Optimal Resource Allocation in a Dynamic and Stochastic Environment: A Mathematical Programming Approach

by

José Niño Mora

Licenciado in Mathematical Sciences (1989)
Universidad Complutense de Madrid, Spain

Submitted to the Sloan School of Management
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Operations Research

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1995

Author ....
Sloan School of Management
May 25, 1995

Certified by ....
Dimitris Bertsimas
Professor of Operations Research
Thesis Supervisor

Accepted by ....
Thomas L. Magnanti
George Eastman Professor of Management Science
Co-director, Operations Research Center

Optimal Resource Allocation in a Dynamic and Stochastic Environment: A Mathematical Programming Approach

by

José Niño Mora

Submitted to the Sloan School of Management
on May 25, 1995, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Operations Research

Abstract

We develop a mathematical programming approach for formulating and solving optimal dynamic and stochastic resource allocation problems. The approach is based on formulating an optimal resource allocation problem as a mathematical program over the region spanned by performance vectors achievable under admissible resource allocation policies (performance region). These formulations are constructed in a unifying way, by identifying physical laws (conservation laws) satisfied by the underlying system, and expressing them as linear constraints on performance vectors. The set of constraints so obtained defines a polyhedron that contains the performance region, thus yielding a relaxed formulation for the problem. In some cases, that polyhedron coincides with the performance region, in which case the formulation is exact. Solving the resulting mathematical program yields bounds on the optimal value of the problem, which can be used to assess the degree of suboptimality of proposed resource allocation policies. It also yields, when the formulations are exact, optimal policies for the problem. The approach is developed in three problem areas in the field of stochastic scheduling, with an increasing level of complexity: scheduling problems solved by simple priority-index rules, such as multi-armed bandit problems, restless bandit problems, and problems of dynamic scheduling in Markovian multiclass queueing networks. For the first class of problems, a unified framework for constructing and solving exact linear programming formulations is developed, based on a generalized concept of conservation laws. It is shown that the strong structural properties of these linear programs translate into corresponding properties of the scheduling problems. For the restless bandit problem, a sequence of increasingly stronger linear programming relaxations is constructed, by applying conservation laws satisfied by underlying Markov decision chains. A primal-dual heuristic is proposed, based on the optimal solution to the first-order relaxation. For dynamic scheduling problems in multiclass queueing networks, we identify a set of flow conservation laws, and apply them for constructing a sequence of increasingly stronger linear programming relaxations.

Thesis Supervisor: Dimitris Bertsimas
Title: Professor of Operations Research

A mis padres

Acknowledgments

I would like to express my gratitude and appreciation to the many people who have contributed to turning my years as a PhD student at MIT into a great growing experience.

Especial thanks to my PhD Advisor, Prof. Dimitris Bertsimas, for his support and encouragement during the ups and downs of my apprenticeship as a budding scientist. Thanks also for his unfailing enthusiasm for new research ideas, insistence on high quality standards, and generosity in sharing his ideas and giving me opportunities for growing as a scientist.

Thanks to Prof. Thomas Magnanti and Prof. John Tsitsiklis who, as members of my PhD Thesis Committee, provided constructive criticism on the research that led to this dissertation.

Thanks to the faculty, staff and fellow students, that make of the Operations Research Center at MIT a great and unique learning environment. Especial thanks to Paulette, Cheryl and Laura, and to Chungpiaw, Hitendra, Eri, Gina, Andrew, Rafael, Augusto, Kerry, Joe, David, Sarah, Thalia, Elaine, Charu and Mitchell.

Thanks to my friends in Boston, especially to Pratap, Hulya, Ray and Tara Misra, for their hospitality in making me feel at home since my arrival to MIT; thanks too to Yuri, María, Mila, Juan Carlos, Gemma, Isela, Marga, Mauricio, Joaquín, Guillermo, José Luis, Olivera, Tambre and many other friends, for sharing this adventure and enriching my life with new perspectives.

A warm thank you to Susan, for bringing new life to my musical, culinary, and so many other interests.

A especial thank you to my aunt Mary, my uncle Pepe, and my cousins Mary Angeles, Mary Cinti, and Mary Jose, for their constant encouragement, in spite of all those miles in between us.

Last, but first for me, I want to thank my parents, for the completion of this doctorate is a success for them as much as it is for me: it is really a dream come true. I thank them for always believing in me, for their unfailing love, support and encouragement, and for teaching me, by example, the most important things.

This research was partially supported by a Ministry of Education and Science of Spain / Fulbright Doctoral Fellowship, and by NSF grant DDM-9158118.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and background

1.1 Introduction

In a wide variety of real-world settings one has to decide how to allocate over time scarce resources to alternative activities with uncertain outcomes. Consider the problem faced by the manager of a commercial laboratory with several research projects vying for the attention of the scientists at her disposal. She must determine how to allocate research effort to the projects, in a sequential manner, in order to maximize the resulting expected reward. Similar problems are faced by a police department that tries to control a number of illegal drug markets by deploying dynamically its police units, or by the manager of a manufacturing facility or a computer-communication network, whose goal is to optimize system performance by controlling the way jobs are scheduled for processing.

It is to problems like these that the methods developed in this dissertation can be applied. They concern a system in which effort may be spent over time in alternative ways, thus guiding its evolution. Relevant features of the long-run operation of the system are captured by suitable measures of performance. Furthermore, rewards and/or costs are associated with effort allocation decisions. The two fundamental questions we shall address are:

1. How should effort be allocated in order to achieve a specified performance objective?

2. What is the best performance objective that can be achieved?

Answers to these questions are important for improving the productivity of technological systems: an answer to the first question would provide guidelines for designing good resource

allocation policies: an answer to the second question could be used to assess a priori the relative performance of alternative system designs, as well as a benchmark for evaluating the performance of proposed policies.

Real-world resource allocation problems are represented mathematically by dynamic and stochastic optimization models, which have traditionally been formulated within a dynamic programming framework. The resulting formulations, based on Bellman's (1957) optimality equation, exhibit typically a prohibitively large —even infinite— size, which hinders their application.

Consequently, progress in the area of stochastic resource allocation has been hard won. In a number of important problems, researchers have overcome that difficulty, obtaining insightful structural results that have led to efficient solution procedures. A celebrated example of these successes was the solution by Gittins and Jones (1974) of the classical multi-armed bandit problem by a simple priority-index policy. This and other results have been derived using an array of techniques, such as interchange arguments, that exploit the structure of the problem at hand. A disadvantage of this ad hoc approach is that every problem seems to require its own solution techniques. Moreover, when heuristic policies are proposed, their performance is often compared with that of other heuristics, instead of with the optimal performance itself. This unsatisfactory state of affairs has prompted Lawler, Lenstra, Rinnooy Kan and Shmoys (1989) to remark:

> The results in stochastic scheduling are scattered and they have been obtained through a considerable and sometimes disheartening effort. In the words of Coffman, Hofri and Weiss (1989), there is great need for new mathematical techniques useful for simplifying the derivation of results.

It is this need for new simplifying techniques that motivates the present work. Since dynamic programming formulations are often too general to exploit special structure, and ad hoc techniques are too specialized, the approach we shall develop pursues a middle ground: to identify physical properties shared by relevant resource allocation models, and to apply them in order to construct improved mathematical programming formulations. The properties we have sought to identify are conservation laws: physical relations in the underlying systems that remain invariant under different resource allocation policies.

Given a stochastic resource allocation problem, the solution approach we propose may be articulated as follows:

1. Define suitable performance measures. The objective to be optimized must be a function of the corresponding performance vector.

2. Identify physical laws (conservation laws) satisfied by the system. These conservation laws must be expressed as linear equalities or inequalities satisfied by performance vectors.

3. Formulate the problem as a mathematical program. The feasible region of this program is a polyhedron defined by the set of linear constraints given by the conservation laws. This polyhedron contains or, in some cases, coincides with, the region spanned by performance vectors achievable under admissible resource allocation policies (performance region).

4. Translate a solution to the mathematical programming formulation into a corresponding solution to the resource allocation problem. The formulation should yield bounds on the optimal problem value, as well as optimal or heuristic resource allocation policies.

The purpose of this dissertation is to demonstrate the effectivenes of such an approach. We have thus tested it in three important problems areas, with an increasing level of complexity, in the field of stochastic resource allocation:

1. Stochastic scheduling problems solved by simple priority-index rules, such as the multi-armed bandit problem.

2. Restless bandit problems.

3. Optimal scheduling problems in queueing networks with multiple job classes.

The conservation laws we identify yield exact formulations in the well-solved problems in the first area, whereas they provide relaxed formulations in the computationally intractable problems in the second and third areas.

The thesis is structured as follows: In the remainder of this chapter we survey previous work in the mathematical programming approach to stochastic scheduling problems. Chapter 2 develops structural and algorithmic properties of certain polyhedra (extended polymatroids) that arise as the performance region in a variety of scheduling problems, thus laying the groundwork for the approach to be presented in the next chapter.

Chapter 3 addresses stochastic scheduling problems that are solved by priority-index rules (indexable scheduling problems). A unified framework for formulating exactly such problems as linear programs is developed, based on a generalized notion of conservation laws. Properties of the resulting linear programs are shown to translate into corresponding properties of the scheduling problems.

Chapter 4 applies that framework to a wide variety of indexable scheduling problems, which correspond to special cases of the versatile model of branching bandits. Several performance measures for branching bandits are shown to satisfy generalized conservation laws, thus casting the corresponding problems into the framework presented in Chapter 3. The resulting exact formulations and solutions are developed in the special cases of multi-armed bandits, multiclass queues with feedback, and scheduling problems, obtaining new results and insights in the process.

Chapter 5 deals with a computationally intractable variant of the classical multi-armed bandit problem: the restless bandit problem. A sequence of increasingly stronger compact linear programming formulations is developed, by identifying conservation laws that account for increasingly higher-order interactions in the system. This sequence of formulations is interpreted geometrically as corresponding to a lift-and-project procedure: each formulation is lifted into a higher-dimensional space (introducing new variables), and then projected back into the original space, thus obtaining the next one. A priority-index heuristic policy is also proposed, where the indices are computed from the optimal solution to the first-order formulation. The quality of the performance bounds given by the formulations and of the heuristic is investigated empirically.

Chapter 6 is concerned with optimal scheduling problems in queueing networks with multiple job classes. A classical flow conservation law of queueing systems is applied to construct a sequence of increasingly stronger compact linear programming formulations, which yield correspondingly tighter bounds on optimal performace. Analogously as in the previous chapter, these sequence of formulations is shown to correspond geometrically to a lift-and-project procedure.

Chapter 7 provides a summary of our work, includes our conclusions and discusses directions for further research.

Appendix A reviews some basic concepts and results from the Palm calculus of point processes. These results are applied in Chapter 6 in order to translate certain flow conser-

vation laws into corresponding linear constraints on performance vectors.

1.2 Background

The mathematical programming approach to stochastic resource allocation problems was pioneered by Gelenbe and Mitrani (1980), and Coffman and Mitrani (1980). Investigating the optimal dynamic scheduling problem in a multiclass $M/G/1$ queue with linear delay costs, they associated with every scheduling policy u a corresponding performance vector of mean delays $(\bar{W}_1^u, \ldots, \bar{W}_n^u)$ for all customer classes. By identifying work-conservation laws of the system and expressing them as linear constraints on performance vectors, they were able to characterize the performance region as a polyhedron well known in polyhedral combinatorics (a polymatroid). The vertices of such a polyhedron were shown to correspond to performance vectors of index-priority rules, thus providing a new geometric explanation for the optimality of such policies.

Federgruen and Groenevelt (1988a), (1988b) extended the approach to multiclass $M/G/c$ and $G/M/c$ queues, showing that their performance regions are also polymatroids. A unified axiomatic framework for these earlier results was developed by Shanthikumar and Yao (1992), who identified physical properties of the system (strong conservation laws) that guarantee the performance region to be a polymatroid.

Tsoucas (1991) addressed a problem that does not fit in the Shanthikumar and Yao framework: The optimal dynamic scheduling of a multiclass $M/G/1$ queue with Bernouilli feedback, first solved by Klimov (1974) by a priority-index rule. By identifiying work-conservation laws and expressing them as linear constraints, he characterized the performance region of mean delays as a new kind of polyhedron, with properties that generalize those of polymatroids (a so-called extended polymatroid). He then rederived Klimov's algorithm for finding the optimal priority ranking using linear programming arguments.

Bertsimas, Paschalidis and Tsitsiklis (1994) investigated the optimal scheduling problem in Markovian multiclass queueing networks. Using potential function ideas, they developed an algebraic procedure for constructing a sequence of increasingly stronger linear and nonlinear programming relaxations for the problem. Kumar and Kumar (1994) developed independently some of their results. The constraints that appear in these linear programs are not given, however, a physical interpretation.

Chapter 2

Extended polymatroids

2.1 Introduction

Extended polymatroids are polyhedra whose role in the field of stochastic scheduling is analogous to that played by classical polymatroids (see Edmonds (1970)) in combinatorial optimization. Polymatroids arise as the convex hull of feasible solutions in combinatorial optimization problems solved by a greedy algorithm, such as that of finding the minimum spanning tree in a graph. Similarly, extended polymatroids appear as the convex hull of performance vectors achievable under admissible scheduling policies (i.e., as the performance region) in stochastic scheduling problems solved by priority-index policies, including multi-armed bandit problems. Problems such as finding a minimum spanning tree or scheduling optimally a multi-armed bandit can thus be formulated as linear programs over polymatroids or extended polymatroids, respectively. These linear programs possess strong structural and algorithmic properties, which explain in a unifying way the optimality of greedy-like solution schemes for the problems they represent. They further provide insight on how to solve variations on the original problems, such as incorporating a nonlinear objective function, or imposing side constraints.

In the remainder of this section we establish the notation to be used, and present the formal definition of extended polymatroid. Section 2.2 develops the theory of linear programs over such polyhedra. Section 2.3 presents a class of extended polymatroids which can be reformulated explicitly as the projection of a higher dimensional polyhedron with a polynomial — as opposed to exponential — number of facets.

Let $\mathcal{N} = \{1, \ldots, n\}$ be a finite set. Let x denote a real n-vector, with components x_i,

for $i \in \mathcal{N}$. For $S \subseteq N$, let $S^c = \mathcal{N} \setminus S$, and let $|S|$ denote the cardinality of S. Let x_S be the subvector of x corresponding to components in S, i.e., $x_S = (x_i)_{i \in S}$. Let $2^{\mathcal{N}}$ denote the class of all subsets of $\mathcal{N}$. Let $b: 2^{\mathcal{N}} \to \Re_+$ be a nonnegative set function that satisfies $b(\emptyset) = 0$. Let $A = (A_i^S)_{i \in \mathcal{N},\ S \subseteq \mathcal{N}}$ be a matrix that satisfies

$$A_i^S > 0, \qquad \text{for } i \in S \text{ and } S \subseteq \mathcal{N}. \tag{2.1}$$

Given a permutation $\pi = (\pi_1, \ldots, \pi_n)$ of $\mathcal{N}$, and a vector $x = (x_1, \ldots, x_n)'$ let us write $x_\pi = (x_{\pi_1}, \ldots, x_{\pi_n})'$, and

$$b_\pi = (b(\{\pi_1, \ldots, \pi_n\}, \ldots, b(\{\pi_{n-1}, \pi_n\}), b(\{\pi_n\}))' .$$

Let A_π be the upper triangular submatrix of A given by

$$A_\pi = \begin{pmatrix} A_{\pi_1}^{\{\pi_1,\ldots,\pi_n\}} & \cdots & A_{\pi_{n-1}}^{\{\pi_1,\ldots,\pi_n\}} & A_{\pi_n}^{\{\pi_1,\ldots,\pi_n\}} \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & A_{\pi_{n-1}}^{\{\pi_{n-1},\pi_n\}} & A_{\pi_n}^{\{\pi_{n-1},\pi_n\}} \\ 0 & \cdots & 0 & A_{\pi_n}^{\{\pi_n\}} \end{pmatrix}.$$

Let $v(\pi)$ denote the unique solution of triangular system

$$\begin{array}{rcccl} A_{\pi_1}^{\{\pi_1,\ldots,\pi_n\}} x_{\pi_1} + \cdots + & A_{\pi_{n-1}}^{\{\pi_1,\ldots,\pi_n\}} x_{\pi_{n-1}} & + & A_{\pi_n}^{\{\pi_1,\ldots,\pi_n\}} x_{\pi_n} & = b(\{\pi_1, \ldots, \pi_n\}) \\ \ddots & \vdots & & \vdots & \vdots \\ & A_{\pi_{n-1}}^{\{\pi_{n-1},\pi_n\}} x_{\pi_{n-1}} & + & A_{\pi_n}^{\{\pi_{n-1},\pi_n\}} x_{\pi_n} & = b(\{\pi_{n-1}, \pi_n\}) \\ & & & A_{\pi_n}^{\{\pi_n\}} x_{\pi_n} & = b(\{\pi_n\}) \end{array} \tag{2.2}$$

Consider now the following polyhedra associated with matrix A and set function b:

$$\mathcal{P}_c(A, b) = \left\{ x \in \Re_+^n : \sum_{i \in S} A_i^S x_i \geq b(S), \quad \text{for } S \subset \mathcal{N} \quad \text{and} \quad \sum_{i \in \mathcal{N}} A_i^{\mathcal{N}} x_i = b(\mathcal{N}) \right\}$$

and

$$\mathcal{P}(A, b) = \left\{ x \in \Re_+^n : \sum_{i \in S} A_i^S x_i \leq b(S), \quad \text{for } S \subset \mathcal{N} \quad \text{and} \quad \sum_{i \in \mathcal{N}} A_i^{\mathcal{N}} x_i = b(\mathcal{N}) \right\}.$$

Polyhedra $\mathcal{P}_c(A, b)$ and $\mathcal{P}(A, b)$ possess, under the consistency assumption on parameters A

and b we present next, strong structural and algorithmic properties, that generalize those of classical polymatroids.

