delta_u \\
&= 2\sum_{u=\tau}^{T} \alpha^{u-\tau+1}\delta_u. \qquad (4.41)
\end{aligned}$$

Plugging in $t = 0$ in (4.39) completes the proof.

■

We show a tight example for the undiscounted case stated in Corollary 4.2, which can be generalized to the discounted case. This is given by a deterministic (meaning all transitions occur deterministically) T-stage process consisting of *continue* states and *stop* states. Figure 4.3 shows the process for $T = 3$. The system starts at a continue state C_0. From a given continue state C_t, we can move either to the following continue state C_{t+1} (action C) or the following stop state P_{t+1} (action P) for $t = 0, \ldots, T-1$.

The terminal continue state C_T is *absorbing*, meaning the process ends in this state. Each stop state P_t for $t = 1, \dots, T$ is also absorbing, so once a stop state is selected, the process stops.

The reward function is defined as follows. If at any continue state C_t we decide to stop, the immediate reward is equal to zero. If we continue, the immediate reward is equal to ϵ, where $0 < \epsilon \ll \delta$. Thus,

$$R_t(C_t, x_t) = \begin{cases} 0, & x_t = P, \\ \epsilon, & x_t = C, \end{cases} \qquad t = 0, \dots, T-1. \tag{4.42}$$

The true and estimated values for the stop states are given by

$$\begin{aligned} V_t(P_t) &= 2\delta(T-t) + 2\delta, & t &= 1, \dots, T, \\ \widehat{V}_t(P_t) &= 2\delta(T-t) + \delta, & t &= 1, \dots, T. \end{aligned} \tag{4.43}$$

The true and estimated values for the continue states are given by

$$\begin{aligned} V_t(C_t) &= 2\delta(T-t), & t &= 0, \dots, T, \\ \widehat{V}_t(C_t) &= 2\delta(T-t) + \delta, & t &= 1, \dots, T. \end{aligned} \tag{4.44}$$

Note that the approximate value function satisfies $|V_t(S) - \widehat{V}_t(S)| \le \delta$ for $t = 1, \dots, T$. Let $S_{t+1}(C_t, x_t)$ denote the state encountered at stage $t+1$, resulting from decision $x_t \in \{C, P\}$ at state C_t. At a given continue state C_t, the approximate policy faces the following optimization problem,

$$\begin{aligned} &\max_{x_t \in \{C,P\}} \Big(R_t(C_t, x_t) + \widehat{V}_{t+1}(S_{t+1}(C_t, x_t)) \Big) \\ =\ &\max\left(2\delta(T-t) - \delta,\ \ 2\delta(T-t) - \delta + \epsilon\right). \end{aligned} \tag{4.45}$$

Clearly, the estimated value of choosing to continue is better by ϵ. The approximate policy always chooses to continue and realizes a total value of $T\epsilon$. The optimal policy, however, is to stop at stage C_0 and obtain a value equal to $2\delta T$. As we let ϵ become arbitrarily small, the loss of the approximate policy approaches $2\delta T$.

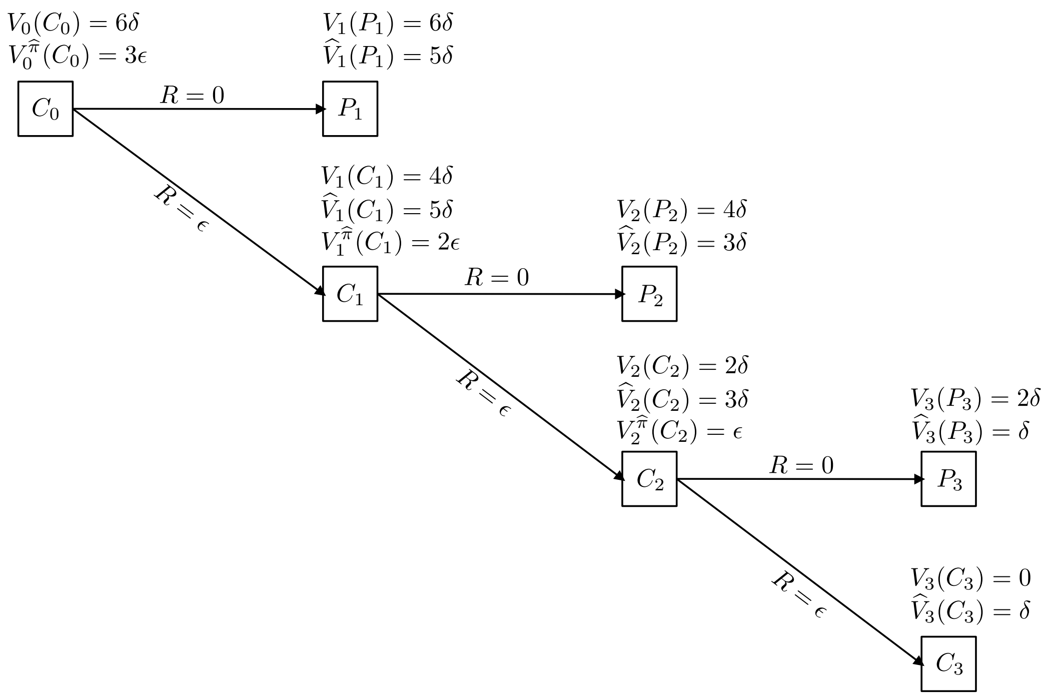

$V_0(C_0) = 6\delta$
$V_0^{\widehat{\pi}}(C_0) = 3\epsilon$
$V_1(P_1) = 6\delta$
$\widehat{V}_1(P_1) = 5\delta$
C_0
$R = 0$
P_1
$R = \epsilon$
$V_1(C_1) = 4\delta$
$\widehat{V}_1(C_1) = 5\delta$
$V_1^{\widehat{\pi}}(C_1) = 2\epsilon$
$V_2(P_2) = 4\delta$
$\widehat{V}_2(P_2) = 3\delta$
C_1
$R = 0$
P_2
$R = \epsilon$
$V_2(C_2) = 2\delta$
$\widehat{V}_2(C_2) = 3\delta$
$V_2^{\widehat{\pi}}(C_2) = \epsilon$
$V_3(P_3) = 2\delta$
$\widehat{V}_3(P_3) = \delta$
C_2
$R = 0$
P_3
$R = \epsilon$
$V_3(C_3) = 0$
$\widehat{V}_3(C_3) = \delta$
C_3

Figure 4.3. Tight example for Corollary 4.2 with $T = 3$.

4.4 Approximate Backward Induction

In this section, we analyze losses from an approximate value function that is generated by some form of approximate backward induction. This setting is also relevant to approximate dynamic programming scenarios, and it will provide intuition for understanding uncertain transition probabilities in the following section.

The backward induction process at each stage introduces its own error, but it also uses value function estimates for the following stage that have their own errors. Despite this accumulation of errors, the total loss incurred will not be as significant as one might expect. This results from the fact that the *relative error* in the value function introduced at each stage ultimately determines the loss, not the total error. This will become clear when we analyze the tight example.

We model the approximate backward induction process via backward induction equation (4.14) with the inclusion of some *error function*

$$E_t : \mathcal{S}_t \times \mathcal{X}_t(\mathcal{S}_t) \to \mathbb{R}, \quad t = 0, \ldots, T-1. \tag{4.46}$$

The *approximate value function* here is thus generated by

$$\begin{aligned} \widehat{V}_t(S_t) \quad &:= \quad \max_{x_t \in \mathcal{X}_t(S_t)} \Big(R_t(S_t, x_t) + \alpha \mathbb{E}[\widehat{V}_{t+1}(S_{t+1}) | S_t, x_t] + E_t(S_t, x_t) \Big), \\ & \qquad t = 0, \ldots, T-1, \end{aligned} \tag{4.47}$$

where $\widehat{V}_T(\cdot) = V_T(\cdot)$. The *approximate policy* in this section makes decisions $\widehat{x}_t(\cdot)$, where

$$\widehat{x}_t(S_t) := \operatorname*{argmax}_{x_t \in \mathcal{X}_t(S_t)} \Big(R_t(S_t, x_t) + \alpha \mathbb{E}[\widehat{V}_{t+1}(S_{t+1})|S_t, x_t] + E_t(S_t, x_t) \Big). \tag{4.48}$$

The value of a state under the approximate policy is again given by

$$V_t^{\widehat{\pi}}(S_t) := R_t(S_t, \widehat{x}_t) + \alpha \mathbb{E}[V_{t+1}^{\widehat{\pi}}(S_{t+1})|S_t, \widehat{x}_t]. \tag{4.49}$$

We use e_t to denote the maximum error at stage t:

$$e_t := \max_{S_t \in \mathcal{S}_t} \max_{x_t \in \mathcal{X}_t(S_t)} |E_t(S_t, x_t)|, \quad t = 0, \dots, T-1. \tag{4.50}$$

Recall that a state loss is

$$L_t(S_t) := V_t(S_t) - V_t^{\widehat{\pi}}(S_t), \quad t = 0, \dots, T, \tag{4.51}$$

and that the total loss of the approximate policy is

$$\mathcal{L} := L_0(S_0) = V_0(S_0) - V_0^{\widehat{\pi}}(S_0). \tag{4.52}$$

We state the main result and then present the analysis. Again, the infinite horizon case corresponds to the limit $T \to \infty$.

Theorem 4.2. *For a T-stage Markov decision process solved with approximate backward induction, if for all stages $t = 0, \dots, T-1$, states $S_t \in \mathcal{S}_t$, and actions $x_t \in \mathcal{X}_t(S_t)$, the error function satisfies $|E(S_t, x_t)| \le e_t$, then the total loss of the approximate policy satisfies*

$$\mathcal{L} \le 2 \sum_{t=0}^{T-1} \alpha^t e_t. \tag{4.53}$$

Corollary 4.3. *For an infinite horizon discounted ($\alpha < 1$) Markov decision process solved with approximate backward induction, if for all stages $t = 0, \dots, T-1$, states $S_t \in \mathcal{S}_t$, and actions $x_t \in \mathcal{X}_t(S_t)$, the error function satisfies $|E(S_t, x_t)| \le e_t$, then the total loss of the approximate policy satisfies*

$$\mathcal{L} \le \frac{2\alpha e}{1-\alpha}. \tag{4.54}$$

Corollary 4.4. *For a T-stage undiscounted ($\alpha = 1$) Markov decision process solved with approximate backward induction, if for all stages $t = 0, \dots, T-1$, states $S_t \in \mathcal{S}_t$, and actions $x_t \in \mathcal{X}_t(S_t)$, the error function satisfies $|E(S_t, x_t)| \le e$, then the total loss of the approximate policy satisfies*

$$\mathcal{L} \le 2eT. \tag{4.55}$$

We introduce a few definitions and lemmas to prove Theorem 4.2. Define the *estimation error* $F_t(S_t)$ for state S_t as

$$F_t(S_t) := V_t(S_t) - \widehat{V}_t(S_t), \quad t = 0, \dots, T. \tag{4.56}$$

Also define the *approximate policy error* $G_t(S_t)$ for state S_t as

$$G_t(S_t) := \widehat{V}_t(S_t) - V_t^{\widehat{\pi}}(S_t), \quad t = 0, \dots, T. \tag{4.57}$$

It is easy to see that with these definitions, the loss of a state is given by

$$L_t(S_t) = F_t(S_t) + G_t(S_t). \tag{4.58}$$

The maximum estimation error and approximate policy error at each stage are denoted by f_t and g_t, respectively;

$$\begin{aligned} f_t &:= \max_{S_t \in \mathcal{S}_t} |F_t(S_t)|, & (4.59)\\ g_t &:= \max_{S_t \in \mathcal{S}_t} |G_t(S_t)|. & (4.60)\end{aligned}$$

We will derive bounds for f_t and g_t in terms of e_t, f_{t+1}, and g_{t+1} in the following lemmas. To be explicit with our development, we first state two propositions.

Proposition 4.1. *For two bounded functions $H_1(x)$ and $H_2(x)$ and a finite domain $\mathcal{X}$,*

$$\max_{x \in \mathcal{X}} \left(H_1(x) + H_2(x)\right) \leq \max_{x \in \mathcal{X}} \left(H_1(x)\right) + \max_{x \in \mathcal{X}} \left(H_2(x)\right). \tag{4.61}$$

Proof. This holds by definition of the maximum. ■

Proposition 4.2. *For two bounded functions $H_1(x)$ and $H_2(x)$ and a finite domain $\mathcal{X}$,*

$$\max_{x \in \mathcal{X}} \left(H_1(x) + H_2(x)\right) \geq \max_{x \in \mathcal{X}} \left(H_1(x)\right) + \min_{x \in \mathcal{X}} \left(H_2(x)\right). \tag{4.62}$$

Proof. Define

$$\begin{aligned} x_{12}^{\max} &:= \operatorname*{argmax}_{x \in \mathcal{X}} \left(H_1(x) + H_2(x)\right), & (4.63)\\ x_1^{\max} &:= \operatorname*{argmax}_{x \in \mathcal{X}} \left(H_1(x)\right), & (4.64)\\ x_2^{\min} &:= \operatorname*{argmin}_{x \in \mathcal{X}} \left(H_2(x)\right). & (4.65)\end{aligned}$$

Then,

$$\begin{aligned} H_1(x_{12}^{\max}) + H_2(x_{12}^{\max}) &\geq H_1(x_1^{\max}) + H_2(x_1^{\max}) \\ &\geq H_1(x_1^{\max}) + H_2(x_2^{\min}). & (4.66)\end{aligned}$$

■

We first determine a recursive relationship for bounds on the estimation error.

Lemma 4.1. *For $t = 0, \dots, T-1$, the bounds on estimation errors satisfy*

$$f_t \leq \alpha f_{t+1} + e_t. \tag{4.67}$$

Proof. Starting with the negative estimation error,

$$\begin{aligned}
\widehat{V}_t(S_t) - V_t(S_t) &= \max_{x_t \in \mathcal{X}_t(S_t)} \Big(R_t(S_t, x_t) + \alpha \mathbb{E}[\widehat{V}_{t+1}(S_{t+1})|S_t, x_t] + E_t(S_t, x_t)\Big) - V_t(S_t) \\
&= \max_{x_t \in \mathcal{X}_t(S_t)} (R_t(S_t, x_t) + \alpha \mathbb{E}[V_{t+1}(S_{t+1})|S_t, x_t] - \alpha \mathbb{E}[F_{t+1}(S_{t+1})|S_t, x_t] \\
&\quad + E_t(S_t, x_t)) - V_t(S_t) \\
&\leq \max_{x_t \in \mathcal{X}_t(S_t)} (R_t(S_t, x_t) + \alpha \mathbb{E}[V_{t+1}(S_{t+1})|S_t, x_t]) \\
&\quad + \max_{x_t \in \mathcal{X}_t(S_t)} (-\alpha \mathbb{E}[F_{t+1}(S_{t+1})|S_t, x_t] + E_t(S_t, x_t)) - V_t(S_t) \\
&\leq \alpha f_{t+1} + e_t.
\end{aligned} \tag{4.68}$$

The first equality above follows from the definition of the approximate value function, and the second equality follows from the definition of the estimation error. The first inequality follows from Proposition 4.1. The second inequality follows from canceling like terms as well as the bounds on estimation error and the error function.

Now for the positive estimation error,

$$\begin{aligned}
V_t(S_t) - \widehat{V}_t(S_t) &= V_t(S_t) - \max_{x_t \in \mathcal{X}_t(S_t)} \Big(R_t(S_t, x_t) + \alpha \mathbb{E}[\widehat{V}_{t+1}(S_{t+1})|S_t, x_t] + E_t(S_t, x_t)\Big) \\
&= V_t(S_t) - \max_{x_t \in \mathcal{X}_t(S_t)} (R_t(S_t, x_t) + \alpha \mathbb{E}[V_{t+1}(S_{t+1})|S_t, x_t] \\
&\quad - \alpha \mathbb{E}[F_{t+1}(S_{t+1})|S_t, x_t] + E_t(S_t, x_t)) \\
&\leq V_t(S_t) - \max_{x_t \in \mathcal{X}_t(S_t)} (R_t(S_t, x_t) + \alpha \mathbb{E}[V_{t+1}(S_{t+1})|S_t, x_t]) \\
&\quad - \min_{x_t \in \mathcal{X}_t(S_t)} (\alpha \mathbb{E}[F_{t+1}(S_{t+1})|S_t, x_t] + E_t(S_t, x_t)) \\
&\leq \alpha f_{t+1} + e_t.
\end{aligned} \tag{4.69}$$

The development above follows the same progression as the previous expression, except the first inequality uses Proposition 4.2. ■

The same relationship holds for the approximate policy error.

Lemma 4.2. *For $t = 0, \dots, T-1$, the bounds on approximate policy error satisfy*

$$g_t \leq \alpha g_{t+1} + e_t. \tag{4.70}$$

Proof. We have for the positive approximate policy error,

$$\begin{aligned}\widehat{V}_t(S_t) - V_t^{\widehat{\pi}}(S_t) &= R_t(S_t,\widehat{x}_t) + \alpha\mathbb{E}[\widehat{V}_{t+1}(S_{t+1})|S_t,\widehat{x}_t] + E_t(S_t,\widehat{x}_t)\\ &\quad -R_t(S_t,\widehat{x}_t) - \alpha\mathbb{E}[V_{t+1}^{\widehat{\pi}}(S_{t+1})|S_t,\widehat{x}_t]\\ &= \alpha\mathbb{E}[G_{t+1}(S_{t+1})|S_t,\widehat{x}_t] + E_t(S_t,\widehat{x}_t)\\ &\le \alpha g_{t+1} + e_t. \end{aligned} \tag{4.71}$$

The first equality uses substitution of the approximate value function, and the second equality follows from canceling like terms and the definition of the approximate policy error. The inequality follows from bounds on the error terms.

Likewise, for the negative approximate policy error,

$$\begin{aligned}V_t^{\widehat{\pi}}(S_t) - \widehat{V}_t(S_t) &= R_t(S_t,\widehat{x}_t) + \alpha\mathbb{E}[V_{t+1}^{\widehat{\pi}}(S_{t+1})|S_t,\widehat{x}_t)]\\ &\quad -R_t(S_t,\widehat{x}_t) - \alpha\mathbb{E}[\widehat{V}_{t+1}(S_{t+1})|S_t,\widehat{x}_t] - E_t(S_t,\widehat{x}_t)\\ &= -\alpha\mathbb{E}[G_{t+1}(S_{t+1})|S_t,\widehat{x}_t] - E_t(S_t,\widehat{x}_t)\\ &\le \alpha g_{t+1} + e_t. \end{aligned} \tag{4.72}$$

■

We can now prove Theorem 4.2.

Proof of Theorem 4.2. Using induction on the above lemmas, we have $g_0 = \sum_{t=0}^{T-1}\alpha^t e_t + \alpha^T g_T$ and $f_0 = \sum_{t=0}^{T-1}\alpha^t e_t + \alpha^T f_T$. With $g_T = f_T = 0$,

$$\mathcal{L} = V_0(S_0) - V_0^{\widehat{\pi}}(S_0) = G_0(S_0) + F_0(S_0) \le g_0 + f_0 = 2\sum_{t=0}^{T-1}\alpha^t e_t. \tag{4.73}$$

■

We show a tight example for Corollary 4.4. The example is simpler than the one in the previous section and is shown for $T = 3$ in Figure 4.4. The example consists of only one decision at the beginning of the process. The decision is between two deterministic paths: one with maximum estimation error and one with maximum approximate policy error. The path with zero future value appears better than the path with maximum value, only by a difference of ϵ, so the approximate policy chooses the path obtaining only ϵ value. The optimal decision is to choose the other path.

Specifically, starting in state S_0, we must choose between state A_1 (action A) and state B_1 (action B). Thereafter, for $t = 1,\dots,T-1$, state A_t deterministically leads to state A_{t+1} and state B_t deterministically leads to state B_{t+1}. All rewards are equal to zero, except for the initial decision:

$$R_0(S_0,x_0) = \begin{cases} 0, & x_0 = A,\\ \epsilon, & x_0 = B,\end{cases} \tag{4.74}$$

where $0 < \epsilon \ll e$. The error function is minimum for path A and maximum for path B:

$$E_0(S_0, x_0) = \begin{cases} -e, & x_0 = A, \\ e, & x_0 = B, \end{cases} \tag{4.75}$$

$$E_t(S_t, \cdot) = \begin{cases} -e, & S_t = A_t, \\ e, & S_t = B_t, \end{cases} \qquad t = 1, \ldots, T-1. \tag{4.76}$$

The states in path A and path B have values

$$\begin{aligned} V_t(A_t) &= 2eT, & t &= 1, \ldots, T, \\ V_t(B_t) = V_t^{\widehat{\pi}}(B_t) &= 0, & t &= 1, \ldots, T. \end{aligned} \tag{4.77}$$

Note since there are no choices to be made along these paths, the values stated above hold for any policy. The stated error functions yield the following estimated values:

$$\begin{aligned} \widehat{V}_t(A_t) &= e(T+t), & t &= 1, \ldots, T-1, \\ \widehat{V}_t(B_t) &= e(T-t), & t &= 1, \ldots, T-1. \end{aligned} \tag{4.78}$$

In effect, states A_1 and B_1 appear to have the same value of eT from the perspective of the approximate policy in state S_0. Since path B gives the extra reward ϵ, the approximate policy chooses this path and realizes a total value of ϵ. The optimal decision is to choose path A, which has value $2eT$.

Notice in the tight example that

$$\delta_t := \max_{S_t \in \mathcal{S}_t} |V_t(S_t) - \widehat{V}_t(S_t)| = e(T-t), \quad t = 1, \ldots, T. \tag{4.79}$$

Using these values in Theorem 4.1 from the previous section gives the total loss bound $\mathcal{L} \leq e(T^2 - T)$, which grows quadratically in T, and is clearly not as strong as the bound $\mathcal{L} \leq 2eT$ derived here. Coincidentally, the two bounds are equal for the case shown in Figure 4.4, where $T = 3$.

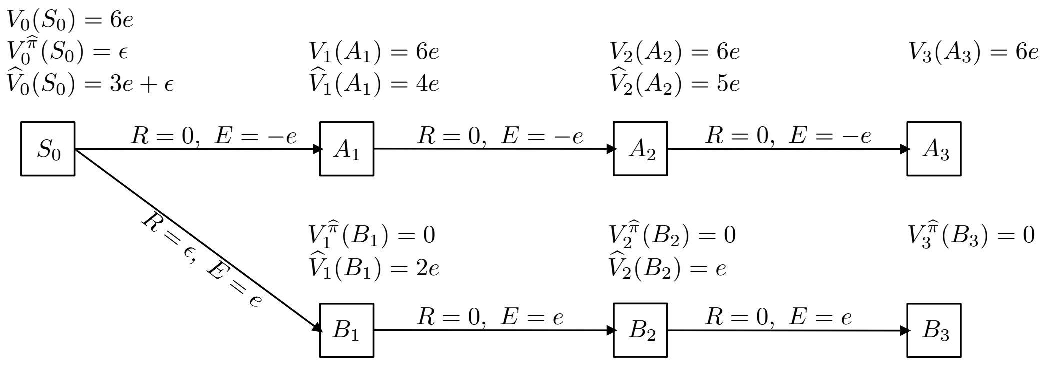

$V_0(S_0) = 6e$
$V_0^{\widehat{\pi}}(S_0) = \epsilon$
$\widehat{V}_0(S_0) = 3e + \epsilon$
$V_1(A_1) = 6e$
$\widehat{V}_1(A_1) = 4e$
$V_2(A_2) = 6e$
$\widehat{V}_2(A_2) = 5e$
$V_3(A_3) = 6e$
S_0
$R = 0,\ E = -e$
A_1
$R = 0,\ E = -e$
A_2
$R = 0,\ E = -e$
A_3
$R = \epsilon,\ E = e$
$V_1^{\widehat{\pi}}(B_1) = 0$
$\widehat{V}_1(B_1) = 2e$
$V_2^{\widehat{\pi}}(B_2) = 0$
$\widehat{V}_2(B_2) = e$
$V_3^{\widehat{\pi}}(B_3) = 0$
B_1
$R = 0,\ E = e$
B_2
$R = 0,\ E = e$
B_3

Figure 4.4. Tight example for Corollary 4.4 with $T = 3$. The policy values for states at stages $t = 1, \ldots, T$ hold for any policy because there is only a single choice for the following state.

4.5 Uncertainty in Transition Probabilities

We now focus on the main topic of this chapter, deriving a general upper bound on the the loss resulting from uncertain transition probabilities. This section is structured as follows. First, we restate some definitions in the context of uncertain transition probabilities. The first subsection states the results, and the second subsection shows the analysis. The third subsection gives a tight example for the undiscounted finite horizon case.

In this section, we are given *estimated transition probability functions*

$$\widehat{\mathbb{P}}_t : \mathcal{S}_{t+1} \times \mathcal{S}_t \times \mathcal{X}_t(\mathcal{S}_t) \to [0,1], \quad t = 0, \dots, T-1, \tag{4.80}$$

where $\widehat{\mathbb{P}}_t(S_{t+1}|S_t, x_t)$ indicates the estimated probability that we transition to state S_{t+1} given that we are in state S_t and select action x_t. These functions satisfy

$$\sum_{S_{t+1} \in \mathcal{S}_{t+1}} \widehat{\mathbb{P}}_t(S_{t+1}|S_t, x_t) = 1, \quad x_t \in \mathcal{X}_t(S_t), \quad S_t \in \mathcal{S}_t, \quad t = 0, \dots, T-1. \tag{4.81}$$

We also assume that these functions satisfy an L_1-norm error bound of 2η with respect to the true transition probability functions.

Assumption 4.1. *For all stages $t = 0, \dots, T-1$, states $S_t \in \mathcal{S}_t$, and actions $x_t \in \mathcal{X}_t(S_t)$, the estimated transition probability functions satisfy*

$$\sum_{S_{t+1} \in \mathcal{S}_{t+1}} \left| \widehat{\mathbb{P}}_t(S_{t+1}|S_t, x_t) - \mathbb{P}_t(S_{t+1}|S_t, x_t) \right| \le 2\eta, \tag{4.82}$$

where $0 < \eta < 1$.

The above error bound can also be stated in terms of the total variation distance between the true and estimated probability distributions. Consider two probability distributions P and Q, defined on a common finite set S. The *total variation distance* between P and Q is the maximum difference in probability that the two distributions assign to the same event, i.e.

$$\eta_{\mathrm{TV}}(P, Q) := \max_{S \in \mathcal{S}} |P(S) - Q(S)|. \tag{4.83}$$

The total variation distance relates to the L_1-norm error as follows (see, e.g. [111], for the short proof).

Proposition 4.3.

$$\eta_{\mathrm{TV}}(P, Q) = \max_{S \in \mathcal{S}} |P(S) - Q(S)| = \frac{1}{2} \sum_{S \in \mathcal{S}} |P(S) - Q(S)|. \tag{4.84}$$

Thus, Assumption 4.1 is equivalent to stating that all estimated transition probability functions have a total variation error of at most η.

The *approximate value function* in this section is now generated using backward induction with the estimated transition probabilities; that is,

$$\begin{aligned} \widehat{V}_t(S_t) \quad := \quad & \max_{x_t \in \mathcal{X}_t(S_t)} \left(R_t(S_t, x_t) + \alpha \sum_{\mathcal{S}_{t+1}} \widehat{\mathbb{P}}_t(S_{t+1}|S_t, x_t) \widehat{V}_{t+1}(S_{t+1}) \right), \\ & t = 0, \dots, T-1, \end{aligned} \tag{4.85}$$

where $\widehat{V}_T(\cdot) = V_T(\cdot)$. We use the shorthand notation $\widehat{\mathbb{E}}[\widehat{V}_{t+1}(S_{t+1})|S_t, x_t]$ in place of $\sum_{\mathcal{S}_{t+1}} \widehat{\mathbb{P}}_t(S_{t+1}|S_t, x_t)\widehat{V}_{t+1}(S_{t+1})$, giving

$$\widehat{V}_t(S_t) = \max_{x_t \in \mathcal{X}_t(S_t)} \left(R_t(S_t, x_t) + \alpha \widehat{\mathbb{E}} \left[\widehat{V}_{t+1}(S_{t+1})|S_t, x_t \right] \right). \tag{4.86}$$

The *approximate policy* in this section makes decisions using the approximate value function as well as estimated transition probabilities:

$$\widehat{x}_t(S_t) := \operatorname*{argmax}_{x_t \in \mathcal{X}_t(S_t)} \left(R_t(S_t, x_t) + \alpha \widehat{\mathbb{E}} \left[\widehat{V}_{t+1}(S_{t+1})|S_t, x_t \right] \right). \tag{4.87}$$

The *approximate policy value* of a state S_t is denoted by $V_t^{\widehat{\pi}}(S_t)$ and, since it represents value obtained from the true system behavior, is defined via the correct transition probabilities:

$$V_t^{\widehat{\pi}}(S_t) := R_t(S_t, \widehat{x}_t(S_t)) + \alpha \mathbb{E}[V_{t+1}^{\widehat{\pi}}(S_{t+1})|S_t, \widehat{x}_t(S_t)]. \tag{4.88}$$

As in the previous sections, we use x_t in place of $x_t(S_t)$ for various policies; the state of interest should be clear from context:

$$V_t^{\widehat{\pi}}(S_t) = R_t(S_t, \widehat{x}_t) + \alpha \mathbb{E}[V_{t+1}^{\widehat{\pi}}(S_{t+1})|S_t, \widehat{x}_t]. \tag{4.89}$$

The *loss* of a state under the approximate policy is defined by

$$L_t(S_t) := V_t(S_t) - V_t^{\widehat{\pi}}(S_t), \quad t = 0, \dots, T. \tag{4.90}$$

The *total loss* of the approximate policy $\mathcal{L}$ is given by the loss of the unique starting state,

$$\mathcal{L} := L_0(S_0) = V_0(S_0) - V_0^{\widehat{\pi}}(S_0). \tag{4.91}$$

To obtain a finite bound on the loss of the approximate policy, it is necessary to have bounds on the rewards for all stages. We state this with the following assumption, where $\overline{R}$ is a nonnegative real number.