Assumption 1 *For every permutation π of $\mathcal{N}$, $v(\pi) \in \mathcal{P}(A,b)$.*

Definition 1 (Extended Polymatroid) We say that polyhedron $\mathcal{P}(A,b)$ is an *extended polymatroid* with ground set $\mathcal{N}$, if Assumption 1 holds:
If Assumption 1 holds for polyhedron $\mathcal{P}_c(A,b)$, we say that $\mathcal{P}_c(A,b)$ is an *extended contra-polymatroid.*

Remark. Extended polymatroids were introduced by Tsoucas (1991), who characterized the region of achievable performance in Klimov's problem (see Klimov (1974)) as a polyhedron with special structure, not previously identified in the literature. He established that the optimality of static-priority policies under linear performance objectives, first proven by Klimov (1974) using dynamic programming arguments, follows from structural properties of such polyhedra. Bhattacharya, Georgiadis and Tsoucas (1992) called this polyhedron an *extended polymatroid*, and developed further its properties.

2.2 Linear programming over extended polymatroids

Extended polymatroids are polyhedra defined by an exponential number of inequality constraints. This exponential-size representation turns out to be "nice," however, as linear programs over extended polymatroids possess strong structural properties that allow them to be efficiently solved.

In this section we develop structural properties of such linear programs. First, we present a new duality proof that such a linear program is solved by a one-pass adaptive greedy algorithm. It is then shown that its optimal solutions are characterized by certain *allocation indices* defined as sums of optimal dual variables. Finally, we identify a condition under which these indices exhibit a decomposition property, which simplifies their computation. The significance of these results in the field of stochastic scheduling is that they explain in a unifying way corresponding properties of indexable scheduling problems, as the next section will demonstrate.

Let us thus consider the problem of maximizing a given linear reward objective $\sum_{i\in\mathcal{N}} r_i x_i$

over extended contra-polymatroid $\mathcal{P}_c(A,b)$,

$$
\begin{aligned}
(LP_c) \qquad Z = \quad \max \quad & \sum_{i\in\mathcal{N}} r_i x_i \\
\text{subject to} \quad & \\
& \sum_{i\in S} A_i^S x_i \geq b(S), \qquad \text{for } S \subset \mathcal{N} \\
& \sum_{i\in\mathcal{N}} A_i^{\mathcal{N}} x_i = b(\mathcal{N}) \\
& x_i \geq 0, \qquad \text{for } i \in \mathcal{N}.
\end{aligned}
$$

Since $\mathcal{P}_c(A,b)$ is a nonempty bounded polyhedron, linear program (LP_c) must have a finite optimal solution. Therefore, its dual program —by strong linear programming duality— will have the same optimum value Z. We shall have a dual variable y^S for every $S \subseteq \mathcal{N}$. The dual program of (LP_c) is

$$
\begin{aligned}
(LD_c) \qquad Z = \quad \min \quad & \sum_{S\subseteq\mathcal{N}} b(S) y^S \\
\text{subject to} \quad & \\
& \sum_{S:\mathcal{N}\supseteq S\ni i} A_i^S y^S \geq r_i, \qquad \text{for } i \in \mathcal{N} \\
& y^S \leq 0, \qquad \text{for } S \subset \mathcal{N}.
\end{aligned}
$$

Adaptive greedy algorithm

We present next a one-pass adaptive greedy algorithm for solving linear program (LP_c) and its dual (LD_c). The input data for the algorithm consists of reward vector $r = (r_i)_{i\in\mathcal{N}}$, and an *oracle* (see Grötschel, Lovász and Schrijver (1988)) that produces the value A_i^S when called with input (i,S). Its output includes a ranking permutation π of ground set $\mathcal{N}$, an optimal dual solution $\bar{y}$, and optimal allocation indices $\{\gamma_i\}_{i\in\mathcal{N}}$ that, as will be seen, characterize optimal solutions to (LP_c).

As shown in Figure 2-1, the algorithm constructs in n steps a vector $\bar{y} = (\bar{y}^S)_{S\subseteq\mathcal{N}}$, which will later be proved to be an optimal solution to program (LD_c). This dual solution has at most n nonzero components, which correspond to a nested (laminar) family (see e.g. Schrijver (1986))

$$S_1 \subset S_2 \subset \cdots \subset S_n = \mathcal{N}.$$

The algorithm identifies in an adaptive greedy manner a permutaton $\pi = (\pi_1, \ldots, \pi_n)$ of

Input: (r, A).

Output: $(\pi, \bar{y}, \gamma)$, where $\pi = (\pi_1, \ldots, \pi_n)$ is a permutation of $\mathcal{N}$, $\bar{y} = \left(\bar{y}^S\right)_{S \subseteq \mathcal{N}}$ and $\gamma = (\gamma_1, \ldots, \gamma_n)$.

Step 0. Set $S_1 = \mathcal{N}$; set $\bar{y}^{S_1} = \max\left\{ \frac{r_i}{A_i^{S_1}} : i \in S_1 \right\}$;

pick $\pi_1 \in \operatorname{argmax}\left\{ \frac{r_i}{A_i^{S_1}} : i \in S_1 \right\}$;

set $\gamma_{\pi_1} = \bar{y}^{S_1}$.

Step k. For $k = 2, \ldots, n$:

set $S_k = S_{k-1} \setminus \{\pi_{k-1}\}$; set $\bar{y}^{S_k} = \max\left\{ \frac{r_i - \sum_{j=1}^{k-1} A_i^{S_j} \bar{y}^{S_j}}{A_i^{S_k}} : i \in S_k \right\}$;

pick $\pi_k \in \operatorname{argmax}\left\{ \frac{r_i - \sum_{j=1}^{k-1} A_i^{S_j} \bar{y}^{S_j}}{A_i^{S_k}} : i \in S_k \right\}$;

set $\gamma_{\pi_k} = \gamma_{\pi_{k-1}} + \bar{y}^{S_k}$.

Step n. For $S \subseteq \mathcal{N}$: set

$$\bar{y}^S = 0, \qquad \text{if } S \notin \{S_1, \ldots, S_n\}.$$

Figure 2-1: Adaptive greedy algorithm $\mathcal{AG}$.

$\mathcal{N}$ such that, defining $S_k = \{\pi_k, \ldots, \pi_n\}$ for $k = 1, \ldots, n$, the unique solution of triangular system

$$\begin{array}{rcl} A_{\pi_1}^{S_1} y^{S_1} & = & r_{\pi_1} \\ \vdots \quad \ddots & & \vdots \\ A_{\pi_{n-1}}^{S_1} y^{S_1} + \cdots + A_{\pi_{n-1}}^{S_{n-1}} y^{S_{n-1}} & = & r_{\pi_{n-1}} \\ A_{\pi_n}^{S_1} y^{S_1} + \cdots + A_{\pi_n}^{S_{n-1}} y^{S_{n-1}} + A_{\pi_n}^{S_n} y^{S_n} & = & r_{\pi_n} \end{array} \tag{2.3}$$

satisfies

$$\bar{y}^{S_k} \leq 0, \qquad \text{for } k = 2, \ldots, n.$$

A corresponding primal solution is obtained by complementary slackness, as the solution $v(\pi)$ of system (2.2). We present next an optimality proof, based on linear programming duality.

Proposition 1 (Optimality of adaptive greedy algorithm) *Let $(\pi, \bar{y}, \gamma)$ be an output of algorithm $\mathcal{AG}$. Then $v(\pi)$ and $\bar{y}$ are an optimal primal-dual pair for linear programs (LP_c) and (LD_c).*

Proof

We shall show that $v(\pi)$ and $\bar{y}$ are primal and dual feasible solutions, respectively, and that they satisfy complementary slackness. We shall first show by induction that $\bar{y}$ is dual

feasible. By definition of $\bar{y}^{S_1}$ in $\mathcal{AG}$, it follows that

$$r_i - A_i^{S_1}\bar{y}^{S_1} \leq 0, \qquad \text{for } i \in S_1,$$

and since $S_2 \subset S_1$ we must have $\bar{y}^{S_2} \leq 0$.

Similarly, for $k = 2, \ldots, n$, we have, by definition of $\bar{y}^{S_k}$,

$$r_i - \sum_{j=1}^{k} A_i^{S_j}\bar{y}^{S_j} \leq 0, \qquad \text{for } i \in S_k,$$

and since $S_{k+1} \subset S_k$, it follows that $\bar{y}^{S_{k+1}} \leq 0$. Therefore $\bar{y}^{S_j} \leq 0$, for $j = 2, \ldots, n$, and by definition of $\overline{y}$, it follows that $\overline{y}^S \leq 0$, for $S \subset \mathcal{N}$.

We also have, by construction,

$$\sum_{S:\mathcal{N}\supseteq S\ni\pi_k} A_{\pi_k}^S \overline{y}^S = \sum_{j=1}^{k} A_{\pi_k}^{S_j}\bar{y}^{S_j} = r_{\pi_k}, \qquad \text{for } k = 1, \ldots, n.$$

Therefore $\overline{y}$ is a dual feasible solution.

Now, by definition of extended contra-polymatroid, $v(\pi)$ is a primal feasible solution. Let $\overline{x} = v(\pi)$. Let us show that $\overline{x}$ and $\bar{y}$ satisfy complementary slackness. Assume $\overline{y}^S \neq 0$ for some $S \subseteq \mathcal{N}$. Then, by construction it must be the case that $S = S_k = \{\pi_k, \ldots, \pi_n\}$, for some $k \in \mathcal{N}$. Now, since $\overline{x}$ is the solution of system (2.2), it follows that

$$\sum_{i\in S} A_i^S \overline{x}_i = \sum_{j=k}^{n} A_{\pi_j}^{\{\pi_k,\ldots,\pi_n\}}\overline{x}_{\pi_j} = b(S).$$

Therefore, by strong linear programing duality, $v(\pi)$ and $\overline{y}$ are an optimal primal-dual pair, which completes the proof. □

Remarks:

1. The result that adaptive greedy algorithm $\mathcal{AG}$ solves program (LP_c) was first established by Tsoucas (1991), who provided a direct optimality proof — not involving duality theory—. He showed that when the underlying extended polymatroid corresponds to the performance region in Klimov's (1974) queueing model, algorithm $\mathcal{AG}$ yields classical Klimov's algorithm for computing the optimal priority scheduling

policy.

2. The running time of algorithm $\mathcal{AG}$ is polynomial in a well-defined sense. Given its input (r, A), the algorithm performs $O(n^3)$ multiplications and $O(n^2)$ pairwise comparisons. The number of required calls to the oracle that produces the A_i^S's is $O(n^2)$. Therefore, $\mathcal{AG}$ is an *oracle-polynomial-time* algorithm (see Grötschel et al. (1988)). In some applications the oracle that returns the A_i^S's runs in polynomial time in the size of the model data, in which case the algorithm is polynomial in the usual sense.

3. Recently, Bertsimas and Teo (1994) have developed a unified primal-dual approximation algorithm for problems of the covering type. When specialized to the case of extended polymatroids, this algorithm coincides with adaptive greedy algorithm $\mathcal{AG}$.

The optimality of adaptive greedy algorithm allows us to characterize explicitly the vertices of an extended polymatroid.

Theorem 1 (Characterization of extreme points) *The set of extreme points of extended contra-polymatroid $\mathcal{P}_c(A, b)$ is*

$$\{\, v(\pi) : \pi \textit{ is a permutation of } \mathcal{N} \,\}.$$

Proof.

First, it is easy to see that for any permutation π of $\mathcal{N}$, $v(\pi)$ is an extreme point of $\mathcal{P}_c(A, b)$. Second, we shall show that any extreme point of $\mathcal{P}_c(A, b)$ is of the form $v(\pi)$ for some permutation π. This follows from the well known result that every extreme point of a polyhedron is the unique maximizer of some linear objective, and the fact that algorithm $\mathcal{AG}$ produces an optimal primal solution of such form. □

Remark. Edmonds (1970) proved the classical result that the *greedy* algorithm solves the linear programming problem over a polyhedron for every linear objective function if and only if the polyhedron is a polymatroid. Now, in the case that $A_i^S = 1$, for $i \in S$ and $S \subseteq \mathcal{N}$, it is easy to see that adaptive greedy algorithm $\mathcal{AG}$ reduces indeed to the classical greedy algorithm that sorts the r_i's in nonincreasing order. Since we know that algorithm $\mathcal{AG}$ solves problem (LP_c) optimally it follows that, in this special case, $\mathcal{P}_c(A, b)$ is indeed a polymatroid, and therefore function b is submodular (see also Dunstan and Welsh (1973)).

Extended polymatroids are therefore natural generalizations of polymatroids, and adaptive greedy algorithm $\mathcal{AG}$ is a natural extension of the greedy algorithm.

Indexability

The optimality of adaptive greedy algorithm $\mathcal{AG}$ leads naturally to the definition of certain *allocation indices*, that characterize the optimal solutions of a linear programs over an extended polymatroid. We will show later that these allocation indices correspond to the well-known Gittins indices in stochastic scheduling problems.

Definition 2 (Allocation indices) Let $\overline{y}$ be the optimal dual solution produced by adaptive greedy algorithm $\mathcal{AG}$. Let

$$\gamma_i = \sum_{S:\ \mathcal{N} \supseteq S \ni i} \overline{y}^S, \qquad \text{for } i \in \mathcal{N}.$$

We say that $\gamma_1, \ldots, \gamma_n$ are the *allocation indices* of linear program (LP_c).

Remarks:

1. If permutation π is produced by algorithm $\mathcal{AG}$, then

$$\gamma_{\pi_i} = \overline{y}^{\{\pi_1,\ldots,\pi_n\}} + \cdots + \overline{y}^{\{\pi_i,\ldots,\pi_n\}}, \qquad \text{for } i \in \mathcal{N}. \tag{2.4}$$

2. Notice that the allocation indices of linear program (LP_c) depend only on (r, A) (not on the right-hand side b).

3. Notice that in order for the allocation indices of linear program (LP_c) to be well defined, the optimal dual solution $\overline{y}$ computed by algorithm $\mathcal{AG}$ must be uniquely determined by its input (r, A).

Consistency of the definition of allocation indices. We address next the issue of whether the allocation indices $\gamma_1, \ldots, \gamma_n$ of linear program (LP_c) are indeed uniquely determined by the input (r, A) of algorithm $\mathcal{AG}$. This question arises because ties may occur in some of the maximizations performed by the algorithm. In the presence of ties, the permutation π produced by the algorithm is not uniquely determined by its input (r, A): it clearly depends on the way ties are broken. We shall establish next that, however, the optimal dual

Input: (r, A).

Output: $(P, \hat{y})$, where $P = \{P_1, \ldots, P_m\}$ is a partition of $\mathcal{N}$ and $\hat{y} = \left(\hat{y}^S\right)_{S \subseteq \mathcal{N}}$.

Step 1. Set $k := 1$; set $U_1 = \mathcal{N}$;

set $\hat{y}^{U_1} = \max\left\{ \frac{r_i}{A_i^{\mathcal{N}}} : i \in U_1 \right\}$ and $P_1 = \operatorname{argmax}\left\{ \frac{r_i}{A_i^{\mathcal{N}}} : i \in U_1 \right\}$.

Step k. While $U_k \neq P_k$ do:

begin

Set $k := k + 1$; set $U_k = U_{k-1} \setminus P_{k-1}$;

set $\hat{y}^{U_k} = \max\left\{ \frac{r_i - \sum_{j=1}^{k-1} A_i^{U_j} \hat{y}^{U_j}}{A_i^{U_k}} : i \in U_k \right\}$ and

$P_k = \operatorname{argmax}\left\{ \frac{r_i - \sum_{j=1}^{k-1} A_i^{U_j} \hat{y}^{U_j}}{A_i^{U_k}} : i \in U_k \right\}$.

end {while}

Step m. Set $m = k$;

for $S \subseteq \mathcal{N}$: set

$$\hat{y}^S = 0, \qquad \text{if } S \notin \{U_1, \ldots, U_m\}.$$

Figure 2-2: Algorithm $\mathcal{AG}'$: Unambiguous version of adaptive greedy algorithm $\mathcal{AG}$.

solution produced by the algorithm remains invariant under different tie-breaking rules, and therefore our definition of allocation indices of linear program (LP_c) is consistent. We shall also characterize the structure of the permutations that can produced by algorithm $\mathcal{AG}$.

In order to prove that the dual solution produced by algorithm $\mathcal{AG}$ is uniquely determined by its input, we introduce next algorithm $\mathcal{AG}'$, which is simply an unambigous version of the former — its output is uniquely determined by its input—. Algorithm $\mathcal{AG}'$, shown in Figure 2-2, produces a partition $P = \{P_1, \ldots, P_m\}$ of the ground set, in addition to a dual vector $\hat{y}$. Each subset in that partition groups together elements of the ground set that would attain the same maximum in the maximizations performed by algorithm $\mathcal{AG}$, thus eliminating the ambiguity due to different tie-breaking rules.

The next result, which is easily seen to hold by induction, shows the relation between the outputs of both algorithms. It establishes that the dual solution returned by adaptive greedy $\mathcal{AG}$ is invariant under tie-breaking rules. It also characterizes the structure of the permutations that can be returned by algorithm $\mathcal{AG}$.

Proposition 2 *Let $(\pi, \bar{y}, \gamma)$ and $(P, \hat{y})$ be outputs of algorithms $\mathcal{AG}$ and $\mathcal{AG}'$, respectively, corresponding to the same input (r, A). Then*

(a) $\hat{y} = \bar{y}$.

(b) *Permutation π satisfies*

$$P_k = \{\pi_{|P_1 \cup \ldots \cup P_{k-1}|+1}, \ldots, \pi_{|P_1 \cup \ldots \cup P_k|}\}, \qquad \text{for } k = 1, \ldots, m; \tag{2.5}$$

Optimality conditions. We present next several equivalent optimality conditions for a linear program over an extended polymatroid, including one based on ranking its indices. Let $\Re_- = \{x \in \Re : x \leq 0\}$. Let $\gamma_1, \ldots, \gamma_n$ be the allocation indices of program (LP_c). Let T be the following $n \times n$ upper triangular matrix:

$$T = \begin{pmatrix} 1 & 1 & \ldots & 1 \\ 0 & 1 & \ldots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & 1 \end{pmatrix}.$$

Proposition 3 (Equivalent optimality conditions) *The following statements are equivalent:*

(a) *Permutation π satisfies* (2.5);

(b) *permutation π is produced by algorithm $\mathcal{AG}$;*

(c) $r_\pi A_\pi^{-1} \in \Re \times \Re_-^{n-1}$, *and then the allocation indices are given by* $\gamma_\pi = r_\pi A_\pi^{-1} T$;

(d) $\gamma_{\pi_n} \leq \gamma_{\pi_{n-1}} \leq \cdots \leq \gamma_{\pi_1}$.

Proof

(a) $\Rightarrow$ (b): This is the result in Proposition 2(b).

(b) $\Rightarrow$ (c): It is clear, by construction in $\mathcal{AG}$, that

$$\left(\bar{y}^{\{\pi_1,\ldots,\pi_n\}}, \ldots, \bar{y}^{\{\pi_{n-1},\pi_n\}}, \bar{y}^{\{\pi_n\}}\right) = r_\pi A_\pi^{-1}. \tag{2.6}$$

Now, in the proof of Proposition 1 we showed that

$$\bar{y}^S \leq 0, \qquad \text{for } S \subset \mathcal{N},$$

which together with (2.6) implies $r_\pi A_\pi^{-1} \in \Re \times \Re_-^{n-1}$. Furthermore, by (2.4) we have

$$\gamma_\pi = \left(\bar{y}^{\{\pi_1,\ldots,\pi_n\}}, \ldots, \bar{y}^{\{\pi_{n-1},\pi_n\}}, \bar{y}^{\{\pi_n\}}\right) T,$$

and by (2.6) it follows that

$$\gamma_\pi = r_\pi A_\pi^{-1} T.$$

(c) $\Rightarrow$ (d): By (c) we have

$$(\gamma_{\pi_1}, \gamma_{\pi_2} - \gamma_{\pi_1}, \ldots, \gamma_{\pi_n} - \gamma_{\pi_{n-1}}) T^{-1} = r_\pi A_\pi^{-1} \in \Re \times \Re_-^{n-1},$$

whence the result follows.

(d) $\Rightarrow$ (a): By construction of $\hat{y}$ in algorithm $\mathcal{AG}'$, the fact that $\bar{y} = \hat{y}$ and the definition of allocation indices, it follows that

$$\gamma_i = \bar{y}^{U_1} + \cdots + \bar{y}^{U_k}, \qquad \text{for } i \in U_k \quad \text{and } k = 1, \ldots, m,$$

where the U_k's are as constructed in algorithm $\mathcal{AG}'$. Also, it is easy to see that $\bar{y}^{U_j} < 0$, for $j \geq 2$. These two facts imply that π must satisfy (2.5), which completes the proof. □

Combining the result that algorithm $\mathcal{AG}$ solves linear program (LP_c) optimally with the equivalent conditions in Proposition 3, we obtain the sufficient optimality conditions presented next.

Theorem 2 (Sufficient optimality conditions and indexability) *Assume that any of the conditions* (a)-(d) *of Proposition 3 holds. Then $v(\pi)$ solves linear program (LP_c) optimally.*

The following results follow from our previous analysis:

1. **Indexability:** Optimality condition (d) of Proposition 3 shows that any permutation that sorts the allocation indices of linear program (LP_c) provides a corresponding optimal solution. Condition (d) therefore shows that this class of linear programs has an *indexability* property.

2. **Sensitivity analysis:** Optimality condition (c) of Proposition 3 is specially well suited for performing sensitivity analysis. Consider the following question: given a permutation π of $\mathcal{N}$, for what vectors r and matrices A does $v(\pi)$ solve problem (LP_c)

optimally? We know that $v(\pi)$ is optimal for r and A that satisfy the condition

$$r_\pi A_\pi^{-1} \in \Re \times \Re_-^{n-1}.$$

We may also ask: For which permutations π is it guaranteed that $v(\pi)$ is optimal? By Proposition 3(d), we know that $v(\pi)$ is optimal for permutations π that satisfy

$$\gamma_{\pi_n} \leq \gamma_{\pi_{n-1}} \leq \cdots \leq \gamma_{\pi_1},$$

thus providing an $O(n \log n)$ optimality test for π. Glazebrook (1994) has recently applied these polyhedral results to provide a range of index-based suboptimality bounds for general policies in a variety of stochastic scheduling problems (see also Glazebrook (1987)).

3. **Closed formulae for allocation indices:** Proposition 3(c) provides a closed formula for the vector of allocation indices. It shows that the indices are piecewise linear functions of the reward vector.

4. **Optimal objective value.** The optimal objective value, Z, is given by:

$$\begin{aligned} Z &= r_\pi x_\pi \\ &= r_\pi A_\pi^{-1} b_\pi \\ &= \gamma_\pi T^{-1} b_\pi \\ &= (\gamma_{\pi_1}, \ \gamma_{\pi_2}, \ \ldots, \ \gamma_{\pi_n}) \begin{pmatrix} b(\{\pi_1, \ldots, \pi_n\}) - b(\{\pi_2, \ldots, \pi_n\}) \\ \vdots \\ b(\{\pi_{n-1}, \pi_n\}) - b(\{\pi_n\}) \\ b(\{\pi_n\}) \end{pmatrix} \end{aligned} \tag{2.7}$$

Index decomposition

We show in this section that the allocation indices of linear program (LP_c) possess, under a certain assumption, a strong *decomposition* property. In that case, the indices can be computed by solving a number of smaller subproblems, thus reducing the amount of computations required. We will show later that this property explains an analogous decomposition property of the optimal priority-indices in some stochastic scheduling problems,

including multi-armed bandits.

In this setting the ground set $\mathcal{N}$ is partitioned into subsets $\mathcal{N}_1, \ldots, \mathcal{N}_K$. Let A^k denote the submatrix of A corresponding to subsets of $\mathcal{N}_k$, i.e., $A^k = (A_i^S)_{i \in \mathcal{N}_k, S \in \mathcal{N}_k}$. Let $r = (r_i)_{i \in \mathcal{N}}$, and $r^k = (r_i)_{i \in \mathcal{N}_k}$, for $k = 1, \ldots, K$. Let γ be the vector of allocation indices of linear program (LP_c), and let γ^k be the index vector produced by adaptive greedy algorithm $\mathcal{AG}$ when fed with input (r^k, A^k).

The required decomposition assumption on the parameters of matrix A is:

Assumption 2

$$A_i^S = A_i^{S \cap \mathcal{N}_k}, \qquad \text{for } i \in S \cap \mathcal{N}_k \quad \text{and } S \subseteq \mathcal{N}. \tag{2.8}$$

We show next that, under Assumption 2, vector of allocation indices γ extends vectors $\gamma^1, \ldots, \gamma^K$ over their respective ranges.

Theorem 3 (Index decomposition) *If Assumption 2 holds, then*

$$\gamma_i = \gamma_i^k, \qquad \text{for } i \in \mathcal{N}_k \quad \text{and } k = 1, \ldots, K.$$

Proof

Let us define $b^k : 2^{\mathcal{N}_k} \to \Re_+$ by $b^k(S) = b(S)$, for $S \subseteq \mathcal{N}_k$. Since $\mathcal{P}_c(A, b)$ is an extended contra-polymatroid, it is easily seen that $\mathcal{P}_c(A^k, b^k)$ is also an extended contra-polymatroid — with ground set $\mathcal{N}_k$. For $k = 1, \ldots, K$, let us write $x^k = (x_i^k)_{i \in \mathcal{N}_k}$, and let (LP_k) be the linear program

$$(LP_k) \qquad \max \left\{ \sum_{i \in \mathcal{N}_k} r_i x_i^k : x^k \in \mathcal{P}_c(A^k, b^k) \right\}.$$

By definition, the allocation indices of linear program (LP_k) are obtained by running algorithm $\mathcal{AG}$ with input (r^k, A^k), and are therefore given by vector γ^k.

Let us define

$$g_i = \gamma_i^k, \qquad \text{for } i \in \mathcal{N}_k \quad \text{and } k = 1, \ldots, K. \tag{2.9}$$

Let us renumber, for simplicity, the elements of $\mathcal{N}$, so that

$$g_n \leq g_{n-1} \leq \cdots \leq g_1. \tag{2.10}$$

Let $\pi = (1, \ldots, n)$. Permutation π of $\mathcal{N}$ induces permutations π^k of $\mathcal{N}_k$, for $k = 1, \ldots, K$,

that satisfy

$$\gamma^k_{\pi^k_{|\mathcal{N}_k|}} \leq \cdots \leq \gamma^k_{\pi^k_1}.$$

Hence, by Proposition 3 it follows that

$$\gamma^k_{\pi^k} = r^k_{\pi^k}(A^k_{\pi^k})^{-1}T_k, \qquad \text{for } k = 1, \ldots, K$$

or, equivalently,

$$(\gamma^1_{\pi_1}, \gamma^2_{\pi_2}, \ldots, \gamma^k_{\pi^k}) \begin{pmatrix} T_1^{-1}A^1_{\pi^1} & 0 & \ldots & 0 \\ 0 & T_2^{-1}A^2_{\pi^2} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & T_k^{-1}A^k_{\pi^k} \end{pmatrix} = (r^1_{\pi^1}, r^2_{\pi^2}, \ldots, r^k_{\pi^k}), \qquad (2.11)$$

where T_k is an $|\mathcal{N}_k| \times |\mathcal{N}_k|$ matrix with the same structure as matrix T, for $k = 1, \ldots, K$. On the other hand, we have

$$\begin{aligned} T^{-1}A_\pi &= \begin{pmatrix} 1 & -1 & 0 & \ldots & 0 & 0 & 0 \\ 0 & 1 & -1 & \ldots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ddots & 0 & 1 & -1 \\ 0 & 0 & 0 & \ldots & 0 & 0 & 1 \end{pmatrix} A_\pi \\ &= \begin{pmatrix} A_1^{\{1,\ldots,n\}} & \ldots & A_{n-1}^{\{1,\ldots,n\}} - A_{n-1}^{\{1,\ldots,n-1\}} & A_n^{\{1,\ldots,n\}} - A_n^{\{1,\ldots,n-1\}} \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \ldots & A_{n-1}^{\{n-1,n\}} & A_n^{\{n-1,n\}} - A_n^{\{n\}} \\ 0 & \ldots & 0 & A_n^{\{n\}} \end{pmatrix}. \end{aligned}$$

Now, notice that if $i \in \mathcal{N}_k$, $j \in \mathcal{N} \setminus \mathcal{N}_k$ and $i > j$ then, by Assumption 2 it follows that

$$A_i^{\{j,\ldots,n\}} = A_i^{\{j,\ldots,n\}\cap\mathcal{N}_k} = A_i^{\{j+1,\ldots,n\}\cap\mathcal{N}_k} = A_i^{\{j+1,\ldots,n\}}. \qquad (2.12)$$

Hence, by (2.9) and (2.12) it follows that system (2.11) can be written equivalently as

$$g_\pi T^{-1}A_\pi = r_\pi. \qquad (2.13)$$

Problem	Allocation indices	Optimal solution
$(LP)\ \min_{x \in \mathcal{P}(A,b)} hx$	$(h, A) \xrightarrow{\mathcal{AG}} \gamma$	$\gamma_{\pi_n} \leq \cdots \leq \gamma_{\pi_1}$ $\overline{x}_\pi = A_\pi^{-1} b_\pi$
$(LP_c)\ \max_{x \in \mathcal{P}_c(A,b)} rx$	$(r, A) \xrightarrow{\mathcal{AG}} \gamma$	$\gamma_{\pi_n} \leq \cdots \leq \gamma_{\pi_1}$ $\overline{x}_\pi = A_\pi^{-1} b_\pi$

Table 2.1: Linear programming over extended polymatroids.

Now, (2.10) and (2.13) imply that

$$r_\pi A_\pi^{-1} = g_\pi T^{-1} = (\, g_1, \quad g_2 - g_1, \quad \ldots, \quad g_n - g_{n-1}\,) \in \Re \times \Re_-^{n-1},$$

and by Proposition 3 it follows that the allocation indices of linear program (LP_c) satisfy

$$\gamma_\pi = r_\pi A_\pi^{-1} T.$$

Therefore, by (2.13),

$$g_i = \gamma_i, \qquad \text{for } i \in \mathcal{N},$$

which completes the proof. □

A useful consequence of Theorems 2 and 3 is the following:

Corollary 1 *Under the assumptions of Theorem 3, an optimal solution of linear program (LP_c) can be computed by running algorithm $\mathcal{AG}$ with inputs (r^k, A^k), for $k = 1, \ldots, K$.*

Remark: It is important to emphasize that the index decomposition property is much stronger that the indexability property. We will see later in the context of stochastic scheduling that the classical multi-armed bandit problem exhibits the index decomposition property. This condition is not satisfied in general, however, by the optimal priority-indices in Klimov's problem (see Klimov (1974)).

We have focused our discussion in properties of linear programs over extended contra-polymatroids. The properties of linear programs over extended polymatroids are analogous. In Table 2.1 it is shown how to solve linear program (LP) — that minimizes a cost function hx over extended polymatroid $\mathcal{P}(A, b)$— by running algorithm $\mathcal{AG}$ with input (h, A), thus obtaining corresponding allocation indices that characterize the optimal solution

2.3 A polynomial projection representation for a class of extended polymatroids

As will be shown in Chapters 3 and 4, extended polymatroids arise in applications as the performance region corresponding to a variety of stochastic scheduling problems. For example, the region spanned by the vector of expected flow times in a multiclass queue with feedback (see Klimov (1974)) is an extended polymatroid, as established by Tsoucas (1991). It may be of interest in applications to impose side performance constraints to such systems, that may represent, e.g., bounds on expected flow times (see e.g. Nain and Ross (1986), and Makowski and Shwartz (1993)). The region of achievable performance corresponding to the modified system is still a polyhedron, which may be represented by the constraints that define the extended polymatroid, augmented with a polynomial number of side constraints.

As a consequence of the polynomial-time solvability of a linear program over an extended polymatroid (see Section 2.2), and of the polynomial-time equivalence between linear programming separation and optimization (see Grötschel et al. (1988)), it follows that a linear program over such an augmented polyhedron can be solved in polynomial time. This theoretical result relies, however, on using the ellipsoid method for solving the separation problem over an extended polymatroid, which hinders its application, as that method has not proven to be efficient in practice (see e.g. Nemhauser and Wolsey (1988)). There is thus a need for practically efficient methods for solving those linear programs.

In this section we present a compact (polynomial-size) projection representation for a class of extended polymatroids, that includes those arising in stochastic scheduling applications such as Klimov's (1974) problem. This polynomial-size reformulation is obtained by representing the extended polymatroid as the projection of a higher dimensional polyhedron (introducing additional variables). A linear program over these extended polymatroids, augmented with a polynomial number of side constraints, can thus be reformulated as a polynomial-size linear program, which can then be solved in polynomial time by practically efficient interior-point methods for linear programming (see e.g. Nesterov and Nevirovskii (1994)).

Our work generalizes results of Bertsimas, Paschalidis and Tsitsiklis (1994a), and Bertsimas, Paschalidis and Tsitsiklis (1994b), in which they present polynomial-size reformulations of the performance region corresponding to Klimov's problem, and branching bandits,

respectively.

We present next the class of extended polymatroids mentioned above. Let $\mathcal{N} = \{1, \ldots, n\}$. Let $p = (p_i)_{i \in \mathcal{N}}$ be a vector with positive components, and let $Q = (q_{ij})_{i,j \in \mathcal{N}}$ and $B = (b_{ij})_{i,j \in \mathcal{N}}$, be matrices with nonnegative components, with Q symmetric.

We shall define a matrix $A = (A_i^S)_{i \in \mathcal{N}, S \subseteq \mathcal{N}}$ and a set function $b : 2^{\mathcal{N}} \to \Re_+$ as functions of p, Q and B. The following notation will be used throughout the section: given subsets S, T of $\mathcal{N}$, we write $p_S = (p_i)_{i \in \mathcal{N}}$, $A_T^S = (A_i^S)_{i \in T}$, $Q_{ST} = (q_{ij})_{i \in S, j \in T}$ (B_{ST} is defined similarly). For every $S \subseteq \mathcal{N}$, let us define vector $A_{\mathcal{N}}^S$ as the solution of the linear system

$$A_S^S = p_S + Q_{SS} A_S^S,$$

$$A_{S^c}^S = p_{S^c} + Q_{S^c S} A_S^S,$$

and let

$$b(S) = \frac{1}{2} A_S^{S\prime} B_{SS} A_S^S.$$

We can now consider the polyhedron $\mathcal{P}_c(A, b)$, defined earlier as

$$\mathcal{P}_c(A,b) = \left\{ x \in \Re_+^n : \sum_{i \in S} A_i^S x_i \geq b(S), \quad \text{for } S \subset \mathcal{N} \quad \text{and} \quad \sum_{i \in \mathcal{N}} A_i^{\mathcal{N}} x_i = b(\mathcal{N}) \right\}.$$

Let us introduce a related polyhedron $\mathcal{P}(p, Q, B)$, in higher dimension, as

$$\mathcal{P}(p,Q,B) = \left\{ (x, X) \in \Re_+^n \times \Re_+^{n^2} : (I - Q)'X + X'(I - Q) = B, \quad \text{and} \quad x' = p'X \right\}.$$

We shall prove next that, under a certain consistency assumption on parameters p, Q and B, which is equivalent to Assumption 1, polyhedron $\mathcal{P}_c(A, b)$ is an extended contra-polymatroid. We shall further show that, in that case, the projection of polyhedron $\mathcal{P}(p, Q, B)$ over the space of the x-variables, denoted $\text{proj}_x(\mathcal{P}(p, Q, B))$, is precisely extended contra-polymatroid $\mathcal{P}_c(A, b)$, thus providing a polynomial-size projection representation.

Let us first present the consistency assumption. Given a permutation π of $\mathcal{N}$ and matrix

X, let us denote

$$X_\pi = \begin{pmatrix} x_{\pi_1\pi_1} & 0 & \cdots & 0 \\ x_{\pi_2\pi_1} & x_{\pi_2\pi_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ x_{\pi_n\pi_1} & x_{\pi_n\pi_{n-1}} & \cdots & x_{\pi_n\pi_n} \end{pmatrix}.$$

Assumption 3 *For every permutation π of $\mathcal{N}$, linear system*

$$(I - Q)'X_\pi + X_\pi'(I - Q) = B \tag{2.14}$$

has a nonnegative solution.

We can now state the main result of this section.

Theorem 4 (Polynomial projection representation for extended polymatroids) *Under Assumption 3, the following results hold:*

(a) *Polyhedron $\mathcal{P}_c(A, b)$ is an extended contra-polymatroid;*

(b) $\mathcal{P}_c(A, b) = \text{proj}_x(\mathcal{P}(p, Q, B))$.