Assumption 4.2. *For all stages $t = 0, \ldots, T-1$, states $S_t \in \mathcal{S}_t$, and actions $x_t \in \mathcal{X}_t(S_t)$, the reward function satisfies*

$$0 \leq R_t(S_t, x_t) \leq \overline{R}. \tag{4.92}$$

Additionally, the terminal rewards for all states $S_T \in \mathcal{S}_T$ satisfy

$$0 \leq V_T(S_T) \leq \overline{R}. \tag{4.93}$$

4.5.1 Results

The general upper bound that we derive reduces to the following three cases.

Theorem 4.3. *For a T-stage discounted ($\alpha < 1$) Markov decision process solved with backward induction using uncertain transition probabilities, if for all stages, states, and actions, the transition probability total variation error is at most η, then the total loss of the approximate policy satisfies*

$$\mathcal{L} \leq 2\overline{R}\left(\frac{\alpha\eta - \alpha^{T+1} + \alpha^{T+2}(1-\eta) + \alpha^{T+1}(1-\eta)^{T+1}(1-\alpha)}{(1-\alpha)(1-\alpha(1-\eta))}\right), \tag{4.94}$$

for $0 < \eta < 1$.

Theorem 4.4. *For an infinite horizon discounted ($\alpha < 1$) Markov decision process solved with backward induction using uncertain transition probabilities, if for all stages, states, and actions, the transition probability total variation error is at most η, then the total loss of the approximate policy satisfies*

$$\mathcal{L} \leq \frac{2\overline{R}\alpha\eta}{(1-\alpha)(1-\alpha(1-\eta))}, \tag{4.95}$$

for $0 < \eta < 1$.

Theorem 4.5. *For a T-stage undiscounted ($\alpha = 1$) Markov decision process solved with backward induction using uncertain transition probabilities, if for all stages, states, and actions, the transition probability total variation error is at most η, then the total loss of the approximate policy satisfies*

$$\mathcal{L} \leq \frac{2\overline{R}}{\eta}\left(-1 + \eta(T+1) + (1-\eta)^{T+1}\right), \tag{4.96}$$

for $0 < \eta < 1$.

It can be shown that the loss function is concave in η. Thus, first-order loss bounds can be found by evaluating $\lim_{\eta \to 0} \frac{\partial \mathcal{L}}{\partial \eta}$. This gives the following corollaries.

Corollary 4.5. *For a finite or infinite horizon discounted problem* $(\alpha < 1)$*, the total loss of the approximate policy satisfies*

$$\mathcal{L} \leq \frac{2\overline{R}\alpha\eta}{(1-\alpha)^2}. \tag{4.97}$$

Corollary 4.6. *For a finite horizon undiscounted problem* $(\alpha = 1)$*, the total loss of the approximate policy satisfies*

$$\mathcal{L} \leq \eta\overline{R}T(T+1). \tag{4.98}$$

Figure 4.5 shows the loss bound for the infinite horizon discounted case as a function of η for two fixed values of α and $\overline{R} = 1$. Both the bounds from the full expression (4.95) and the first-order expression (4.97) are shown. To visualize the full expression (4.95) as a function of α, we define the *normalized loss*, which is simply the loss divided by the maximum possible total reward of $\overline{R}/(1-\alpha)$,

$$\widetilde{\mathcal{L}} := \frac{\mathcal{L}}{\overline{R}/(1-\alpha)}. \tag{4.99}$$

Figure 4.6 shows a plot of the normalized loss bound for various values of η as a function of α.

Similar results for the finite horizon undiscounted case are shown in Figure 4.7 and Figure 4.8. Figure 4.7 shows bounds for $T = 10$ and $T = 5$ as a function of η, along with first-order approximations. The normalized loss for the finite horizon case is defined as

$$\widetilde{\mathcal{L}} := \frac{\mathcal{L}}{\overline{R}(T+1)}. \tag{4.100}$$

Figure 4.8 shows a plot of the normalized loss bound for various values of η as a function of T. It is clear that the bound exceeds the bound on maximum total reward for large enough values of η and T. This is true for $\eta \geq 0.2$ and $T \geq 8$ as well as $\eta \geq 0.1$ and $T \geq 15$, for instance.

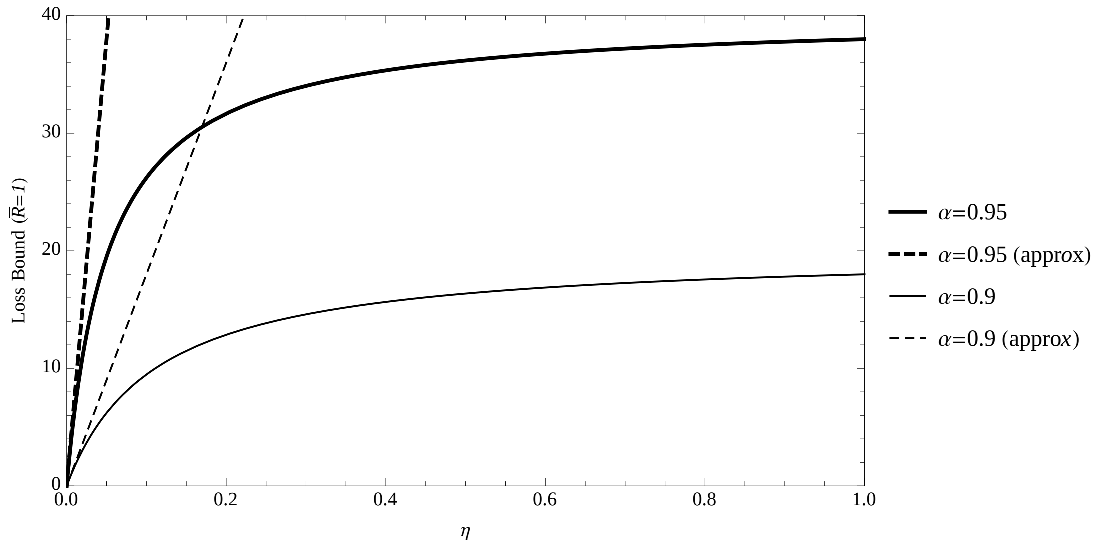


Figure 4.5. Loss bound for infinite horizon discounted case (4.95) as a function of η for two values of α fixed and $\overline{R} = 1$. First-order approximations are shown as dashed.

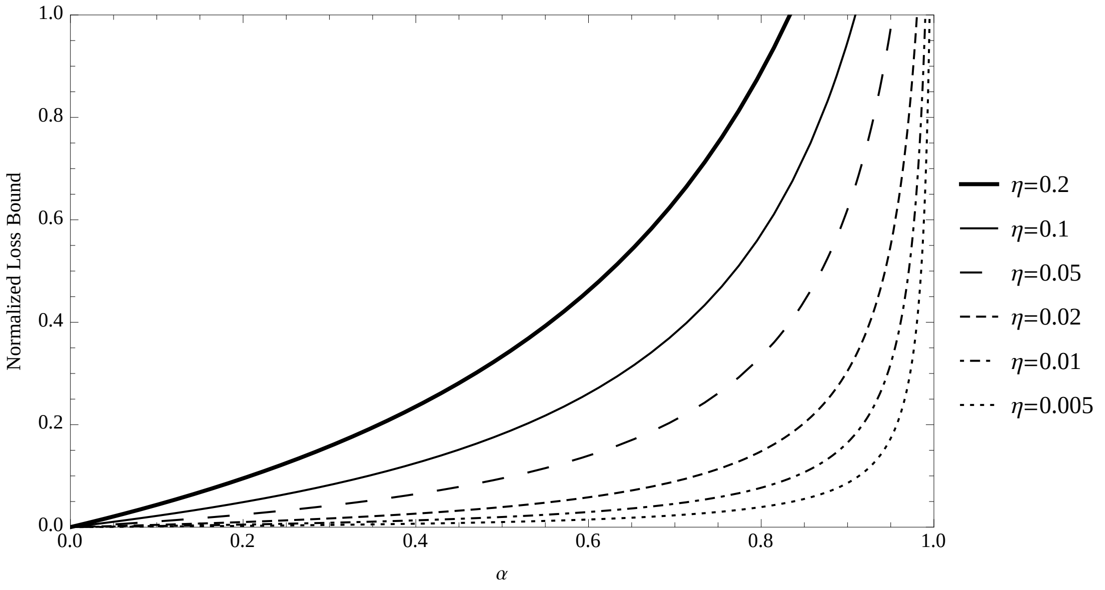


Figure 4.6. Normalized loss bound for infinite horizon discounted case as a function of discount factor α for various values of η.

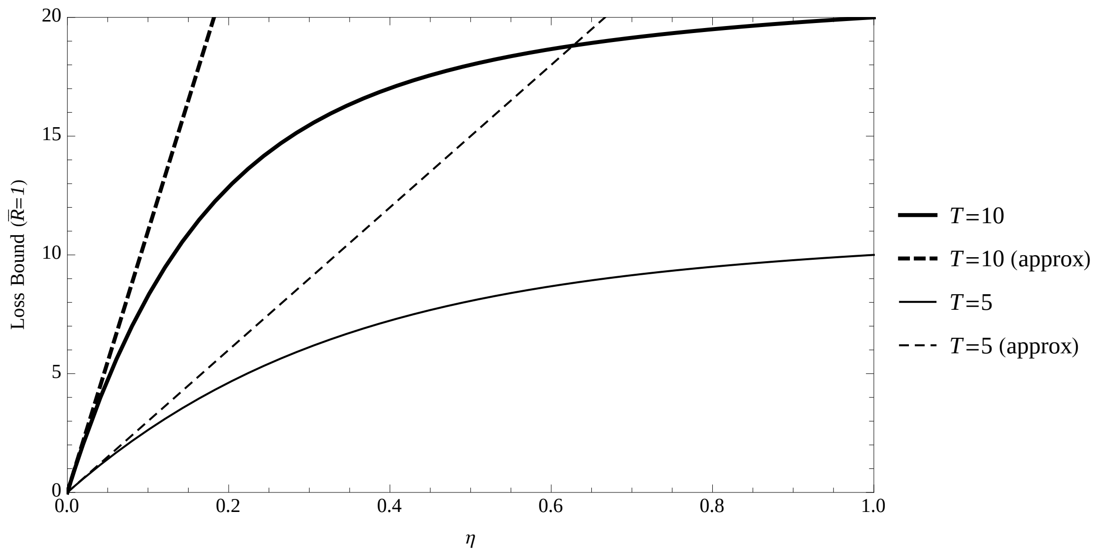


Figure 4.7. Loss bound for undiscounted case (4.96) as a function of η for two values of T fixed and $\overline{R} = 1$. First-order approximations are shown as dashed.

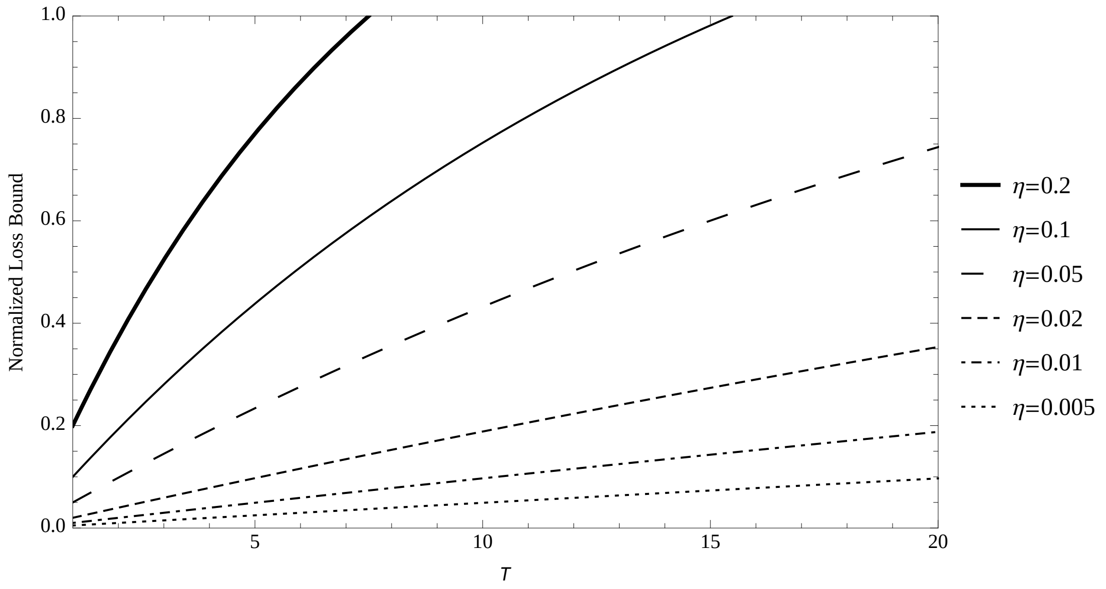


Figure 4.8. Normalized loss bound for undiscounted case as a function of horizon length T for various values of η.

4.5.2 Analysis

The overarching proof approach is similar to the analysis in the previous section. We will use the same definitions for estimation error and approximate policy error. Instead of the additive error considered in the previous section, the errors here will accumulate from calculating expected future values using an estimated transition probability distribution. To reiterate, the *estimation error* for a state S_t is

$$F_t(S_t) := V_t(S_t) - \widehat{V}_t(S_t), \quad t = 0, \ldots, T. \tag{4.101}$$

The *approximate policy error* for a state S_t is

$$G_t(S_t) := \widehat{V}_t(S_t) - V_t^{\widehat{\pi}}(S_t), \quad t = 0, \ldots, T. \tag{4.102}$$

The maximum estimation error at stage t is

$$f_t \quad := \quad \max_{S_t \in \mathcal{S}_t} |F_t(S_t)|, \quad t = 0, \ldots, T, \tag{4.103}$$

and the maximum approximate policy error is

$$g_t \quad := \quad \max_{S_t \in \mathcal{S}_t} |G_t(S_t)|, \quad t = 0, \ldots, T. \tag{4.104}$$

We define the *difference* between the true and estimated distributions as

$$D(S_{t+1}|S_t, x_t) := \widehat{\mathbb{P}}_t(S_{t+1}|S_t, x_t) - \mathbb{P}_t(S_{t+1}|S_t, x_t). \tag{4.105}$$

We can view $D(\cdot|S_t, x_t)$ as vector of length $|\mathcal{S}_{t+1}|$ with entries that sum to zero; we refer to $D(\cdot|S_t, x_t)$ as a *difference vector*. In terms of the transition probability error bound from Assumption 4.1, for all stages, states, and actions,

$$\sum_{\mathcal{S}_{t+1}} |D(S_{t+1}|S_t, x_t)| \leq 2\eta. \tag{4.106}$$

Before bounding the growth of maximum estimation and approximate policy errors, we observe a few implications following from the bounds on rewards. Recall Assumption 4.2, which states that rewards at each stage must be nonnegative and at most $\overline{R}$. We define the the maximum possible state value at stage t,

$$\overline{V}_t := \overline{R} \sum_{u=t}^{T} \alpha^{u-t}. \tag{4.107}$$

This immediately gives

$$0 \leq V_t(S_t) \leq \overline{V}_t, \quad \forall S_t \in \mathcal{S}_t, \quad t = 0, \ldots, T. \tag{4.108}$$

The same holds for the value of any policy, including the approximate policy,

$$0 \leq V_t^{\widehat{\pi}}(S_t) \leq \overline{V}_t, \quad \forall S_t \in \mathcal{S}_t, \quad t = 0, \ldots, T. \tag{4.109}$$

An equivalent statement holds for the approximate value function, as given by the following lemma.

Lemma 4.3. *Under the backward induction process with estimated transition probabilities, for all stages $t = 0, \ldots, T$ and states $S_t \in \mathcal{S}_t$,*

$$0 \leq \widehat{V}_t(S_t) \leq \overline{V}_t. \tag{4.110}$$

Proof. Intuitively, each state value estimate is formed by taking a convex combination of state values for the following stage, so the property holds by induction. More explicitly, we show that if the property holds for all states at stage $t = \tau + 1$, then it holds for all states at stage $t = \tau$. For a given state S_τ,

$$\begin{aligned}
\widehat{V}_\tau(S_\tau) &= \max_{x_\tau} \left(R_\tau(S_\tau, x_\tau) + \alpha \sum_{\mathcal{S}_{\tau+1}} \widehat{\mathbb{P}}(S_{\tau+1}|S_\tau, x_\tau) \widehat{V}_{\tau+1}(S_{\tau+1}) \right) \\
&\leq \overline{R} + \alpha \overline{V}_{\tau+1} \\
&= \overline{R} + \alpha \overline{R} \sum_{u=\tau+1}^{T} \alpha^{u-\tau-1} \\
&= \overline{R} \left(1 + \sum_{u=\tau+1}^{T} \alpha^{u-\tau} \right) \\
&= \overline{R} \sum_{u=\tau}^{T} \alpha^{u-\tau} = \overline{V}_\tau.
\end{aligned} \tag{4.111}$$

The second line uses Assumption 4.2 and the inductive hypothesis. The remaining lines follow from manipulation and use of the maximum value expression (4.107). A similar argument gives that $\widehat{V}_\tau(S_\tau) \geq 0$. Since the terminal state values satisfy $V_T(S_T) = \widehat{V}_T(S_T)$, the inductive argument is complete. ■

We now focus on bounding the growth of errors. We start by assuming that we are given the maximum estimation error f_{t+1} and we wish to find an upper bound on f_t. In the backward induction process with estimated transition probabilities, recall that the estimated value of a state is

$$\widehat{V}_t(S_t) = \max_{x_t \in \mathcal{X}_t(S_t)} \left(R_t(S_t, x_t) + \alpha \sum_{\mathcal{S}_{t+1}} \widehat{\mathbb{P}}_t(S_{t+1}|S_t, x_t) \widehat{V}_{t+1}(S_{t+1}) \right). \tag{4.112}$$

From the definition of the difference vector,

$$\begin{aligned}
\widehat{V}_t(S_t) = \max_{x_t \in \mathcal{X}_t(S_t)} &\left(R_t(S_t, x_t) + \alpha \sum_{\mathcal{S}_{t+1}} \mathbb{P}_t(S_{t+1}|S_t, x_t) \widehat{V}_{t+1}(S_{t+1}) \right. \\
&\left. + \alpha \sum_{\mathcal{S}_{t+1}} D(S_{t+1}|S_t, x_t) \widehat{V}_{t+1}(S_{t+1}) \right).
\end{aligned} \tag{4.113}$$

Starting with the negative estimation error,

$$\begin{aligned}
-F_t(S_t) &= \widehat{V}_t(S_t) - V_t(S_t) \\
&= \max_{x_t \in \mathcal{X}_t(S_t)} \Bigg(R_t(S_t, x_t) + \alpha \sum_{\mathcal{S}_{t+1}} \mathbb{P}_t(S_{t+1}|S_t, x_t) \widehat{V}_{t+1}(S_{t+1}) \\
&\quad + \alpha \sum_{\mathcal{S}_{t+1}} D(S_{t+1}|S_t, x_t) \widehat{V}_{t+1}(S_{t+1}) \Bigg) - V_t(S_t) \\
&= \max_{x_t \in \mathcal{X}_t(S_t)} \Bigg(R_t(S_t, x_t) + \alpha \sum_{\mathcal{S}_{t+1}} \mathbb{P}_t(S_{t+1}|S_t, x_t) V_{t+1}(S_{t+1}) \\
&\quad - \alpha \sum_{\mathcal{S}_{t+1}} \mathbb{P}_t(S_{t+1}|S_t, x_t) F_{t+1}(S_{t+1}) + \alpha \sum_{\mathcal{S}_{t+1}} D(S_{t+1}|S_t, x_t) V_{t+1}(S_{t+1}) \\
&\quad - \alpha \sum_{\mathcal{S}_{t+1}} D(S_{t+1}|S_t, x_t) F_{t+1}(S_{t+1}) \Bigg) - V_t(S_t) \\
&\leq \alpha \max \Bigg(- \sum_{\mathcal{S}_{t+1}} \mathbb{P}_t(S_{t+1}|S_t, x_t) F_{t+1}(S_{t+1}) + \sum_{\mathcal{S}_{t+1}} D(S_{t+1}|S_t, x_t) V_{t+1}(S_{t+1}) \\
&\quad - \sum_{\mathcal{S}_{t+1}} D(S_{t+1}|S_t, x_t) F_{t+1}(S_{t+1}) \Bigg). \qquad (4.114)
\end{aligned}$$

The third equality above follows from substitution of the estimation error for stage $t+1$, and the inequality follows from Proposition 4.1 and canceling like terms. The last maximum is taken over all possible probability distributions, difference vectors, and value functions. Since x_t simply indexes distributions, difference vectors, and value functions, this generalization is justified.

We wish to find an upper bound on (4.114) that holds for all instances. This requires formulating a multilinear program with constraints representing our stated assumptions. To simplify notation for the program, we abbreviate $F_{t+1}(S_{t+1})$, $V_{t+1}(S_{t+1})$, $\mathbb{P}_t(S_{t+1}|S_t, x_t)$, $D(S_{t+1}|S_t, x_t)$ by $F(s)$, $V(s)$, $P(s)$, $D(s)$, respectively, where s is used in place of S_{t+1}. Also denote the set $\mathcal{S}_{t+1}$ by $\mathcal{S}$, the bound f_{t+1} by f, and the bound $\overline{V}_{t+1}$ by $\overline{V}$. With further abbreviation, we consider F, V, P, D all as vectors in $\mathbb{R}^{|\mathcal{S}|}$. Temporarily ignoring the α term in (4.114), we have the multilinear program

$$
\begin{aligned}
\underset{F,V,P,D}{\text{maximize}} \quad & Z_1 = \sum_{s\in\mathcal{S}} \left(-P(s)F(s) + D(s)V(s) - D(s)F(s)\right) \\
\text{subject to} \quad & \sum_{s\in\mathcal{S}} |D(s)| \le 2\eta, \\
& \sum_{s\in\mathcal{S}} P(s) = 1, \\
& \sum_{s\in\mathcal{S}} D(s) = 0, \\
& 0 \le V(s) \le \overline{V}, && \forall s \in \mathcal{S}, \\
& 0 \le V(s) - F(s) \le \overline{V}, && \forall s \in \mathcal{S}, \\
& |F(s)| \le f, && \forall s \in \mathcal{S}, \\
& 0 \le P(s) \le 1, && \forall s \in \mathcal{S}, \\
& 0 \le P(s) + D(s) \le 1, && \forall s \in \mathcal{S}.
\end{aligned}
\tag{4.115}
$$

Implicitly, we are also considering sets $\mathcal{S}$ of all finite sizes. For the constraints with sums, the first one represents the bound on total variation error, and the second and third are necessary for both the true and estimated transition probabilities to be valid. The two constraints thereafter ensure that both the true and estimated state values do not violate the bounds given by (4.108) and Lemma 4.3. Of the last three constraints, the first constraint is the bound on estimation error, and the remaining two bounds are required to ensure valid probability distributions.

Let Z_1^* denote the optimal objective value for (4.115). In an effort to find an upper bound for this program, we first show that we can impose additional assumptions on the probability distribution $P(s)$ without affecting the final bound. We then find optimal choices of variables when other variables are fixed to obtain the bound. Define the states with the maximum and minimum $V(s) - F(s)$ values as

$$
s^+ := \underset{s\in\mathcal{S}}{\operatorname{argmax}} \left(V(s) - F(s)\right), \tag{4.116}
$$

$$
s^- := \underset{s\in\mathcal{S}}{\operatorname{argmin}} \left(V(s) - F(s)\right). \tag{4.117}
$$

Consider an instance where $P(s)$, $V(s)$, and $F(s)$ are given and where $P(s^-) \ge \eta$ and $P(s^+) \le 1 - \eta$. We have left to choose $D(s)$. The result has a simple structure.

Lemma 4.4. *For instances of (4.115) where P, V, and F are fixed, and where $P(s^-) \ge \eta$ and $P(s^+) \le 1 - \eta$, the optimal choice of $D(s)$ is given by*

$$
D(s) = \begin{cases} \eta, & s = s^+, \\ -\eta, & s = s^-, \\ 0, & \text{otherwise.} \end{cases} \tag{4.118}
$$

Proof. The non-constant part of the objective function is $\sum_{s\in\mathcal{S}} D(s)\left(V(s)-F(s)\right)$. Let $\mathcal{S}^+ := \{s : D(s) > 0\}$ and $\mathcal{S}^- := \{s : D(s) < 0\}$. Define $\Delta^+ := \sum_{s\in\mathcal{S}^+} D(s)$ and $\Delta^- := \sum_{s\in\mathcal{S}^-} D(s)$. For any given Δ^+ and Δ^-, it is optimal to choose $\mathcal{S}^+ = \{s^+\}$ and $\mathcal{S}^- = \{s^-\}$ as these provide the largest and smallest multipliers for Δ^+ and Δ^-, respectively. Now letting Δ^+, Δ^- vary, we must have $\Delta^+ = -\Delta^-$, and the resulting objective function is increasing in Δ^+, assuming $V(s^+) - F(s^+) \neq V(s^-) - F(s^-)$. Making Δ^+ as large as possible gives $\Delta^+ = \eta$ and $\Delta^- = -\eta$. ■

The assumption on $P(s^+)$ and $P(s^-)$ values is without loss of generality, as shown by the following lemma. Define an *instance* of (4.115) as a set of states $\mathcal{S}$ with given P, F, V vectors. Optimization is over D. Let $\mathcal{P}$ be the set of all problem instances, and let $\overline{\mathcal{P}}$ be the set of problem instances that satisfy $P(s^+) \leq (1-\eta)$ and $P(s^-) \geq \eta$.

Lemma 4.5. *For every problem instance $\mathcal{I} \in \mathcal{P}\setminus\overline{\mathcal{P}}$ with optimal value Z^*, there exists a problem instance $\overline{\mathcal{I}} \in \overline{\mathcal{P}}$ with optimal value $\overline{Z}^*$ such that $Z^* \leq \overline{Z}^*$.*

Proof. The presence of $\sum_{s\in\mathcal{S}} -P(s)F(s)$ in the objective function requires the proof to be non-trivial. Returning to the analysis in Lemma 4.4, if $P(s^-) < -\Delta^-$, it is optimal to add states to $\mathcal{S}^-$ in increasing order of $V(s) - F(s)$ until $\sum_{s\in\mathcal{S}^-} P(s) \geq -\Delta^-$. If $P(s^+) > 1-\eta$, then $\Delta^+ = 1 - P(s^+)$. With this in mind, we use a problem transformation algorithm to produce the instance $\overline{\mathcal{I}}$ given $\mathcal{I}$ and show that during each step of the algorithm, the optimal objective value does not decrease. The procedure for generating the new problem instance is shown in Algorithm 14.

Beginning with line 4, if $P(s^+) > 1-\eta$, the algorithm adjusts the values of states s^+ and s^-. Under the optimal solution, if $P(s^+)$ is decreased by c, then $D(s^+)$ increases by c. Let $V'(\cdot)$ $F'(\cdot)$, $D'(\cdot)$ refer to state properties after lines 4-10 of the algorithm have been executed. We use the shorthand notation P_+ for $P(s^+)$, F'_- for $F'(s^-)$, etc. The change in the optimal objective value ΔZ^* for line 6 is

$$\begin{aligned}\Delta Z^* &= -(P_+ - c)F_+ + (D_+ + c)V_+ - (D_+ + c)F_+ \\ &\quad + P_+F_+ - D_+V_+ + D_+F_+ \\ &= cV_+, \end{aligned} \tag{4.119}$$

which is always nonnegative. The change in optimal objective value for lines 7-9 is

$$\begin{aligned}\Delta Z^* &= -P'_-F'_- + D'_-V'_- - D'_-F'_- \\ &\quad + P_-F_- - D_-V_- + D_-F_- \\ &= P_-V_-, \end{aligned} \tag{4.120}$$

where we have used $P_- = -D_-$ under the optimal solution.

Lines 11-19 of the algorithm are used to aggregate states in an iterative fashion until $P(s^-) \geq \eta$. Define $\mathcal{S}^-(\mathcal{A})$ as the set of states in instance $\mathcal{A}$ that are assigned negative $D(s)$ values under the optimization of (4.115), as explained in Lemma 4.4. At each iteration of the process, the algorithm aggregates the two smallest $V(s) - F(s)$ states

```
Input: $\mathcal{I} \in \mathcal{P}$
Output: $\overline{\mathcal{I}} \in \overline{\mathcal{P}}$
 1: $\mathcal{A} \leftarrow \mathcal{I}$
 2: $s^+ \leftarrow \operatorname{argmax}_{s \in \mathcal{A}} V(s) - F(s)$
 3: $s^- \leftarrow \operatorname{argmin}_{s \in \mathcal{A}} V(s) - F(s)$
 4: if $P(s^+) > 1 - \eta$ then
 5:     $c \leftarrow P(s^+) - (1 - \eta)$
 6:     $P(s^+) \leftarrow P(s^+) - c$
 7:     $P(s^-) \leftarrow P(s^-) + c$
 8:     $F(s^-) \leftarrow 0$
 9:     $V(s^-) \leftarrow 0$
10: end if
11: while $\sum_{s \in \mathcal{S}^-(\mathcal{A})} P(s) \leq \Delta^-$ do
12:     $r_1 \leftarrow \operatorname{argmin}_{s \in \mathcal{A}} V(s) - F(s)$
13:     $r_2 \leftarrow \operatorname{argmin}_{s \in \mathcal{A} \setminus \{r_1\}} V(s) - F(s)$
14:     $P(r') \leftarrow P(r_1) + P(r_2)$
15:     $D(r') \leftarrow D(r_1) + D(r_2)$
16:     $F(r') \leftarrow \max(F(r_2) - V(r_2), -f)$
17:     $V(r') \leftarrow 0$
18:     $\mathcal{A} \leftarrow (\mathcal{A} \setminus \{r_1, r_2\}) \cup \{r'\}$
19: end while
20: $\overline{\mathcal{I}} \leftarrow \mathcal{A}$
```

Algorithm 14. PROBLEM-TRANSFORMATION

to produce a new state r' with $V(r') - F(r')$ value smaller than the remaining states. At any point in the process, let r_1 and r_2 be the states with the smallest and second smallest $V(s) - F(s)$ values, respectively. Initially, $r_1 = s^-$. For other iterations, $r_1 = r'$ from the previous iteration. During the aggregation process, we wish only to increase the objective value, so adding the aggregated state and removing the two original states always creates a positive change in the objective function:

$$\begin{aligned} -P'F' + D'(V' - F') \quad &\geq \quad -P_1F_1 - P_2F_2 + D_1(V_1 - F_1) \\ &\qquad +D_2(V_2 - F_2), \end{aligned} \tag{4.121}$$

where $P_1 = P(r_1)$, $P_2 = P(r_2)$, and V', F', D' now refer to state values obtained after one iteration of the aggregation process. The D' and P' values are given by

$$D' = D_1 + D_2, \tag{4.122}$$

$$P' = P_1 + P_2. \tag{4.123}$$

The V' value is always equal to zero, and F' is determined according to

$$F' = \max(F_2 - V_2, -f), \tag{4.124}$$

which results from the constraints that $V' - F'$ must be positive and $V' - F' \leq V_2 - F_2$. The algorithm repeats the process while $P(r') \leq \eta$, so $D_1 = -P_1$ always holds, as the optimal $D(s)$ places the maximum possible weight on the lowest $V(s) - F(s)$ coefficient before placing weight on other states. The change in objective value ΔZ^* for the aggregation process is always nonnegative. For each iteration, there are two cases; $-f > F_2 - V_2$ and $-f \leq F_2 - V_2$. For the first case, we have

$$\begin{aligned}
\Delta Z^* &= -P'F' + D'(V' - F') + P_1F_1 + P_2F_2 - D_1V_1 + D_1F_1 - D_2V_2 + D_2F_2 \\
&= (P_1 + P_2)f + (D_1 + D_2)f + P_1F_1 + P_2F_2 - D_1V_1 + D_1F_1 - D_2V_2 + D_2F_2 \\
&= P_1(F_1 + f) + P_2(F_2 + f) + D_1(f + F_1 - V_1) + D_2(f + F_2 - V_2). \qquad (4.125)
\end{aligned}$$

Since $P_1 = -D_1$,

$$\Delta Z^* = P_1V_1 + P_2(F_2 + f) + D_2(f + F_2 - V_2). \qquad (4.126)$$

The terms D_2 and $f + F_2 - V_2$ are nonpositive (the latter by definition of the case), and the remaining terms are all nonnegative. The second case gives

$$\begin{aligned}
\Delta Z^* &= -P'F' + D'(V' - F') + P_1F_1 + P_2F_2 - D_1V_1 + D_1F_1 - D_2V_2 + D_2F_2 \\
&= (P_1 + P_2)(V_2 - F_2) + (D_1 + D_2)(V_2 - F_2) + P_1F_1 + P_2F_2 - D_1V_1 \\
&\quad + D_1F_1 - D_2V_2 + D_2F_2 \\
&= P_1(V_2 - F_2 + F_1) + P_2V_2 + D_1(V_2 - F_2 - V_1 + F_1). \qquad (4.127)
\end{aligned}$$

Again using $P_1 = -D_1$,

$$\Delta Z^* = P_1V_1 + P_2V_2, \qquad (4.128)$$

which is always nonnegative. ■

The above lemma shows that we can assume $P(s^+) \leq (1-\eta)$ and $P(s^-) \geq \eta$ without loss of generality. Using this observation with Lemma 4.4 allows us to formulate the following multilinear program, which has optimal objective value Z_2^*. As we show in the lemma below, $Z_1^* \leq Z_2^*$.

$$\begin{aligned}
\underset{F,V,P}{\text{maximize}} \quad & Z_2 = \eta\left(V(s^+) - F(s^+)\right) - \eta\left(V(s^-) - F(s^-)\right) - \sum_{s \in \mathcal{S}} P(s)F(s) \\
\text{subject to} \quad & \sum_{s \in \mathcal{S}} P(s) = 1, \\
& P(s^+) \leq 1 - \eta, \\
& P(s^-) \geq \eta, \\
& 0 \leq V(s) \leq \overline{V}, && \forall s \in \mathcal{S}, \\
& 0 \leq V(s) - F(s) \leq \overline{V}, && \forall s \in \mathcal{S}, \\
& |F(s)| \leq f, && \forall s \in \mathcal{S}, \\
& 0 \leq P(s) \leq 1, && \forall s \in \mathcal{S}, \\
& V(s) - F(s) \leq V(s^+) - F(s^+), && \forall s \in \mathcal{S}, \\
& V(s) - F(s) \geq V(s^-) - F(s^-), && \forall s \in \mathcal{S}.
\end{aligned} \qquad (4.129)$$

Lemma 4.6. *The optimal objective values for the multilinear programs* (4.115) *and* (4.129) *satisfy*

$$Z_1^* \le Z_2^*. \tag{4.130}$$

Proof. Lemma 4.5 states that we can assume $P(s^-) \ge \eta$ and $P(s^+) \le 1 - \eta$ without loss of generality, and Lemma 4.4 indicates the optimal values for the D vector in such cases. ∎

We now assume that V and P are fixed and we would like to calculate the optimal F values. We rewrite the objective function in (4.129) as

$$Z_2 = \eta V_+ - \eta V_- - (P_+ + \eta)F_+ - (P_- - \eta)F_- - \sum_{s \in \mathcal{S} \setminus \{s^-, s^+\}} P(s)F(s), \tag{4.131}$$

where $P_+ := P(s^+)$, $P_- := P(s^-)$, $F_+ := F(s^+)$, $F_- := F(s^-)$, $V_+ := V(s^+)$, and $V_- := V(s^-)$. Since the coefficients for all $F(s)$ values are nonnegative, the $F(s)$ values for all $s \in \mathcal{S}$ should be made as small as possible. The resulting objective function is bounded by the following lemma.

Lemma 4.7. *The optimal objective value of (4.129) satisfies*

$$Z_2^* \le \eta \overline{V} + (1 - \eta) f. \tag{4.132}$$

Proof. Using the bounds on $F(s)$ gives

$$Z_2^* \le \eta V_+ - \eta V_- - (P_+ + \eta)F_+ - (P_- - \eta)F_- + f(1 - P_+ - P_-). \tag{4.133}$$

We evaluate cases based on values for V_+, V_-.

Case 1: $V_+ \ge \overline{V} - f,\ V_- \ge \overline{V} - f$
The smallest that F_+ and F_- can be is $V_+ - \overline{V}$ and $V_- - \overline{V}$, respectively. This gives

$$\begin{aligned} Z_2^* &\le \eta V_+ - \eta V_- - (p_+ + \eta)(V_+ - \overline{V}) - (p_- - \eta)(V_- - \overline{V}) + f(1 - p_+ - p_-) \\ &= -p_+V_+ + p_+\overline{V} - p_-V_- + p_-\overline{V} + f - fp_+ - fp_- \\ &= p_+(\overline{V} - V_+ - f) + p_-(\overline{V} - V_- - f) + f \\ &\le f, \end{aligned} \tag{4.134}$$

where we have used that both $\overline{V} - V_+ - f$ and $\overline{V} - V_+ - f$ are nonpositive by definition of the case. The other cases follow similar reasoning.

Case 2: $V_+ \geq \overline{V} - f,\ V_- \leq \overline{V} - f$
We set $F_+ = V_+ - \overline{V}$ and $F_- = -f$. Then,

$$\begin{aligned} Z_2^* &\leq \eta V_+ - \eta V_- - (p_+ + \eta)(V_+ - \overline{V}) + (p_- - \eta)f + f(1 - p_+ - p_-) \\ &= -\eta V_- + \eta\overline{V} - p_+ V_+ + p_+\overline{V} - \eta f + f - f p_+ \\ &= -\eta V_- + \eta\overline{V} + p_+(\overline{V} - V_+ - f) + f(1-\eta) \\ &\leq \eta\overline{V} + (1-\eta)f, \end{aligned} \tag{4.135}$$

where we have set $V_- = 0$ and used that $\overline{V} - V_+ - f \leq 0$.

Case 3: $V_+ \leq \overline{V} - f,\ V_- \leq \overline{V} - f$
Both F_+ and F_- are set equal to $-f$. Then,

$$\begin{aligned} Z_2^* &\leq \eta V_+ - \eta V_- + (p_+ + \eta)f + (p_- - \eta)f + f(1 - p_+ - p_-) \\ &= \eta V_+ - \eta V_- + f \\ &\leq \eta(\overline{V} - f) + f \\ &= \eta\overline{V} + (1-\eta)f. \end{aligned} \tag{4.136}$$

We have set V_+ equal to its maximum possible value of $\overline{V} - f$ and $V_- = 0$.

Case 4: $V_+ \leq \overline{V} - f,\ V_- \geq \overline{V} - f$
It is optimal to set $F_+ = -f$ and $F_- = V_- - V_+ + F_+$, where the latter is the smallest value of F_- permitted from the constraint $V_+ - F_+ \geq V_- - F_-$. Then,

$$\begin{aligned} Z_2^* &\leq \eta V_+ - \eta V_- + (p_+ + \eta)f - (p_- - \eta)(V_- - V_+ - f) + f(1 - p_+ - p_-) \\ &= p_-(V_+ - V_-) + f \\ &\leq f, \end{aligned} \tag{4.137}$$

where we have used that $V_+ - V_-$ is nonpositive by definition of the case. The maximum bounds are achieved by the second and third cases. ■

An example of a tight solution (i.e. satisfying Lemma 4.7 with equality) using only three states is shown below.

	V	F	P	D
s_0	0	$-f$	η	$-\eta$
s_1	0	$-f$	$1-\eta$	0
s_2	$\overline{V}$	0	0	η

The solution provides an intuitive understanding of the bound. Consider an adversary who wishes to construct an approximate value as large as possible for a state with zero value. The adversary has a total probability weight of 2η that may be added/subtracted from various state probabilities in the following stage. To make the approximate state value large, the adversary adds weight η to the state s_2 with maximum value, yielding an objective increase of $\eta\overline{V}$, and subtracts η weight from the minimum value state s_0, which has zero value. Since adding η weight to $\overline{V}$ leaves at most $(1-\eta)$ remaining weight for the estimated distribution, this weight is associated with state s_1, as it carries maximum (negative) estimation error. This solution is used as a building block for the tight example shown in the next subsection.

Returning to our original analysis, we have shown using (4.114), (4.115), Lemma 4.6, and Lemma 4.7 that

$$Z_1^* \leq Z_2^* \leq \eta\overline{V} + (1-\eta)f \tag{4.138}$$

and replacing the α term,

$$\widehat{V}_t(S_t) - V_t(S_t) \leq \alpha\eta\overline{V}_{t+1} + \alpha(1-\eta)f_{t+1}. \tag{4.139}$$

Finding a lower bound for the estimation error $\widehat{V}_t(S_t) - V_t(S_t)$ follows a similar approach.

$$\begin{aligned}
F_t(S_t) &= V_t(S_t) - \widehat{V}_t(S_t) \\
&= V_t(S_t) - \max_{x_t \in \mathcal{X}_t(S_t)} \left(R_t(S_t, x_t) + \alpha \sum_{\mathcal{S}_{t+1}} \mathbb{P}_t(S_{t+1}|S_t, x_t)\widehat{V}_{t+1}(S_{t+1}) \right. \\
&\qquad \left. + \alpha \sum_{\mathcal{S}_{t+1}} D(S_{t+1}|S_t, x_t)\widehat{V}_{t+1}(S_{t+1}) \right) \\
&= V_t(S_t) - \max_{x_t \in \mathcal{X}_t(S_t)} \left(R_t(S_t, x_t) + \alpha \sum_{\mathcal{S}_{t+1}} \mathbb{P}_t(S_{t+1}|S_t, x_t)V_{t+1}(S_{t+1}) \right. \\
&\qquad - \alpha \sum_{\mathcal{S}_{t+1}} \mathbb{P}_t(S_{t+1}|S_t, x_t)F_{t+1}(S_{t+1}) + \alpha \sum_{\mathcal{S}_{t+1}} D(S_{t+1}|S_t, x_t)V_{t+1}(S_{t+1}) \\
&\qquad \left. - \alpha \sum_{\mathcal{S}_{t+1}} D(S_{t+1}|S_t, x_t)F_{t+1}(S_{t+1}) \right) \\
&\leq -\alpha \min \left(- \sum_{\mathcal{S}_{t+1}} \mathbb{P}_t(S_{t+1}|S_t, x_t)F_{t+1}(S_{t+1}) + \sum_{\mathcal{S}_{t+1}} D(S_{t+1}|S_t, x_t)V_{t+1}(S_{t+1}) \right. \\
&\qquad \left. - \sum_{\mathcal{S}_{t+1}} D(S_{t+1}|S_t, x_t)F_{t+1}(S_{t+1}) \right).
\end{aligned} \tag{4.140}$$

The third equality above follows from substitution of the estimation error for stage $t+1$, and the inequality follows from Proposition 4.2 and canceling like terms. The last minimum is taken over all probability distributions, difference vectors, and value functions. Simplifying notation and ignoring α gives the multilinear program

$$
\begin{aligned}
\underset{F,V,P,D}{\text{minimize}} \quad & Z_3 = \sum_{s\in\mathcal{S}} (-P(s)F(s) + D(s)V(s) - D(s)F(s)) \\
\text{subject to} \quad & \sum_{s\in\mathcal{S}} |D(s)| \leq 2\eta, \\
& \sum_{s\in\mathcal{S}} P(s) = 1, \\
& \sum_{s\in\mathcal{S}} D(s) = 0, \\
& 0 \leq V(s) \leq \overline{V}, && \forall s \in \mathcal{S}, \\
& 0 \leq V(s) - F(s) \leq \overline{V}, && \forall s \in \mathcal{S}, \\
& |F(s)| \leq f, && \forall s \in \mathcal{S}, \\
& 0 \leq P(s) \leq 1, && \forall s \in \mathcal{S}, \\
& 0 \leq P(s) + D(s) \leq 1, && \forall s \in \mathcal{S}.
\end{aligned}
\tag{4.141}
$$

Substituting $F(s) = -F'(s)$ and $W(s) = \overline{V} - V(s)$ for all $s \in \mathcal{S}$ gives the program

$$
\begin{aligned}
\underset{F',W,P,D}{\text{minimize}} \quad & Z_4 = \sum_{s\in\mathcal{S}} (P(s)F'(s) - D(s)W(s) + D(s)F'(s)) \\
\text{subject to} \quad & \sum_{s\in\mathcal{S}} |D(s)| \leq 2\eta, \\
& \sum_{s\in\mathcal{S}} P(s) = 1, \\
& \sum_{s\in\mathcal{S}} D(s) = 0, \\
& 0 \leq W(s) \leq \overline{V}, && \forall s \in \mathcal{S}, \\
& 0 \leq W(s) - F'(s) \leq \overline{V}, && \forall s \in \mathcal{S}, \\
& |F'(s)| \leq f, && \forall s \in \mathcal{S}, \\
& 0 \leq P(s) \leq 1, && \forall s \in \mathcal{S}, \\
& 0 \leq P(s) + D(s) \leq 1, && \forall s \in \mathcal{S},
\end{aligned}
\tag{4.142}
$$

where we have used that $\sum_s D(s)\overline{V} = 0$ in the objective function. Negating the objective function and changing from minimization to maximization makes the problem identical to (4.115) with $W(s) = V(s)$ and $F'(s) = F(s)$. Recalling the -1 coefficient in (4.140), we thus have that

$$
-Z_3^* = -Z_4^* = Z_1^* \leq \eta\overline{V} + (1-\eta)f, \tag{4.143}
$$

where the inequality follows from Lemma 4.6 and Lemma 4.7. Hence, replacing the α factor that we have omitted,

$$
V_t(S_t) - \widehat{V}_t(S_t) \leq \alpha\eta\overline{V}_{t+1} + \alpha(1-\eta)f_{t+1}. \tag{4.144}
$$

The following lemma then holds.

Lemma 4.8. *For $t = 0, \dots, T-1$, the bounds on estimation errors satisfy*

$$f_t \leq \alpha\eta\overline{V}_{t+1} + \alpha(1-\eta)f_{t+1}. \tag{4.145}$$

Proof. Summarizing the arguments we have made above, we used (4.114), (4.115), Lemma 4.6, and Lemma 4.7 to show that

$$-F_t(S_t) = \widehat{V}_t(S_t) - V_t(S_t) \leq \alpha\eta\overline{V}_{t+1} + \alpha(1-\eta)f_{t+1}. \tag{4.146}$$

Then we used (4.140) and showed the equivalences of the programs (4.141), (4.142), and (4.115). Using again Lemma 4.6 and Lemma 4.7 with the equivalences of the programs, we showed

$$F_t(S_t) = V_t(S_t) - \widehat{V}_t(S_t) \leq \alpha\eta\overline{V}_{t+1} + \alpha(1-\eta)f_{t+1}. \tag{4.147}$$

■

We now move to bounding the approximate policy error, $G_t(S_t)$, starting with the upper bound. Note again that the decision $\widehat{x}_t$ chosen by the approximate policy is given by

$$\begin{aligned}\widehat{x}_t(S_t) \quad = \quad & \operatorname*{argmax}_{x_t \in \mathcal{X}_t(S_t)} \Bigg(R_t(S_t, x_t) + \alpha \sum_{\mathcal{S}_{t+1}} \mathbb{P}_t(S_{t+1}|S_t, x_t)\widehat{V}_{t+1}(S_{t+1}) \\ & + \alpha \sum_{\mathcal{S}_{t+1}} D(S_{t+1}|S_t, x_t)\widehat{V}_{t+1}(S_{t+1}) \Bigg)\end{aligned} \tag{4.148}$$

and that the value of a state under the approximate policy is given by

$$V_t^{\widehat{\pi}}(S_t) = R_t(S_t, \widehat{x}_t) + \alpha \sum_{\mathcal{S}_{t+1}} \mathbb{P}_t(S_{t+1}|S_t, \widehat{x}_t)V_{t+1}^{\widehat{\pi}}(S_{t+1}). \tag{4.149}$$

Expanding the approximate policy error,

$$\begin{aligned}
G_t(S_t) &= \widehat{V}_t(S_t) - V_t^{\widehat{\pi}}(S_t) \\
&= R_t(S_t,\widehat{x}_t) + \alpha \sum_{\mathcal{S}_{t+1}} \mathbb{P}_t(S_{t+1}|S_t,\widehat{x}_t)\widehat{V}_{t+1}(S_{t+1}) + \alpha \sum_{\mathcal{S}_{t+1}} D(S_{t+1}|S_t,\widehat{x}_t)\widehat{V}_{t+1}(S_{t+1}) \\
&\quad - R_t(S_t,\widehat{x}_t) - \alpha \sum_{\mathcal{S}_{t+1}} \mathbb{P}_t(S_{t+1}|S_t,\widehat{x}_t)V_{t+1}^{\widehat{\pi}}(S_{t+1}) \\
&= \alpha \sum_{\mathcal{S}_{t+1}} \mathbb{P}_t(S_{t+1}|S_t,\widehat{x}_t)V_{t+1}^{\widehat{\pi}}(S_{t+1}) + \alpha \sum_{\mathcal{S}_{t+1}} \mathbb{P}_t(S_{t+1}|S_t,\widehat{x}_t)G_{t+1}(S_{t+1}) \\
&\quad + \alpha \sum_{\mathcal{S}_{t+1}} D(S_{t+1}|S_t,\widehat{x}_t)V_{t+1}^{\widehat{\pi}}(S_{t+1}) + \alpha \sum_{\mathcal{S}_{t+1}} D(S_{t+1}|S_t,\widehat{x}_t)G_{t+1}(S_{t+1}) \\
&\quad - \alpha \sum_{\mathcal{S}_{t+1}} \mathbb{P}_t(S_{t+1}|S_t,\widehat{x}_t)V_{t+1}^{\widehat{\pi}}(S_{t+1}) \\
&= \alpha \sum_{\mathcal{S}_{t+1}} \mathbb{P}_t(S_{t+1}|S_t,\widehat{x}_t)G_{t+1}(S_{t+1}) + \alpha \sum_{\mathcal{S}_{t+1}} D(S_{t+1}|S_t,\widehat{x}_t)V_{t+1}^{\widehat{\pi}}(S_{t+1}) \\
&\quad + \alpha \sum_{\mathcal{S}_{t+1}} D(S_{t+1}|S_t,\widehat{x}_t)G_{t+1}(S_{t+1}). \qquad (4.150)
\end{aligned}$$

Above, the third inequality follows from substituting the approximate policy error for stage $t+1$ and canceling the identical reward terms, and the fourth equality follows from canceling like terms. To find the maximum value of (4.150), we adopt the shorthand notation we used before along with $V'(s) = V^{\widehat{\pi}}(S_{t+1})$, $G(s) = G_{t+1}(S_{t+1})$, $g = g_{t+1}$. This gives the multilinear program

$$\begin{aligned}
\underset{G,V',P,D}{\text{maximize}} \quad & Z_5 = \sum_{s\in\mathcal{S}} (P(s)G(s) + D(s)V'(s) + D(s)G(s)) \\
\text{subject to} \quad & \sum_{s\in\mathcal{S}} |D(s)| \le 2\eta, \\
& \sum_{s\in\mathcal{S}} P(s) = 1, \\
& \sum_{s\in\mathcal{S}} D(s) = 0, \\
& 0 \le V'(s) \le \overline{V}, \qquad \forall s \in \mathcal{S}, \\
& 0 \le V'(s) + G(s) \le \overline{V}, \quad \forall s \in \mathcal{S}, \\
& |G(s)| \le g, \qquad \forall s \in \mathcal{S}, \\
& 0 \le P(s) \le 1, \qquad \forall s \in \mathcal{S}, \\
& 0 \le P(s) + D(s) \le 1, \quad \forall s \in \mathcal{S}.
\end{aligned} \qquad (4.151)$$

Substituting $G(s)$ with $-G'(s)$ gives the same program in (4.115) with $F(s) = G'(s)$ and $V(s) = V'(s)$. The bounds for $V_{t+1}(\cdot)$ are the same as the bounds for $V_{t+1}^{\widehat{\pi}}(\cdot)$ as we noted in (4.108) and (4.109). Thus, for equal bounds $f = g$,

$$Z_5^* = Z_1^*, \tag{4.152}$$

and using our bound on Z_1^* from Lemma 4.6 and Lemma 4.7,

$$Z_5^* \le \eta \overline{V}_{t+1} + (1-\eta) g_{t+1}. \tag{4.153}$$

For the lower bound of (4.150), we have the problem

$$\begin{array}{lll} \underset{G,V',P,D}{\text{minimize}} & Z_6 = \sum_{s\in\mathcal{S}} (P(s)G(s) + D(s)V'(s) + D(s)G(s)) & \\ \\ \text{subject to} & \sum_{s\in\mathcal{S}} |D(s)| \le 2\eta, & \\ & \sum_{s\in\mathcal{S}} P(s) = 1, & \\ & \sum_{s\in\mathcal{S}} D(s) = 0, & \\ & 0 \le V'(s) \le \overline{V}, & \forall s \in \mathcal{S}, \\ & 0 \le V'(s) + G(s) \le \overline{V}, & \forall s \in \mathcal{S}, \\ & |G(s)| \le g, & \forall s \in \mathcal{S}, \\ & 0 \le P(s) \le 1, & \forall s \in \mathcal{S}, \\ & 0 \le P(s) + D(s) \le 1, & \forall s \in \mathcal{S}. \end{array} \tag{4.154}$$

Again using substitution with $G'(s) = -G(s)$, we have an instance of (4.141) with $F(s) = G'(s)$ and $V(s) = V'(s)$. This gives, for equal bounds $f = g$,

$$Z_6^* = Z_3^*, \tag{4.155}$$

and using (4.143),

$$Z_6^* \ge -\eta \overline{V} - (1-\eta) g. \tag{4.156}$$

We have the following lemma.

Lemma 4.9. *For $t = 0, \ldots, T-1$, the bounds on approximate policy errors satisfy*

$$g_t \le \alpha \eta \overline{V}_{t+1} + \alpha (1-\eta) g_{t+1}. \tag{4.157}$$

Proof. We used (4.150), the equivalence of (4.151) and (4.115), Lemma 4.6, and Lemma 4.7 to show that

$$G_t(S_t) = \widehat{V}_t(S_t) - V_t^{\widehat{\pi}}(S_t) \le \alpha \eta \overline{V}_{t+1} + \alpha (1-\eta) g_{t+1}. \tag{4.158}$$

We then used (4.150), the equivalences of (4.154), (4.141), (4.115), and again Lemma 4.6 and Lemma 4.7 to show that

$$G_t(S_t) = \widehat{V}_t(S_t) - V_t^{\widehat{\pi}}(S_t) \geq -\alpha\eta\overline{V}_{t+1} - \alpha(1-\eta)g_{t+1}. \tag{4.159}$$

■

Equipped with bounds on the growth of estimation errors and approximate policy errors from Lemma 4.8 and Lemma 4.9, we can now prove the final results.