Proof

(a) We shall first show that $\text{proj}_x(\mathcal{P}(p, Q, B)) \subseteq \mathcal{P}_c(A, b)$. Let (x, X) be such that $x \in \text{proj}_x(\mathcal{P}(p, Q, B))$, and let $S \subseteq \mathcal{N}$. We have

$$\begin{aligned} b(S) &= \frac{1}{2} A_S^{S\prime} B_{SS} A_S^S \\ &= \frac{1}{2} \begin{pmatrix} A_S^{S\prime}, & 0 \end{pmatrix} ((I - Q)'X + X'(I - Q)) \begin{pmatrix} A_S^S \\ 0 \end{pmatrix} \\ &= \left\{ (I - Q) \begin{pmatrix} A_S^S \\ 0 \end{pmatrix} \right\}' X \begin{pmatrix} A_S^S \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} (I_S - Q_{SS}) A_S^S \\ -Q_{S^cS} A_S^S \end{pmatrix}' X \begin{pmatrix} A_S^S \\ 0 \end{pmatrix} \\ &= \left\{ p - \begin{pmatrix} 0 \\ A_{S^c}^S \end{pmatrix} \right\}' X \begin{pmatrix} A_S^S \\ 0 \end{pmatrix} \\ &= \sum_{i \in S} A_i^S x_i - A_{S^c}^{S\prime} X_{S^cS} A_S^S \qquad (2.15) \\ &\leq \sum_{i \in S} A_i^S x_i, \qquad (2.16) \end{aligned}$$

which proves that $\text{proj}_x(\mathcal{P}(p, Q, B)) \subseteq \mathcal{P}_c(A, b)$.

We shall now show that $\mathcal{P}_c(A, b)$ is an extended contra-polymatroid. Let $\pi = (\pi_1, \ldots, \pi_n)$ be a permutation of $\mathcal{N}$. Since Assumption 3 holds, we can pick a nonnegative solution $\bar{X}_\pi$ to system (2.14). Letting $\bar{x}'_\pi = p'_\pi \bar{X}_\pi$, and substituting into equation (2.15) we obtain that $\bar{x}$ is the solution of system (2.2). This solution, by (2.16), lies in $\mathcal{P}_c(A, b)$, which proves that $\mathcal{P}_c(A, b)$ is an extended contra-polymatroid.

(b) In the proof of part (a) we established that $\mathcal{P}_c(A, b)$ is an extended contra-polymatroid whose vertices lie in $\text{proj}_x(\mathcal{P}(p, Q, B))$. Therefore, $\mathcal{P}_c(A, b) \subseteq \text{proj}_x(\mathcal{P}(p, Q, B))$, thus completing the proof. □

Chapter 3

A unified approach to indexable scheduling problems

Researchers have established over the last four decades that a wide variety of stochastic scheduling problems are solved optimally by simple priority-index rules. In these problems a numerical index may be associated with each kind of job, in such a way that the policy that serves at each time a job with largest current index is optimal. Well known examples include Smith's rule (see Smith (1956)) in single-machine deterministic scheduling, the $c\mu$ rule (see Cox and Smith (1961)) and Klimov's rule (see Klimov (1974)) in dynamic scheduling of multiclass queues, and the Gittins index policy (see Gittins and Jones (1974)) in multi-armed bandit problems.

These results have been obtained through a variety of techniques, such as dynamic programming formulations and interchange arguments, tailored to each particular problem. This approach is not completely satisfactory, both from a theoretical and from a practical viewpoint. On the theoretical side, it would be desirable to understand in a unifying way the reason that some scheduling problems are *indexable* (i.e., solved optimally by priority-index rules). Such understanding should provide a method for testing whether a given scheduling problem is indexable, and a unified index-computing algorithm. On the practical side, one often needs to consider variations on the original problem, such as incorporating a nonlinear performance objective, or imposing side constraints on performance measures. Ad hoc techniques for analyzing the original problem do not usually yield insight on how to solve such variations.

This chapter develops a mathematical programming approach, as outlined in Chapter 1, to the solution of indexable scheduling problems. A unified framework is thus obtained for formulating and solving such problems as linear programs with special structure. In an abstract setting of general multiclass queueing systems, we introduce the concept of generalized conservation laws, and prove that the performance region corresponding to a performance measure that satisfies these laws is necessarily a specific kind of polyhedron: an extended polymatroid. The problem of optimizing a linear performance objective by controlling the dynamic scheduling policy used is thus translated into a linear program over an extended polymatroid. The structural and algorithmic properties of these underlying linear programs, developed in Chapter 2, are shown to correspond to related properties of the scheduling problems, such as the optimality of priority-index rules.

The chapter is structured as follows: Section 3.1 introduces the application of the approach through an example: a simple two-armed bandit problem. Section 3.2 presents a unified framework for formulating and solving indexable scheduling problems, which is based on exploiting the relation between physical conservation laws and the geometrical structure of the performance region.

3.1 Optimal dynamic scheduling of a two-armed bandit

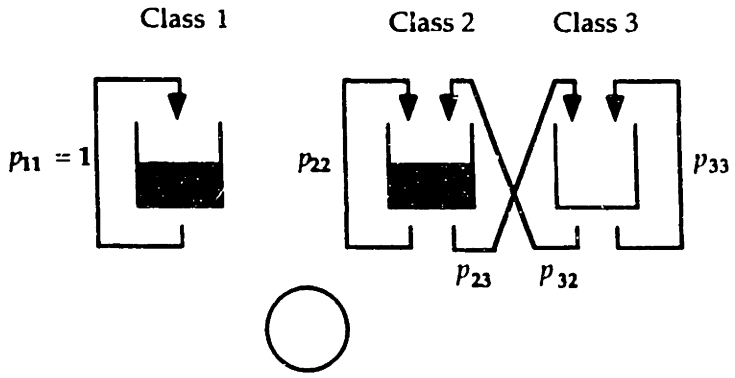

Figure 3-1: A two-armed bandit.

This section introduces the main ideas of the mathematical programming approach to stochastic scheduling, as described in Chapter 1, through an example: a simple two-armed bandit problem.

Let us first describe the example problem. Although it may be regarded as a two-armed bandit problem (see e.g. Gittins (1989)), we shall introduce and analyze it as a scheduling

problem in a multiclass queue, for consistency with the general framework to be presented later. In the terminology of queueing theory, the problem may be stated as the optimal dynamic scheduling of the closed queue in discrete time with three job classes and two jobs shown in Figure 3-1. Initially one of the jobs is in class 1 and the other in class 2. Jobs may change class after completing service, according to Markovian transition probabilities. A class 2 job may thus either remain in the same class, with probability p_{22}, or transfer to class 3 with probability $p_{23} = 1 - p_{22}$. A class 3 job behaves analogously, as shown in Figure 3-1. The class 1 job, however, always remains in the same class. There is a discounted reward structure associated with service completions: Each time a class-j job completes its service, a reward r_j is earned, discounted in time by a discount factor $0 < \beta < 1$. The problem is to find a scheduling policy that decides which job to serve at each time, in order to maximize the expected present value of the rewards earned over an infinite horizon.

Two natural requirements are imposed on the class of scheduling policies that may be used. First, a policy must be nonanticipative (i.e., decisions may not be based on future information on the evolution of the system), thus obeying the principle of causality. Second, a policy has to be nonidling (i.e., the server never stops working while there are jobs in the system). We shall refer to the class $\mathcal{U}$ of policies that satisfy these two conditions as the class of *admissible* policies. By defining the indicator

$$I_j(t) = \begin{cases} 1, & \text{if a class-}j\text{ job is serviced at time } t; \\ 0, & \text{otherwise.} \end{cases}$$

we can write this optimal dynamic scheduling problem as

$$(SP) \qquad Z = \max \left\{ E_u \left[\sum_{t=0}^{\infty} \sum_{j=1}^{3} r_j I_j(t) \beta^t \right] : u \in \mathcal{U} \right\}. \tag{3.1}$$

3.1.1 Performance measures

The first step of the approach involves expressing the objective to be optimized as a function of suitably defined performance measures. For the above problem, a natural performance measure is the total expected discounted number of service completions for each job class,

$$\lambda_j^u = E_u \left[\sum_{t=0}^{\infty} I_j(t) \beta^t \right], \qquad \text{for } j = 1, 2, 3. \tag{3.2}$$

As scheduling policies range over the class of admissible policies, the corresponding performance vectors span the region of achievable performance

$$\Lambda = \{(\lambda_1^u, \lambda_2^u, \lambda_3^u) : u \in \mathcal{U}\}.$$

The problem of finding an optimal performance vector can now be written as the mathematical program

$$\begin{aligned} (MP) \quad Z = \quad & \max \quad r_1\lambda_1 + r_2\lambda_2 + r_3\lambda_3 \\ & \text{subject to} \\ & (\lambda_1, \lambda_2, \lambda_3) \in \Lambda. \end{aligned} \tag{3.3}$$

3.1.2 Conservation laws

We shall construct a complete polyhedral description of performance region Λ, by identifying conservation laws satisfied by the system and expressing them as linear constraints on performance vectors. First, since at each time exactly one job completes its service, it follows that the total expected discounted number of completed jobs is the same under any admissible scheduling policy. This conservation law may be written as

$$\lambda_1^u + \lambda_2^u + \lambda_3^u = \frac{1}{1-\beta}, \qquad \text{for } u \in \mathcal{U}. \tag{3.4}$$

We shall outline next how to construct a family of conservation laws associated with subsets of job classes. Consider, for example, the subset $\{2\}$ corresponding to class 2 jobs. Let u_2 be a scheduling policy that gives priority to class-2 jobs over jobs in other classes (i.e., the job in service must be a class-2 job whenever there is one present). Under such a policy, the conservation law

$$\lambda_2^{u_2} = \frac{1}{1-\beta p_{22}} + \beta p_{32} \frac{1}{1-\beta p_{22}} \lambda_3^{u_2} \tag{3.5}$$

holds. Equation (3.5) expresses the intuitive fact that the total expected discounted number of class-2 jobs completed, $\lambda_2^{u_2}$, can be decomposed into two terms: A first constant term, $1/(1-\beta p_{22})$, that accounts for the class 2 jobs completed until the job that was initially in class 2 transfers for the first time to class 3; and a second term, that accounts for the

class 2 jobs completed afterwards, using the following accounting argument: After a class 3 job completes its service at time t, the expected discounted number of successive class 2 job completions is

$$\beta^{t+1} p_{32}/(1-\beta p_{22}).$$

It should be intuitively clear that, under other admissible policies, the right-hand side of equation (3.5) overestimates the expected discounted number of successive class 2 job completions, i.e.,

$$\lambda_2^u \leq \frac{1}{1-\beta p_{22}} + \beta p_{32} \frac{1}{1-\beta p_{22}} \lambda_3^u, \qquad \text{for } u \in \mathcal{U}. \tag{3.6}$$

Using conservation law (3.4), we may rewrite inequality (3.6) as

$$\lambda_1^u + (1 + \beta \frac{p_{32}}{1-\beta p_{22}}) \lambda_3^u \geq \frac{1}{1-\beta} - \frac{1}{1-\beta p_{22}}, \qquad \text{for } u \in \mathcal{U}. \tag{3.7}$$

with equality holding under any policy u that gives priority to class 2 jobs. Inequality (3.7) yields a corresponding linear inequality constraint on performance region Λ.

By applying a similar argument to policies that give priority to other subsets of job classes, we may derive corresponding conservation laws and linear constraints on performance vectors.

3.1.3 Linear programming formulation

As will be shown later, the linear constraints obtained through the procedure outlined above represent a complete polyhedral description of performance region Λ. Proceeding in this way, we obtain the following explicit linear programming formulation of problem (MP):

$$
\begin{array}{lllllllll}
Z = \max & r_1\lambda_1 & + & r_2\lambda_2 & + & r_3\lambda_3 & & & \\
\text{subject to} & & & & & & & & \\
 & & & \lambda_2 & + & \lambda_3 & \geq & 0 & \\
 & \lambda_1 & + & & & (1+\beta\frac{p_{32}}{1-\beta p_{22}})\lambda_3 & \geq & \frac{1}{1-\beta}-\frac{1}{1-\beta p_{22}} & \\
 & \lambda_1 & + & (1+\beta\frac{p_{23}}{1-\beta p_{33}})\lambda_2 & & & \geq & \frac{1}{1-\beta} & \\
 & \lambda_1 & & & & & \geq & 0 & \\
 & & & \lambda_2 & & & \geq & 0 & \\
 & & & & & \lambda_3 & \geq & 0 & \\
 & \lambda_1 & + & \lambda_2 & + & \lambda_3 & = & \frac{1}{1-\beta}, &
\end{array}
$$

where each inequality constraint holds with equality under a scheduling policy that gives priority to the corresponding subset of job classes.

It will be shown in Chapter 4 that the feasible region of linear program (MP) is an extended polymatroid. as defined in Chapter 2.

3.1.4 Optimal solution

We will show in the next section that the extreme points of that extended polymatroid are the performance vectors achieved under static-priority scheduling policies (i.e., the service priority of a class remains constant through time). Since the optimal value of a linear program is attained by some extreme point in its feasible region. it follows that static-priority policies are optimal for this scheduling problem. In the next section we will also see how to identify the optimal extreme point, and its associated optimal static-priority policy, by running a one-pass algorithm.

Figure 3-2 depicts an extended polymatroid in dimension three, representing the performance region of a 3-class bandit problem. Notice that it has $3! = 6$ extreme points, corresponding to all static-priority policies.

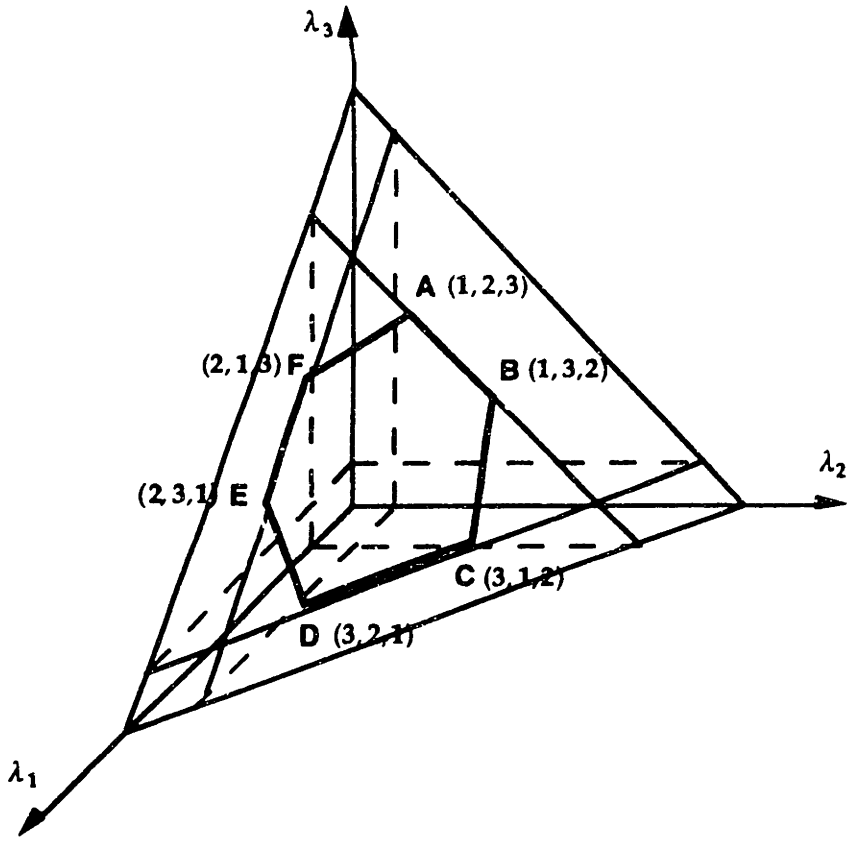

Figure 3-2: An extended polymatroidal performance region (priority policies are associated with its extreme points, with job classes ranked by increasing priority).

3.2 A mathematical programming approach to indexable scheduling problems

In this section we develop a unified framework for formulating and solving stochastic scheduling problems solved by priority-index rules. This framework is obtained by applying the mathematical programming approach to stochastic resource allocation problems described in Chapter 1.

Section 3.2.1 introduces a concept of conservation laws that generalizes other definitions used in the literature. We show that the performance region corresponding to a scheduling problem that satisfies these laws is necessarily a polyhedron with special structure: an extended polymatroid. The vertices of the performance region are shown to be achieved by static-priority scheduling policies.

Section 3.2.3 shows how to formulate the problem of optimizing a linear performance objective, in a system that satisfies generalized conservation laws, as a linear program over an extended polymatroid, and how to obtain an optimal scheduling policy from the

solution the formulation. We establish the optimality of priority-index rules and we present an assumption under which the optimal indices satisfy a decomposition property, which simplifies their computation.

3.2.1 Generalized conservation laws

The wide variety of stochastic scheduling problems solved optimally by priority-index rules leads us naturally to consider the question: What physical properties of the system account for that indexability property? We provide in this section an answer to that question, by introducing and applying a general concept of conservation laws. We thus show that the performance region corresponding to a performance measure that satisfies such laws is necessarily an extended polymatroid, whose vertices are achievable under static-priority scheduling policies. As will be shown in the next section, the indexability property of linear programs over extended polymatroids (see Appendix A) translates into the optimality of priority-index policies in the scheduling problems that satisfy those laws.

Consider a general dynamic and stochastic multiclass queueing system. There are n job classes, which we label $i \in \mathcal{N} = \{1, \ldots, n\}$. Jobs have to be scheduled for service in the system under an admissible scheduling policy. Let $\mathcal{U}$ denote the class of admissible scheduling policies. Let x_i^u be a performance measure of class-i jobs under scheduling policy u. We assume that x_i^u is an expectation. Let x^u denote the corresponding performance vector. Let x^π be the performance vector corresponding to a *static-priority* policy (i.e., the service priority of a job depends only on its class and does not change over time) that assigns priorities to the job classes according to permutation $\pi = (\pi_1, \ldots, \pi_n)$ of $\mathcal{N}$, where class-π_1 has the highest priority.