Proof of Theorem 4.3. From Lemma 4.8,

$$f_t \leq \alpha\eta\overline{V}_{t+1} + \alpha(1-\eta)f_{t+1}, \tag{4.160}$$

where $f_T = 0$. Our inductive hypothesis is

$$f_t \leq \eta \sum_{u=t+1}^{T} \alpha^{u-t}(1-\eta)^{u-t-1}\overline{V}_u, \quad t = 0, \ldots, T-1. \tag{4.161}$$

For $t = T-1$, substituting gives

$$f_{T-1} \leq \alpha\eta\overline{V}_T, \tag{4.162}$$

which is correct. Now assuming that (4.161) holds for $t = \tau$, we show that it holds for $t = \tau - 1$.

$$\begin{aligned} f_{\tau-1} &\leq \alpha\eta\overline{V}_\tau + \alpha(1-\eta)\eta \sum_{u=\tau+1}^{T} \alpha^{u-\tau}(1-\eta)^{u-\tau-1}\overline{V}_u \\ &= \alpha\eta\overline{V}_\tau + \alpha\eta \sum_{u=\tau+1}^{T} \alpha^{u-\tau}(1-\eta)^{u-\tau}\overline{V}_u \\ &= \eta\sum_{u=\tau}^{T} \alpha^{u-\tau+1}(1-\eta)^{u-\tau}\overline{V}_u. \end{aligned} \tag{4.163}$$

Since Lemma 4.9 gives the same recursive relationship, the same expression holds for g_t using $g_T = 0$:

$$g_t \leq \eta \sum_{u=t+1}^{T} \alpha^{u-t}(1-\eta)^{u-t-1}\overline{V}_u, \quad t = 0, \ldots, T-1. \tag{4.164}$$

We have

$$f_0 \leq \eta\sum_{t=1}^{T} \alpha^t(1-\eta)^{t-1}\overline{V}_t. \tag{4.165}$$

Note that

$$\begin{aligned}\overline{V}_t &= \overline{R}\sum_{u=t}^{T}\alpha^{u-t} \\ &= \frac{\overline{R}}{\alpha^t}\frac{(\alpha^t-\alpha^{T+1})}{1-\alpha} \\ &= \frac{\overline{R}}{1-\alpha}-\frac{\overline{R}\alpha^{T-t+1}}{1-\alpha}. \qquad (4.166)\end{aligned}$$

With the assumption that $\eta < 1$,

$$\begin{aligned}f_0 &\leq \frac{\eta\overline{R}}{1-\alpha}\sum_{t=1}^{T}\alpha^t(1-\eta)^{t-1}-\frac{\eta\alpha^{T+1}\overline{R}}{1-\alpha}\sum_{t=1}^{T}\alpha^t(1-\eta)^{t-1} \\ &= \frac{\eta\overline{R}}{(1-\alpha)(1-\eta)}\left(\frac{(\alpha(1-\eta))-(\alpha(1-\eta))^{T+1}}{1-\alpha(1-\eta)}\right) \\ &\quad -\frac{\eta\alpha^{T+1}\overline{R}}{(1-\alpha)(1-\eta)}\left(\frac{(1-\eta)-(1-\eta)^{T+1}}{\eta}\right) \\ &= \overline{R}\left(\frac{\alpha\eta-\alpha^{T+1}+\alpha^{T+2}(1-\eta)+\alpha^{T+1}(1-\eta)^{T+1}(1-\alpha)}{(1-\alpha)(1-\alpha(1-\eta))}\right). \qquad (4.167)\end{aligned}$$

The same expression also holds for g_0. Recalling that

$$\mathcal{L} = V_0(S_0) - V_0^{\widehat{\pi}}(S_0) = G_0(S_0) + F_0(S_0) \leq g_0 + f_0, \qquad (4.168)$$

the proof is complete. ■

Proof of Theorem 4.4. There is a bounded reward per stage, so the limit of (4.94) as $T \to \infty$ is well defined [27]. ■

Proof of Theorem 4.5. Note that

$$\overline{V}_t = (T-t+1)\overline{R}. \qquad (4.169)$$

Using the same approach as the discounted case, we have

$$\begin{aligned}
f_0 &\leq \eta \sum_{t=1}^{T} (1-\eta)^{t-1} \overline{V}_t \\
&= \eta(T+1)\overline{R} \sum_{t=1}^{T} (1-\eta)^{t-1} - \eta\overline{R} \sum_{t=1}^{T} (1-\eta)^{t-1} t \\
&= \frac{\eta(T+1)\overline{R}}{1-\eta} \left(\frac{(1-\eta) - (1-\eta)^{T+1}}{1-(1-\eta)} \right) \\
&\quad - \frac{\eta\overline{R}}{1-\eta} \left(\frac{(1-\eta) - (T+1)(1-\eta)^{T+1} + T(1-\eta)^{T+2}}{\eta^2} \right) \\
&= \frac{\overline{R}}{\eta} \left(-1 + \eta(T+1) + (1-\eta)^{T+1} \right). \qquad (4.170)
\end{aligned}$$

The same expression is valid for g_0, yielding

$$\mathcal{L} \leq \frac{2\overline{R}}{\eta} \left(-1 + \eta(T+1) + (1-\eta)^{T+1} \right). \qquad (4.171)$$

■

Proof of Corollary 4.5. Let

$$\mathcal{L}_u = \frac{2\overline{R}\alpha\eta}{(1-\alpha)(1-\alpha(1-\eta))}. \qquad (4.172)$$

We first show that $\mathcal{L}_u$ is concave in η and then find $\lim_{\eta\to 0} \frac{\partial \mathcal{L}_u}{\partial \eta}$. We have

$$\frac{\partial^2 \mathcal{L}_u}{\partial \eta^2} = \frac{4\overline{R}\alpha^2}{(1-\alpha)(1-\alpha(1-\eta))^2} \left(\frac{\alpha\eta}{(1-\alpha(1-\eta))} - 1 \right). \qquad (4.173)$$

The first term

$$\frac{4\overline{R}\alpha^2}{(1-\alpha)(1-\alpha(1-\eta))^2} \qquad (4.174)$$

is clearly nonnegative, and the fact that

$$\frac{\alpha\eta}{(1-\alpha(1-\eta))} - 1 \qquad (4.175)$$

is nonpositive follows from $1-\alpha > 0$. Next, the first derivative is

$$\frac{\partial \mathcal{L}_u}{\partial \eta} = \frac{2\overline{R}\alpha}{(1-\alpha)(1-\alpha(1-\eta))} \left(\frac{\alpha\eta}{(1-\alpha(1-\eta))} - 1 \right), \qquad (4.176)$$

and

$$\lim_{\eta \to 0} \frac{\partial \mathcal{L}_u}{\partial \eta} = \frac{2\overline{R}\alpha}{(1-\alpha)^2}. \tag{4.177}$$

■

Proof of Corollary 4.6. Let

$$\mathcal{L}_u = \frac{2\overline{R}}{\eta}\left(-1 + \eta(T+1) + (1-\eta)^{T+1}\right). \tag{4.178}$$

The second derivative is given by

$$\frac{\partial^2 \mathcal{L}_u}{\partial \eta^2} = \frac{2\overline{R}}{\eta^2}\left(\frac{-2}{\eta} + \frac{2(1-\eta)^T}{\eta} + (1-\eta)^T T + (1-\eta)^{T-1} T + (1-\eta)^{T-1}\eta T^2\right). \tag{4.179}$$

The term $\frac{2\overline{R}}{\eta^2}$ is clearly nonnegative, so it is sufficient to show that the remaining expression is nonpositive to establish concavity. We wish to show that

$$\frac{2(1-\eta)^T}{\eta} + (1-\eta)^T T + (1-\eta)^{T-1} T + (1-\eta)^{T-1}\eta T^2 \le \frac{2}{\eta}, \tag{4.180}$$

or equivalently,

$$(1-\eta) + \frac{\eta T}{2} + \frac{\eta(1-\eta)T}{2} + \frac{\eta^2 T^2}{2} \le (1-\eta)^{-(T-1)}. \tag{4.181}$$

We will determine the Taylor expansion for $(1-\eta)^{-(T-1)}$. First note that

$$\frac{\partial^k}{\partial \eta^k}(1-\eta)^{-(T-1)} = (1-\eta)^{-(T-1+k)} \prod_{u=0}^{k-1}(T-1+u), \quad k \ge 1. \tag{4.182}$$

The Taylor series around $\eta = 0$ gives

$$(1-\eta)^{-(T-1)} = 1 + \sum_{k=1}^{\infty} \frac{\eta^k}{k!} \prod_{u=0}^{k-1}(T-1+u). \tag{4.183}$$

Each term in the expansion is nonnegative, and evaluating the expression through $k = 2$ gives precisely the left hand side of (4.181). This shows that $\mathcal{L}_u$ is a concave function of η.

The first derivative of the loss bound is

$$\frac{\partial \mathcal{L}_u}{\partial \eta} = \frac{2\overline{R}}{\eta^2}\left(1 - (1-\eta)^T - (1-\eta)^T \eta T\right). \tag{4.184}$$

Taking the limit as $\eta \to 0$ gives

$$\lim_{\eta \to 0} \frac{\partial \mathcal{L}_u}{\partial \eta} = \overline{R}T(T+1). \tag{4.185}$$

■

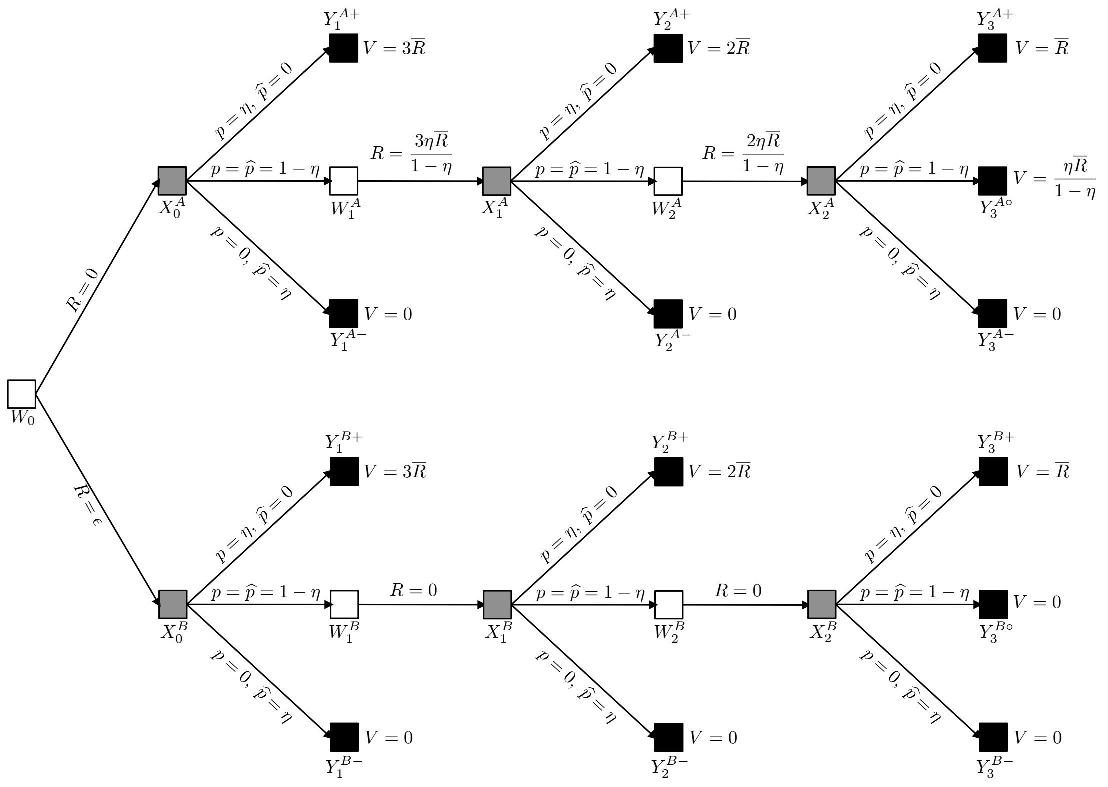


Figure 4.9. Tight example for undiscounted case with $T = 3$. White nodes are pre-decision states, gray nodes are post-decision states, and black nodes are terminal states. Terminal states with nonzero values do not violate the bound on reward $\overline{R}$ because they can be interpreted as paths where a reward $\overline{R}$ is received at each following stage.

4.5.3 Tight Example

We show a tight example for the undiscounted case assuming that $T \leq \frac{(1-\eta)}{\eta}$. Tight examples for the discounted cases can be derived using similar structure. We introduce a few terms to describe the example. A *post-decision* state $S_t^x = (S_t, x_t)$ is defined by a state S_t and a feasible decision $x_t \in \mathcal{X}_t(S_t)$ for the state [122]. Post-decision states are essentially what we have until now called decisions, but we refer to them here as "states" since we assign values and approximate values to them as we would with ordinary states. We refer to value functions and approximate value functions of post-decision states with the notation $V_t^x(\cdot)$ and $\widehat{V}_t^x(\cdot)$, respectively. A *pre-decision* state is what we have until now referred to as a state.

The example is described with a directed tree structure. Nodes in the tree correspond to pre-decision states (denoted by W), post-decision states (denoted by X)[3], and

[3]The post-decision states for the example should not be confused with the our notation for a policy, which also uses the symbol X.

absorbing and terminal states (denoted by Y). Arcs correspond to transitions between sequential states that occur by decision or randomly. The example for $T = 3$ is shown in Fig. 4.9. The only decision takes place at $t = 0$ from the unique pre-decision state W_0. The decision is between two post-decision states X_0^A and X_0^B corresponding to path A and path B, respectively. Path A, which has a large expected reward, is defined as the set of all node descendants of X_0^A. Path B, which has a negligible expected reward, is defined as the set of all node descendants of X_0^B.

For $t = 1, \dots, T-1$, there are two pre-decision and two post-decision states labeled $W_t^A,\ W_t^B,\ X_t^A,\ X_t^B$. The notation W_t^A denotes the pre-decision state at stage t on path A, for example. For $t = 1, \dots, T$, there are four absorbing states, two for each path, which are denoted by $Y_t^{A+},\ Y_t^{A-},\ Y_t^{B+},\ Y_t^{B-}$. For $t = T$, these four absorbing states are terminal states, and there are two additional absorbing/terminal states $Y_T^{A\circ},\ Y_T^{B\circ}$. The outgoing arcs for nodes are given as follows, where $\delta^+(S)$ denotes the set of nodes connected to node S with an outgoing arc.

$$\delta^+(W_0) = \{X_0^A,\ X_0^B\}, \tag{4.186}$$

$$\delta^+(X_t^Q) = \{W_{t+1}^Q,\ Y_{t+1}^{Q+},\ Y_{t+1}^{Q-}\},\quad Q = A,\ B,\quad t = 0, \dots, T-2, \tag{4.187}$$

$$\delta^+(W_t^Q) = \{X_t^Q\},\quad Q = A,\ B,\quad t = 1, \dots, T-1, \tag{4.188}$$

$$\delta^+(X_T^Q) = \{Y_T^{Q+},\ Y_T^{Q\circ}\ , Y_T^{Q-}\},\quad Q = A,\ B. \tag{4.189}$$

Like our previous examples, arc weights exiting pre-decision states correspond to rewards, and arc weights exiting post-decision states correspond to probabilities. The reward function for the starting pre-decision state W_0 is

$$R_0(W_0, \cdot) = \begin{cases} 0, & \text{choose } X_0^A, \\ \epsilon, & \text{choose } X_0^B, \end{cases} \tag{4.190}$$

where $0 < \epsilon \ll \overline{R}$. Since all other pre-decision states have only one decision (one exiting arc), we can simply specify the corresponding rewards as a function of the state for $t = 1, \dots, T-1$:

$$R_t(W_t^Q) = \begin{cases} \frac{(T-t+1)\eta\overline{R}}{1-\eta}, & Q = A, \\ 0, & Q = B. \end{cases} \tag{4.191}$$

With the assumption that $T \leq \frac{(1-\eta)}{\eta}$, these rewards do not violate the reward bound $\overline{R}$. Absorbing state values for $t = 1, \dots, T$ are given by

$$V_t(S_t) = \begin{cases} (T-t+1)\overline{R}, & S_t = Y_t^{Q+}, \\ 0, & S_t = Y_t^{Q-}, \end{cases} \quad Q = A,\ B,$$

$$V_T(Y_T^{A\circ}) = \frac{\eta\overline{R}}{(1-\eta)}, \quad V_T(Y_T^{B\circ}) = 0. \tag{4.192}$$

Note that the absorbing states with nonzero values do not violate the upper bound on rewards; they can be interpreted as paths where the maximum reward is obtained at each following stage for the remainder of the horizon. Transition and estimated transition probabilities are given by

$$\mathbb{P}(S_{t+1}|X_0^A) = \begin{cases} \eta, & S_{t+1} = Y_{t+1}^{A+}, \\ (1-\eta), & S_{t+1} = W_{t+1}^{A}, \ Y_T^{A\circ}, \\ 0, & S_{t+1} = Y_{t+1}^{A-}, \end{cases} \tag{4.193}$$

$$\mathbb{P}(S_{t+1}|X_0^B) = \begin{cases} 0, & S_{t+1} = Y_{t+1}^{B+}, \\ (1-\eta), & S_{t+1} = W_{t+1}^{B}, \ Y_T^{B\circ}, \\ \eta, & S_{t+1} = Y_{t+1}^{B-}, \end{cases} \tag{4.194}$$

$$\widehat{\mathbb{P}}(S_{t+1}|X_0^A) = \begin{cases} 0, & S_{t+1} = Y_{t+1}^{A+}, \\ (1-\eta), & S_{t+1} = W_{t+1}^{A}, \ Y_T^{A\circ}, \\ \eta, & S_{t+1} = Y_{t+1}^{A-}, \end{cases} \tag{4.195}$$

$$\widehat{\mathbb{P}}(S_{t+1}|X_0^B) = \begin{cases} \eta, & S_{t+1} = Y_{t+1}^{B+}, \\ (1-\eta), & S_{t+1} = W_{t+1}^{B}, \ Y_T^{B\circ}, \\ 0, & S_{t+1} = Y_{t+1}^{B-}, \end{cases} \tag{4.196}$$

for $t = 0, \dots, T-1$. Since we have fully specified the construction of the tight example, we now prove that it is indeed tight.

Lemma 4.10. *Based on the construction of the tight example* (4.186) - (4.196)*, the true and estimated values of the post-decision states at the starting stage satisfy*

$$V_0^x(X_0^A) = 2\eta\sum_{u=1}^{T}(1-\eta)^{u-1}\overline{V}_u, \tag{4.197}$$

$$\widehat{V}_0^x(X_0^A) = \eta\sum_{u=1}^{T}(1-\eta)^{u-1}\overline{V}_u, \tag{4.198}$$

$$\widehat{V}_0^x(X_0^B) = \eta\sum_{u=1}^{T}(1-\eta)^{u-1}\overline{V}_u, \tag{4.199}$$

$$V_0^x(X_0^B) = 0. \tag{4.200}$$

Proof. Starting with the true value for path A, our inductive hypothesis is

$$V_t^x(X_t^A) = 2\eta\sum_{u=t+1}^{T}(1-\eta)^{u-t-1}\overline{V}_u, \quad t = 0, \dots, T-1. \tag{4.201}$$

Plugging in $t = T - 1$ gives

$$V^x_{T-1}(X^A_{T-1}) = 2\eta \overline{V}_T = 2\eta \overline{R}, \tag{4.202}$$

which matches the construction of the tight example. We now verify that the hypothesis holds for $t = \tau - 1$ given that it holds for $t = \tau$. We have

$$\begin{aligned} V^x_{\tau-1}(X^A_{\tau-1}) &= \eta \overline{V}_\tau + (1-\eta)\left(\frac{(T-\tau+1)\eta\overline{R}}{(1-\eta)} + 2\eta \sum_{u=\tau+1}^{T} (1-\eta)^{u-\tau-1}\overline{V}_u\right) \\ &= \eta \overline{V}_\tau + (T-\tau+1)\eta\overline{R} + 2\eta \sum_{u=\tau+1}^{T} (1-\eta)^{u-\tau}\overline{V}_u \\ &= 2\eta \overline{V}_\tau + 2\eta \sum_{u=\tau+1}^{T} (1-\eta)^{u-\tau}\overline{V}_u \\ &= 2\eta \sum_{u=\tau}^{T} (1-\eta)^{u-\tau}\overline{V}_u. \end{aligned} \tag{4.203}$$

On the first line, we used the definitions for the true transition probabilities (4.193), reward function (4.191), and terminal state values (4.192). The third line follows by noting that $\overline{V}_\tau = (T - \tau + 1)$. For the estimated state value of path A, the inductive hypothesis is

$$\widehat{V}^x_t(X^A_t) = \eta \sum_{u=t+1}^{T} (1-\eta)^{u-t-1}\overline{V}_u, \quad t = 0, \ldots, T-1. \tag{4.204}$$

For $t = T - 1$,

$$\widehat{V}^x_{T-1}(X^A_{T-1}) = \eta \overline{V}_T = \eta \overline{R}. \tag{4.205}$$

Now assuming that it holds for $t = \tau$,

$$\begin{aligned} \widehat{V}^x_{\tau-1}(X^A_{\tau-1}) &= \eta \cdot 0 + (1-\eta)\left(\frac{(T-\tau+1)\eta\overline{R}}{(1-\eta)} + \eta \sum_{u=\tau+1}^{T} (1-\eta)^{u-\tau-1}\overline{V}_u\right) \\ &= (T-\tau+1)\eta\overline{R} + \eta \sum_{u=\tau+1}^{T} (1-\eta)^{u-\tau}\overline{V}_u \\ &= \eta \overline{V}_\tau + \eta \sum_{u=\tau+1}^{T} (1-\eta)^{u-\tau}\overline{V}_u \\ &= \eta \sum_{u=\tau}^{T} (1-\eta)^{u-\tau}\overline{V}_u. \end{aligned} \tag{4.206}$$

Moving to the estimated value for path B, the inductive hypothesis is again

$$\widehat{V}_t^x(X_t^B) = \eta \sum_{u=t+1}^{T} (1-\eta)^{u-t-1}\overline{V}_u, \quad t = 0, \ldots, T-1, \tag{4.207}$$

which satisfies $\widehat{V}_{T-1}^x(X_{T-1}^B) = \eta\overline{V}_T = \eta\overline{R}$. Assuming it is valid for $t = \tau$,

$$\begin{aligned}\widehat{V}_{\tau-1}^x(X_{\tau-1}^B) &= \eta\overline{V}_\tau + (1-\eta)\eta \sum_{u=\tau+1}^{T} (1-\eta)^{u-\tau-1}\overline{V}_u \\ &= \eta\overline{V}_\tau + \eta \sum_{u=\tau+1}^{T} (1-\eta)^{u-\tau}\overline{V}_u \\ &= \eta \sum_{u=\tau}^{T} (1-\eta)^{u-\tau}\overline{V}_u. \end{aligned} \tag{4.208}$$

Finally, it can be verified by inspection that $V_0^x(X_0^B) = 0$. ■

Theorem 4.6. *The total loss of the tight example* (4.186) - (4.196) *for* $0 < \eta < 1$ *is*

$$\mathcal{L} = \frac{2\overline{R}}{\eta}\left(-1 + \eta(T+1) + (1-\eta)^{T+1}\right) - \epsilon. \tag{4.209}$$

Proof. Consider Lemma 4.10 and the rewards for the initial stage (4.190). The optimal choice for the problem is to choose path A and obtain the expected value given by (4.197),

$$V_0^x(X_0^A) = 2\eta \sum_{u=1}^{T} (1-\eta)^{u-1}\overline{V}_u. \tag{4.210}$$

The estimated post-decision values for both paths are equal to $V_0^x(X_0^A)/2$. Since the immediate reward for choosing path B is larger by ϵ, however, the approximate policy chooses path B and realizes value ϵ. The expression (4.210) for $V_0^x(X_0^A)$ evaluates to the upper bound on the loss using the development in (4.170). ■

4.6 Discussion

We have presented a general loss bound for Markov decision processes solved using exact dynamic programming with estimated transition probabilities. The bound applies to discounted finite and infinite horizon scenarios as well as undiscounted finite horizon scenarios. We also presented a tight example that can be generalized to these scenarios.

One major drawback of our uncertainty model is that it is relatively pessimistic: the total variation description of probability uncertainty leads to a boundary solution

(Lemma 4.4) where probability weight is placed on paths with maximum and zero future value. This behavior is more adversarial than one might expect from natural uncertainty occurring in a system.

This motivates two questions for future research. First, is there a class of problems that naturally prohibits the existence of states with maximum future value (i.e. where the upper bound on reward is obtained at each future stage) so that an improved bound for this class of problems is closer to real-life systems? Next, what bounds can be obtained for other characterizations of probability uncertainty? Our result applies to some other measures of probability distance simply by known identities relating the measures. For instance, Pinsker's inequality gives, for an upper bound on the Kullback–Leibler (KL) divergence between two distributions, an upper bound on the total variation distance between two distributions [45]. Letting transition probabilities themselves be random variables allows for interesting modeling options. Probabilities could have additive Gaussian noise, for example. However, this average-case analog of our work here is likely to be difficult to analyze. While our worst-case analysis requires bounds on rewards, the average-case analysis would probably require assumptions on the *distribution* of rewards, which is rather constricting.

Understanding the effects of uncertainty with conventional dynamic programming is helpful, but approximate dynamic programming methods are used more frequently for large systems. It would be interesting to see which approximation methods are more robust in the presence of uncertain transition probabilities. Some approximation methods, for example, may be better able to capture correlations in uncertain transition probabilities across stages.

Chapter 5

Randomized Minmax Regret for Combinatorial Optimization

MANY optimization applications involve cost coefficients that are not fully known. When information on cost coefficients is available in the form of probability distributions (e.g. from historical data or other estimates), stochastic programming is often an appropriate modeling choice [37, 131]. In other cases, costs may only be known to be contained in intervals (i.e. each cost has a known lower and upper bound) or to be a member of a finite set of scenarios, and one is more interested in worst-case performance. Robust optimization formulations are desirable here as they employ a minmax-type objective and do not require knowledge of cost distributions [31, 84, 93].

In a general robust optimization problem with cost uncertainty, one must select a set of items from some feasible *solution set* such that item costs are contained in some *uncertainty set*. The basic problem of selecting an optimal solution from the solution set when costs are known is referred to as the *nominal problem*. When only the uncertainty set is known, the goal under the *minmax* criterion (also referred to as absolute robustness) is to select a solution that gives the best upper bound on objective cost over all possible costs from the uncertainty set [145] (this is assuming that the nominal problem is a minimization problem). That is, one must select the solution that, when item costs are chosen to maximize the cost of the selected solution, is minimum. Under the *minmax regret* criterion (sometimes called the robust deviation model), the goal is instead to select the solution that minimizes the maximum possible regret, defined as the difference between the cost of the selected solution and the optimal solution [127].

The motivation for choosing the minmax regret criterion over the minmax criterion is shown by the following example; the example involves a nominal problem with a maximizing objective, so the criterion analogous to minmax is actually *maxmin* here. Consider the choice between two items, the values of which are unknown but are contained within known bounds: the first item has a value $v_1 \in [0, 100]$ while the second item as a value $v_2 \in [1, 2]$. The maxmin solution is to choose the second item since it guarantees a value of at least 1 as opposed to 0. This choice, however, ignores the potential value of up to 100 that could be obtained in choosing the first item. The minmax regret solution is to choose the first item since this choice has a maximum regret of 2 (the value of the maximum possible difference $v_2 - v_1$), whereas choosing the second item has a maximum regret of 99 (the value of the maximum possible difference $v_1 - v_2$). The minmax regret criterion thus seeks to exploit potential gains in value, while the minmax/maxmin criterion is more risk averse.

This example, as well as other robust optimization problems under cost uncertainty, can be interpreted as a two-stage game played between an optimizing player and an adversary. In the first stage, the optimizing player selects some solution deterministically from the solution set. In the second stage, the adversary observes the solution chosen by the optimizing player and selects costs from the uncertainty set. The adversary selects costs either to give the worst objective value for the optimizing player (under the minmax/maxmin criterion) or to maximize the regret of the optimizing player (under the minmax regret criterion). We refer to this model as the *deterministic* model. A slightly different model is to allow the optimizing player to select a probability distribution over solutions and require the adversary to select costs with knowledge of the optimizing player's distribution, but not its realization. We refer to this model as the *randomized* model, and this chapter focuses on the randomized model under the minmax regret criterion.

To illustrate the randomized model, we again consider our item selection example but with value bounds $v_1 \in [0, 100]$ and $v_2 \in [10, 60]$. The deterministic minmax regret solution is still to choose the first item, guaranteeing a minmax regret of 60, while the maxmin solution is to choose the second item. The randomized minmax regret solution, however, is to choose the first item with probability 9/15 and the second item with probability 6/15. For any distribution over values within bounds chosen by the adversary, assuming it is chosen without knowledge of the selected item, the resulting expected regret is no greater than 36. The randomized model, from the perspective of the optimizing player, is thus a game played against a less powerful adversary. This model is more realistic for many applications where the adversary is nature.

Remarkably, we show that under this randomized model, the minmax regret version of any polynomial solvable 0–1 integer linear programming problem is polynomial solvable. This holds true for both interval and discrete scenario representations of uncertainty[1]. Our crucial observation is that the randomized model corresponds to the linear programming relaxation of the mixed integer program for the deterministic model. This leads to some useful insights. First, the minmax expected regret in the randomized model is upper bounded by the minmax regret in the deterministic model, a property that was satisfied in our example. Next, the linear program formulation can be used to create an approximation algorithm for the deterministic problem. We show that existing approximation algorithms for deterministic minmax regret problems, which have been proved using combinatorial arguments, can in fact be derived using primal-dual

[1]We have illustrated *interval uncertainty* with the example item selection problem. The same problem under *discrete scenario uncertainty* might have, for example, three scenarios corresponding to costs $(v_1, v_2) = (0, 100), (10, 60), (20, 30)$.

methods [6, 85]. Our analysis here leads to lower bounds on randomized minmax regret with respect to the deterministic minmax regret, effectively stating limits on the power of using randomization.

Given that the randomized model makes many minmax regret problems polynomial solvable for interval uncertainty and discrete scenario uncertainty, it is natural to ask if polynomial solvability remains in the presence of slightly more elaborate uncertainty sets. We show that for general convex uncertainty sets, however, both the deterministic and randomized minmax regret problems are NP-hard[2].

5.1 Related Work

One of the first studies of minmax regret from both an algorithmic and complexity perspective was that of Averbakh [14]. He looked at the minmax regret version of the simple problem of selecting k items out of n total items, where the cost of each item is uncertain, and the goal is to select the set of items with minimum total cost. For interval uncertainty, he derived a polynomial time algorithm based on interchange arguments. However, he demonstrated that for the discrete scenario representation of uncertainty, the minmax regret problem becomes NP-hard, even for the case of only two scenarios. It is interesting to contrast these results with the case of minmax regret linear programming, which as shown by Averbakh and Lebedev [16], is NP-hard for interval uncertainty but polynomial solvable for discrete scenario uncertainty.

Apart from the item selection problem, most polynomial solvable minmax regret combinatorial problems are NP-hard, both for interval and discrete scenario uncertainty. This is true for the shortest path, minimum spanning tree, assignment, and minimum s-t cut problems [4, 5, 15, 93, 150]. One exception is the minimum cut problem, the minmax regret version of which is polynomial solvable both for interval and discrete scenario uncertainty [5]. The survey paper of Aissi et al. [8] provides a comprehensive summary of results related to both minmax and minmax regret combinatorial problems. For problems that are already NP-complete, most of their minmax regret versions are Σ_2^p-complete (meaning that they lie at the second level of the polynomial hierarchy) [50, 80]. To solve minmax regret problems in practice, the book by Kasperski reviews standard mixed integer program (MIP) formulations for both interval and discrete scenario uncertainty [84].

General approximation algorithms for deterministic minmax regret are known for both types of uncertainty. Kasperski and Zieliński [85] proved a 2-approximation algorithm based on midpoint costs under interval uncertainty, and Aissi et al. [6] gave a k-approximation algorithm using mean costs under discrete scenario uncertainty, where k is the number of scenarios. Under interval uncertainty, fully polynomial time approximation schemes are known for many problems [84, 86]. For discrete scenario uncertainty, Kasperski et al. [87] looked at the minmax regret item selection problem, which models special cases of many combinatorial problems. They showed that for a non-constant

[2]For an introduction to NP-hardness and other complexity topics, see [69, 112].

number of scenarios, the problem is not approximable within any constant factor unless P=NP. If the number of scenarios is constant, fully polynomial time approximation schemes are known for some problems [7, 9].

The application of a game-theoretic model with mixed strategies to robust optimization problems was introduced by Bertsimas et al. [35]. They focused on the minmax criterion, and their analysis was motivated by adversarial models used for online optimization algorithms. As described by Ben-David et al. [23] (see also Borodin and El-Yaniv [42]), the three types of adversaries are the *oblivious adversary*, the *adaptive online adversary*, and the *adaptive offline adversary*. The adaptive offline adversary is the analog of the conventional deterministic minmax regret problem, while the adaptive online adversary corresponds to the randomized model. The analog of the oblivious adversary, which we do not study, is the model where the adversary first selects costs, and the optimizing player then selects the solution after viewing these costs.

For the randomized (corresponding to the adaptive online adversary) minmax problem, Bertsimas et al. [35] showed that if it is possible to optimize over both the solution set and the uncertainty set in polynomial time, then an optimal mixed strategy solution can be computed in polynomial time, and that the expected cost under the randomized model is no greater than the cost for the deterministic model. This holds despite the fact that solving the minmax version of many polynomial solvable problems is NP-hard for the deterministic case [32]. They also gave lower bounds on the improvement gained from randomization for various uncertainty sets. Our work is similar to theirs, but we focus on the minmax regret criterion instead of the minmax criterion.

Another line of research that is related to ours is in security applications, where the adversarial model is realistically motivated. Korzhyk et al. [92] considered assignment-type problems where defensive resources, such as security guards, must be assigned to valued targets. They followed a Stackelberg model where the defending player has the power to commit to a mixed strategy; the attacker then observes this mixed strategy (though not the realization) and decides which targets to attack. They used linear programming formulations along with the Birkhoff-von Neumann theorem to find polynomial-sized optimal mixed strategies. It is also worth mentioning the work of Bertismas et al. on randomized strategies for network interdiction [36].

5.2 Definitions

We consider a general combinatorial optimization problem where we are given a set of n items $E = \{e_1, e_2, \ldots, e_n\}$ and a set $\mathcal{F}$ of feasible subsets of E. Each item $e \in E$ has a cost $c_e \in \mathbb{R}$. Given the vector $c = (c_1, \ldots, c_n)$, the goal of the optimization problem is to select the feasible subset of items that minimizes the total cost; we refer to this as the *nominal problem*:

$$F^*(c) := \min_{T \in \mathcal{F}} \sum_{e \in T} c_e. \tag{5.1}$$

Let $x = (x_1, \ldots, x_n)$ be a characteristic vector for some set $T \in \mathcal{F}$, so that $x_e = 1$ if $e \in T$ and $x_e = 0$ otherwise. Also let $\mathcal{X} \subseteq \{0,1\}^n$ denote the set of all characteristic vectors corresponding to feasible sets $T \in \mathcal{F}$. We assume that $\mathcal{X}$ is described in size m (e.g. with m linear inequalities). We can equivalently write the nominal problem with a linear objective function:

$$F^*(c) = \min_{x \in \mathcal{X}} \sum_{e \in E} c_e x_e. \tag{5.2}$$

Throughout the chapter, we will use both set notation and characteristic vectors for ease of presentation.

We will review the conventional regret definitions for the deterministic minmax regret framework and then present the analogous definitions for our randomized model. For some cost vector $c \in \mathcal{C}$, the deterministic cost of a solution $T \in \mathcal{F}$ is

$$F(T, c) := \sum_{e \in T} c_e. \tag{5.3}$$

The regret of a solution T under some cost vector c is the difference between the cost of the solution and the optimal cost:

$$R(T, c) := F(T, c) - F^*(c). \tag{5.4}$$

The *maximum regret problem* for a solution T is

$$R_{\max}(T) := \max_{c \in \mathcal{C}} R(T, c) = \max_{c \in \mathcal{C}} \left(F(T, c) - F^*(c)\right). \tag{5.5}$$

The *deterministic minmax regret problem* is then

$$Z_{\mathrm{D}} := \min_{T \in \mathcal{F}} R_{\max}(T) = \min_{T \in \mathcal{F}} \max_{c \in \mathcal{C}} (F(T, c) - F^*(c)). \tag{5.6}$$

In the remainder of the chapter, we will frequently abuse the notation $F(\cdot, c)$, $R(\cdot, c)$ and $R_{\max}(\cdot)$ by replacing set arguments with vectors (e.g. $F(x, c)$ in place of $F(T, c)$), but we will follow the convention of using capital letters for sets and lowercase letters for vectors.

We now move to the randomized framework, where the optimizing player selects a distribution over solutions and the adversary selects a distribution over costs. Starting with the optimizing player, for some set $T \in \mathcal{F}$, let y_T denote the probability that the optimizing player selects set T. Let $y = (y_T)_{T \in \mathcal{F}}$ be the vector of length $|\mathcal{F}|$ specifying the set selection distribution; we will refer to y simply as a solution. Define the feasible region for y as

$$\mathcal{Y} := \{y | y \geq \mathbf{0}, \mathbf{1}^\top y = 1\}, \tag{5.7}$$

where the notation $\mathbf{0}$ and $\mathbf{1}$ indicates a full vector of zeros and ones, respectively, and $\top$ denotes the transpose operation. We similarly define a distribution over costs for the adversary. The set $\mathcal{C}$ may in general be infinite, but we will only consider strategies with finite support; for now we will assume that such strategies are sufficient. Thus consider a finite set $\mathcal{C}_f \subseteq \mathcal{C}$, and for some $c \in \mathcal{C}_f$, let w_c denote the probability that the adversary selects costs c. Then let $w = (w_c)_{c \in \mathcal{C}_f}$ and define the feasible region

$$\mathcal{W} := \{w | w \geq \mathbf{0}, \mathbf{1}^\top w = 1\}. \tag{5.8}$$

We are interested in succinct descriptions of strategies for both players. We define for the optimizing player a *mixed strategy encoding* $\mathcal{M} = (\Theta, Y)$ as a set of deterministic solutions $\Theta = \{T_i \in \mathcal{F} \mid i = 1, \ldots, \mu\}$ that should be selected with nonzero probability and the corresponding probabilities $Y = \{y_{T_i} \in [0,1] \mid i = 1, \ldots, \mu\}$ that satisfy $\sum_{i=1}^{\mu} y_{T_i} = 1$. Here μ is the support size of the mixed strategy (i.e., the number of deterministic solutions with nonzero probability). Likewise, define an adversarial mixed strategy encoding $\mathcal{L} = (C, W)$ as a set of costs $C = \{c^j \in \mathcal{C}_f \mid j = 1, \ldots, \eta\}$ to be selected with corresponding probabilities $W = \{w_{c^j} \in [0,1] \mid j = 1, \ldots, \eta\}$ satisfying $\sum_{j=1}^{\eta} w_{c^j} = 1$.

The expected regret under y and w is simply

$$\overline{R}(y, w) := \sum_{T \in \mathcal{F}} \sum_{c \in \mathcal{C}_f} y_T w_c R(T, c) = \sum_{T \in \mathcal{F}} \sum_{c \in \mathcal{C}_f} y_T w_c (F(T, c) - F^*(c)). \tag{5.9}$$

For a given y, the *maximum expected regret problem* is

$$\begin{aligned} \overline{R}_{\max}(y) \quad &:= \quad \max_{w \in \mathcal{W}} \sum_{c \in \mathcal{C}_f} w_c \sum_{T \in \mathcal{F}} y_T R(T, c) \\ &= \quad \max_{c \in \mathcal{C}_f} \sum_{T \in \mathcal{F}} y_T R(T, c). \end{aligned} \tag{5.10}$$

The above equality follows using the standard observation used in game theory: the optimization of $w \in \mathcal{W}$ is maximization of the function $G(y, c) = \sum_{T \in \mathcal{F}} y_T R(T, c)$ over the convex hull of $\mathcal{C}_f$, which is equivalent to optimizing over $\mathcal{C}_f$ itself. The minmax expected regret problem, which we refer to as the *randomized minmax regret problem*, is

$$Z_{\mathrm{R}} := \min_{y \in \mathcal{Y}} \overline{R}_{\max}(y) = \min_{y \in \mathcal{Y}} \max_{c \in \mathcal{C}} \left(\sum_{T \in \mathcal{F}} y_T (F(T, c) - F^*(c)) \right), \tag{5.11}$$

where we have replaced $\mathcal{C}_f$ with $\mathcal{C}$ under the assumption that $\mathcal{C}_f$ contains the maximizing cost vector.

The above minmax expected regret problem is the problem faced by the optimizing player. The adversary, however, is interested in solving the maxmin expected regret problem, defined as follows. First, the *minimum expected regret problem* for a given w

is

$$\begin{aligned}\overline{R}_{\min}(w) &:= \min_{y\in\mathcal{Y}} \sum_{T\in\mathcal{F}} y_T \sum_{c\in\mathcal{C}_f} w_c R(T,c) \\ &= \min_{T\in\mathcal{F}} \sum_{c\in\mathcal{C}_f} w_c R(T,c), \end{aligned} \tag{5.12}$$

where we have once again used the fact that optimizing over the convex hull of the set of solutions is equivalent to optimizing over the set of solutions. The *adversarial randomized maxmin regret problem* is

$$Z_{\text{AR}} := \max_{w\in\mathcal{W}} \overline{R}_{\min}(w) = \max_{w\in\mathcal{W}} \min_{T\in\mathcal{F}} \left(\sum_{c\in\mathcal{C}_f} w_c(F(T,c) - F^*(c)) \right). \tag{5.13}$$

It is often the case that the minmax value of the game is equal to the maxmin value; that is, $Z_{\text{R}} = Z_{\text{AR}}$. Von Neumann's minimax theorem states that this holds for two-person zero-sum games with a finite number of pure strategies [144]. In the following sections, we will show that this identity holds for discrete scenario uncertainty and interval uncertainty, following from linear programming duality.

5.3 Discrete Scenario Uncertainty

Under discrete scenario uncertainty, we are given a finite set $\mathcal{S}$ of $|\mathcal{S}| = k$ scenarios. For each $S \in \mathcal{S}$, there exists a cost vector $c^S = (c_e^S)_{e\in E}$. The adversary's mixed strategy is a probability distribution over scenarios, so we are not concerned with complications arising from infinite sets. This section is divided into three parts. We first determine computation of the optimal randomized strategy for the optimizing player, and we then look at computation of the adversary's optimal strategy. Thereafter, we use the randomized model to devise a primal-dual approximation scheme for the deterministic minmax regret problem. We restate and clarify some notation in the context of discrete scenario uncertainty throughout our development.

5.3.1 Optimizing Player

We first make some observations regarding the deterministic minmax regret problem that will be helpful in making comparisons with the randomized model. Under discrete scenario uncertainty, the deterministic maximum regret problem is

$$R_{\max}(T) = \max_{S\in\mathcal{S}} R(T, c^S) = \max_{S\in\mathcal{S}} \left(F(T,c^S) - F^*(c^S)\right). \tag{5.14}$$

The deterministic minmax regret problem is

$$Z_{\text{D}} = \min_{T\in\mathcal{F}} R_{\max}(T) = \min_{T\in\mathcal{F}} \max_{S\in\mathcal{S}} (F(T,c^S) - F^*(c^S)). \tag{5.15}$$

Lemma 5.1. *For discrete scenario uncertainty, the deterministic minmax regret problem is equivalent to the following mixed integer program.*

$$\begin{aligned} Z_{\mathrm{D}} \;=\; \min_{z,x} \quad & z \\ \text{s.t.} \quad & \sum_{e\in E} c_e^S x_e - F^*(c^S) \le z, \qquad \forall S \in \mathcal{S}, \qquad (5.16)\\ & x \in \mathcal{X}, \\ & z \text{ free.} \end{aligned}$$

Proof. Slightly abusing the notation for maximum regret, we have with vector notation

$$R_{\max}(x) = \max_{S\in\mathcal{S}} \left(\sum_{e\in E} c_e^S x_e - F^*(c^S) \right). \tag{5.17}$$

The program then follows by definition of the maximum. ∎

For the randomized model, recall that the optimizing player's distribution over solutions is denoted by $y = (y_T)_{T\in\mathcal{F}}$ and that $\mathcal{Y}$ denotes the set of valid probability distributions. The maximum expected regret problem is

$$\overline{R}_{\max}(y) = \max_{S\in\mathcal{S}} \sum_{T\in\mathcal{F}} y_T R(T, c^S) = \max_{S\in\mathcal{S}} \left(\sum_{T\in\mathcal{F}} y_T F(T, c^S) - F^*(c^S) \right). \tag{5.18}$$

We define the expected value of a solution for a distribution y and cost vector c^S to simplify notation:

$$\overline{F}(y, c^S) := \sum_{T\in\mathcal{F}} y_T F(T, c^S). \tag{5.19}$$

The maximum expected regret problem can then be stated as

$$\overline{R}_{\max}(y) = \max_{S\in\mathcal{S}} (\overline{F}(y, c^S) - F^*(c^S)). \tag{5.20}$$

The randomized minmax regret problem is

$$Z_{\mathrm{R}} = \min_{y\in\mathcal{Y}} \overline{R}_{\max}(y) = \min_{y\in\mathcal{Y}} \max_{S\in\mathcal{S}} (\overline{F}(y, c^S) - F^*(c^S)). \tag{5.21}$$

To solve the randomized minmax regret problem, it is possible to write a linear program analogous to the above integer program using variables y_T. This would, however, have $|\mathcal{F}|$ variables, which may grow exponentially in n. Instead, we note that for the maximum regret expected regret problem,

$$\begin{aligned} \overline{R}_{\max}(y) &= \max_{S\in\mathcal{S}} \left(\sum_{T\in\mathcal{F}} y_T \sum_{e\in T} c_e^S - F^*(c^S) \right) \\ &= \max_{S\in\mathcal{S}} \left(\sum_{e\in E} c_e^S \sum_{T\in\mathcal{F}: e\in T} y_T - F^*(c^S) \right). \qquad (5.22) \end{aligned}$$

The change in summation order motivates the substitution

$$p_e := \sum_{T \in \mathcal{F} : e \in T} y_T, \quad e \in E. \tag{5.23}$$

Let $p = (p_1, \ldots, p_n)$; we refer to this as the *marginal probability vector.* The substitution is a mapping from $\mathcal{Y}$ to the convex hull of $\mathcal{X}$. The following is the minmax regret analog of an observation made by Bertsimas et al. [35].

Lemma 5.2. *For discrete scenario uncertainty, the objective value* Z_{R} *of the randomized minmax regret problem* (5.11) *is equal to that of the problem*

$$\min_{p \in \mathrm{CH}(\mathcal{X})} \max_{S \in \mathcal{S}} \left(\sum_{e \in E} c_e^S p_e - F^*(c^S) \right), \tag{5.24}$$

where $\mathrm{CH}(\mathcal{X})$ *denotes the convex hull of* $\mathcal{X}$*.*

Proof. We use the same arguments presented in [35]. By definition of the substitution (5.23), the vector p must lie in the convex hull of $\mathcal{X}$. Carathéodory's theorem [44] states that any $p \in \mathrm{CH}(\mathcal{X})$ can be represented by a convex combination of at most $n+1$ points in $\mathcal{X}$, so there exists a surjective mapping from $\mathcal{Y}$ to $\mathrm{CH}(\mathcal{X})$. ■

Since we will use the simplified formulation given in Lemma 5.2 to solve the randomized minmax regret problem, we address the problem of recovering a vector y given a solution p. In the proof of the lemma, we have used Carathéodory's theorem, which proves the existence of such a mapping but does not give its construction. Recall that we defined for the optimizing player a *mixed strategy encoding* $\mathcal{M} = (\Theta, Y)$ as a set of deterministic solutions $\Theta = \{T_i \in \mathcal{F} \mid i = 1, \ldots, \mu\}$ that should be selected with nonzero probability and the corresponding probabilities $Y = \{y_{T_i} \in [0,1] \mid i = 1, \ldots, \mu\}$ that satisfy $\sum_{i=1}^{\mu} y_{T_i} = 1$. Here μ is the support size of the mixed strategy (i.e. the number of deterministic solutions with nonzero probability). For a given vector p, we are interested in solving the following constraint satisfaction program[3]:

$$\begin{aligned} \min_{y} \quad & 0 \\ \text{s.t.} \quad & \sum_{T \in \mathcal{F} : e \in T} y_T = p_e, \qquad \forall e \in E, \\ & \sum_{T \in \mathcal{F}} y_T = 1, \\ & y \geq \mathbf{0}. \end{aligned} \tag{5.25}$$

[3]We have written a 0 in the objective function since all objective coefficients have zero value.

Consider the dual program of (5.25), which has variables $u = (u_1, \dots, u_e)$ and w:

$$\begin{aligned}
\max_{w,u} \quad & w - \sum_{e\in E} p_e u_e \\
\text{s.t.} \quad & w - \sum_{e\in T} u_e \leq 0, \qquad \forall T \in \mathcal{F}, \\
& u, w \text{ free.}
\end{aligned} \tag{LPM}$$

Recall that the region $\mathcal{X}$ is described in size m.

Lemma 5.3. *For any given $p \in \mathrm{CH}(\mathcal{X})$, a corresponding mixed strategy encoding $\mathcal{M} = (\Theta, Y)$ of size polynomial in n can be found via the linear programming formulation (LPM). Furthermore, if the nominal problem $F^*(c)$ can be solved in time polynomial in n and m, then $\mathcal{M}$ can be found in time polynomial in n and m.*

Proof. Notice that while the primal program has an exponential number of variables and a linear number of constraints, the opposite holds true for the dual. The primal program is bounded since all objective coefficients are equal to zero and is feasible due to Carathéodory's theorem. Therefore the dual program must be feasible and bounded.

To guarantee a polynomial sized solution, note that the separation problem for the constraints in (LPM) is simply the nominal problem with costs u, so the dual program can be solved via the ellipsoid method. If the nominal problem can be solved in polynomial time, then the constraints (LPM) can be generated in polynomial time, giving a polynomial time solution for the entire dual program.

From a practical perspective, a separation oracle for the constraints in (LPM) gives an efficient method for performing row generation with the simplex method. Each row i generated while solving the dual problem gives a solution $T_i \in \mathcal{F}$, and its dual variable is the corresponding probability y_{T_i}. ■

Using Lemma 5.2, we can now formulate a linear program to solve the randomized minmax regret problem.

$$\begin{aligned}
\min_{p,z} \quad & z \\
\text{s.t.} \quad & \sum_{e\in E} c_e^S p_e - F^*(c^S) \leq z, \qquad \forall S \in \mathcal{S}, \\
& p \in \mathrm{CH}(\mathcal{X}), \\
& z \text{ free.}
\end{aligned} \tag{LPD}$$

This leads to the important result that the randomized minmax regret problem is polynomial solvable for any polynomial solvable nominal problem. Also, the minmax expected regret is upper bounded by the minmax regret in the deterministic case. Recall that the feasible region $\mathcal{X}$ is described in size m.

```
Input: Nominal combinatorial problem, cost vectors $(c^S)_{S\in\mathcal{S}}$
Output: Optimizing player's optimal mixed strategy $\mathcal{M}^* = (\Theta^*, Y^*)$ where $\Theta^* = (T_1, \ldots, T_\mu)$ and $Y^* = (y_{T_1}, \ldots, y_{T_\mu})$
1: Solve linear program (LPD) to get probability vector $p^* = (p_1^*, \ldots, p_n^*)$.
2: Solve linear program (LPM) with $p = p^*$ to generate constraints indexed $i = 1, \ldots, \mu$. Each constraint $i$ corresponds to a set $T_i \in \mathcal{F}$ and dual variable $y_{T_i}$, indicating that $T_i$ is an element in the optimal mixed strategy and has probability $y_{T_i}$.
```

Algorithm 15. RAND-MINMAX-REGRET (discrete scenario uncertainty)

Theorem 5.1. *For discrete scenario uncertainty, if the nominal problem $F^*(c)$ can be solved in time polynomial in n and m, then the corresponding randomized minmax regret problem $\min_{y\in\mathcal{Y}} \max_{S\in\mathcal{S}}(\overline{F}(y, c^S) - F^*(c^S))$ can be solved in time polynomial in n, m, and k.*

Proof. The linear program (LPD) is

$$\min_{p,z} \quad z \tag{5.26}$$

$$\text{s.t.} \quad \sum_{e\in E} c_e^S p_e - F^*(c^S) \leq z, \qquad \forall S \in \mathcal{S}, \tag{5.27}$$

$$p \in \mathrm{CH}(\mathcal{X}), \tag{5.28}$$

$$z \text{ free.} \tag{5.29}$$

Since for all $S \in \mathcal{S}$, the value $F^*(c^S)$ is polynomial solvable, each constraint (5.27) can be enumerated in polynomial time. If we can optimize over $\mathcal{X}$ in polynomial time, then we can separate over $\mathrm{CH}(\mathcal{X})$ in polynomial time via the result of [73]. This gives the separation oracle for (5.28). ■

Corollary 5.1. *For discrete scenario uncertainty, $Z_{\mathrm{R}} \leq Z_{\mathrm{D}}$.*

Proof. The program (5.26) - (5.29) is the linear programming relaxation of (5.16). ■

5.3.2 Adversary

Moving to the perspective of the adversary under discrete scenario uncertainty, the adversary must select a mixed strategy over scenarios. The finite number of scenarios naturally requires the adversary's distribution to have finite support. Specifically, the adversary selects a distribution over costs $w = (w_S)_{S\in\mathcal{S}}$. The minimum expected regret problem for a given w is

$$\begin{aligned} \overline{R}_{\min}(w) &= \min_{T\in\mathcal{F}} \sum_{S\in\mathcal{S}} w_S R(T, c^S) \\ &= \min_{T\in\mathcal{F}} \sum_{S\in\mathcal{S}} w_S \left(F(T, c^S) - F^*(c^S)\right). \end{aligned} \tag{5.30}$$

Recall that $\mathcal{W}$ indicates valid probability distributions for w. The adversarial randomized maxmin regret problem is

$$Z_{\text{AR}} = \max_{w \in \mathcal{W}} \overline{R}_{\min}(w) = \max_{w \in \mathcal{W}} \min_{T \in \mathcal{F}} \sum_{S \in \mathcal{S}} w_S \left(F(T, c^S) - F^*(c^S) \right). \tag{5.31}$$

From the above definition, we formulate a linear program to solve the adversarial randomized maxmin regret problem:

$$\max_{w,z} \quad z \tag{5.32}$$

$$\text{s.t.} \quad \sum_{S \in \mathcal{S}} w_S(F(T, c^S) - F^*(c^S)) \geq z, \qquad \forall T \in \mathcal{F}, \tag{5.33}$$

$$w \in \mathcal{W}, \tag{5.34}$$

$$z \text{ free.} \tag{5.35}$$

The linear program has an exponential number of constraints, but the nominal problem gives a separation oracle. Since determining the adversary's optimal mixed strategy requires only solving this linear program, it is not necessary to enumerate an algorithm.

Theorem 5.2. *For discrete scenario uncertainty, if the nominal problem $F^*(c)$ can be solved in time polynomial in n and m, then the corresponding randomized adversarial maxmin regret problem $\max_{w \in \mathcal{W}} \min_{T \in \mathcal{F}} \sum_{S \in \mathcal{S}} w_S \left(F(T, c^S) - F^*(c^S) \right)$ can be solved in time polynomial in n, m, and k.*

Proof. The separation oracle for (5.33) is given by the nominal problem. First, notice that $F^*(c^S)$ for $S \in \mathcal{S}$ can be computed once at initialization and then stored for easy computation of $\sum_{S \in \mathcal{S}} w_S F^*(c^S)$ for any w. Next, we have

$$\sum_{S \in \mathcal{S}} w_S F(T, c^S) = \sum_{S \in \mathcal{S}} w_S \sum_{e \in T} c_e^S = \sum_{e \in T} \left(\sum_{S \in \mathcal{S}} w_S c_e^S \right). \tag{5.36}$$

This means that solving nominal problem with costs $d = (d_1, \ldots, d_n)$ where

$$d_e = \sum_{S \in \mathcal{S}} w_S c_e^S \tag{5.37}$$

and comparing the solution with z and $\sum_{S \in \mathcal{S}} w_S F^*(c^S)$ gives the oracle. ■

Corollary 5.2. *For discrete scenario uncertainty, $Z_{\text{R}} = Z_{\text{AR}}$.*

Proof. Using the substitution of the marginal probability vector in (5.23), it can be verified that the linear program solved by the adversary (5.32) - (5.35) is the dual of the program solved by the optimizing player (5.26) - (5.29). The result holds by strong duality. ■

5.3.3 Primal-Dual Approximation

As noted in the above corollary, the linear program solved by the adversary (5.32) - (5.35) is the dual of the program solved by the optimizing player (5.26) - (5.29). These linear programs correspond to the relaxation of the deterministic minmax regret problem and can thus be used to develop a primal-dual approximation scheme. We will refer to the program solved by the optimizing player as the primal linear program and the problem solved by the adversary as the dual linear program.

We rewrite the dual program (5.32) - (5.35) as

$$\max_{w,z} \quad z - \sum_{S\in\mathcal{S}} w_S F^*(c^S) \tag{5.38}$$

$$\text{s.t.} \quad \sum_{S\in\mathcal{S}} w_S F(T, c^S) \geq z, \qquad \forall T \in \mathcal{F}, \tag{5.39}$$

$$w \in \mathcal{W}, \tag{5.40}$$

$$z \text{ free.} \tag{5.41}$$

A simple feasible solution to this program is given first by setting $w_S = 1/k$ for each $S \in \mathcal{S}$. Using the standard approach for primal-dual algorithms [146], we then start with a sufficiently small value of z and increase it until a constraint becomes tight. The set corresponding to the tight solution is then added to the primal solution. The constraint (5.39) can be written as

$$\sum_{S\in\mathcal{S}} w_S F(T, c^S) = \sum_{S\in\mathcal{S}} w_S \sum_{e\in T} c_e^S = \sum_{e\in T} \left(\frac{1}{k} \sum_{S\in\mathcal{S}} c_e^S \right). \tag{5.42}$$

The first constraint that becomes tight corresponds to the set M that minimizes the mean costs over all scenarios,

$$M := \operatorname*{argmin}_{T\in\mathcal{F}} \sum_{e\in T} \left(\frac{1}{k} \sum_{S\in\mathcal{S}} c_e^S \right). \tag{5.43}$$

The set M, which is a complete primal feasible solution, is added to the primal problem. Additionally, we have a feasible solution to the adversarial (dual) linear program with objective value

$$\left(\frac{1}{k}\right) \sum_{S\in\mathcal{S}} \left(\sum_{e\in M} c_e^S - F^*(c^S) \right), \tag{5.44}$$

which is a lower bound on the optimal objective value Z_{R}. Using the same observations made in [6], this gives a k-approximation algorithm for the minmax regret problem. We refer to this as the MEAN-COST-APPROXIMATION algorithm, shown in Algorithm 16. The result given by the primal-dual framework is stronger than the result proved in [6] since it states that $R_{\max}(M) \leq kZ_{\mathrm{R}}$ instead of $R_{\max}(M) \leq kZ_{\mathrm{D}}$.

```
Input: Nominal combinatorial problem, cost vectors $(c^S)_{S\in\mathcal{S}}$
Output: Feasible solution $M \in \mathcal{F}$ satisfying $R_{\max}(M) \leq kZ_{\mathrm{D}}$.
1: Determine mean costs for each item: $d_e \leftarrow \frac{1}{k}\sum_{S\in\mathcal{S}} c_e^S, \quad \forall e \in E$.
2: Solve nominal problem with mean costs: $M \leftarrow \operatorname{argmin}_{T\in\mathcal{F}} \sum_{e\in T} d_e$.
```

Algorithm 16. Mean-Cost-Approximation (Aissi et al. [6])

Theorem 5.3. *For discrete scenario uncertainty, the solution to the nominal problem with mean costs is a k-approximation algorithm for the deterministic minmax regret problem.*

Proof. Using the construction above for a lower bound on Z_{R}, we have

$$\frac{Z_{\mathrm{D}}}{k} \leq \left(\frac{1}{k}\right) \max_{S\in\mathcal{S}} \left(\sum_{e\in M} c_e^S - F^*(c^S)\right) \leq \left(\frac{1}{k}\right) \sum_{S\in\mathcal{S}} \left(\sum_{e\in M} c_e^S - F^*(c^S)\right) \leq Z_{\mathrm{R}}. \quad (5.45)$$

The first inequality follows by definition of the deterministic minmax regret, the second inequality by a simple identity between the sum of a set of values and the maximum (since the regret is always nonnegative), and the third inequality from the linear program. ■

An interesting corollary of this analysis is a tight bound on the power of randomization in the minmax regret problem. For any nominal problem, moving from the deterministic model to the randomized model allows the optimizing player to reduce the expected regret by at most a factor of k. Equivalently, the *integrality gap*, defined as the largest possible ratio of the optimal objective value of a program to its optimal linear programming relaxation, is at most k. We state this with the following theorem and show that it is tight.

Theorem 5.4. *For discrete scenario uncertainty and all nominal problems,*

$$Z_{\mathrm{R}} \geq \frac{Z_{\mathrm{D}}}{k}, \quad (5.46)$$

where $k = |\mathcal{S}|$ is the number of scenarios. Equivalently, the integrality gap of the mixed integer program (5.16) *is equal to k.*

Proof. The inequality follows from the primal-dual analysis. We construct a tight example using $n = k$ items, where the goal of the problem is simply to select the single item with lowest cost. For each item, there exists a scenario where the item has cost $c_e = 1$ and all other items have costs $c_e = 0$. The deterministic minmax regret is equal to 1 for the problem. In the randomized problem, the optimizing player selects each item with probability $1/k$, and the adversary selects each scenario with probability $1/k$. The expected regret is equal to the probability that the optimizing player selects the same item to which the adversary assigns unit cost, which is equal to $1/k$. ■

5.4 Interval Uncertainty

In this section we assume that cost uncertainty is characterized by interval uncertainty, meaning that the cost of each item is independently contained within known lower and upper bounds:

$$c_e \in [c_e^-, c_e^+], \quad \forall e \in E. \tag{5.47}$$

Define the region

$$\mathcal{I} := \{c | c_e \in [c_e^-, c_e^+], e \in E\}. \tag{5.48}$$

The set $\mathcal{I}$ is in general infinite. Since we wish to use a distribution over $\mathcal{I}$ with finite support, we loosely define the set $\mathcal{I}_f$ to be some subset $\mathcal{I}_f \subset \mathcal{I}$ with finite cardinality over which a probability distribution will be defined. The exact construction of $\mathcal{I}_f$ will become clear during the analysis, but a sufficient example to consider at this point is the set of cost vectors where costs are set equal to their lower or upper bounds, $\mathcal{I}_f = \{c | c_e = c_e^- \text{ or } c_e = c_e^+, e \in E\}$.

We proceed in the same way as the last section, studying the optimal policy for the optimizing player and then the adversary, followed by a primal-dual approximation algorithm for the deterministic problem. We restate notation and definitions throughout.