Definition 3 (Generalized conservation laws) Performance vector x^u is said to satisfy *generalized conservation laws* if there exist a set function $b: 2^{\mathcal{N}} \rightarrow \Re_+$ and a matrix $A = (A_i^S)_{i \in \mathcal{N}, S \subseteq \mathcal{N}}$ that satisfies $A_i^S > 0$, for $S \subseteq \mathcal{N}$, such that:

(a)

$$b(S) = \sum_{i \in S} A_i^S x_i^\pi, \qquad \text{for all } \pi : \{\pi_1, \ldots, \pi_{|S^c|}\} = S^c \quad \text{and} \quad S \subseteq \mathcal{N}. \tag{3.8}$$

(b) For every admissible policy $u \in \mathcal{U}$,

$$\sum_{i \in S} A_i^S x_i^u \geq b(S), \quad \text{for all } S \subset \mathcal{N} \qquad \text{and} \qquad \sum_{i \in \mathcal{N}} A_i^{\mathcal{N}} x_i^u = b(\mathcal{N}), \tag{3.9}$$

or respectively,

$$\sum_{i \in S} A_i^S x_i^u \leq b(S), \quad \text{for all } S \subset \mathcal{N} \qquad \text{and} \qquad \sum_{i \in \mathcal{N}} A_i^{\mathcal{N}} x_i^u = b(\mathcal{N}). \tag{3.10}$$

Remarks:

1. In words, a performance vector is said to satisfy generalized conservation laws if there exist weights A_i^S such that the total weighted performance over all job classes is invariant under any admissible policy, and the minimum (or maximal) weighted performance over job classes in any subset $S \subset \mathcal{N}$ is achieved by any static-priority policy that gives priority to all other classes (in S^c) over classes in S.

2. Shanthikumar and Yao (1992) formalized a definition of *strong conservation laws* for performance measures in general multiclass queues, that implies a polymatroidal structure in the performance space. These laws correspond to the special case that all weights are $A_i^S = 1$ in Definition 3.

The connection between generalized conservation laws and extended polymatroids is given by the following theorem:

Theorem 5 (Performance region characterization) *Assume that performance vector x^u satisfies generalized conservation laws (3.8) and (3.9) (resp. (3.8) and (3.10)). Then*
(a) The performance vectors corresponding to static-priority policies are the vertices of $\mathcal{P}_c(A,b)$ (resp. $\mathcal{P}(A,b)$), and $x^\pi = v(\pi)$, for every permutation π of $\mathcal{N}$.
(b) The performance region is the extended contra-polymatroid $\mathcal{P}_c(A,b)$ (resp. the extended polymatroid $\mathcal{P}(A,b)$).

Proof

We shall prove the theorem in the case that x^u satisfies generalized conservation laws (3.8) and (3.9). The other case ((3.8) and (3.10)) is analogous.
(a) By (3.8) it follows that $x^\pi = v(\pi)$. By Theorem 1 the result follows.
(b) Let $\mathcal{P} = \{ x^u : u \in \mathcal{U} \}$ be the performance space. Let $\mathcal{P}_c^v(A,b)$ be the set of extreme points of $\mathcal{P}_c(A,b)$. By (3.9) it follows that $\mathcal{P} \subseteq \mathcal{P}_c(A,b)$. By (a), $\mathcal{P}_c^v(A,b) \subseteq X$. Hence, since $\mathcal{P}$ is a convex set ($\mathcal{U}$ contains randomized policies) we have

$$\mathcal{P}_c(A,b) = \text{conv}(\mathcal{P}_c(A,b)) \subseteq X,$$

where conv$(\cdot)$ denotes the convex hull operator. Hence $\mathcal{P} = \mathcal{P}_c(A, b)$, which completes the proof. □

Consider the problem of designing an admissible policy that achieves a given performance vector x. It easily follows from Theorem 5 and Carathéodory theorem (see e.g. Bazaraa and Shetty (1979)) that any given performance vector x can be achieved by a randomization of at most n static-priority rules.

3.2.2 Strong flow conservation laws

The generalized conservation laws presented in the previous section may often be interpreted in applications in terms of physical work conservation relations satisfied by the underlying system. We will see examples of this fact in the applications developed in Chapter 4. In some stochastic scheduling models, though, such as the ones considered in Chapter 6, it may be easier to establish that a system satisfies a different set of conservation laws, that represent physical *flow conservation* relations. These laws, when translated into linear constraints on performance measures, give rise to a set of equality constraints satisfied by the performance vector to be optimized and a certain auxiliary performance matrix.

In this section we present a general definition of *strong flow conservation laws* and prove that they imply the generalized conservation laws introduced in Section 3.2.1. Let us consider a general multiclass queueing system, with job classes $i \in \mathcal{N} = \{1, \ldots, n\}$, as described in Section 3.2.1. Let $x^u = (x_i^u)_{i \in \mathcal{N}}$ and $X^u = (x_{ij}^u)_{i,j \in \mathcal{N}}$ be a vector and a matrix, respectively, of associated performance measures. Let $p = (p_i)_{i \in \mathcal{N}}$ be an n-vector with positive components, and let $Q = (q_{ij})_{i,j \in \mathcal{N}}$ and $B = (b_{ij})_{i,j \in \mathcal{N}}$ be $n \times n$ matrices with nonnegative matrices, with B symmetric.

Definition 4 (Strong flow conservation laws) *We say that performance measures x^u, X^u satisfy strong flow conservation laws if the following conditions hold:*

(i) *For any admissible scheduling policy $u \in \mathcal{U}$,*

$$(I - Q)' X^u + X^{u\prime}(I - Q) = B.$$

$$x^{u\prime} = p' X^u.$$

and

(ii) *For any pair of job classes $i, j \in \mathcal{N}$, and any admissible scheduling policy u that gives priority to class-i over class-j jobs,*

$$x^u_{ij} = 0$$

.

The following result shows that a system that satisfies strong flow conservation laws must necessarily satisfy generalized conservation laws. As in Section 2.3, we shall define a matrix $A = (A^S_i)_{i\in\mathcal{N}, S\subseteq\mathcal{N}}$ and a set function $b : 2^{\mathcal{N}} \to \Re_+$ as functions of p, Q and B. We shall also write, for subsets S, T of $\mathcal{N}$, $p_S = (p_i)_{i\in\mathcal{N}}$, $A^S_T = (A^S_i)_{i\in T}$, $Q_{ST} = (q_{ij})_{i\in S, j\in T}$ (B_{ST} is defined similarly).

For every $S \subseteq \mathcal{N}$, let us define vector $A^S_{\mathcal{N}}$ as the solution of the linear system

$$A^S_S = p_S + Q_{SS}A^S_S,$$

$$A^S_{S^c} = p_{S^c} + Q_{S^cS}A^S_S,$$

and let

$$b(S) = \frac{1}{2}A^{S\prime}_S B_{SS}A^S_S.$$

Theorem 6 *If performance measures x^u, X^u satisfy strong flow conservation laws with parameters p, Q, B, then performance vector x^u satisfies the following generalized conservation laws:*

(a) For every policy $u \in \mathcal{U}$ that gives priority to jobs whose class belongs in S^c,

$$b(S) = \sum_{i\in S} A^S_i x^u_i, \qquad \text{for } S \subseteq \mathcal{N};$$

(b) For every scheduling policy $u \in \mathcal{U}$,

$$\sum_{i\in S} A^S_i x^u_i \geq b(S), \qquad \text{for all } S \subset \mathcal{N} \qquad \text{and} \qquad \sum_{i\in\mathcal{N}} A^{\mathcal{N}}_i x^u_i = b(\mathcal{N}),$$

Proof

Let $S \subseteq \mathcal{N}$. The same algebraic manipulations carried out in the proof of Theorem 4 yield

that, for any admissible policy u,

$$\begin{aligned} b(S) &= \sum_{i \in S} A_i^S x_i^u - \sum_{i \in S^c} \sum_{j \in S} A_i^S x_{ij}^u A_j^S \\ &\leq \sum_{i \in S} A_i^S x_i^u, \end{aligned} \tag{3.11}$$

with equality being achieved at (3.11) by any scheduling policy u that gives priority to jobs with classes in S^c (S^c-jobs) over S-jobs, since by definition $x_{ij}^u = 0$ if job class i has priority over job class j. □

Theorem 6 together with Theorem 5 imply directly that the performance region spanned by performance vectors x^u under admissible scheduling policies is the extended contra-polymatroid

$$\mathcal{P}_c(A,b) = \left\{ x \in \Re_+^n : \sum_{i \in S} A_i^S x_i \geq b(S), \quad \text{for } S \subset \mathcal{N} \quad \text{and} \quad \sum_{i \in \mathcal{N}} A_i^{\mathcal{N}} x_i = b(\mathcal{N}) \right\}.$$

A second consequence of Theorem 6 is that the performance region of performance vectors x^u's can be represented as the projection of a higher dimensional polyhedron defined by a polynomial number of constraints. Let us define polyhedron $\mathcal{P}(p,Q,B)$ by

$$\mathcal{P}(p,Q,B) = \left\{ (x,X) \in \Re_+^n \times \Re_+^{n^2} : (I-Q)'X + X'(I-Q) = B, \quad \text{and} \quad x' = p'X \right\}.$$

Theorem 7 *If performance measures x^u, X^u satisfy strong flow conservation laws with parameters p, Q, B, then*

(a) *The performance region $\{x^u : u \in \mathcal{U}\}$ is the extended contra-polymatroid $\mathcal{P}_c(A,b)$;*

(b) $\mathcal{P}_c(A,b) = \text{proj}_x(\mathcal{P}(p,Q,B))$.

Proof

Part (a) follows directly from Theorems 6 and 5.

Part (b) follows from Theorem 4, and the fact that for any permutation π of $\mathcal{N}$, the performance matrix $X^{u(\pi)}$ corresponding to the static-priority scheduling policy $u(\pi)$ that assigns priorities to job classes according to permutation π, with class π_1 having the highest priority, is easily seen to be a nonnegative solution of system

$$(I-Q)'X_\pi + X_\pi'(I-Q) = B.$$

Therefore, Assumption 3 holds, and Theorem 4 yields the required result. □

3.2.3 Optimization over systems that satisfy conservation laws

Let x^u be a performance vector for a multiclass queueing system that satisfies generalized conservation laws (3.8) and (3.9). Suppose that we want to find an admissible policy u that maximizes a linear reward function $\sum_{i\in\mathcal{N}} r_i x_i^u$. This optimal scheduling control problem can be expressed as

$$(LP_{\mathcal{U}}) \qquad \max\left\{\sum_{i\in\mathcal{N}} r_i x_i^u : u \in \mathcal{U}\right\}.$$

By Theorem 5, problem $(LP_{\mathcal{U}})$ can be translated into linear program

$$(LP_c) \qquad \max\left\{\sum_{i\in\mathcal{N}} r_i x_i : x \in \mathcal{P}_c(A,b)\right\}.$$

The strong structural properties of extended polymatroids lead to strong structural properties in the control problem. Suppose that to each job class-i we attach an index, γ_i. A policy that selects for service at each decision epoch a job of currently largest index will be referred to as a *priority-index* policy. Let $\gamma_1, \ldots, \gamma_n$ be the allocation indices of linear program (LP_c), obtained by running adaptive-greedy algorithm with input (r, A), as described in Chapter 2. As a direct consequence of the results of Section 2.2 we obtain that control problem $(LP_{\mathcal{U}})$ is solved by an index policy, with optimal priority indices given by $\gamma_1, \ldots, \gamma_n$.

Theorem 8 (Indexability under generalized conservation laws) *(a) Let $v(\pi)$ be an optimal solution of linear program (LP_c); then the static-priority policy that assigns priorities to job classes according to permutation π (class π_1 has highest priority) solves problem $(LP_{\mathcal{U}})$ optimally;*

(b) A policy that selects at each decision epoch a job of currently largest allocation index solves problem $(LP_{\mathcal{U}})$ optimally.

Index decomposition

A stronger *index decomposition* property holds under certain conditions. Assume $\mathcal{N}_1, \ldots, \mathcal{N}_K$ is a given partition of the set of job classes $\mathcal{N}$. Job classes in $\mathcal{N}_k$ may be interpreted as corresponding to a class-k *project*. Assume also that submatrix $A^k = (A_i^S)_{i\in\mathcal{N}_k, S\subseteq\mathcal{N}_k}$ of

A (as a function on the parameters of the system) depends only on characteristics of job classes in $\mathcal{N}_k$ (i.e., of project class-k). If Assumption 2 holds, i.e.,

$$A_i^S = A_i^{S\cap\mathcal{N}_k}, \quad \text{for } i \in S \cap \mathcal{N}_k \quad \text{and } S \subseteq \mathcal{N}.$$

then the Index decomposition theorem for extended polymatroids (Theorem 3) applies, and therefore

$$\gamma_i = \gamma_i^k, \qquad \text{for } i \in \mathcal{N}_k,$$

where the $\gamma_i^k(r^k, A^k)$'s, for $i \in \mathcal{N}_k$, are the indices obtained by algorithm $\mathcal{AG}$ with input (r^k, A^k), and $r^k = (r_i)_{i\in\mathcal{N}_k}$. Combining this result with Theorem 8 we obtain:

Theorem 9 (Index decomposition for multiclass queueing systems) *The allocation indices corresponding to job classes in $\mathcal{N}_k$ only depend on characteristics of project class-k.*

The previous theorem identifies a sufficient condition for the indices of an indexable system to have a strong *decomposition* property. Therefore, systems that in addition to generalized conservation laws further satisfy Assumption 2 are *decomposable*. For such systems the solution of optimal scheduling control problem (LP_U) can be obtained by solving K smaller independent subproblems using algorithm $\mathcal{AG}$.

An example of a decomposable system is the multiclass-$M/G/1$ queue with performance measure

$$x_i^u = \frac{N_i^u}{\mu_i}, i \in \mathcal{N}, \tag{3.12}$$

where N_i^u is the time average number of class-i jobs (with mean processing time $\frac{1}{\mu_i}$) in the system under scheduling policy u. Given a holding cost c_i per unit time for each job class-i, the goal is to assign the jobs to the server according to a nonanticipative, nonidling and nonpreemptive scheduling policy in order to minimize time average holding cost:

$$\min_u \sum_{i\in\mathcal{N}} c_i N_i^u = \sum_{i\in\mathcal{N}} (c_i\mu_i) x_i^u.$$

Gelenbe and Mitrani (1980) first showed that performance measure x^u given by (3.12) satisfies the following conservation laws:

$$\sum_{i\in S} x_i^u \geq b(S), \quad \text{for } S \subset \mathcal{N}, \tag{3.13}$$

$$\sum_{i \in \mathcal{N}} x_i^u = b(\mathcal{N}).$$

with equality in (3.13) if policy u gives priority to jobs with classes in S. Assumption 2 clearly holds in this case, since all $A_i^S = 1$, for $i \in S$ and $S \subseteq \mathcal{N}$. This is the reason that the optimal index for each job class depends only on characteristics of that job class: the allocation indices are given by $\gamma_i = c_i \mu_i$, thus explaining the optimality of the $c\mu$ rule from a linear programming perspective.

Another well known example of decomposable system is the multi-armed bandit problem. We will see later that in this case the allocation indices reduce indeed to the original Gittins indices. Furthermore, Theorem 9 explains the fact that the Gittins indices corresponding to the states of project k only depend on characteristics of that project.

Let us consider briefly the problem of optimizing a nonlinear cost function on the performance vector. Analogously as what we did in the linear rewards case, the optimal scheduling control problem in the case of a nonlinear reward function can be translated into a nonlinear program whose feasible region is an extended polymatroid. See Bhattacharya et al. (1991) for a discussion of algorithmic methods for solving separable convex, min-max, lexicographic and semi-separable convex optimization problems over an extended polymatroid.

Chapter 4

Applications: multi-armed bandits and other indexable scheduling problems

In this chapter we apply the framework developed in Chapter 3 to formulate and solve in a unifying way a variety of stochastic scheduling problems whose optimal policies are known to be of priority-index type. They include the classical multi-armed bandit problem (see e.g. Gittins (1989)), Klimov's problem (see Klimov (1974)), and the minimum weighted flow time problem in deterministic scheduling (see Smith (1956)). These and other indexable scheduling problems correspond to special cases of the branching bandit problem (see Meilijson and Weiss (1977), and Weiss (1988)).

The chapter is structured as follows: Section 4.1 introduces the branching bandit model, defines associated optimal scheduling problems, and shows how to cast them in the framework presented in Chapter 3. Suitable performance measures are defined, and it is established that they satisfy generalized conservation laws. The scheduling problems are thus formulated as linear programs over extended polymatroids, and they are shown to be solved by a priority-index policy, with the optimal indices being computed by a one-pass adaptive-greedy algorithm.

Section 4.2 specializes the previous results to several cases of the branching bandit model, which correspond to some classical problems in stochastic scheduling. The cases developed include the multi-armed bandit problem, the problem of optimal dynamic scheduling in a

multiclass queue with feedback, and deterministic scheduling problems.

4.1 Branching bandit problems

A branching bandit is a versatile model of a multiclass single-server queue. The systems that can be modeled as branching bandits include multiclass queues in discrete or continuous time, and with or without arrivals, as well as multi-armed bandits. This model was first introduced by Meilijson and Weiss (1977).

In a branching bandit model, a single server must be allocated over time to jobs demanding its service attention. Jobs are classified in a finite number of job classes $i \in \mathcal{N} = \{1, \ldots, n\}$. We associate with job class i a random service time v_i and random arrivals $(N_{ij})_{j \in \mathcal{N}}$. When a class-$i$ job completes its service, it is replaced by new jobs N_{ij} of class-j, for $j \in \mathcal{N}$. Given the job class-i, the service time and the descendants $(v_i, (N_{ij})_{j \in \mathcal{N}})$ are random variables with an arbitrary joint distribution, independent of all other jobs. Jobs are to be selected for service under an *admissible* policy u, which must be nonanticipative (decisions may be based on past and present information on the evolution of the system, but not on future information, such as the service time of the next job to be serviced), nonidling (the server is busy as long as there are jobs in the system) and nonpreemptive (the service of a job, once started, must proceed without interruption until its completion). Let us denote $\mathcal{U}$ the class of admissible policies. The decision epochs are $t = 0$ and the instants at which a job is completed and there is some other job present.

We next introduce some notation and concepts that will be useful for analyzing the sample-path of a branching bandit process. Let $S \subseteq \mathcal{N}$ be a subset of job classes. We shall refer to jobs whose classes are in S as S-jobs. We are interested in studying the busy period of a branching bandit process. This busy period may be represented as a tree. We say that S-job i_1 is an S-descendant of job i_0 if i_1 belongs in the subtree of the busy period that is rooted at i_0. Given a job i in a busy period, the union of intervals where S-descendants of i are being processed is called an (i, S)-period. Notice that under a policy that gives complete priority to S-jobs, these intervals will be consecutive. Let T_i^S be the duration (possibly infinite) of an (i, S)-period. We define $T_{L(0)}^S$ as the time needed for first clearing the system of S-jobs, under a policy that gives priority to S-jobs. The distributions of T_i^S and of $T_{L(0)}^S$ are independent of the admissible policy used, as long as it gives priority to S-

jobs. Notice that an $(i,\emptyset)$-period is distributed as the service time v_i. It will be convenient to introduce the following additional notation:

v_{ik} = service time corresponding to the kth selection of a class-i job; notice that the distribution of $v_{i,k}$ is independent of k (v_i).