5.4.1 Optimizing Player

Under interval uncertainty, we have the deterministic maximum regret problem

$$R_{\max}(T) = \max_{c \in \mathcal{I}} R(T, c) = \max_{c \in \mathcal{I}} (F(T, c) - F^*(c)) \tag{5.49}$$

and the deterministic minmax regret problem

$$Z_{\mathrm{D}} = \min_{T \in \mathcal{F}} R_{\max}(T) = \min_{T \in \mathcal{F}} \max_{c \in \mathcal{I}} (F(T, c) - F^*(c)). \tag{5.50}$$

The deterministic minmax regret problem is well studied and can be solved with a mixed integer program [84]. We use an unconventional formulation that (potentially) has an exponential number of constraints. We will ultimately show that the randomized minmax regret problem corresponds to the linear programming relaxation of this formulation.

Lemma 5.4. *For interval uncertainty, the deterministic minmax regret problem (5.6) is equivalent to the following mixed integer program.*

$$\begin{aligned} Z_{\mathrm{D}} = \min_{x,z} \quad & z \\ \text{s.t.} \quad & \sum_{e \in E \setminus T} c_e^+ x_e - \sum_{e \in T} c_e^- (1 - x_e) \le z, \qquad \forall T \in \mathcal{F}, \\ & x \in \mathcal{X}, \\ & z \text{ free.} \end{aligned} \tag{5.51}$$

Proof. From the maximum regret definition (5.49) and using vector notation instead of set notation,

$$\begin{aligned} R_{\max}(x) &= \max_{c \in \mathcal{I}} (F(x,c) - F^*(c)) \\ &= \max_{c \in \mathcal{I}} \left(\sum_{e \in E} c_e x_e - \min_{T \in \mathcal{F}} \sum_{e \in T} c_e \right) \\ &= \max_{T \in \mathcal{F}} \max_{c \in \mathcal{I}} \left(\sum_{e \in E} c_e x_e - \sum_{e \in T} c_e \right) \\ &= \max_{T \in \mathcal{F}} \max_{c \in \mathcal{I}} \left(\sum_{e \in E \setminus T} c_e x_e - \sum_{e \in T} c_e (1 - x_e) \right) \\ &= \max_{T \in \mathcal{F}} \left(\sum_{e \in E \setminus T} c_e^+ x_e - \sum_{e \in T} c_e^- (1 - x_e) \right), \end{aligned} \tag{5.52}$$

where in the third equality we have used that the expression $\sum_{e \in E} c_e x_e$ is not a function of T, and the last equality follows since $x_e \in \{0,1\}$. The program is then valid by the definition of the maximum. ■

In the randomized model, the maximum expected regret is

$$\overline{R}_{\max}(y) = \max_{c \in \mathcal{I}} \sum_{T \in \mathcal{F}} y_T R(T,c) = \max_{c \in \mathcal{I}} \left(\sum_{T \in \mathcal{F}} y_T F(T,c) - F^*(c) \right). \tag{5.53}$$

As with the discrete scenario uncertainty case, we define the expected value of a solution for a distribution y and cost vector c,

$$\overline{F}(y,c) := \sum_{T \in \mathcal{F}} y_T F(T,c), \tag{5.54}$$

so the maximum expected regret can be stated as

$$\overline{R}_{\max}(y) = \max_{c \in \mathcal{I}} (\overline{F}(y,c) - F^*(c)). \tag{5.55}$$

The randomized minmax regret problem is thus

$$Z_{\mathrm{R}} = \min_{y \in \mathcal{Y}} \overline{R}_{\max}(y) = \min_{y \in \mathcal{Y}} \max_{c \in \mathcal{I}} (\overline{F}(y,c) - F^*(c)). \tag{5.56}$$

Starting with analysis of the maximum expected regret problem (5.53), we use the same substitution that we used in the previous section. Specifically, we let

$$p_e = \sum_{U \in \mathcal{F} : e \in U} y_U, \quad e \in E \tag{5.57}$$

and define the marginal probability vector $p = (p_1, \ldots, p_n)$. Slightly abusing notation, we write $\overline{R}_{\max}(p)$ in place of $\overline{R}_{\max}(y)$ via this substitution.

Lemma 5.5. *For interval uncertainty, the maximum expected regret problem (5.53) is equivalent to the problem*

$$\overline{R}_{\max}(p) = \max_{T\in\mathcal{F}} \left(\sum_{e\in E\setminus T} c_e^+ p_e - \sum_{e\in T} c_e^- \left(1-p_e\right) \right). \tag{5.58}$$

Proof. We start with (5.53) and use the substitution of p. The analysis is nearly identical to the proof of Lemma 5.4.

$$\begin{aligned}\overline{R}_{\max}(y) &= \max_{c\in\mathcal{I}}(\overline{F}(y,c) - F^*(c))\\ &= \max_{c\in\mathcal{I}} \left(\sum_{U\in\mathcal{F}} y_U \sum_{e\in U} c_e - \min_{T\in\mathcal{F}} \left(\sum_{e\in T} c_e \right) \right)\\ &= \max_{c\in\mathcal{I}} \left(\sum_{e\in E} c_e \sum_{U\in\mathcal{F}:e\in U} y_U - \min_{T\in\mathcal{F}} \left(\sum_{e\in T} c_e \right) \right)\\ &= \max_{c\in\mathcal{I}} \left(\sum_{e\in E} c_e p_e - \min_{T\in\mathcal{F}} \left(\sum_{e\in T} c_e \right) \right). \end{aligned} \tag{5.59}$$

Now using the notation $\overline{R}_{\max}(p)$,

$$\begin{aligned}\overline{R}_{\max}(p) &= \max_{c\in\mathcal{I}} \left(\sum_{e\in E} c_e p_e - \min_{T\in\mathcal{F}} \left(\sum_{e\in T} c_e \right) \right)\\ &= \max_{c\in\mathcal{I}} \max_{T\in\mathcal{F}} \left(\sum_{e\in E} c_e p_e - \sum_{e\in T} c_e \right)\\ &= \max_{T\in\mathcal{F}} \max_{c\in\mathcal{I}} \left(\sum_{e\in E\setminus T} c_e p_e - \sum_{e\in T} c_e \left(1-p_e\right) \right), \end{aligned} \tag{5.60}$$

where in the first equality we have used the fact that the expression $\sum_{e\in E} c_e p_e$ is not a function of T, and the other equalities follow from rearranging terms. Notice in (5.60) that p_e is simply the total probability that item e is selected, so for $y \in \mathcal{Y}$, we must have $p_e \in [0,1]$. This makes it easy to see that for a given $T \in \mathcal{F}$,

$$\max_{c\in\mathcal{I}} \left(\sum_{e\in E\setminus T} c_e p_e - \sum_{e\in T} c_e \left(1-p_e\right) \right) = \sum_{e\in E\setminus T} c_e^+ p_e - \sum_{e\in T} c_e^- \left(1-p_e\right). \tag{5.61}$$

Substituting (5.61) into (5.60) then gives an optimization problem with a finite number of feasible solutions,

$$\overline{R}_{\max}(p) = \max_{T\in\mathcal{F}} \left(\sum_{e\in E\setminus T} c_e^+ p_e - \sum_{e\in T} c_e^- \left(1-p_e\right) \right), \tag{5.62}$$

which completes the proof. ■

An immediate corollary of Lemma 5.5 is that we can solve the maximum expected regret problem for a given y by enumerating all $|\mathcal{F}|$ subsets (potentially an exponential number of them) and choosing the one that maximizes the argument of (5.62). This allows the entire randomized minmax regret problem to be restated.

Lemma 5.6. *For interval uncertainty, the objective value Z_{R} of the randomized minmax regret problem* (5.11) *is equal to that of the problem*

$$\min_{p\in \mathrm{CH}(\mathcal{X})} \max_{T\in\mathcal{F}} \left(\sum_{e\in E\setminus T} c_e^+ p_e - \sum_{e\in T} c_e^- (1-p_e) \right), \tag{5.63}$$

where $\mathrm{CH}(\mathcal{X})$ *denotes the convex hull of* $\mathcal{X}$*.*

Proof. By the same argument as the proof of Lemma 5.2. ■

Using Lemma 5.6, we can now formulate a linear program to solve the randomized minmax regret problem:

$$\begin{aligned} \min_{p,z} \quad & z \\ \text{s.t.} \quad & \sum_{e\in E\setminus T} c_e^+ p_e - \sum_{e\in T} c_e^- (1-p_e) \le z, \qquad \forall T\in\mathcal{F}, \\ & p \in \mathrm{CH}(\mathcal{X}), \\ & z \text{ free.} \end{aligned} \tag{LPI}$$

While the above program may have an exponential number of constraints, it can be solved efficiently via the ellipsoid algorithm if a separation oracle is available for the constraints. This brings us to our main result and Algorithm 17.

```
Input: Nominal combinatorial problem, item cost bounds $(c_e^-, c_e^+)$, $e \in E$.
Output: Optimizing player's optimal mixed strategy $\mathcal{M}^* = (\Theta^*, Y^*)$ where $\Theta^* = (T_1, \ldots, T_\mu)$ and $Y^* = (y_{T_1}, \ldots, y_{T_\mu})$
1: Solve linear program (LPI) to get probability vector $p^* = (p_1^*, \ldots, p_n^*)$.
2: Solve linear program (LPM) with $p = p^*$ to generate constraints indexed $i = 1, \ldots, \mu$. Each constraint $i$ corresponds to a set $T_i \in \mathcal{F}$ and dual variable $y_{T_i}$, indicating that $T_i$ is an element in the optimal mixed strategy and has probability $y_{T_i}$.
```

Algorithm 17. RAND-MINMAX-REGRET (interval uncertainty)

Theorem 5.5. *For interval uncertainty, if the nominal problem $F^*(c)$ can be solved in time polynomial in n and m, then the corresponding randomized minmax regret problem $\min_{y\in\mathcal{Y}} \max_{c\in\mathcal{I}} (\overline{F}(y,c) - F^*(c))$ can be solved in time polynomial in n and m.*

Proof. The linear program (LPI) is

$$\min_{p,z} \quad z \tag{5.64}$$

$$\text{s.t.} \quad \sum_{e \in E \setminus T} c_e^+ p_e - \sum_{e \in T} c_e^- (1 - p_e) \le z, \qquad \forall T \in \mathcal{F}, \tag{5.65}$$

$$p \in \mathrm{CH}(\mathcal{X}), \tag{5.66}$$

$$z \text{ free}. \tag{5.67}$$

The separation oracle for the constraints (5.66) is given by the equivalence of optimization and separation [73]. To see the separation oracle for the constraint (5.65), we define the item cost vector $d = (d_1, \ldots, d_n)$ where

$$d_e = c_e^- + p_e(c_e^+ - c_e^-), \quad e \in E, \tag{5.68}$$

and then solve

$$z_d = \min_{T \in \mathcal{F}} \sum_{e \in T} d_e. \tag{5.69}$$

Let T_d be the set that minimizes the above expression. If $\sum_{e \in E} c_e^+ p_e - z_d \le z$, then we are guaranteed feasibility, otherwise the separating hyperplane (5.65) is generated where $T = T_d$. To see the validity of this approach, we have

$$\begin{aligned}
\sum_{e \in E} c_e^+ p_e - z_d &= \sum_{e \in E} c_e^+ p_e - \min_{T \in \mathcal{F}} \sum_{e \in T} d_e \\
&= \sum_{e \in E} c_e^+ p_e - \min_{T \in \mathcal{F}} \sum_{e \in T} (c_e^- + p_e(c_e^+ - c_e^-)) \\
&= \max_{T \in \mathcal{F}} \left(\sum_{e \in E} c_e^+ p_e - \sum_{e \in T} (c_e^- + p_e(c_e^+ - c_e^-)) \right) \\
&= \max_{T \in \mathcal{F}} \left(\sum_{e \in E \setminus T} c_e^+ p_e - \sum_{e \in T} c_e^- (1 - p_e) \right).
\end{aligned} \tag{5.70}$$

The solution to the linear program (5.64) - (5.67) is a vector p, which can then be used to find a mixed strategy y in polynomial time using Lemma 5.3. ∎

Corollary 5.3. *For interval uncertainty,* $Z_\mathrm{R} \le Z_\mathrm{D}$.

Proof. This follows simply by noting that the program (LPI) is the linear programming relaxation of (5.51). ∎

5.4.2 Adversary

The set $\mathcal{I}_f$ is necessary for describing the distribution of the adversary. The distribution over costs is $w = (w_c)_{c\in\mathcal{I}_f}$, and $\mathcal{W}$ again indicates the set of valid distributions. The minimum expected regret problem is

$$\begin{aligned} \overline{R}_{\min}(w) &= \min_{T\in\mathcal{F}} \sum_{c\in\mathcal{I}_f} w_c R(T,c) \\ &= \min_{T\in\mathcal{F}} \sum_{c\in\mathcal{I}_f} w_c \left(F(T,c) - F^*(c)\right). \end{aligned} \tag{5.71}$$

The adversarial randomized maxmin regret problem is then

$$Z_{\text{AR}} = \max_{w\in\mathcal{W}} \overline{R}_{\min}(w) = \max_{w\in\mathcal{W}} \min_{T\in\mathcal{F}} \sum_{c\in\mathcal{I}_f} w_c \left(F(T,c) - F^*(c)\right). \tag{5.72}$$

We can directly formulate a linear program for the adversarial maxmin regret problem, explicitly writing the constraints for $w \in \mathcal{W}$.

$$\max_{w,z} \quad z \tag{5.73}$$

$$\text{s.t.} \quad \sum_{c\in\mathcal{I}_f} w_c \left(\sum_{e\in T} c_e - F^*(c)\right) \geq z, \qquad \forall T \in \mathcal{F}, \tag{5.74}$$

$$\sum_{c\in\mathcal{I}_f} w_c = 1, \tag{5.75}$$

$$w \geq \mathbf{0}, \tag{5.76}$$

$$z \text{ free.} \tag{5.77}$$

Since this program has an exponential number of constraints and potentially an exponential number of variables, we consider its dual. We expect this "dual of the dual" program to be the primal linear program solved by the optimizing player; this will indeed be the case after some manipulation. The dual of (5.73) - (5.77) is

$$\min_{\alpha,\beta} \quad \beta \tag{5.78}$$

$$\text{s.t.} \quad \sum_{T\in\mathcal{F}} \alpha_T \left(\sum_{e\in T} c_e - F^*(c)\right) \leq \beta, \qquad \forall c \in \mathcal{I}_f, \tag{5.79}$$

$$\sum_{T\in\mathcal{F}} \alpha_T = 1, \tag{5.80}$$

$$\alpha \geq \mathbf{0}, \tag{5.81}$$

$$\beta \text{ free.} \tag{5.82}$$

To simplify this program, note that for a feasible $\alpha = (\alpha_T)_{T\in\mathcal{F}}$, we have $\sum_{T\in\mathcal{F}} \alpha_T F^*(c) = F^*(c)$. Furthermore,

$$\sum_{T\in\mathcal{F}} \alpha_T \sum_{e\in T} c_e = \sum_{e\in E} c_e \sum_{T\in\mathcal{F}:e\in T} \alpha_T. \tag{5.83}$$

We use the substitution

$$q_e := \sum_{T\in\mathcal{F}:e\in T} \alpha_T, \quad e \in E. \tag{5.84}$$

Let $q = (q_1, \dots, q_n)$. This substitution is a mapping from $\mathcal{X}$ to the convex hull of $\mathcal{X}$, much like the definition of the marginal probability vector. Taking the substitution into account gives the program

$$\begin{aligned} \min_{q,\beta} \quad & \beta && (5.85)\\ \text{s.t.} \quad & \sum_{e\in E} c_e q_e - F^*(c) \le \beta, \qquad \forall c \in \mathcal{I}_f, && (5.86)\\ & q \in \mathrm{CH}(\mathcal{X}), && (5.87)\\ & \beta \text{ free.} \end{aligned}$$

This program no longer has an exponential number of variables, and the exponential number of constraints can be handled via separation, which we describe shortly. First, for some set $A \in \mathcal{F}$, define the cost vector $c^A = (c^A_e)_{e\in E}$ where

$$c^A_e := \begin{cases} c^-_e, & e \in A, \\ c^+_e, & e \in E \setminus A. \end{cases} \tag{5.88}$$

That is, c^A is the cost vector where all costs are equal to their upper bound, except for costs in the set A, which are equal to their lower bound. The analysis below shows that without loss of generality, we can can consider cost vectors of this form for separation of the constraint (5.86). This allows us to define $\mathcal{I}_f$ as the set of all cost vectors $\{c^A |\ A \in \mathcal{F}\}$. Since this may still be an exponentially sized set, recall that we defined an adversarial mixed strategy encoding $\mathcal{L} = (C, W)$ as a set of costs $C = \{c^{A_j} \in \mathcal{I}_f \mid j = 1, \dots, \eta\}$ to be selected with corresponding probabilities $W = \{w_{c^{A_j}} \in [0,1] \mid j = 1, \dots, \eta\}$ satisfying $\sum_{j=1}^{\eta} w_{c^{A_j}} = 1$.

Theorem 5.6. *For interval uncertainty, if the nominal problem $F^*(c)$ can be solved in time polynomial in n and m, then the corresponding randomized adversarial maxmin regret problem $\max_{w\in\mathcal{W}} \min_{T\in\mathcal{F}} \sum_{c\in\mathcal{I}_f} w_S\left(F(T,c) - F^*(c)\right)$ can be solved in time polynomial in n and m.*

Proof. The constraint (5.86) is simply the maximum regret problem for a given vector q and can be generated via the nominal problem,

$$\max_{c\in\mathcal{I}_f} \left(\sum_{e\in E} c_e q_e - F^*(c) \right) = \max_{T\in\mathcal{F}} \left(\sum_{e\in E\setminus T} c^+_e q_e - \sum_{e\in T} c^-_e (1 - q_e) \right), \tag{5.89}$$

where we have used the analysis in Lemma 5.5. This allows us to write the linear program as

$$\begin{aligned}
\min_{q,\beta} \quad & \beta && (5.90)\\
\text{s.t.} \quad & \sum_{e\in E\setminus T} c_e^+ q_e - \sum_{e\in T} c_e^-(1-q_e) \le \beta, \qquad \forall T\in\mathcal{F}, && (5.91)\\
& q \in \mathrm{CH}(\mathcal{X}), && (5.92)\\
& \beta \text{ free.} && (5.93)
\end{aligned}$$

Note that this is precisely the linear program (5.64) - (5.67) solved by the optimizing player. This justifies the assumption of the finite set $\mathcal{I}_f$: only a polynomial number of separating cost vectors will be generated, and they will be of the form c^A as defined in (5.88). The adversary is of course interested in the dual variables of the linear program (5.90) - (5.93). Each separating hyperplane generated for the constraint (5.91) gives a set $T \in \mathcal{F}$ for which the adversary adds the cost vector c^T (based on the notation in (5.88)) to his mixed strategy. This cost vector has probability w_{c^T} in the mixed strategy, given by the corresponding dual variable. ■

Corollary 5.4. *For interval uncertainty, $Z_{\mathrm{R}} = Z_{\mathrm{AR}}$.*

Proof. This holds by strong duality, since the optimizing player solves (5.64)-(5.67), which is the dual of the linear program (5.73) - (5.77) for the adversary. ■

5.4.3 Primal-Dual Approximation

The primal linear program for the optimizing player is (5.64)-(5.67), and the dual linear program for the adversary is (5.73) - (5.77). Using a similar approach to the previous section, we devise a primal-dual approximation algorithm for the deterministic minmax regret problem.

We rewrite the dual linear program as

$$\begin{aligned}
\max_{w,z} \quad & z - \sum_{c\in\mathcal{I}_f} w_c F^*(c) && (5.94)\\
\text{s.t.} \quad & \sum_{c\in\mathcal{I}_f} w_c \sum_{e\in U} c_e \ge z, \qquad \forall U\in\mathcal{F}, && (5.95)\\
& \sum_{c\in\mathcal{I}_f} w_c = 1, && (5.96)\\
& w \ge \mathbf{0}, && (5.97)\\
& z \text{ free.} && (5.98)
\end{aligned}$$

We must select a feasible solution for w; we will make the simple choice of setting $w_c = 1/2$ for two cost vectors. Recall the definition of the cost vector c^A. For some set

$A \in \mathcal{F}$, we have $c^A = (c_e^A)_{e \in E}$ where

$$c_e^A = \begin{cases} c_e^-, & e \in A, \\ c_e^+, & e \in E \setminus A. \end{cases} \tag{5.99}$$

Additionally, we define $c^{\overline{A}} = (c_e^{\overline{A}})_{e \in E}$ where

$$c_e^{\overline{A}} = \begin{cases} c_e^+, & e \in A, \\ c_e^-, & e \in E \setminus A. \end{cases} \tag{5.100}$$

We set $w_c = 1/2$ for $c = c^A$ and $c = c^{\overline{A}}$, so the linear program becomes

$$\max_z \quad z - \frac{1}{2}\left(F^*(c^A) - F^*(c^{\overline{A}})\right) \tag{5.101}$$

$$\text{s.t.} \quad \sum_{e \in U} \left(\frac{c_e^- + c_e^+}{2}\right) \geq z, \qquad \forall U \in \mathcal{F}, \tag{5.102}$$

$$z \text{ free.} \tag{5.103}$$

Under the primal-dual approach, we increase z until one of the constraints becomes tight. The first tight constraint corresponds to the primal solution M that has minimum total cost under the midpoint costs:

$$M := \operatorname*{argmin}_{U \in \mathcal{F}} \sum_{e \in U} \left(\frac{c_e^- + c_e^+}{2}\right). \tag{5.104}$$

This gives the objective value

$$\frac{1}{2}\left(\sum_{e \in M} (c_e^- + c_e^+) - F^*(c^A) - F^*(c^{\overline{A}})\right). \tag{5.105}$$

By choosing the set A to be the midpoint cost minimizing set M, we can write the resulting objective value in terms of the maximum deterministic regret.

Lemma 5.7. *For $A = M$,*

$$\sum_{e \in M} (c_e^- + c_e^+) - F^*(c^A) - F^*(c^{\overline{A}}) = R_{\max}(M). \tag{5.106}$$

Proof. The maximum regret for the set M can be expressed as

$$R_{\max}(M) = \sum_{e \in M} c_e^+ - \min_{T \in \mathcal{F}} \left(\sum_{e \in T \cap M} c_e^+ + \sum_{e \in T \setminus M} c_e^-\right); \tag{5.107}$$

this can be shown by taking the expression in (5.52), plugging in the characteristic vector corresponding to M, and using simple manipulations. Also note that

$$F^*(c^{\overline{A}}) = \min_{T\in\mathcal{F}} \left(\sum_{e\in T\cap A} c_e^+ + \sum_{e\in T\setminus A} c_e^- \right). \tag{5.108}$$

Thus for $A = M$, we have

$$R_{\max}(M) = \sum_{e\in M} c_e^+ - F^*(c^{\overline{A}}). \tag{5.109}$$

It is left to show that $F^*(c^M) = \sum_{e\in M} c_e^-$. This, however, immediately follows with a simple argument. Since the set M is minimum for midpoint costs, it must also be minimum for costs c^M (i.e., the costs where $c_e = c_e^-$ for all $e \in M$ and $c_e = c_e^+$ for all $e \in E \setminus M$).

To verify that M is indeed the minimum solution under costs c^M, consider the set M along with some other set $U \in \mathcal{F}$ that differs from M by at least one element. We look at the change in cost when moving from midpoint costs to the cost vector c^M for both sets. For M, the change in cost is

$$-\sum_{e\in M} \frac{(c_e^- + c_e^+)}{2}. \tag{5.110}$$

For U, the change in cost is

$$-\sum_{e\in U\cap M} \frac{(c_e^- + c_e^+)}{2} + \sum_{e\in U\setminus M} \frac{(c_e^- + c_e^+)}{2}. \tag{5.111}$$

It is clear that (5.110) is no greater than (5.111). Starting with midpoint costs, M is the lowest cost solution, and when moving to the cost vector c^M, the decrease in cost is at least as significant for M as it is for any other set U. Hence, M must also be a minimum cost solution under costs c^M. ■

This gives a new proof that solving the nominal problem with midpoint costs gives a 2-approximation algorithm for the deterministic minmax regret problem. Once again, the result here is stronger than the result of [85] since it states that $R_{\max}(M) \leq 2Z_{\mathrm{R}}$ instead of $R_{\max}(M) \leq 2Z_{\mathrm{D}}$.

Theorem 5.7. *For interval uncertainty, the solution to the nominal problem with midpoint costs is a 2-approximation algorithm for the deterministic minmax regret problem.*

Proof. The linear program (5.94)-(5.98) is the dual of the problem solved by the optimizing player in the randomized framework. By weak duality, any feasible solution to (5.94)-(5.98) gives a lower bound on the value of the game in the randomized framework

```
Input: Nominal combinatorial problem, item cost bounds $(c_e^-, c_e^+)$, $e \in E$.
Output: Feasible solution $M \in \mathcal{F}$ satisfying $R_{\max}(M) \le 2Z_{\mathrm{D}}$.
1: Determine midpoint costs for each item: $d_e \leftarrow \left(\frac{c_e^- + c_e^+}{2}\right), \quad \forall e \in E$.
2: Solve nominal problem with midpoint costs: $M \leftarrow \operatorname{argmin}_{T \in \mathcal{F}} \sum_{e \in T} d_e$.
```

Algorithm 18. Midpoint-Cost-Approximation (Kasperski and Zieliński [85])

Z_{R}. The construction described above using costs c^A and $c^{\overline{A}}$ gives a feasible solution to the program, and Lemma 5.7 allows us to express the resulting objective value in terms of the maximum deterministic regret for a solution set M using $A = M$. Specifically,

$$\frac{Z_{\mathrm{D}}}{2} \quad \le \quad \frac{R_{\max}(M)}{2} \le Z_{\mathrm{R}} \le Z_{\mathrm{D}}, \tag{5.112}$$

where the first inequality follows from the definition of the deterministic minmax regret problem, the second inequality follows using Lemma 5.7 with the feasible linear programming solution, and the third inequality follows from Corollary 5.3. ■

The potential gain from using randomization under interval uncertainty is not as significant as with discrete scenario uncertainty.

Theorem 5.8. *For interval uncertainty and all nominal problems,*

$$Z_{\mathrm{R}} \ge \frac{Z_{\mathrm{D}}}{2}. \tag{5.113}$$

Equivalently, the integrality gap of the mixed integer program (5.51) *is equal to* 2.

Proof. The inequality follows from the primal-dual analysis. A tight example is easily constructed. Consider a problem with two items $E = \{e_1, e_2\}$ where the optimizing player must choose one item. Let $(c_e^+, c_e^-) = (0, 1)$ for both items $e = e_1, e_2$. It can be verified that $Z_{\mathrm{D}} = 1$ and $Z_{\mathrm{R}} = 1/2$. ■

5.4.4 Minimum Assignment Example

We apply the above theory to a small instance of the minimum assignment problem. Given a bipartite graph $G = (V_1, V_2, E)$ with edge costs c_{ij} for each edge in $(i, j) \in E$, the minimum assignment problem asks for a perfect matching with minimum total cost.

This is stated by the following integer program:

$$\min \sum_{(i,j)\in E} c_{ij}x_{ij} \tag{5.114}$$

$$\text{s.t.} \sum_{j:(i,j)\in E} x_{ij} = 1, \qquad \forall i \in V_1, \tag{5.115}$$

$$\sum_{i:(i,j)\in E} x_{ij} = 1, \qquad \forall j \in V_2, \tag{5.116}$$

$$x_{ij} \in \{0,1\}, \qquad \forall (i,j) \in E. \tag{5.117}$$

The minimum assignment problem is polynomial solvable, but its deterministic minmax regret version is NP-hard [84]. A useful feature of the minimum assignment problem for our work is that the linear programming relaxation of the above program directly gives the convex hull of solutions.

We look at a simple problem instance with a complete graph where $|V_1| = |V_2| = 3$, shown in Figure 5.1 (a). Since bipartite graphs are easily described in terms of adjacency matrices, we will use use matrix notation here in place of the vector notation used in the analysis. The cost bounds for the problem are given by

$$c^- = \begin{bmatrix} 4 & 5 & 0 \\ 4 & 4 & 2 \\ 1 & 2 & 5 \end{bmatrix}, \qquad c^+ = \begin{bmatrix} 5 & 7 & 6 \\ 7 & 6 & 7 \\ 7 & 6 & 6 \end{bmatrix}, \tag{5.118}$$

where the (i,j)th entry of c^- indicates c^-_{ij}, the lower bound on the cost of edge (i,j). It can be verified either by enumeration or with the integer program that the deterministic minmax regret solution is

$$x = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \tag{5.119}$$

which is shown by the bold solution in Figure 5.1 (b). With knowledge of this solution selected by the optimizing player, the adversary sets all costs for edges in the solution equal to their maximum value and all other costs equal to their minimum value; the resulting costs are

$$c = \begin{bmatrix} 4 & 5 & 6 \\ 4 & 6 & 2 \\ 7 & 2 & 5 \end{bmatrix}. \tag{5.120}$$

These costs are shown with their corresponding edges in Figure 5.1 (b). The value of the solution selected by the optimizing player is equal to 19, while the optimal solution, indicated by the dashed solution in Figure 5.1 (b), has a cost of 8. The deterministic minmax regret value for the problem is thus $Z_{\mathrm{D}} = 11$.