τ_{ik} = time at which the kth selection of a class-i job occurs:

ν_i = number of times a class-i job is selected (can be infinite);

$\{T^S_{ik}\}_{k\geq 1}$= duration of the (i,S)-period that starts with the kth selection of a class-i job for the kth time. Notice that the distribution of $T^S_{i,k}$ is independent of k (T^S_i).

$I_i(t)$ = 1, if a class-i job is in service at time t; 0, otherwise.

$L_i(t)$ = number of class-i jobs in the system at time t. We denote $L(t) = (L_i(t))_{i\in\mathcal{N}}$.

$T^S_{L(0)}$ = time until the system is first cleared of S-jobs (can be infinite) under a policy that gives priority to S-jobs: notice that $T^{\mathcal{N}}_{L(0)}$ is the length of the busy period.

The busy period of a branching bandit process has a simple structure under priority policies. This fact, which will be needed later for proving that certain performance measures satisfy conservation laws, is made precise and shown next. Let $S \subseteq \mathcal{N}$.

Proposition 4 *Under an admissible policy that gives priority to S^c-jobs, the busy period $[0, T^{\mathcal{N}}_{L(0)})$ can be partitioned as follows:*

$$[0, T^{\mathcal{N}}_{L(0)}) = [0, T^{S^c}_{L(0)}) \bigcup_{i\in S} \bigcup_{k=1}^{\nu_i} [\tau_{ik}, \tau_{ik} + T^{S^c}_{ik}). \tag{4.1}$$

Proof outline

Identity (4.1) simply expresses the intuitive fact that under a policy that gives complete priority to S^c-jobs, the busy period is partitioned into: (1) an initial interval, in which the system is first cleared of S^c-jobs; and (2) a sequence of consecutive intervals, each of which starts with the service of an S-job, and lasts until the system is first cleared of its descendant S^c-jobs. □

4.1.1 Discounted branching bandits

The discounted reward-tax problem

Consider the following linear discounted reward-tax structure on a branching bandit process: an instantaneous reward r_i is received at the completion epoch of a class-i job. In addition, a holding tax αh_i is incurred continuously while a class-i job is in the system. Rewards and taxes are discounted in time with a discount factor $\alpha > 0$.

The discounted reward-tax problem consists in finding an admissible scheduling policy that maximizes the total expected discounted value of rewards earned minus taxes incurred. This problem was first introduced and shown to be solved by a priority-index policy by Weiss (1988).

Let us define the objective to be maximized as $Z_u^{(r,h)}(\alpha)$, where

$Z_u^{(r,h)}(\alpha) =$ expected total discounted value of rewards received minus taxes incurred under scheduling policy u.

The problem can now be written as

$$\max_{u \in \mathcal{U}} Z_u^{(r,h)}(\alpha).$$

We shall show how to formulate and solve this problem in the framework developed in Chapter 3.

Performance measures

We introduce next two families of performance measures for branching bandits, $\{\lambda^u(\alpha)\}_{\alpha>0}$ and $\{L_j^{*u}(\alpha)\}_{\alpha>0}$, that are appropriate for modeling the linear discounted reward-tax structure described above. For a given $\alpha > 0$, we define performance measure $\lambda_j^u(\alpha)$ of class-j jobs under policy u to be total expected discounted number of class-j job service completions, i.e.,

$$\begin{aligned} \lambda_j^u(\alpha) &= E_u\left[\sum_{k=1}^{\nu_j} e^{-\alpha(\tau_{jk}+v_{jk})}\right] \\ &= E\left[e^{-\alpha v_i}\right] E_u\left[\sum_{k=1}^{\nu_j} e^{-\alpha \tau_{jk}}\right], \qquad \text{for } j \in \mathcal{N}, \end{aligned} \tag{4.2}$$

and we shall write $\lambda^u(\alpha) = (\lambda_j^u(\alpha))_{j\in\mathcal{N}}$.

Let us define $L_j^{*u}(\alpha)$ as the total expected discounted number of class-j jobs in the system under policy $u \in \mathcal{U}$, i.e.,

$$L_j^{*u}(\alpha) = E_u\left[\int_0^{T^{\mathcal{N}}_{L(0)}} L_j(t)\, e^{-\alpha t}\, dt\right], \qquad \text{for } j \in \mathcal{N}. \tag{4.3}$$

and let $L^{*u}(\alpha) = \left(L_j^{*u}(\alpha)\right)_{j\in\mathcal{N}}$.

We shall show next that the objective to be maximized, $Z_u^{(r,h)}(\alpha)$, can be expressed as a linear function of performance vector $\lambda^u(\alpha)$.

In the pure rewards case, i.e. when $h = 0$.

$$\begin{aligned} Z_u^{(r,0)}(\alpha) &= E_u\left[\sum_{i\in\mathcal{N}}\sum_{k=1}^{\nu_i} r_i e^{-\alpha(\tau_{ik}+v_{ik})}\right] \\ &= \sum_{i\in\mathcal{N}} r_i \lambda_i^u(\alpha). \end{aligned} \tag{4.4}$$

We show now how to reduce the general reward-tax problem to the pure rewards case, using an accounting argument introduced by Bell (1971). The total expected discounted value of holding taxes is the same whether they are charged continuously in time, or according to the following charging scheme: At the arrival epoch of a class-i job, charge the system with an instantaneous *entrance charge* of h_i, equal to the total discounted holding cost that would be incurred if the job remained within the system forever; at the departure epoch of the job (if it ever departs), credit the system with an instantaneous *departure refund* of h_i, thus refunding that portion of the entrance cost corresponding to residence beyond the departure epoch. We can thus write

$$\begin{aligned} Z_u^{(r,h)}(\alpha) &= E_u[\text{Rewards}] - E_u[\text{Charges at } t=0] + \\ &\quad (E_u[\text{Departure refunds}] - E_u[\text{Entrance charges}]) \\ &= Z_u^{(r,0)}(\alpha) - \sum_{i\in\mathcal{N}} h_i L_i(0) + Z_u^{(r_\alpha,v)}(\alpha) \\ &= Z_u^{(r+r_\alpha,0)}(\alpha) - \sum_{i\in\mathcal{N}} h_i L_i(0) \\ &= \sum_{i\in\mathcal{N}} (r_i + r_{i,\alpha})\, \lambda_i^u(\alpha) - \sum_{i\in\mathcal{N}} h_i L_i(0). \end{aligned} \tag{4.5}$$

where the components of vector $r_\alpha = (r_{i,\alpha})_{i\in\mathcal{N}}$ are given by

$$r_{i,\alpha} = h_i - \sum_{j\in\mathcal{N}} \frac{E\left[N_{ij}e^{-\alpha v_i}\right]}{E\left[e^{-\alpha v_j}\right]} h_j. \qquad \text{for } j \in \mathcal{N}. \tag{4.6}$$

Notice that by letting $r = 0$, $h_j = 1$ and $h_i = 0$ for $i \neq j$ in (4.5) we obtain a linear relation between performance vectors $L^{*u}(\alpha)$ and $\lambda^u(\alpha)$:

$$\alpha L_j^{*u}(\alpha) = L_j(0) - \lambda_j^u(\alpha) + \sum_{i\in\mathcal{N}} \frac{E\left[N_{ij}e^{-\alpha v_i}\right]}{E\left[e^{-\alpha v_j}\right]} \lambda_i^u(\alpha), \qquad \text{for } j \in \mathcal{N}. \tag{4.7}$$

Generalized conservation laws

We show in this section that the performance measure for branching bandits $\lambda^u(\alpha)$ satisfies generalized conservation laws. Let us define, for $S \subseteq \mathcal{N}$,

$$A_{i,\alpha}^S = \frac{\alpha E\left[\int_0^{T_i^{S^c}} e^{-\alpha t}\, dt\right]}{E\left[e^{-\alpha v_i}\right]}. \qquad \text{for } i \in S, \tag{4.8}$$

and

$$b_\alpha(S) = \alpha E\left[\int_0^{T_{L(0)}^{\mathcal{N}}} e^{-\alpha t}\, dt\right] - \alpha E\left[\int_0^{T_{L(0)}^{S^c}} e^{-\alpha t}\, dt\right]. \tag{4.9}$$

The conservation laws we present next represent physical work conservation relations in a branching bandit process. In particular, the total expected discounted amount of work performed by the server during the busy period is

$$E\left[\int_0^{T_{L(0)}^{\mathcal{N}}} e^{-\alpha t}\, dt\right],$$

under any admissible scheduling policy $u \in \mathcal{U}$.

Notice that coefficient $A_{i,\alpha}^S$ may be interpreted as the total expected discounted amount of work performed by the server during and (i, S^c)-period under a policy that gives priority to S^c-jobs (and hence under which that (i, S^c)-period is a single interval). The sum

$$\sum_{i\in S} A_{i,\alpha}^S \lambda_i^u(\alpha)$$

is thus an overestimate of the total expected discounted work performed after the server first starts servicing S-jobs. This estimate is exact under a policy that gives priority to

S^c-jobs. In this case, the expected discounted work done until the system is first cleared of S^c-jobs is

$$E\left[\int_0^{T^{S^c}_{L(0)}} e^{-\alpha t}\,dt\right].$$

The remaining work is, in that case,

$$E\left[\int_0^{T^{\mathcal{N}}_{L(0)}} e^{-\alpha t}\,dt\right] - E\left[\int_0^{T^{S^c}_{L(0)}} e^{-\alpha t}\,dt\right].$$

which is precisely $b_\alpha(S)$. This intuitive interpretation is made precise in the proof of the next result.

Theorem 10 (Generalized conservation laws for discounted branching bandits) *The performance vector for branching bandits $\lambda^u(\alpha)$ satisfies the following generalized conservation laws:*

(a) $\sum_{i\in S} A^S_{i,\alpha}\lambda^u_i(\alpha) \geq b_\alpha(S)$, *for $S \subset \mathcal{N}$, with equality if policy u gives complete priority to S^c-jobs.*

(b) $\sum_{i\in\mathcal{N}} A^{\mathcal{N}}_{i,\alpha}\lambda^u_i(\alpha) = b_\alpha(\mathcal{N})$.

Proof

Let $S \subseteq \mathcal{N}$. Let us assume that jobs are selected under an admissible policy u. Let us define two random vectors, $(r_i)_{i\in\mathcal{N}}$ and $(d^S_i)_{i\in S}$, as functions of the sample path of the generated branching bandit process as follows:

$$\begin{aligned} r_i &= \int_0^\infty I_i(t)e^{-\alpha t}\,dt = \sum_{k=1}^{\nu_i}\int_{\tau_{ik}}^{\tau_{ik}+v_{ik}} e^{-\alpha t}\,dt \\ &= \sum_{k=1}^{\nu_i} e^{-\alpha\tau_{ik}}\int_0^{v_{ik}} e^{-\alpha t}\,dt, \end{aligned} \tag{4.10}$$

and

$$d^S_i = \sum_{k=1}^{\nu_i} e^{-\alpha\tau_{ik}}\int_0^{T^{S^c}_{ik}} e^{-\alpha t}\,dt, \quad i \in S.$$

Now, we have

$$\begin{aligned} E_u[r_i] &= E_u\left[\sum_{k=1}^{\nu_i} e^{-\alpha\tau_{ik}}\int_0^{v_{ik}} e^{-\alpha t}\,dt\right] \\ &= E_u\left[\sum_{k=1}^{\nu_i} e^{-\alpha\tau_{ik}}\int_0^{v_{ik}} e^{-\alpha t}\,dt \mid \nu_i\right] \end{aligned}$$

$$= E_u\left[\sum_{k=1}^{\nu_i} E[e^{-\alpha\tau_{ik}} \mid \nu_i]\, E\left[\int_0^{\upsilon_{ik}} e^{-\alpha t}\, dt\right]\right] \tag{4.11}$$

$$= E\left[\int_0^{\upsilon_i} e^{-\alpha t}\, dt\right] E_u\left[\sum_{k=1}^{\nu_i} e^{-\alpha\tau_{ik}}\right]$$

$$= \frac{E\left[\int_0^{\upsilon_i} e^{-\alpha t}\, dt\right]}{E\left[e^{-\alpha\upsilon_i}\right]}\, \lambda_i^u(\alpha). \tag{4.12}$$

Notice that equality (4.11) holds because, since u is nonanticipative, τ_{ik} and υ_{ik} are independent random variables. Furthermore,

$$E_u[d_i^S] = \mathrm{E}_u\left[\sum_{k=1}^{\nu_i} e^{-\alpha\tau_{ik}} \int_0^{T_{ik}^{S^c}} e^{-\alpha t}\, dt\right] = E_u\left[E\left[\sum_{k=1}^{\nu_i} e^{-\alpha\tau_{ik}} \int_0^{T_{ik}^{S^c}} e^{-\alpha t}\, dt \mid \nu_i\right]\right]$$

$$= E_u\left[\sum_{k=1}^{\nu_i} E\left[\int_0^{T_i^{S^c}} e^{-\alpha t}\, dt\right] E\left[e^{-\alpha\tau_{ik}} \mid \nu_i\right]\right] \tag{4.13}$$

$$= E\left[\int_0^{T_i^{S^c}} e^{-\alpha t}\, dt\right] E_u\left[\sum_{k=1}^{\nu_i} e^{-\alpha\tau_{ik}}\right]$$

$$= A_{i,\alpha}^S\, E\left[\int_0^{\upsilon_i} e^{-\alpha t}\, dt\right] E_u\left[\sum_{k=1}^{\nu_i} e^{-\alpha\tau_{ik}}\right]. \tag{4.14}$$

Notice that equality (4.13) holds because, since u is nonanticipative, τ_{ik} and $T_{ik}^{S^c}$ are independent. Hence, by (4.12) and (4.14)

$$E_u[d_i^S] = A_{i,\alpha}^S \lambda_i^u(\alpha), \qquad \text{for } i \in S,$$

and we thus obtain

$$E_u\left[\sum_{i\in S} d_i^S\right] = \sum_{i\in S} A_{i,\alpha}^S \lambda_i^u(\alpha). \tag{4.15}$$

We first show that if policy $u = \pi$ gives complete priority ty S^c-jobs then generalized conservation law (a) holds with equality. Applying Proposition 4, we obtain

$$\int_0^{T_{L(0)}^{\mathcal{N}}} e^{-\alpha t}\, dt = \int_0^{T_{L(0)}^{S^c}} e^{-\alpha t}\, dt + \sum_{i\in S}\sum_{k=1}^{\nu_i} \int_{\tau_{ik}}^{\tau_{ik}+T_{ik}^{S^c}} e^{-\alpha t}\, dt$$

$$= \int_0^{T_{L(0)}^{S^c}} e^{-\alpha t}\, dt + \sum_{i\in S}\sum_{k=1}^{\nu_i} e^{-\alpha\tau_{ik}} \int_0^{T_{ik}^{S^c}} e^{-\alpha t}\, dt$$

$$= \int_0^{T_{L(0)}^{S^c}} e^{-\alpha t}\, dt + \sum_{i\in S} d_i^S. \tag{4.16}$$

Hence, taking expectations and using equation (4.15) we obtain

$$E\left[\int_0^{T_{L(0)}^{N}} e^{-\alpha t}\,dt\right] = E\left[\int_0^{T_{L(0)}^{S^c}} e^{-\alpha t}\,dt\right] + \sum_{i\in S} A_{i,\alpha}^{S}\lambda_i^{\pi}(\alpha)$$

or equivalently, by (4.9),

$$\sum_{i\in S} A_{i,\alpha}^{S}\lambda_i^{\pi}(\alpha) = b_\alpha(S),$$

which proves that generalized conservation law (a) holds with equality. Notice that by letting $S = \emptyset$ we obtain the conservation law in part (b).

We next show that generalized conservation law (a) is satisfied in the inequality case. We will use a sample path interchange argument.

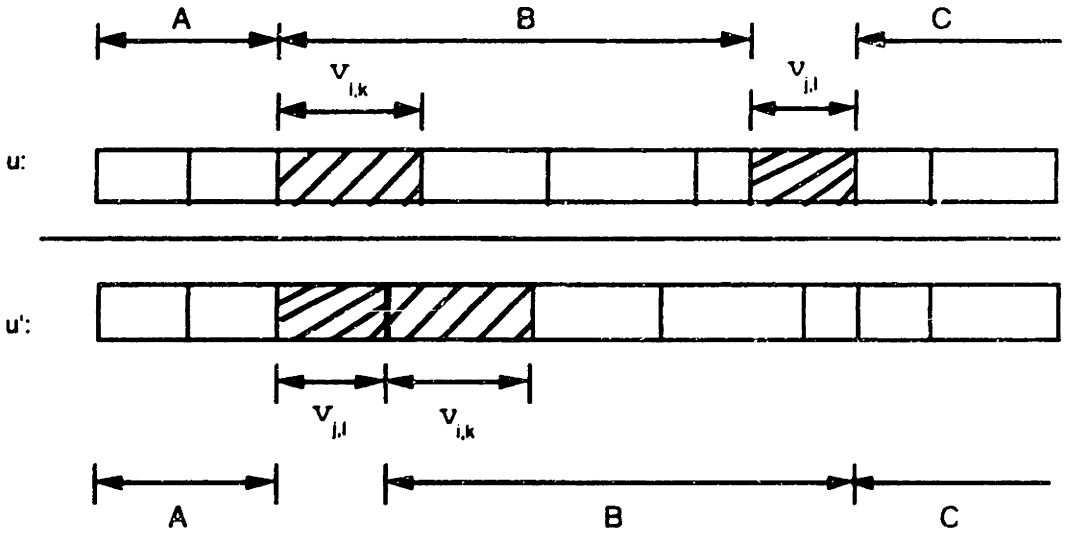

Figure 4-1: Interchange argument.

Let the jobs be selected under an admissible policy u. For a given sample path of the branching bandit process let us consider the sum

$$d^S = \sum_{i\in S} d_i^S = \sum_{i\in S}\sum_{k=1}^{\nu_i}\int_{\tau_{ik}}^{\tau_{ik}+T_{ik}^{S^c}} e^{-\alpha t}\,dt.$$

Suppose that at time τ_{i^*,k^*} a class-i^* job, with $i^* \in S$ is selected for the k^*th time. Suppose that at that time a class-j^* job, with $j^* \in S^c$ is also available, but it is selected later, at time τ_{j^*,l^*}. Let us consider the effect of selecting instead that class-j^* job in S^c at time τ_{i^*,k^*}, and selecting immediately afterwards the class-i^* job. Let us call the corresponding policy u'. Let A, B and C be the segments shown in Figure 4-1. The sum d^S can be decomposed

as

$$
\begin{aligned}
d^S &= \sum_{i\in S}\sum_{k=1:\tau_{ik}\in A}^{\nu_i}\int_{\tau_{ik}}^{\tau_{ik}+T_{ik}^{S^c}} e^{-\alpha t}\,dt + \\
&\quad \sum_{i\in S}\sum_{k=1:\tau_{ik}\in B}^{\nu_i}\int_{\tau_{ik}}^{\tau_{ik}+T_{ik}^{S^c}} e^{-\alpha t}\,dt + \\
&\quad \sum_{i\in S}\sum_{k=1:\tau_{ik}\in C}^{\nu_i}\int_{\tau_{ik}}^{\tau_{ik}+T_{ik}^{S^c}} e^{-\alpha t}\,dt. \qquad (4.17)
\end{aligned}
$$

Let d'^S be the sum corresponding to policy u'. It is clear from Figure 4-1, (4.17) and the fact that the function $e^{-\alpha t}$ is decreasing in t that

$$
\begin{aligned}
d'^S &= \sum_{i\in S}\sum_{k=1:\tau_{ik}\in A}^{\nu_i}\int_{\tau_{ik}}^{\tau_{ik}+T_{ik}^{S^c}} e^{-\alpha t}\,dt + \\
&\quad \sum_{i\in S}\sum_{k=1:\tau_{ik}\in B}^{\nu_i}\int_{\tau_{ik}+v_{j^*,l^*}}^{\tau_{ik}+v_{j^*,l^*}+T_{ik}^{S^c}} e^{-\alpha t}\,dt + \\
&\quad \sum_{i\in S}\sum_{k=1:\tau_{ik}\in C}^{\nu_i}\int_{\tau_{ik}}^{\tau_{ik}+T_{ik}^{S^c}} e^{-\alpha t}\,dt \\
&< d^S. \qquad (4.18)
\end{aligned}
$$

It follows that for this sample path a policy that gives complete priority to S^c-jobs minimizes the sum d^S. Hence this result holds taking expectations, and by equation (4.15) conservation law (a) follows, which completes the proof of the theorem. □

Since performance measure $\lambda^u(\alpha)$ satisfies generalized conservation laws, the results of Section 3.2 apply. Direct application of Theorem 5 yields the next result.

Corollary 2 *The performance region for branching bandits corresponding to performance vector $\lambda^u(\alpha)$ is the extended contra-polymatroid $\mathcal{P}_e(A_\alpha, b_\alpha)$; furthermore, the vertices of $\mathcal{P}_e(A_\alpha, b_\alpha)$ are the performance vectors corresponding to static-priority policies.*

Optimal solution

From equation (4.5) it is clear how to apply the results of Section 3.2 to solve the reward-tax problem: run adaptive-greedy algorithm $\mathcal{AG}$ (see Appendix A) with input $(r + r_\alpha, A_\alpha)$. Let $\gamma_1(\alpha), \ldots, \gamma_n(\alpha)$ be the allocation indices so obtained. Then we have, by Theorem 8,

Theorem 11 (Indexability: discounted branching bandits) *An optimal scheduling policy is to serve at each decision epoch a job with largest current allocation index* $\gamma_i(\alpha)$.

The previous theorem characterizes the structure of the optimal policy. We will see later that we can also compute the performance of the optimal policy, as we will present in Proposition 6 closed formulae for matrix A_α and set function b_α.

Economic interpretation of allocation indices in discounted branching bandits

The original definition of dynamic allocation indices in multi-armed bandit problems (see e.g. Gittins (1989)) characterized them as optimal reward rates with respect to a family of stopping times. Our definition of allocation indices, on the other hand, involves sums of optimal dual variables in a certain linear program. In this section we clarify the relation between these two definitions, and show that they are, indeed, equivalent.

Given a discounted branching bandit problem, as described above, we shall define a modified problem by adding an additional job class, which we label 0, with infinite service time (i.e., $v_0 = \infty$). A reward of r_0, continuously discounted in time, is received for each unit of time that a class-0 job is in service. Notice that the option to serve a class-0 job may be interpreted as an option to retire from the game modeled by the original problem for a *pension* of r_0, discounted in time. Notice also that the modified problem is still a branching bandit problem.

Let us now assume that at time $t = 0$ there are only two jobs present, a class-0 and a class-i job, with $i \in \mathcal{N}$. We may then consider the following question: What is the smallest value of the pension r_0 that makes the option of retirement (serving the class-0 job) preferable to the option of continuation (serving the class-i job)? Let us call this *break-even value* $r_0^*(i)$. Let $\gamma_1, \ldots, \gamma_n$ be the allocation indices corresponding to the original branching bandit problem.

Proposition 5

$$\gamma_i = r_0^*(i).$$

Proof

Let $\gamma_0^0, \gamma_1^0, \ldots, \gamma_n^0$ be the allocation indices for the modified problem. Let us partition the corresponding modified state space as $\hat{\mathcal{N}} = \{0\} \cup \mathcal{N}$. It is easily seen that Assumption 2

holds for the modified problem. Hence Theorem 3 applies, and the problem is decomposable. Consequently,

$$\gamma_0^0 = r_0 \qquad \text{and} \qquad \gamma_j^0 = \gamma_j, \quad j \in \mathcal{N}.$$

Now, since by Theorem 11 it is optimal to serve a job with largest current allocation index, it follows that the break-even reward r_0 which makes the options of continuation and of retirement (with reward r_0) equally attractive is $r_0 = \gamma_0^0$. But, by definition, $r_0^*(i)$ is such a breakpoint. Therefore $r_0^*(i) = \gamma_0^0$, which completes the proof. □

Remark. Whittle (1980), (1982) introduced the idea of a retirement option in his analysis of the multi-armed bandit problem, and provided an interpretation of the Gittins indices as break-even values. Weber (1992) also made use of this characterization of the Gittins indices in his intuitive proof. Here we extend this interpretation to the more general case of branching bandits. From this characterization it follows that the allocation indices coincide with the well known Gittins indices in the classical multi-armed bandit problem.

Parameter computation

The results of the previous sections are structural, but do not lead to explicit computations of matrix A_α and set function b_α appearing in the generalized conservation laws for the branching bandit problem. Our goal in this section is to compute from the model data matrix A_α and set function b_α. Combined with the previous results these computations make possible to evaluate the performance of specific policies.

As generic data for the branching bandit model, we assume that the joint distribution of $(v_i, (N_{ij})_{j \in \mathcal{N}})$ is given by the transform

$$\Phi_i(\alpha, z_1, \ldots, z_n) = \mathrm{E}\left[e^{-\alpha v_i} z_1^{N_{i1}} \ldots z_n^{N_{in}} \right]. \tag{4.19}$$

Notice that

$$E[\, N_{ij} e^{-\alpha v_i} \,] = \frac{\partial}{\partial z_j} \Phi_i(\alpha, 1). \tag{4.20}$$

In addition, we are given the Laplace transform of the marginal distribution of service time v_i,

$$\Psi_i(\alpha) = E\left[e^{-\alpha v_i} \right].$$

Finally the vector $L(0) = (L_1(0), \ldots, L_n(0))$ of jobs present at the start is assumed to be known.

As we saw in the previous section the duration of an (i, S)-period, T_i^S, plays a crucial role. We will compute its Laplace transform function.

$$\Psi_i^S(\alpha) = E[e^{-\alpha T_i^S}].$$

For this reason we decompose the duration of an (i, S)-period as a sum of independent random variables as follows:

$$T_i^S \stackrel{\mathrm{d}}{=} v_i + \sum_{j \in S} \sum_{k=1}^{N_{ij}} T_{j,k}^S, \tag{4.21}$$

where v_i, $\{T_{j,k}^S\}_{k \geq 1}$ are independent. Therefore,

$$\begin{aligned} \Psi_i^S(\alpha) &= E\left[e^{-\alpha v_i} E\left[e^{-\alpha \sum_{j \in S} \sum_{k=1}^{N_{ij}} T_{j,k}^S} \mid v_i\right]\right] \\ &= E\left[e^{-\alpha v_i} \prod_{j \in S} E\left[e^{-\alpha T_j^S}\right]^{N_{ij}}\right] \\ &= \Phi_i\left(\alpha, \Psi_S^S(\alpha), 1_{S^c}\right), \qquad \text{for } i \in \mathcal{N}. \end{aligned} \tag{4.22}$$

Given S, fixed point system (4.22) provides a way to compute the values of $\Psi_i^S(\alpha)$, for $i \in \mathcal{N}$. We can now prove the following result:

Proposition 6 (Computation of A_α and b_α) *For a branching bandit process, matrix A_α and set function b_α satisfy the following relations:*

$$A_{i,\alpha}^S = \frac{1 - \Psi_i^{S^c}(\alpha)}{\Psi_i(\alpha)}, \qquad \text{for } i \in S \text{ and } S \subseteq \mathcal{N}; \tag{4.23}$$

$$b_\alpha(S) = \prod_{S^c} [\Psi_j^{S^c}(\alpha)]^{L_j(0)} - \prod_{j \in \mathcal{N}} [\Psi_j^{\mathcal{N}}(\alpha)]^{L_j(0)}, \quad S \subseteq \mathcal{N} \tag{4.24}$$

Proof

Relation (4.23) follows directly from the definition of $A_{i,\alpha}^S$. Furthermore,

$$T_{L(0)}^S \stackrel{\mathrm{d}}{=} \sum_{i \in S} \sum_{k=1}^{L_i(0)} T_{ik}^S. \tag{4.25}$$

Hence,

$$E\left[\int_0^{T^S_{L(0)}} e^{-\alpha t}\,dt\right] = \frac{1}{\alpha} - \frac{1}{\alpha}E\left[e^{-\alpha\sum_{i\in S}\sum_{k=1}^{L_i(0)}T^S_{ik}}\right]$$
$$= \frac{1}{\alpha} - \frac{1}{\alpha}\prod_{i\in S}[\Psi^S_i(\alpha)]^{L_i(0)}. \qquad (4.26)$$

Therefore, from (4.9), (4.24) follows. □

Remarks:

1. Notice that $A^{\mathcal{N}}_{i,\alpha} = \frac{1-\Psi_i(\alpha)}{\alpha\Psi_i(\alpha)}$, for $i \in \mathcal{N}$, and $b_\alpha(\mathcal{N}) = \frac{1}{\alpha} - \frac{1}{\alpha}\prod_{j\in\mathcal{N}}[\Psi^{\mathcal{N}}_j(\alpha)]^{L_j(0)}$, for $S \subseteq \mathcal{N}$.

2. From Proposition 6 we can compute matrix A_α and set function b_α provided we can solve system (4.22). As an example, we illustrate the form of the equations in the special case, in which the class-j jobs that arrive during the time that we work on class-i job form a Poisson process with rate λ_{ij}, i.e.,

$$\Phi_i(\alpha, z_1, \ldots, z_n) = \mathrm{E}\left[e^{-\upsilon_i(\alpha+\sum_{j\in\mathcal{N}}\lambda_{ij}(1-z_j))}\right] = \Psi_i\left(\alpha + \sum_{j\in\mathcal{N}}\lambda_{ij}(1-z_j)\right).$$

In this case, (4.22) yields

$$\Psi^S_i(\alpha) = \Psi_i\left(\alpha + \sum_{j\in S}\lambda_{ij}\left(1-\Psi^S_j(\alpha)\right)\right), i \in \mathcal{N} \qquad (4.27)$$

As a result, an algorithm to compute $\Psi^S_i(\alpha)$ is as follows:

(1) Find a fixed point for the system of nonlinear equations (4.27) in terms of $\Psi^S_i(\alpha)$. Although in general (4.27) might not have a closed form solution, in special cases (υ_i exponential) a closed form solution can be obtained.

(2) From Proposition 6 compute (A_α, b_α) in terms of $\Psi^{S^c}_i(\alpha)$.

4.1.2 Undiscounted branching bandits

In this section we apply the framework developed in Chapter 3 to the branching bandit problem with a linear undiscounted cost criterion. This problem was first introduced and solved by Meilijson and Weiss (1977) using dynamic programming ideas.

We shall assume in what follows that matrix $E[N] = (E[N_{ij}])_{i,j\in\mathcal{N}}$ satisfies the following

condition:

Assumption 4 *Matrix $E[N]$ has spectral radius smaller than 1.*

Bertsimas, Paschalidis and Tsitsiklis (1994b) proved that under Assumption 4, the branching bandit process is *stable*, in the sense that the first and second moments of its busy period are finite.

The undiscounted tax problem

In the undiscounted tax problem, a holding tax h_i per unit time is incurred continuously while a class-i job is in the system. We do not consider undiscounted rewards (i.e. a reward r_i is earned on completion of a class-i job) since the total expected reward earned is the same under all policies. Let us define the objective to be maximized, Z_u^h, as

$Z_u^h = -$(total expected tax incurred under policy u).

The tax problem can now be written as the optimal control problem

$$\max_{u \in \mathcal{U}} Z_u^h.$$

Performance measures

We introduce next two performance measures, $C^u = (C_j^u)_{j \in \mathcal{N}}$ and $W^u = (W_j^u)_{j \in \mathcal{N}}$, that are appropriate for modeling a linear undiscounted delay cost structure. We assume in what follows that all the expectations that appear are finite. Later we will show necessary and sufficient conditions for this assumption to hold. Using the indicator

$$I_j(t) = \begin{cases} 1, & \text{if a class-}j\text{ job is in service at time } t; \\ 0, & \text{otherwise.} \end{cases}$$

we introduced earlier, let us define performance measure C_j^u as the total expected completion time of class-j jobs under policy u, i.e.,

$$C_j^u = E_u\left[\sum_{k=1}^{\nu_j}(\tau_{jk} + v_{jk})\right]. \tag{4.28}$$

Let us define another performance measure, W_j^u, as the total expected system time of

class-j jobs under policy u, i.e.,

$$W_j^u = E_u\left[\int_0^{T_{L(0)}^{\mathcal{N}}} L_j(t)\,dt\right]. \tag{4.29}$$

We show now how to express objective Z_u^h as a linear function of performance vectors W^u or C^u. First, it is clear from the definition of the undiscounted tax problem that

$$Z_u^h = -\sum_{j\in\mathcal{N}} h_j W_j^u. \tag{4.30}$$

As for the relation with performance vector C^u, it is obtained by taking the limit as $\alpha \searrow 0$ on equation (4.7), which yields

$$L_j^{*u}(0) = -\frac{d}{d\alpha}\lambda_j^u(0) + \sum_{i\in\mathcal{N}} E[N_{ij}]\,\frac{d}{d\alpha}\lambda_i^u(0) + g_j, \qquad \text{for } j \in \mathcal{N},$$

where g_j is given by

$$g_j = \sum_{i\in\mathcal{N}} E[\nu_i]\left(E[v_i]E[N_{ij}] - E\left[v_i N_{ij}\right]\right). \tag{4.31}$$

Since

$$\frac{d}{d\alpha}\lambda_j^u(0) = -C_j^u,$$

we thus obtain the following linear relation between performance vectors W^u and C^u:

$$W_j^u = C_j^u - \sum_{i\in\mathcal{N}} E[N_{ij}]C_i^u + g_j, \qquad \text{for } j \in \mathcal{N}. \tag{4.32}$$

We can also express objective Z_u^h as a linear function of performance vector C^u. By (4.32),

$$Z_u^h = -\sum_{j\in\mathcal{N}} h_j W_j^u \tag{4.33}$$

$$= \sum_{i\in\mathcal{N}}\left(h_i - \sum_{j\in\mathcal{N}} E[N_{ij}]h_j\right) C_i^u - \sum_{i\in\mathcal{N}} h_i g_i, \tag{4.34}$$

Conservation laws for the completion times

We show next that performance measures C^u satisfies generalized conservation laws. Let us define

$$A_i^S = E\left[T_i^{S^c}\right], \qquad \text{for } i \in S, \tag{4.35}$$

and

$$b(S) = \frac{1}{2}E\left[\left(T_{L(0)}^{\mathcal{N}}\right)^2\right] - \frac{1}{2}E\left[\left(T_{L(0)}^{S^c}\right)^2\right] + \sum_{i\in S} b_i(S), \tag{4.36}$$

where

$$b_i(S) = E[\nu_i]\left(E[v_i]E\left[T_i^{S^c}\right] - \frac{1}{2}E\left[(T_i^{S^c})^2\right]\right) \qquad \text{for } i \in S.$$

Theorem 12 (Generalized conservation laws for the completion times) *The performance vector for branching bandits C^u satisfies the following generalized conservation laws:*

(a) $\sum_{i\in S} A_i^S C_i^u \leq b(S)$, *for* $S \subset \mathcal{N}$, *with equality if policy* u *gives complete priority to* S^c*-jobs.*

(b) $\sum_{i\in\mathcal{N}} A_i^{\mathcal{N}} C_i^u = b(\mathcal{N})$.