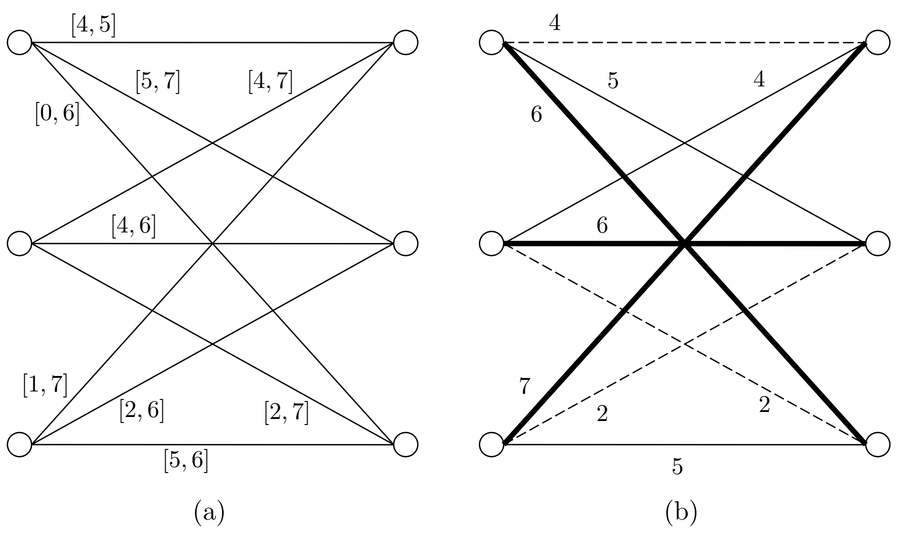


Figure 5.1. Minimum assignment example with (a) cost bounds for each edge labeled as $[c_e^-, c_e^+]$ and (b) deterministic minmax regret solution shown in bold. The edge weights shown in (b) indicate the costs selected by the adversary, which are maximum for the solution selected by the optimizing player and minimum for all other edges. The optimal solution under the selected edge weights is shown by the dashed solution.

Solving the randomized minmax regret problem, first for marginal probabilities, gives the probability matrix

$$p = \begin{bmatrix} 3/8 & 0 & 5/8 \\ 1/6 & 11/24 & 3/8 \\ 11/24 & 13/24 & 0 \end{bmatrix}. \tag{5.121}$$

This is illustrated in Figure 5.3, where each edge thickness is shown in proportion to its probability. Solving the linear program transforming marginal probabilities gives a mixed strategy for the optimizing player consisting of three solutions T_1, T_2, and T_3 with adjacency matrices

$$x^{T_1} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}, \quad x^{T_2} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \quad x^{T_3} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \tag{5.122}$$

and corresponding probabilities

$$y_{T_1} = \frac{11}{24}, \quad y_{T_2} = \frac{3}{8}, \quad y_{T_3} = \frac{1}{6}. \tag{5.123}$$

The solutions are also shown in Figure 5.2.

The adversary's mixed strategy consists of three cost matrices. These cost matrices are c^{T_1}, c^{T_2}, and c^{T_3} for the solutions shown in Figure 5.2; specifically,

$$c^{T_1} = \begin{bmatrix} 5 & 7 & 0 \\ 7 & 4 & 7 \\ 1 & 6 & 6 \end{bmatrix}, \quad c^{T_2} = \begin{bmatrix} 4 & 7 & 6 \\ 7 & 6 & 2 \\ 7 & 2 & 6 \end{bmatrix}, \quad c^{T_3} = \begin{bmatrix} 5 & 7 & 0 \\ 4 & 6 & 7 \\ 7 & 2 & 6 \end{bmatrix}. \tag{5.124}$$

These are selected with probabilities

$$w_{c^{T_1}} = \frac{3}{8}, \quad w_{c^{T_2}} = \frac{11}{24}, \quad w_{c^{T_3}} = \frac{1}{6}. \tag{5.125}$$

The value of the randomized minmax regret problem is $Z_{\mathrm{R}} = 149/24 \approx 6.208$.

It is interesting to note that solving the optimal problem with midpoint costs, that is, with cost matrix

$$c = \begin{bmatrix} 4.5 & 6 & 3 \\ 5.5 & 5 & 4.5 \\ 4 & 4 & 5.5 \end{bmatrix}, \tag{5.126}$$

gives the same solution that solving the deterministic minmax regret problem gives. So for this problem, the midpoint approximation scheme is optimal.

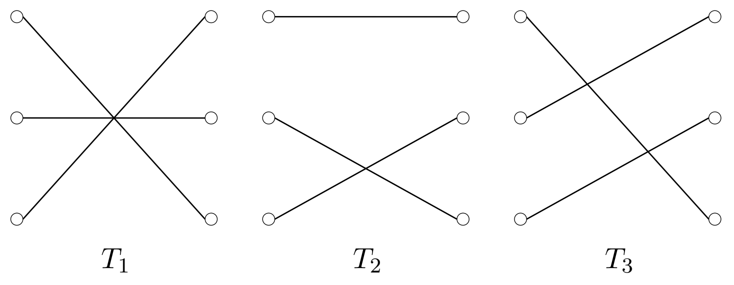


Figure 5.2. Solutions in optimizing player's mixed strategy.

5.5 General Uncertainty Sets

In this section, we show that if the uncertainty set $\mathcal{C}$ is allowed to be a general nonnegative convex set and the nominal problem is polynomial solvable, the maximum expected regret problem becomes NP-hard. Note that the deterministic maximum regret problem is a special case of the maximum expected regret problem. The result of this section thus implies that both randomized and deterministic minmax regret problems are NP-hard under general convex uncertainty, even if the nominal problem is polynomial solvable.

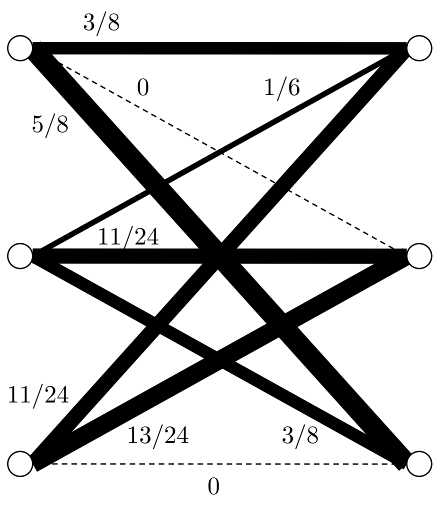


Figure 5.3. Marginal probability solution for randomized minmax regret problem. Edge weights indicate the probabilities; edge thicknesses are also shown in proportion to the probabilities.

We restate the maximum expected regret problem for general uncertainty sets. For a given marginal probability vector p, the maximum expected regret problem is, starting with the first line of (5.60),

$$\begin{aligned}\overline{R}_{\max}(p) &= \max_{c\in\mathcal{C}}\left(\sum_{e\in E} c_e p_e - F^*(c)\right)\\ &= \max_{c\in\mathcal{C}}\left(\sum_{e\in E} c_e p_e - \min_{x\in\mathcal{X}}\sum_{e\in E} c_e x_e\right)\\ &= \max_{c\in\mathcal{C}}\max_{x\in\mathcal{X}}\left(\sum_{e\in E} c_e (p_e - x_e)\right). \qquad (5.127)\end{aligned}$$

Negating the objective function, the maximum expected regret problem is equivalent to

$$-\overline{R}_{\max}(p) = \min_{c\in\mathcal{C}}\min_{x\in\mathcal{X}}\sum_{e\in E} c_e(x_e - p_e) \qquad (5.128)$$

for a given $p \in \mathrm{CH}(\mathcal{X})$.

Before we examine the complexity of (5.128), we consider the following problem that we refer to as the *bilinear combinatorial problem*:

$$\min_{c\in\mathcal{C}}\min_{x\in\mathcal{X}}\sum_{e\in E} c_e x_e. \qquad (5.129)$$

We demonstrate the hardness of this problem via a reduction from the Hamiltonian path problem. The proof is similar to the standard proof for showing that the intersection of three matroids is NP-hard [128]. We then modify the proof slightly to show that the maximum expected regret problem is NP-hard.

Lemma 5.8. *For polynomial solvable nominal problems $F^*(c) = \min_{x\in\mathcal{X}} \sum_{e\in E} c_e x_e$ and nonnegative convex uncertainty sets $\mathcal{C}$, the bilinear combinatorial problem $\min_{c\in\mathcal{C}} \min_{x\in\mathcal{X}} \sum_{e\in E} c_e x_e$ is NP-hard.*

Proof. Recall that the directed Hamiltonian path problem asks the following: given a directed graph $G = (V, E)$ with a designated source node s and terminal node t, does there exist a path starting at s and ending at t that visits each node exactly once? For a given instance of the directed Hamiltonian path problem, we construct an instance of the bilinear combinatorial problem such that it has an optimal objective value of zero if and only if the graph contains a valid Hamiltonian path.

We construct the set $\mathcal{C}$ to indicate the selection of edges so that each vertex has exactly one incoming edge (except for vertex s) and one outgoing edge (except for vertex t). Specifically, we say that an edge e is *selected* if its cost c_e is equal to zero, otherwise we refer to it as *blocked*. The notation $\delta^+(v)$ (respectively $\delta^-(v)$) indicates the set of outgoing (incoming) edges for vertex v. The constraints for the set $\mathcal{C}$ are

$$\begin{aligned}
\sum_{e\in\delta^-(v)} c_e &= |\delta^-(v)| - 1, \quad v \in V \setminus \{s\},\\
\sum_{e\in\delta^-(s)} c_e &= |\delta^-(s)|,\\
\sum_{e\in\delta^+(v)} c_e &= |\delta^+(v)| - 1, \quad v \in V \setminus \{t\},\\
\sum_{e\in\delta^+(t)} c_e &= |\delta^+(t)|,\\
0 \le c_e \le 1, &\qquad e \in E. \qquad (5.130)
\end{aligned}$$

Note that for a given vertex, if one if its incoming edges is selected ($c_e = 0$), then all of the remaining incoming edges must be blocked ($c_e = 1$); the same holds for outgoing edges.

Define $\mathcal{X}$ to indicate the set of all feasible spanning trees for G, so that $\min_{x\in\mathcal{X}} \sum_{e\in E} c_e x_e$ is the minimum spanning tree problem. For an optimal solution $x^* = (x_1^*, \ldots, x_n^*)$ to the bilinear combinatorial problem giving zero objective value, the set $\{e | x_e^* = 0, e \in E\}$ indicates a valid Hamiltonian path. This holds because the construction of $\mathcal{C}$ indicates that all vertices have one selected incoming and outgoing edge (except for s and t), and the spanning tree ensures that no cycles are present. Finally, we have that the construction of the set $\mathcal{C}$ can be done in polynomial time. ■

Theorem 5.9. *For polynomial solvable nominal problems $F^*(c) = \min_{x\in\mathcal{X}} \sum_{e\in E} c_e x_e$ and nonnegative convex uncertainty sets $\mathcal{C}$, the maximum expected regret problem $\max_{c\in\mathcal{C}} \left(\sum_{e\in E} c_e p_e - F^*(c)\right)$ where $p \in \mathrm{CH}(\mathcal{X})$ is* NP*-hard.*

Proof. We modify the reduction used for Lemma 5.8 to account for the presence of some $p \in \mathrm{CH}(\mathcal{X})$ in the objective function of the maximum expected regret problem, which is now

$$\min_{c\in\mathcal{C}} \min_{x\in\mathcal{X}} \sum_{e\in E} c_e(x_e - p_e). \tag{5.131}$$

Again let $\mathcal{X}$ indicate the set of all feasible spanning trees for the directed graph $G = (V, E)$, and let $p \in \mathcal{X}$ be a valid spanning tree. We construct a new graph over the same set of vertices by taking G and duplicating $|V| - 1$ edges. For each edge given by the spanning tree p, we choose a corresponding edge in G (note that there may be more than one option if both edges (v_i, v_j) and (v_j, v_i) are present, for example) and duplicate it. Let this new graph be denoted by $G' = (V, E')$, and let the set of all spanning trees over the new graph be indicated by $\mathcal{X}'$. Let $\widetilde{p} \in \mathcal{X}'$ indicate the set of edges that were constructed via duplication (i.e. the edges $E' \setminus E$), which is a valid spanning tree for G'. We finally construct the set $\mathcal{C}'$ using the inequalities in (5.130) but over E' instead of E.

Now consider the modified maximum expected regret problem

$$\min_{c\in\mathcal{C}'} \min_{x\in\mathcal{X}'} \sum_{e\in E'} c_e(x_e - \widetilde{p}_e) = \min_{c\in\mathcal{C}'} \min_{x\in\mathcal{X}'} \left(\sum_{e\in E} c_e x_e + \sum_{e\in E'\setminus E} c_e(x_e - 1) \right). \tag{5.132}$$

It can be seen that the modified problem has an objective value equal to $-(|V| - 1)$ if and only if G has a Hamiltonian path. This corresponds to the first sum in objective function being equal to zero and the second sum being equal to $-(|V| - 1)$. As before, for an optimal solution $x^* = (x_1^*, \ldots, x_{n+|V|-1}^*)$ to the modified problem, the set $\{e | x_e^* = 0, e \in E'\}$ gives a Hamiltonian path that is valid for both G' and G. Notice that all of the duplicated edges $e \in E' \setminus E$ must be blocked ($c_e = 1$) and not selected by the minimum spanning tree ($x_e = 0$) for the objective value to be equal to $-(|V| - 1)$. To finish the proof, we observe that the construction of $\mathcal{C}'$ and G' can be accomplished in polynomial time.

An example of the reduction is shown in Figure 5.4. Figure 5.4 (a) shows a directed graph with edges $E = \{e_1, e_2, \ldots, e_7\}$, a source node s, and a terminal node t. A spanning tree indicating p is given by the set $\{e_1, e_2, e_3, e_4, e_7\}$, shown in Figure 5.4 (b) in bold. Figure 5.4 (c) shows duplication of edges in the spanning tree, given by the introduction of edges $\{e_8, e_9, e_{10}, e_{11}, e_{12}\}$. Thus $E' = E \cup \{e_8, e_9, e_{10}, e_{11}, e_{12}\}$. With the formulation of the corresponding maximum expected regret problem (5.132), the optimal solution has blocked edges ($c_e = 1$) for the set $\{e_1, e_5, e_8, e_9, e_{10}, e_{11}, e_{12}\}$, shown as dashed in Figure 5.4 (d). The remaining non-blocked edges are in the set $\{e_2, e_3, e_4, e_6, e_7\}$, which is a valid spanning tree and thus a Hamiltonian path. ■

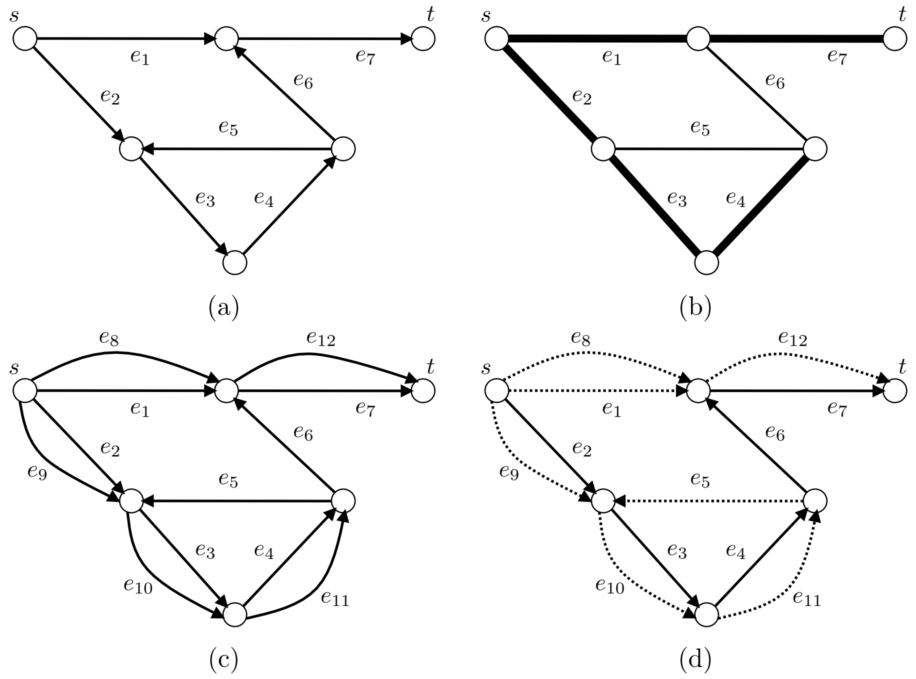


Figure 5.4. Example of the construction shown in Theorem 5.9. (a) A directed graph with source node s and terminal node t. (b) A spanning tree for the corresponding undirected graph shown in bold. (c) Directed graph with edges for the spanning tree duplicated. (d) Optimal solution for edge costs $c \in \mathcal{C}'$, where blocked edges, corresponding to $c_e = 1$, are dashed. The valid Hamiltonian path is given by the non-blocked (solid) edges.

5.6 Discussion

We have shown that for both interval and discrete scenario representations of uncertainty, the randomized minmax regret version of any polynomial solvable combinatorial problem is polynomial solvable. These results are at first glance surprising. For many applications that are not truly adversarial in nature, the randomized minmax regret model is more appropriate than the deterministic minmax regret model. In particular, the deterministic solution may be overly conservative since costs are not actually chosen in an adversarial fashion in response to the selected solution. On the other hand, one must be willing to tolerate higher variance if randomization is used.

Our results on lower bounds for randomized minmax regret in relation to deterministic minmax regret, specifically Theorems 5.4 and 5.8, have important implications for approximating deterministic minmax regret problems. Theorem 5.8 indicates that the integrality gap for the minmax regret problem is equal to 2, and it is easy to create instances of nearly all nominal problems that achieve this gap. The same can be argued for the integrality gap of k under discrete scenario uncertainty. In Kasperski [84], it is posed as an open problem whether there exist approximation algorithms under interval uncertainty that, for some specific nominal problems, achieve an approximation ratio better than 2. We have answered this question in the negative for approximation schemes based on our linear programming relaxations.

An important future step with randomized minmax regret research is to develop approximation algorithms (now in the randomized model) for dealing with nominal problems that are already NP-complete. This problem is non-trivial: an algorithm with an approximation factor α for a nominal problem does not immediately yield an algorithm to approximate the randomized minmax regret problem with a factor α. Another interesting topic to study from an experimental perspective is a hybrid approach that employs both deterministic and randomized minmax regret. For example, one could find a solution that minimizes the maximum expected regret, subject to the maximum regret being no greater than some constant. The algorithm for this problem can be easily constructed by combining our results with existing work, but it may no longer run in polynomial time.

Chapter 6

Conclusion

THIS thesis has addressed four topics on approximation and uncertainty in optimization. In the first part of the thesis, on the theme of approximation, we analyzed greedy-type algorithms on classic combinatorial problems of packing and matching. Our focus was on average-case analysis, and our results included average-case analogues of known theorems for worst-case behavior. On the second theme of uncertainty, we considered the effect of uncertain transition probabilities in Markov decision processes and then explored a randomized model for robust optimization with cost uncertainty. In this chapter, we review our contributions and give recommendations for future research.

6.1 Summary of Contributions

In Chapter 2, we investigated the performance of rollout algorithms on the subset sum problem and 0–1 knapsack problem. As observed by Bertsekas [27], rollout algorithms tend to perform well in simulation, but it has been difficult to obtain theoretical results proving that they perform well. In particular, it has been difficult to prove that they perform strictly better than their base policies. We noted a few exceptions in the literature: Bertsekas [27] described an average-case asymptotic result for the breakthrough problem, and some worst-case results are known for the 0–1 knapsack problem due to Bertazzi and Sahni [26, 125]. To complement this work, we studied the subset sum problem and the 0–1 knapsack problem using average-case analysis. In accordance with two natural ways to modify a greedy algorithm for these problems, we defined the Consecutive-Rollout and Exhaustive-Rollout algorithms, both of which use the simple Blind-Greedy base policy.

We proved that both rollout algorithms perform significantly better than the Blind-Greedy base policy in expectation. The bounds that we derived hold after the first iteration of the algorithms and are valid for small values of n (recall that we only required $n \geq 3$, where n is the number of items). To our knowledge, these are the first non-asymptotic average-case bounds to show a strict improvement of rollout algorithms over a base policy. For the subset sum problem, we employed a graphical diagram to prove performance gains. A similar strategy was used for the 0–1 knapsack problem. The crucial assumption that we used to make the proofs tenable was to only consider the contribution of the first few packed items to the performance improvement. We showed that this gives a close characterization of the first iteration of the rollout algorithms in comparison to simulations. For both problems, we arrived at simple bounds for the Consecutive-Rollout algorithm. On the other hand, for the Exhaustive-Rollout algorithm, we obtained a simple bound for the subset sum problem and an unwieldy expression for the 0–1 knapsack bound.

More generally, we argued that rollout algorithms may perform better in the average case than in the worst case. We pointed out the tight example given by Bertazzi [26] for the 0–1 knapsack problem, which shows that in the worst case, it is not possible to guarantee a performance gain beyond the first iteration of the rollout algorithm. On the other hand, simulations indicate performance gains from additional iterations under stochastic models for knapsack problems. We also looked at the problem of finding a shortest path in a binary decision tree. We showed that even when running every iteration of the rollout algorithm, performance is poor in the worst case but strong on average.

Continuing with the theme of approximation, in Chapter 3, we considered the average-case performance of greedy online matching. The online bipartite matching problem was introduced in the well-known paper of Karp et al. [83], where it was studied using worst-case analysis. Karp et al. presented the Ranking algorithm, which has a competitive ratio of $1 - 1/e \approx 0.632$, and showed that this is the best possible competitive ratio for any online algorithm. We looked at the average-case performance of greedy algorithms on this problem using the Erdős-Rényi binomial random graph $\mathcal{G}(n, n, p)$ and the random regular graph $\mathcal{G}(n, n, r)$. The majority of our analysis was conducted using the differential equation method with Wormald's theorem [148].

We described our results in terms of asymptotic matching sizes and performance ratios, where we defined the performance ratio as an average-case analog of the competitive ratio. Under the $\mathcal{G}(n, n, p)$ model, we showed that the performance ratio for Greedy is at least 0.837 for all monotonic functions $p = p(n)$. Moreover, we showed that Ranking and Greedy are equivalent on $\mathcal{G}(n, n, p)$, indicating that the results for Greedy apply also to Ranking. For the random regular graph model $\mathcal{G}(n, n, r)$, we focused on the case where $r = 2$. We showed that the performance ratio of Greedy on this model is 0.877, larger than the lower bound for $\mathcal{G}(n, n, p)$. We also defined the Degree-Greedy algorithm for random regular graphs and showed that its performance ratio on $\mathcal{G}(n, n, 2)$ is equal to $11/12 \approx 0.917$.

In addition to the conventional online bipartite model, we presented a model for online *non-bipartite* matching, where all vertices in the graph arrive sequentially (as opposed to just one side in the case of the bipartite model). For the binomial random graph $\mathcal{G}(n, p)$, we showed that the non-bipartite Greedy algorithm achieves the same asymptotic fraction of matched vertices that the bipartite Greedy algorithm does. We also showed that it has a performance ratio of at least 0.837 for all monotonic functions $p = p(n)$. For the non-bipartite random regular graph model $\mathcal{G}(n, r)$ with $r = 2$, we showed that the Greedy algorithm has a performance ratio of at least 0.869 and that the Degree-Greedy algorithm has a performance ratio of at least 0.884.

Turning to the second theme of uncertainty, Chapter 4 dealt with uncertainty in Markov decision processes. Under the total expected reward criterion, we reviewed existing loss bounds for uncertain value functions and approximate backwards induction, generalizing these results to account for undiscounted models and showing tight examples. The main topic of the chapter was devoted to losses resulting from uncertain transition probabilities. We examined a general non-stationary Markov decision process with nonnegative, bounded rewards. Our assumption was that for all stages, states, and actions, estimated transition probabilities are available, and that the total variation error on the estimated probabilities is upper bounded by a known value.

Assuming that the Markov decision process is solved with conventional dynamic programming, we derived a general upper bound on the expected total loss as a function of the horizon length, reward bound, discount factor, and bound on total variation error. Our proof technique was to analyze the interplay among the the approximate policy value, the approximate value function, and the optimal value function. We defined the approximate policy error as the difference between the approximate policy value and the approximate value function, and the estimation error as the difference between the approximate value function and the optimal value function. We showed that the growth of these errors while stepping backwards in time is characterized by a multilinear program, and we bounded the multilinear program using the worst-case behavior induced by the total variation distance. We presented a tight example for the undiscounted case and derived first-order approximations of the loss bounds.

Finally, in Chapter 5, we studied a randomized model for robust optimization. We focused on the problem of combinatorial optimization with uncertainty in cost coefficients. We noted that most models existing in the literature, both for the minmax and minmax regret robust criteria, require the optimizing player to make a deterministic selection of a solution; the solution is then viewed by some adversary who selects costs against the optimizing player [84, 93]. An alternative model, first suggested by Bertsimas et al. [35] for the minmax criterion in robust optimization, is to allow the optimizing player to select a probability distribution over solutions and require the adversary to select a probability distribution over costs in response to the player's distribution. We studied this randomized model under the minmax regret criterion. For both discrete scenario uncertainty and interval uncertainty, we proved that the randomized minmax regret version of any polynomial solvable nominal problem is polynomial solvable.

Our key observation was that the randomized minmax regret problem corresponds to the linear programming relaxation of the deterministic minmax regret problem. The algorithm that we presented for solving the randomized problem requires solving two linear programs: one to determine marginal probabilities and another to map marginal probabilities to a distribution over deterministic solutions. While both linear programs may have an exponential number of constraints, we showed that the nominal problem gives a separation oracle for both programs. In addition to calculating the optimal strategy for the optimizing player, we showed how to calculate the adversary's optimal strategy, which corresponds to the dual interpretation of the minmax regret linear program.

We also used the linear programs with a primal-dual interpretation to show new proofs of existing approximation algorithms for the deterministic minmax regret problem. For discrete scenario uncertainty, we showed a new proof that solving the nominal problem with mean costs (averaged uniformly over all scenarios) gives a k-approximation algorithm for the deterministic minmax regret problem, where k is the number of scenarios. Likewise, for interval uncertainty, we gave a new proof that solving the nominal problem with midpoint costs gives a 2-approximation algorithm for the deterministic minmax regret problem. Furthermore, we showed that the integrality gaps of the deterministic minmax regret problems are k and 2 for discrete scenario uncertainty and interval uncertainty, respectively. These integrality gaps give bounds on the reduction in expected regret when moving from the deterministic model to the randomized model. Lastly, we considered the minmax regret problem under general convex uncertainty. We showed that even for polynomial solvable nominal problems, solving the maximum regret problem, which is simpler than the entire minmax regret problem (randomized or deterministic), is NP-hard under convex uncertainty.

6.2 Recommendations for Future Research

We give here a summary of important directions for future research on the four topics that this thesis has addressed. We revisit many of the ideas that we gave in the discussion sections of the previous chapters. We also mention some additional, broader research directions.

6.2.1 Rollout Algorithms

Our work on rollout algorithms, as well as the worst-case results of Bertzzi [26] and Sahni [125], are important first steps in proving strict improvements relative to base policies. Yet, there are many open problems, even just involving the 0–1 knapsack problem and the subset sum problem. The obvious place to start is in proving performance bounds for additional iterations (beyond the first) of the rollout algorithm. We have argued that average-case analysis is the appropriate method for doing this, as opposed to worst-case analysis. This seems difficult for the stochastic model that we considered, but it might be easier with simpler models, such as models where the knapsack capacity is fixed, for example. Determining upper bounds on improvements (i.e. statements that the rollout improvement is no better than some bound) on our stochastic model is also important. We have proved some of such results, but we did not include them for sake of brevity.

An alternative direction is to look at different lookahead lengths. We used a lookahead length equal to one in all of our work. A lookahead length of two, for example, would at every iteration try adding all pairs of available items and using the base policy thereafter. Note that the first iteration of the rollout algorithm with larger lookahead lengths is understood due to partial enumeration results [89, 125].

For more practical applications, it is desirable to understand rollout performance on problems with a muti-dimensional state space. (The subset sum problem and 0–1 knapsack problem both have a single-dimensional state space, namely, the capacity.) Rollout algorithms are, after all, a form of approximate dynamic programming, and approximate dynamic programming techniques are needed for problems suffering from the *curse of dimensionality* [22, 27, 122]. Problems such as the the bin packing problem, the multiple knapsack problem (where one can choose from multiple knapsacks to place items), and the multidimensional knapsack problem (where each item consumes multiple resources) are good candidates for further research. While partial enumeration results are known for these problems [89], general theorems for rollout algorithms on these problems will require some work. Note that in resource allocation problems where individual items only consume a small amount of resources relative to the total supply, model predictive control with fluid models has proved very useful; see [46, 68] for examples. These approaches can be viewed as continuous analogs of the rollout approach.

Theoretically, it would be nice to find a simple problem with a clean expression for rollout performance as a function of the number of iterations run. Again, this would probably have to be some problem analyzed from the average-case perspective. Recall that we did make some progress on this front in Chapter 3, where we showed that for matching on random 2-regular bipartite graphs, running the OBLIVIOUS-ROLLOUT algorithm (at each iteration) improves the performance ratio from $3/4$ to $1 - 1/(4e) \approx 0.908$. This problem is extremely simple, though, so it would be preferable to obtain a result on a slightly more difficult problem. A closed-form expression for the performance of the rollout algorithm on binary decision trees that we considered in Chapter 2 would be an illuminating result. Solving the recursion analytically is probably too lofty a goal, but it may be possible to find an asymptotic solution.

Using the rollout approach in a game-theoretic setting is also interesting, especially since this was one of the early uses of rollout algorithms [140]. A theoretical treatment for a two-player game would require many modeling decisions. The player using the rollout algorithm must have a base policy, or strategy, and she must also have some base policy model for her opponent that she uses in her rollouts. Then, during actual gameplay, the opponent may or may not have knowledge of the approximation algorithm that she is using. An opponent who knows exactly what rollout approach she is using has the opportunity to exploit this in his decisions.

6.2.2 Greedy Online Matching on Random Graphs

In our work on matching on random graphs, we were primarily interested in the expected asymptotic matching sizes given by algorithms, and we were less concerned about how tightly concentrated the matching sizes are around their expected values. Wormald's general theorem was sufficient for our analysis, but it is possible to obtain tighter concentration results, still employing differential equations, but using the so-called *wholistic approach*. This is described in Wormald's tutorial paper [148]. Alternatively, the altogether different approach used by Dyer et al. [56] may be useful for the algorithms we considered. Their results characterize the asymptotic *distribution* of matching sizes, rather than just the expected values.

Building on our results for random regular graphs, it may be possible to analyze the algorithms that we presented on $\mathcal{G}(n, n, r)$ and $\mathcal{G}(n, r)$ for $r = 3$ (recall that we only considered $r = 2$). Some approximations will likely be required, however. Directly generalizing our approach to $r = 3$ for Degree-Greedy on bipartite graphs, for example, requires enumerating five bin labels, leading to 35 joint cases of bin labels to consider in the analysis. Furthermore, the resulting differential equations do not seem to be solvable analytically. A related open problem for random regular graphs is to prove that the asymptotic matching sizes given by our algorithms are monotonically increasing in r for $r \geq 2$.

An important topic in Erdős-Rényi graphs is the tightness of the asymptotic bound for the maximum matching size on $\mathcal{G}(n, n, c/n)$ given by Bollobás and Brightwell [40]. Whether this bound is tight for $c > e$ is an open problem, and this problem could be resolved by analyzing the Karp-Sipser algorithm for bipartite graphs [13, 64, 82]. It would also be useful to understand the asymptotic maximum matching size for unbalanced bipartite graphs (i.e. graphs with more vertices in one partition than the other).

Most importantly, while Erdős-Rényi random graphs and random regular graphs are natural first steps for analyzing matching algorithms, they are not realistic for many applications. Random graphs with node degrees following a power law distribution are more appropriate, but of course are not as easy to deal with analytically. However, it may still be possible to use the differential equation method on such graphs. A reasonable and interesting graph model to start with is the *preferential attachment model* [115, 142].

6.2.3 Uncertainty in Markov Decision Processes

Our perspective on uncertain transition probabilities in Markov decision processes was fairly general. We allowed the process to be non-stationary (meaning the set of states, rewards, and transition probabilities may vary across stages), and our only critical assumption was that rewards are upper bounded by some constant. It would be interesting to see how the loss bounds change when additional restrictions are placed on the MDP. There are likely some classes of problems, for instance, where states with maximum future reward (that we used to construct the tight example) cannot occur. There may also be some problems where the loss bounds for stationary processes are different than bounds for non-stationary process.

The bounds that we derived are pessimistic, in part because of the worst-case behavior induced by the total variation distance. If uncertain transition probabilities are in fact known within a certain total variation error for a given system, the loss in total expected reward is likely to be significantly less than our upper bound. The reason is that uncertain transition probabilities are not truly chosen adversarially. This motivates other models of uncertainty, perhaps where parameters are subject to random (instead of worst-case) fluctuations. One option is to model transition probabilities with additive Gaussian noise and look at the expected loss. The direct analysis of this average-case problem seems difficult, however. It would be necessary to have an understanding of the distribution of rewards, as opposed to just an upper bound, and Gaussian distributions are often not convenient to work with analytically.

We looked at uncertain transition probabilities assuming that the MDP is solved with exact dynamic programming. Exact dynamic programming is not computationally feasible for large systems, though, and approximate dynamic programming methods must be used instead [122, 135]. Some approximate dynamic programming algorithms may be more sensitive to uncertain transition probabilities than others, and there are opportunities for research here.

Developing robust optimization algorithms for dealing with uncertainty in Markov decision processes is a fruitful research area. Even without uncertainty in model parameters, MDPs are difficult to solve, so computational complexity becomes an even more significant concern when a layer of robustness is added. A related topic is *online learning* of uncertain parameters in Markov decision processes; this is a setting where learning takes place as the system is being optimized. This can be viewed as a type of multi-armed bandit problem, where an optimizer must make decisions to balance the objectives of exploration (learning more about the system) and exploitation (making decisions with high value) [45, 70]. Online learning in Markov decision processes is an active research area, and the topic of uncertain transition probabilities has been considered to some extent [1, 59].

6.2.4 Randomized Minmax Regret for Combinatorial Optimization

We showed that the linear programming formulations for the randomized minmax regret problems are effective in deriving approximation algorithms for the deterministic minmax regret problems via the primal-dual approach. However, we only analyzed known approximation algorithms, and there are a variety of other algorithms to consider. For instance, the maximum likelihood solution from the randomized problem – that is, the solution in the mixed strategy with maximum probability – is likely to be a good approximation for the deterministic minmax regret problem. It may also hold that any solution in the support of the mixed strategy is a decent approximation. Of course, none of these solutions will be able to guarantee an approximation ratio better than the integrality gaps that we determined. If many different approximation algorithms are available, however, where each algorithm gives an approximate solution, it is easy to calculate the maximum regret for each solution in a given problem instance and then pick the solution with the smallest maximum regret.

A useful extension for the randomized minmax regret framework is to consider approximation techniques for *nominal problems* that are NP-hard. Our solution approach is still valid for NP-hard nominal problems but of course comes at the expense of NP-hardness. Approximating the randomized minmax regret problem is nontrivial since an α-approximation algorithm for the nominal problem does not give an α-approximation algorithm for the randomized minmax regret problem. An approximation scheme will likely incur losses in two steps: first when solving the minmax regret program for obtaining marginal probabilities, and second when mapping marginal probabilities to a mixed strategy.

For both the minmax and minmax regret criteria, the randomized and deterministic models give two useful models of nature, the latter giving nature more power. However, the randomized model may still be too pessimistic, and for many applications, it may generate solutions that are too conservative. Other models with adversaries of varying power would thus be useful to consider. For interval uncertainty, an *indifferent adversary*, for example, might randomly select costs with a uniform distribution over the specified intervals. It is easy to see that the optimal minmax strategy for the optimizing player in this setting is to solve the nominal problem with midpoint costs. Yet, it might be possible to define adversaries that lie between this indifferent adversary and the one in the randomized model in terms of power. For instance, an adversary might play the indifferent strategy with some probability and the randomized maximum regret strategy with remaining probability. Another type of adversary is a *computationally bounded adversary* [63, 116]. This is relevant to minmax regret problems under general convex uncertainty sets, where the maximum regret problem is NP-hard. Such an adversary, for example, may only be able to solve polynomial problems and *approximate* NP-hard problems.

Bibliography

[1] Yasin Abbasi, Peter Bartlett, Varun Kanade, Yevgeny Seldin, and Csaba Szepesvári. Online learning in markov decision processes with adversarially chosen transition probability distributions. In *Advances in Neural Information Processing Systems*, pages 2508–2516, 2013.

[2] Gagan Aggarwal, Gagan Goel, Chinmay Karande, and Aranyak Mehta. Online vertex-weighted bipartite matching and single-bid budgeted allocations. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1253–1264. SIAM, 2011.

[3] Asrar Ahmed, Pradeep Varakantham, Yossiri Adulyasak, and Patrick Jaillet. Regret based robust solutions for uncertain markov decision processes. In *Advances in Neural Information Processing Systems*, pages 881–889, 2013.

[4] Hassene Aissi, Cristina Bazgan, and Daniel Vanderpooten. Complexity of the min–max and min–max regret assignment problems. *Operations Research Letters*, 33(6):634–640, 2005.

[5] Hassene Aissi, Cristina Bazgan, and Daniel Vanderpooten. Complexity of the min-max (regret) versions of cut problems. In *Algorithms and Computation*, pages 789–798. Springer, 2005.

[6] Hassene Aissi, Cristina Bazgan, and Daniel Vanderpooten. Approximating min-max (regret) versions of some polynomial problems. In *Computing and Combinatorics*, pages 428–438. Springer, 2006.

[7] Hassene Aissi, Cristina Bazgan, and Daniel Vanderpooten. Approximation of min–max and min–max regret versions of some combinatorial optimization problems. *European Journal of Operational Research*, 179(2):281–290, 2007.

[8] Hassene Aissi, Cristina Bazgan, and Daniel Vanderpooten. Min–max and min–max regret versions of combinatorial optimization problems: A survey. *European Journal of Operational Research*, 197(2):427–438, 2009.

[9] Hassene Aissi, Cristina Bazgan, and Daniel Vanderpooten. General approximation schemes for min–max (regret) versions of some (pseudo-) polynomial problems. *Discrete Optimization*, 7(3):136–148, 2010.

[10] Réka Albert, Hawoong Jeong, and Albert-Lászlό Barabási. Internet: Diameter of the world-wide web. *Nature*, 401(6749):130–131, 1999.

[11] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, pages 20–29. ACM, 1996.

[12] Jonathan Aronson, Martin Dyer, Alan Frieze, and Stephen Suen. Randomized greedy matching II. *Random Structures & Algorithms*, 6(1):55–73, 1995.

[13] Jonathan Aronson, Alan Frieze, and Boris G Pittel. Maximum matchings in sparse random graphs: Karp-Sipser revisited. *Random Structures and Algorithms*, 12(2): 111–177, 1998.

[14] Igor Averbakh. On the complexity of a class of combinatorial optimization problems with uncertainty. *Mathematical Programming*, 90(2):263–272, 2001.

[15] Igor Averbakh and Vasilij Lebedev. Interval data minmax regret network optimization problems. *Discrete Applied Mathematics*, 138(3):289–301, 2004.

[16] Igor Averbakh and Vasilij Lebedev. On the complexity of minmax regret linear programming. *European Journal of Operational Research*, 160(1):227–231, 2005.

[17] Kazuoki Azuma. Weighted sums of certain dependent random variables. *Tohoku Mathematical Journal*, 19(3):357–367, 1967.

[18] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[19] Albert-László Barabási, Réka Albert, and Hawoong Jeong. Scale-free characteristics of random networks: the topology of the world-wide web. *Physica A: Statistical Mechanics and its Applications*, 281(1):69–77, 2000.

[20] Nicole Bäuerle and Ulrich Rieder. *Markov Decision Processes with Applications to Finance*. Springer, 2011.

[21] Rene Beier and Berthold Vöcking. Random knapsack in expected polynomial time. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, pages 232–241. ACM, 2003.

[22] Richard Ernest Bellman. *Dynamic Programming*. Dover Publications, Incorporated, 2003.

[23] Shai Ben-David, Allan Borodin, Richard Karp, Gabor Tardos, and Avi Wigderson. On the power of randomization in on-line algorithms. *Algorithmica*, 11(1): 2–14, 1994.

[24] Aharon Ben-Tal and Arkadi Nemirovski. Robust convex optimization. *Mathematics of Operations Research*, 23(4):769–805, 1998.

[25] Aharon Ben-Tal and Arkadi Nemirovski. Robust solutions of uncertain linear programs. *Operations Research Letters*, 25(1):1–13, 1999.

[26] Luca Bertazzi. Minimum and worst-case performance ratios of rollout algorithms. *Journal of Optimization Theory and Applications*, 152(2):378–393, 2012.

[27] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 3rd edition, 2007.

[28] Dimitri P. Bertsekas and David A. Castanon. Rollout algorithms for stochastic scheduling problems. *Journal of Heuristics*, 5(1):89–108, 1999.

[29] Dimitri P. Bertsekas and John N. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[30] Dimitri P. Bertsekas, John Tsitsiklis, and Cynara Wu. Rollout algorithms for combinatorial optimization. *Journal of Heuristics*, 3(3):245–262, 1997.

[31] Dimitris Bertsimas and Melvyn Sim. The price of robustness. *Operations Research*, 52(1):35–53, 2004.

[32] Dimitris Bertsimas and Melvyn Sim. Robust discrete optimization under ellipsoidal uncertainty sets. *Technical report, Massachusetts Institute of Technology*, 2004.

[33] Dimitris Bertsimas and John N Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific Belmont, MA, 1997.

[34] Dimitris Bertsimas and Robert Weismantel. *Optimization over Integers*. Dynamic Ideas Belmont, 2005.

[35] Dimitris Bertsimas, Ebrahim Nasrabadi, and James B. Orlin. On the power of nature in robust discrete optimization. *In preparation*, 2012.

[36] Dimitris Bertsimas, Ebrahim Nasrabadi, and James B. Orlin. On the power of randomization in network interdiction. *In preparation*, 2012.

[37] John R Birge and François V Louveaux. *Introduction to Stochastic Programming*. Springer, 1997.

[38] Benjamin Birnbaum and Claire Mathieu. On-line bipartite matching made simple. *ACM SIGACT News*, 39(1):80–87, 2008.

[39] Béla Bollobás. *Random Graphs.* Springer, 1998.

[40] Béla Bollobás and Graham Brightwell. The width of random graph orders. *Mathematical Scientist*, 20(2):69–90, 1995.

[41] K. Borgwardt and B. Tremel. The average quality of greedy-algorithms for the subset-sum-maximization problem. *Mathematical Methods of Operations Research*, 35(2):113–149, 1991.

[42] Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis.* Cambridge University Press Cambridge, 1998.

[43] James M Calvin and Joseph Y-T Leung. Average-case analysis of a greedy algorithm for the 0/1 knapsack problem. *Operations Research Letters*, 31(3):202–210, 2003.

[44] Constantin Carathéodory. Über den variabilitätsbereich der fourierschen konstanten von positiven harmonischen funktionen. *Rendiconti del Circolo Matematico di Palermo (1884-1940)*, 32(1):193–217, 1911.

[45] Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, Learning, and Games.* Cambridge University Press Cambridge, 2006.

[46] Dragos Florin Ciocan and Vivek Farias. Model predictive control for dynamic resource allocation. *Mathematics of Operations Research*, 37(3):501–525, 2012.

[47] Edward Grady Coffman and George S Lueker. *Probabilistic Analysis of Packing and Partitioning Algorithms.* Wiley New York, 1991.

[48] Gianfranco D'Atri and Claude Puech. Probabilistic analysis of the subset-sum problem. *Discrete Applied Mathematics*, 4(4):329–334, 1982.

[49] Brian C Dean, Michel X Goemans, and J Vondrdk. Approximating the stochastic knapsack problem: The benefit of adaptivity. In *Foundations of Computer Science, 2004. Proceedings. 45th Annual IEEE Symposium on*, pages 208–217. IEEE, 2004.

[50] Vladimir G Deineko and Gerhard J Woeginger. Pinpointing the complexity of the interval min–max regret knapsack problem. *Discrete Optimization*, 7(4):191–196, 2010.

[51] Karina Valdivia Delgado, Scott Sanner, and Leliane Nunes de Barros. Efficient solutions to factored mdps with imprecise transition probabilities. *Artificial Intelligence*, 175(910):1498 – 1527, 2011.

[52] Nikhil R. Devanur, Kamal Jain, and Robert D. Kleinberg. Randomized primal-dual analysis of ranking for online bipartite matching. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 101–107, 2013.

[53] Gennady Diubin and Alexander Korbut. The average behaviour of greedy algorithms for the knapsack problem: general distributions. *Mathematical Methods of Operations Research*, 57(3):449–479, 2003.

[54] R. F. Drenick. Multilinear programming: Duality theories. *Journal of Optimization Theory and Applications*, 72:459–486, 1992.

[55] Martin Dyer and Alan Frieze. Randomized greedy matching. *Random Structures & Algorithms*, 2(1):29–45, 1991.

[56] Martin Dyer, Alan Frieze, and Boris Pittel. The average performance of the greedy matching algorithm. *The Annals of Applied Probability*, pages 526–552, 1993.

[57] P Erdős and Alfréd Rényi. On random matrices. *Magyar Tud. Akad. Mat. Kutató Int. Közl*, 8(455-461):1964, 1964.

[58] P Erdős and Alfréd Rényi. On the existence of a factor of degree one of a connected random graph. *Acta Mathematica Hungarica*, 17(3):359–368, 1966.

[59] Eyal Even-Dar, Sham M Kakade, and Yishay Mansour. Online Markov decision processes. *Mathematics of Operations Research*, 34(3):726–736, 2009.

[60] Amir Massoud Farahmand, Rémi Munos, and Csaba Szepesvári. Error propagation for approximate policy and value iteration. In *NIPS*, pages 568–576, 2010.

[61] Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theoretical Computer Science*, 348(2):207–216, 2005.

[62] Ricardo Shirota Filho, Fabio Gagliardi Cozman, Felipe Werndl Trevizan, Cassio Polpo de Campos, and Leliane Nunes de Barros. Multilinear and integer programming for markov decision processes with imprecise probabilities. In *Proceedings of the 5th International Symposium on Imprecise Probability: Theories and Applications (ISIPTA), Prague, Czech Republic*, pages 395–404, 2007.

[63] Yoav Freund, Michael Kearns, Yishay Mansour, Dana Ron, Ronitt Rubinfeld, and Robert E Schapire. Efficient algorithms for learning to play repeated games against computationally bounded adversaries. In *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, pages 332–341. IEEE, 1995.

[64] Alan Frieze. Perfect matchings in random bipartite graphs with minimal degree at least 2. *Random Structures & Algorithms*, 26(3):319–358, 2005.

[65] Alan Frieze and Boris Pittel. Perfect matchings in random graphs with prescribed minimal degree. In *Mathematics and Computer Science III*, pages 95–132. Springer, 2004.

[66] Alan Frieze, AJ Radcliffe, and Stephen Suen. Analysis of a simple greedy matching algorithm on random cubic graphs. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 341–351. Society for Industrial and Applied Mathematics, 1993.

[67] Hiroshi Fujiwara and Kazuo Iwama. Average-case competitive analyses for ski-rental problems. *Algorithmica*, 42(1):95–107, 2005.

[68] Guillermo Gallego and Garrett Van Ryzin. A multiproduct dynamic pricing problem and its applications to network yield management. *Operations Research*, 45 (1):24–41, 1997.

[69] Michael R Garey and David S Johnson. *Computers and Intractability.* W. H. Freeman, 2002.

[70] John Gittins, Kevin Glazebrook, and Richard Weber. *Multi-Armed Bandit Allocation Indices.* John Wiley & Sons, Ltd, 2nd edition, 2011.

[71] Robert Givan, Sonia Leach, and Thomas Dean. Bounded-parameter markov decision processes. *Artificial Intelligence*, 122:71–109, 2000.

[72] Geoffrey Grimmett and David Stirzaker. *Probability and Random Processes.* Oxford University Press, 3rd edition, 2001.

[73] Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, 1981.

[74] Pascal Van Hentenryck and Russell Bent. *Online Stochastic Combinatorial Optimization.* The MIT Press, 2009.

[75] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[76] John E Hopcroft and Richard M Karp. An nˆ5/2 algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973.

[77] WJ Hopp. Sensitivity analysis in discrete dynamic programming. *Journal of Optimization Theory and Applications*, 56(2):257–269, 1988.

[78] Chelsea C. White III and Hany K. Eldeib. Markov decision processes with imprecise transition probabilities. *Operations Research*, 42:739–749, 1994.

[79] Svante Janson, Tomasz Luczak, and Andrzej Rucinski. *Random Graphs.* John Wiley & Sons, 2011.

[80] Berit Johannes and James B. Orlin. Minimax regret problems are harder than minimax problems. *Submitted*, 2012.

[81] Richard M Karp and Judea Pearl. Searching for an optimal path in a tree with random costs. *Artificial Intelligence*, 21(1):99–116, 1983.

[82] Richard M Karp and Michael Sipser. Maximum matching in sparse random graphs. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 364–375. IEEE, 1981.

[83] Richard M Karp, Umesh V Vazirani, and Vijay V Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the Twenty-Second Annual ACM Symposium on Theory of Computing*, pages 352–358. ACM, 1990.

[84] Adam Kasperski. *Discrete Optimization with Interval Data: Minmax Regret and Fuzzy Approach.* Springer, 2008.

[85] Adam Kasperski and Paweł Zieliński. An approximation algorithm for interval data minmax regret combinatorial optimization problems. *Information Processing Letters*, 97(5):177–180, 2006.

[86] Adam Kasperski and Paweł Zieliński. On the existence of an fptas for minmax regret combinatorial optimization problems with interval data. *Operations Research Letters*, 35(4):525–532, 2007.

[87] Adam Kasperski, Adam Kurpisz, and Paweł Zieliński. Approximating the min–max (regret) selecting items problem. *Information Processing Letters*, 113(1): 23–29, 2013.

[88] Michael Kearns, Yishay Mansour, and Andrew Y. Ng. A sparse sampling algorithm for near-optimal planning in large markov decision processes. In *Machine Learning*, pages 1324–1331, 1999.

[89] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack Problems.* Springer, 2004.

[90] Jon M Kleinberg, Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew S Tomkins. The web as a graph: Measurements, models, and methods. In *Computing and Combinatorics*, pages 1–17. Springer, 1999.

[91] Dénes König. Über graphen und ihre anwendung auf determinantentheorie und mengenlehre. *Mathematische Annalen*, 77(4):453–465, 1916.

[92] Dmytro Korzhyk, Vincent Conitzer, and Ronald Parr. Complexity of computing optimal stackelberg strategies in security resource allocation games. In *AAAI*, 2010.

[93] Panos Kouvelis and Gang Yu. *Robust Discrete Optimization and Its Applications.* Springer, 1997.

[94] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. Extracting large-scale knowledge bases from the web. In *VLDB*, volume 99, pages 639–650, 1999.

[95] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. Trawling the web for emerging cyber-communities. *Computer networks*, 31(11): 1481–1493, 1999.

[96] Masami Kurano, Masanori Hosaka, Youqiang Huang, and Jinjie Song. Controlled Markov set-chains with discounting. *Journal of Applied Probability*, 35(2):293–302, 1998.

[97] Thomas G Kurtz. Solutions of ordinary differential equations as limits of pure jump Markov processes. *Journal of Applied Probability*, 7(1):49–58, 1970.

[98] Yun Li, Lucas W Krakow, Edwin KP Chong, and Kenneth N Groom. Approximate stochastic dynamic programming for sensor scheduling to track multiple targets. *Digital Signal Processing*, 19(6):978–989, 2009.

[99] George S Lueker. Average-case analysis of off-line and on-line knapsack problems. In *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 179–188. Society for Industrial and Applied Mathematics, 1995.

[100] Vahideh H Manshadi, Shayan Oveis Gharan, and Amin Saberi. Online stochastic matching: Online actions based on offline statistics. *Mathematics of Operations Research*, 37(4):559–573, 2012.

[101] Silvano Martello and Paolo Toth. Worst-case analysis of greedy algorithms for the subset-sum problem. *Mathematical Programming*, 28(2):198–205, 1984.

[102] Silvano Martello and Paolo Toth. *Knapsack Problems: Algorithms and Computer Implementations.* John Wiley & Sons, Inc., 1990.

[103] Andrew Mastin and Patrick Jaillet. Loss bounds for uncertain transition probabilities in Markov decision processes. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 6708–6715. IEEE, 2012.

[104] Andrew Mastin and Patrick Jaillet. Greedy online bipartite matching on random graphs. *(working paper)*, 2014.

[105] Andrew Mastin and Patrick Jaillet. Average-case performance of rollout algorithms for knapsack problems. *Journal of Optimization Theory and Applications*, pages 1–21, 2014.

[106] Andrew Mastin, Patrick Jaillet, and Sang Chin. Randomized minmax regret for combinatorial optimization under uncertainty. *(working paper)*, 2014.

[107] Aranyak Mehta, Amin Saberi, Umesh Vazirani, and Vijay Vazirani. Adwords and generalized online matching. *Journal of the ACM (JACM)*, 54(5):22, 2007.

[108] Luc Mercier and Pascal Van Hentenryck. Performance analysis of online anticipatory algorithms for large multistage stochastic integer programs. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, IJCAI'07, pages 1979–1984. Morgan Kaufmann Publishers Inc., 2007.

[109] Vahab Mirrokni, Nithum Thain, and Adrian Vetta. On the implications of lookahead search in game playing. *arXiv preprint arXiv:1202.4134*, 2012.

[110] Michael Mitzenmacher. Studying balanced allocations with differential equations. *Combinatorics Probability and Computing*, 8(5):473–482, 1999.

[111] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis.* Cambridge University Press, 2005.

[112] Cristopher Moore and Stephan Mertens. *The Nature of Computation.* Oxford University Press, 2011.

[113] Alfred Müller. How does the value function of a markov decision process depend on the transition probabilities? *Mathematics of Operations Research*, 22(4):872–885, 1997.

[114] Rémi Munos. Error bounds for approximate policy iteration. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 560–567, 2003.

[115] Mark Newman. *Networks: An Introduction.* Oxford University Press, 2010.

[116] Abraham Neyman and Daijiro Okada. Strategic entropy and complexity in repeated games. *Games and Economic Behavior*, 29(1):191–223, 1999.

[117] Arnab Nilim and Laurent El Ghaoui. Robust control of Markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798, 2005.

[118] Ulrich Pferschy. Stochastic analysis of greedy algorithms for the subset sum problem. *Central European Journal of Operations Research*, 7:53–70, 1999.

[119] Boris Pittel. On tree census and the giant component in sparse random graphs. *Random Structures & Algorithms*, 1(3):311–342, 1990.

[120] Boris Pittel, Joel Spencer, and Nicholas Wormald. Sudden emergence of a giant k-core in a random graph. *Journal of Combinatorial Theory, Series B*, 67(1): 111–151, 1996.

[121] Matthias Poloczek and Mario Szegedy. Randomized greedy algorithms for the maximum matching problem with new analysis. In *Foundations of Computer Science (FOCS), 2012 IEEE 53rd Annual Symposium on*, pages 708–717. IEEE, 2012.

[122] Warren B. Powell. *Approximate dynamic programming.* John Wiley and Sons, Inc., 2nd edition, 2011.

[123] Martin L Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* John Wiley & Sons, 2009.

[124] Monika R Henzinger Prabhakar Raghavan. Computing on data streams. In *External Memory Algorithms: DIMACS Workshop External Memory Algorithms and Visualization, May 20-22, 1998*, volume 50, page 107. AMS Bookstore, 1999.

[125] Sartaj Sahni. Approximate algorithms for the 0/1 knapsack problem. *Journal of the ACM (JACM)*, 22(1):115–124, 1975.

[126] Jay K Satia and Roy E Lave Jr. Markovian decision processes with uncertain transition probabilities. *Operations Research*, 21(3):728–740, 1973.

[127] Leonard J Savage. The theory of statistical decision. *Journal of the American Statistical association*, 46(253):55–67, 1951.

[128] Alexander Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency.* Springer, 2003.

[129] Nicola Secomandi. A rollout policy for the vehicle routing problem with stochastic demands. *Operations Research*, 49(5):796–802, 2001.

[130] Robert Sedgewick and Philippe Flajolet. *An Introduction to the Analysis of Algorithms.* Addison-Wesley, 2013.

[131] Alexander Shapiro, Darinka Dentcheva, and Andrzej P Ruszczyński. *Lectures on Stochastic Programming: Modeling and Theory.* SIAM, 2009.

[132] Edward Allan Silver. Markovian decision processes with uncertain transition probabilities or rewards. Technical Report AD0417150, Massachusetts Institute of Technology, August 1963.

[133] David Simchi-Levi, Xin Chen, and Julien Bramel. *The Logic of Logistics: Theory, Algorithms, and Applications for Logistics Management.* Springer, 3rd edition, 2013.

[134] Satinder P. Singh and Richard C. Yee. An upper bound on the loss from approximate optimal-value functions. *Machine Learning*, 16(3):227–233, 1994.

[135] Richard S Sutton and Andrew G Barto. *Introduction to Reinforcement Learning*. MIT Press, 1998.

[136] Krzysztof Szkatula and Marek Libura. Probabilistic analysis of simple algorithms for binary knapsack problem. *Control and Cybernetics*, 12:147–157, 1983.

[137] Krzysztof Szkatula and Marek Libura. On probabilistic properties of greedy-like algorithms for the binary knapsack problem. *Proceedings of Advanced School on Stochastics in Combinatorial Optimization*, pages 233–254, 1987.

[138] Chin Hon Tan and Joseph C. Hartman. *Sensitivity Analysis and Dynamic Programming*. John Wiley & Sons, Inc., 2010.

[139] Chin Hon Tan and Joseph C. Hartman. Sensitivity analysis in markov decision processes with uncertain reward parameters. *Journal of Applied Probability*, 48 (4):954 – 967, 2011.

[140] Gerald Tesauro and Gregory R Galperin. On-line policy improvement using monte-carlo search. *Advances in Neural Information Processing Systems*, pages 1068–1074, 1997.

[141] Fang Tu and Krishna Pattipati. Rollout strategies for sequential fault diagnosis. In *AUTOTESTCON Proceedings*, pages 269–295. IEEE, 2002.

[142] Remco van der Hofstad. *Random Graphs and Complex Networks. Vol. I*. http://www.win.tue.nl/~rhofstad/NotesRGCN.pdf, 2014.

[143] Vijay Vazirani. *Approximation Algorithms*. Springer, 2001.

[144] John von Neumann. Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, 100(1):295–320, 1928.

[145] Abraham Wald. Contributions to the theory of statistical estimation and testing hypotheses. *The Annals of Mathematical Statistics*, 10(4):299–326, 1939.

[146] David P Williamson and David B Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.

[147] Nicholas C Wormald. Differential equations for random processes and random graphs. *The Annals of Applied Probability*, pages 1217–1235, 1995.

[148] Nicholas C Wormald. The differential equation method for random graph processes and greedy algorithms. *Lectures on approximation and randomized algorithms*, pages 73–155, 1999.

[149] Nicholas C Wormald. Models of random regular graphs. *London Mathematical Society Lecture Note Series*, pages 239–298, 1999.

[150] Gang Yu and Jian Yang. On the robust shortest path problem. *Computers & Operations Research*, 25(6):457–468, 1998.