Proof

Let $S \subseteq \mathcal{N}$. Let us assume that jobs are selected under an admissible policy u. This generates a branching bandit process. Let us define two random vectors, $(r_i)_{i\in\mathcal{N}}$ and $(d_i^S)_{i\in S}$, as functions of the sample path as follows:

$$\begin{aligned} r_i &= \int_0^\infty I_i(t)\,t\,dt = \sum_{k=1}^{\nu_i}\int_{\tau_{ik}}^{\tau_{ik}+v_{ik}} t\,dt \\ &= \sum_{k=1}^{\nu_i}\left(v_{ik}\,\tau_{ik} + \frac{v_{ik}^2}{2}\right), \quad i \in \mathcal{N}, \end{aligned} \tag{4.37}$$

and

$$d_i^S = \sum_{k=1}^{\nu_i}\int_{\tau_{ik}}^{\tau_{ik}+T_{ik}^{S^c}} t\,dt, \quad i \in S.$$

Now, we have

$$\begin{aligned} E_u[r_i] &= E_u\left[\sum_{k=1}^{\nu_i} E\left[(v_{ik}\,\tau_{ik} + \frac{v_{ik}^2}{2}) \mid \nu_i\right]\right] \\ &= E_u\left[\sum_{k=1}^{\nu_i}\left(E[v_i]\,E[\tau_{ik}|\nu_i] + \frac{E[v_i^2]}{2}\right)\right] \end{aligned} \tag{4.38}$$

$$= E[v_i]\, E_u\left[\sum_{k=1}^{\nu_i} \tau_{ik}\right] + \frac{E[\nu_i]\, E[v_i^2]}{2}. \tag{4.39}$$

Note that equality (4.38) holds because, since u is nonanticipative, τ_{ik} and v_{ik} are independent random variables. Furthermore,

$$\begin{aligned} E_u[d_i^S] &= E_u\left[\sum_{k=1}^{\nu_i} \int_{\tau_{ik}}^{\tau_{ik}+T_{ik}^{S^c}} t\, dt\right] = E_u\left[E\left[\sum_{k=1}^{\nu_i} \int_{\tau_{ik}}^{\tau_{ik}+T_{ik}^{S^c}} t\, dt \mid \nu_i\right]\right] \\ &= E_u\left[\sum_{k=1}^{\nu_i} E\left[(\tau_{ik}\, T_{ik}^{S^c} + \frac{(T_{ik}^{S^c})^2}{2}) \mid \nu_i\right]\right] \\ &= E[T_i^{S^c}]\, E_u\left[\sum_{k=1}^{\nu_i} \tau_{ik}\right] + \frac{E[\nu_i]\, E[(T_i^{S^c})^2]}{2}. \end{aligned} \tag{4.40}$$

Note that equality (4.40) holds because, since policy u is nonanticipative, τ_{ik} and $T_{ik}^{S^c}$ are independent random variables. Hence, by (4.39) and (4.40),

$$C_i^u - E[\nu_i]E[v_i] = \frac{E_u[\tau_i] - \frac{1}{2}E[\nu_i]\, E[v_i^2]}{E[v_i]} = \frac{E_u[d_i^S] - \frac{1}{2}E[\nu_i]\, \mathrm{E}[(T_i^{S^c})^2]}{E[T_i^{S^c}]}, \quad i \in S$$

and therefore we obtain

$$\sum_{i\in S} A_i^S C_i^u = E_u\left[\sum_{i\in S} d_i^S\right] - \sum_{i\in S} b_i(S). \tag{4.41}$$

We will first show that if policy $u = \pi$ gives complete priority to S^c jobs then generalized conservation law (a) holds with equality. Applying Proposition 4 we obtain:

$$\begin{aligned} \int_0^{T_{L(0)}^{\mathcal{N}}} t\, dt &= \int_0^{T_{L(0)}^{S^c}} t\, dt + \sum_{i\in S}\sum_{k=1}^{\nu_i} \int_{\tau_{ik}}^{\tau_{ik}+T_{ik}^{S^c}} t\, dt \\ &= \frac{(T_{L(0)}^{S^c})^2}{2} + \sum_{i\in S} d_i^S. \end{aligned} \tag{4.42}$$

Hence, taking expectations and using equation (4.41) and the definition of $b(S)$ we obtain

$$\sum_{i\in S} A_i^S C_i^\pi = b(S).$$

which proves that generalized conservation law (a) holds with equality. Notice that by letting $S = \emptyset$ part (b) follows.

We next show that generalized conservation law (a) is satisfied in the inequality case. We will use a sample path interchange argument. Let the jobs be selected under an admissible policy u. For a given sample path of the branching bandit process let us consider the sum

$$d^S = \sum_{i \in S} d_i^S = \sum_{i \in S} \sum_{k=1}^{\nu_i} \int_{\tau_{ik}}^{\tau_{ik}+T_{ik}^{S^c}} t\, dt.$$

Suppose that at time $\tau_{i^\bullet,k^\bullet}$ a class-$i^\bullet$ job, with $i^\bullet \in S$ is selected for the $k^\bullet$th time. Suppose that at that time a class-$j^\bullet$ job, with $j^\bullet \in S^c$ is also available, but it is selected later, at time $\tau_{j^\bullet,l^\bullet}$. Let us consider the effect of selecting instead that class-$j^\bullet$ job in S^c at time $\tau_{i^\bullet,k^\bullet}$, and selecting immediately afterwards the class-$i^\bullet$ job. Let us call the corresponding policy u'. Let A, B and C be the segments shown in Figure 4-1. The sum d^S can be decomposed as

$$\begin{aligned} d^S \;=\; & \sum_{i \in S} \sum_{k=1:\tau_{ik} \in A}^{\nu_i} \int_{\tau_{ik}}^{\tau_{ik}+T_{ik}^{S^c}} t\, dt + \\ & \sum_{i \in S} \sum_{k=1:\tau_{ik} \in B}^{\nu_i} \int_{\tau_{ik}}^{\tau_{ik}+T_{ik}^{S^c}} t\, dt + \\ & \sum_{i \in S} \sum_{k=1:\tau_{ik} \in C}^{\nu_i} \int_{\tau_{ik}}^{\tau_{ik}+T_{ik}^{S^c}} t\, dt. \end{aligned} \tag{4.43}$$

Let d'^S be the sum corresponding to policy u'. It is clear from Figure 4-1, (4.43) and the fact that the function t is increasing in t that

$$\begin{aligned} d'^S \;=\; & \sum_{i \in S} \sum_{k=1:\tau_{ik} \in A}^{\nu_i} \int_{\tau_{ik}}^{\tau_{ik}+T_{ik}^{S^c}} t\, dt + \\ & \sum_{i \in S} \sum_{k=1:\tau_{ik} \in B}^{\nu_i} \int_{\tau_{ik}+v_{j^\bullet,l^\bullet}}^{\tau_{ik}+v_{j^\bullet,l^\bullet}+T_{ik}^{S^c}} t\, dt + \\ & \sum_{i \in S} \sum_{k=1:\tau_{ik} \in C}^{\nu_i} \int_{\tau_{ik}}^{\tau_{ik}+T_{ik}^{S^c}} t\, dt \\ > \;\; & d^S. \end{aligned}$$

It follows that for this sample path a policy that gives complete priority to S^c-jobs maximizes the sum d^S. Hence this result holds taking expectations, and by equation (4.41) conservation law (a) follows, which completes the proof of the theorem. □

Corollary 3 *The performance space for branching bandits corresponding to performance vector C^u is the extended polymatroid $\mathcal{P}(A, b)$; furthermore, the vertices of $\mathcal{P}(A, b)$ are the performance vectors corresponding to static-priority rules.*

Conservation laws for the number in system

We show next that the performance measure for branching bandits W^u defined by (4.29) satisfies generalized conservation laws. Let us define, for $S \subseteq \mathcal{N}$,

$$\bar{A}_i^S = E[T_i^S], \qquad \text{for } i \in S, \tag{4.44}$$

and

$$\bar{b}(S) = b(\mathcal{N}) - b(S^c) + \sum_{j \in S} g_j E[T_j^S]. \tag{4.45}$$

Theorem 13 (Generalized conservation laws for the number in system) *The performance vector for branching bandits W^u satisfies the following generalized conservation laws:*

(a) $\sum_{i \in S} \bar{A}_i^S W_i^u \geq \bar{b}(S)$, *for $S \subset \mathcal{N}$, with equality if policy u gives complete priority to S-jobs.*

(b) $\sum_{i \in \mathcal{N}} \bar{A}_i^{\mathcal{N}} W_i^u = \bar{b}(\mathcal{N})$.

Proof

By applying equation (4.32) for relating C^u with W^u, we obtain

$$\begin{aligned}
\sum_{j \in S} \bar{A}_i^S W_i^u &= C^{u\prime}(I - E[N]) \begin{pmatrix} E\left[T_S^S\right] \\ 0 \end{pmatrix} + g_S{}' E[T_S^S] \\
&= C^{u\prime} \begin{pmatrix} (I_S - E[N_{SS}]) E\left[T_S^S\right] \\ -E[N_{S^cS}]\, E\left[T_S^S\right] \end{pmatrix} + g_S{}' E\left[T_S^S\right] \\
&= C^{u\prime} \begin{pmatrix} 1_S \\ 1_{S^c} - E\left[T_{S^c}^S\right] \end{pmatrix} + g_S{}' E\left[T_S^S\right] \qquad (4.46) \\
&= b(\mathcal{N}) - \sum_{i \in S^c} A_i^{S^c} C_i^u + \sum_{j \in S} g_j E\left[T_j^S\right], \qquad (4.47)
\end{aligned}$$

where (4.46) follows from the results in Section 4.1.2 below. Now, by the conservation laws satisfied by C^u (see Theorem 12) the result follows. □

Corollary 4 *The performance space for branching bandits corresponding to performance vector W'^u is the extended contra-polymatroid $\mathcal{P}_c(\bar{A}, \bar{b})$; furthermore, the vertices of $\mathcal{P}_c(\bar{A}, \bar{b})$ are the performance vectors corresponding to static-priority rules.*

Optimal solution

From equations (4.33) and (4.34) and the conservation laws satisfied by C^u and W'^u we obtain two different algorithms for solving the control problem: the first one corresponds to running algorithm $\mathcal{AG}$ with input $(-h, \bar{A})$, and it is a bottom-up algorithm (priorities are computed from lowest to highest); the second one corresponds to running algorithm $\mathcal{AG}$ with input $(\hat{r}, A)$, where

$$\hat{r}_i = h_i - \sum_{j \in \mathcal{N}} E[N_{ij}] h_j, \tag{4.48}$$

and it is a top-down algorithm (i.e., priorities are computed from highest to lowest).

Parameter computation

In this section we show how to compute matrix A and set function b from the branching model data. Recall that

$$A_i^S = E\left[T_i^{S^c}\right], \qquad \text{for } i \in S.$$

and

$$b(S) = \frac{1}{2} E\left[\left(T_{L(0)}^{\mathcal{N}}\right)^2\right] - \frac{1}{2} E\left[\left(T_{L(0)}^{S^c}\right)^2\right] + \sum_{i \in S} E[v_i] \left(E[v_i] E\left[T_i^{S^c}\right] - \frac{1}{2} E\left[(T_i^{S^c})^2\right]\right).$$

From equation (4.21) we obtain, taking expectations:

$$E\left[T_i^S\right] = E[v_i] + \sum_{j \in S} E[N_{ij}]\, E\left[T_j^S\right], \qquad \text{for } i \in \mathcal{N}. \tag{4.49}$$

Solving this linear system we obtain $E\left[T_i^S\right]$. Note that the computation of A_i^S is much easier in the undiscounted case compared with the discounted case, where we had to solve a system of nonlinear equations. Also, applying the conditional variance formula to (4.21)

we obtain:

$$\mathrm{Var}\left[T_i^S\right] = \mathrm{Var}[v_i] + \left(E\left[T_j^S\right]\right)'_{j\in S} \mathrm{Cov}\left[(N_{ij})_{j\in S}\right] \left(E\left[T_j^S\right]\right)_{j\in S} + \sum_{j\in S} E[N_{ij}]\,\mathrm{Var}\left[T_j^S\right], \; i \in \mathcal{N}. \tag{4.50}$$

Solving this linear system we obtain $\mathrm{Var}\left[T_i^S\right]$ and thus $E\left[(T_i^S)^2\right]$. Moreover, the expected number of class j jobs serviced during the busy period, $E[\nu_j]$, for $j \in \mathcal{N}$, can be obtained by solving the linear system

$$E[\nu_j] = L_j(0) + \sum_{i\in\mathcal{N}} E[N_{ij}]\,E[\nu_i], \qquad \text{for } j \in \mathcal{N}.$$

Furthermore, from equation (4.25) we obtain

$$E\left[T_{L(0)}^S\right] = \sum_{i\in S} L_i(0)\,E\left[T_i^S\right], \tag{4.51}$$

and

$$\mathrm{Var}\left[T_{L(0)}^S\right] = \sum_{i\in S} L_i(0)\,\mathrm{Var}\left[T_i^S\right]. \tag{4.52}$$

For computing $\bar{b}(S)$ the quantity $E[v_i N_{ij}]$ is needed. It is easy to see that

$$E[v_i N_{ij}] = \frac{\partial}{\partial\alpha\,\partial z_j}\Phi_i(1, 1_{\mathcal{N}}).$$

4.1.3 Summary

Table 4.1 summarizes the problems we considered, the performance measures used, the conservation laws, the corresponding performance region, as well as the input to algorithm $\mathcal{AG}$.

4.2 Special cases

In this section we specialize the previous formulations for branching bandit problems to several classical stochastic scheduling problems. For each problem we define suitable performance measures, characterize explicitly the corresponding performance region and show how to compute the optimal priority indices. Table 4.2 summarizes these and other indexable scheduling problems that can be modeled and solved using the mathematical programming

Problem	Performance measure	Perfomance region	Indices
$\max_{u \in \mathcal{U}} Z_u^{(r,h)}(\alpha)$	$\lambda_j^u(\alpha)$	$\mathcal{P}_c(A_\alpha, b_\alpha)$ A_α, b_α: see (4.23), (4.24)	$(r_\alpha, A_\alpha) \xmapsto{\mathcal{AG}} \gamma(\alpha)$ r_α: see (4.6)
$\max_{u \in \mathcal{U}} Z_u^h$	C_j^u	$\mathcal{P}(A, b)$ A, b: see (4.35), (4.36)	$(\hat{r}, A) \xmapsto{\mathcal{AG}} \gamma$ $\hat{r}$: see (4.48)
	W_j^u	$\mathcal{P}_c(\bar{A}, \bar{b})$ $\bar{A}$, $\bar{b}$: see (4.44), (4.45)	$(-h, \bar{A}) \xmapsto{\mathcal{AG}} \gamma$

Table 4.1: Branching bandit problems: formulation and solution.

framework developed in Chapter 3.

System	Criterion	Indexability	Performance Space
Batch of jobs	LC[a]	Smith (1956): D[b] Rothkopf (1966b)	Queyranne (1993): D, P[c] This thesis: P
	DC[d]	Rothkopf (1966a): D Gittins & Jones (1974)	This thesis: P
Batch of jobs with out-tree prec. constraints	LC	Horn (1972): D Meilijson & Weiss (1977)	This thesis: EP[e]
	DC	Glazebrook (1976)	This thesis: EP
Multiclass $M/G/1$	LC	Cox & Smith (1961)	Coffman & Mitrani (1980): P Gelenbe & Mitrani (1980): P
	DC	Harrison (1975a, 1975b)	This thesis: EP
Multiclass [f] $M/G/c$	LC	Federgruen & Groenevelt (1988b) Shanthikumar & Yao (1992)	Federgruen & Groenevelt (1988b): P Shanthikumar & Yao (1992): P
Multiclass $G/M/c$	LC	Federgruen & Groenevelt (1988a) Shanthikumar & Yao (1992)	Federgruen & Groenevelt (1988a): P Shanthikumar & Yao (1992): P
Multiclass Jackson network [g]	LC	Ross & Yao (1989)	Ross & Yao (1989): P
Multiclass $M/G/1$ with feedback	LC	Klimov (1974)	Tsoucas (1991): EP
	DC	Tcha & Pliska (1977)	This thesis: EP
Multi-armed bandits	DC	Gittins & Jones (1974)	This thesis: EP
Branching bandits	LC	Meilijson & Weiss (1977)	This thesis: EP
	DC	Weiss (1988)	This thesis: EP

Table 4.2: Indexable problems and their performance regions.

[a] Linear cost
[b] Deterministic processing times
[c] Polymatroid
[d] Discounted linear reward-cost
[e] Extended polymatroid
[f] Same service time distribution for all classes
[g] Same service time distribution and routing probabilities for all classes (can be node dependent)

4.2.1 The multi-armed bandit problem

Problem definition

The multi-armed bandit problem can be described as follows: There are K parallel projects, indexed $k = 1, \ldots, K$. Project k can be in one of a finite number of states $j_k \in \mathcal{N}_k$. At each instant of discrete time $t = 0, 1, \ldots$ one must work on exactly one project. If one works on project k in state $j_k(t)$ at time t, then an immediate reward of $r_{kj_k(t)}$ is earned. Rewards are additive and are discounted in time by a discount factor $0 < \beta < 1$. The state $j_k(t)$ changes to $j_k(t+1)$ by a homogeneous Markov transition rule, with transition matrix $P^k = (p^k_{ij})_{i,j \in \mathcal{N}_k}$, while the states of the projects one has not engaged remain frozen. The problem is how to allocate one's resources to projects in a sequential manner, according to a policy u which must belong in the class $\mathcal{U}$ of nonanticipative and nonidling policies, in order to maximize the total expected discounted reward earned over an infinite horizon,

$$Z = \max_{u \in \mathcal{U}} E_u \left[\sum_{t=0}^{\infty} \beta^t r_{k(t) j_{k(t)}(t)} \right].$$

Modeling the problem as a branching bandit

We shall model the problem as a discounted branching bandit problem in order to apply the results of Section 4.1.1. We shall thus identify project states with job classes, in such a way that working on a project in state i corresponds to serving a class-i job. Using the notation of Section 4.1.1, let us define $\mathcal{N} = \cup_{k=1}^{K} \mathcal{N}_k$, $e^{-\alpha} = \beta$, and $v_i \equiv 1$. We also define matrix $P = (p_{ij})_{i,j \in \mathcal{N}}$ by

$$p_{ij} = \begin{cases} p^k_{ij}, & \text{if } i, j \in \mathcal{N}_k, \text{ for some } k = 1, \ldots, K; \\ 0, & \text{otherwise.} \end{cases}$$

In this model, a class-i job has only one descendant, which is of class-j with probability p_{ij}. Let us also define the discrete-time indicator

$$I_j(t) = \begin{cases} 1, & \text{if a project in state } j \text{ is engaged at time } t; \\ 0, & \text{otherwise.} \end{cases}$$

Performance measures

The performance measure $\lambda^u(\alpha) = \left(\lambda_j^u(\alpha)\right)_{i\in\mathcal{N}}$ for discounted branching bandits introduced in Section 4.1.1 is easily seen to simplify into

$$\lambda_j(\alpha)^u = \beta E_u\left[\sum_{t=0}^{\infty} I_j(t)\beta^t\right], \qquad \text{for } j \in \mathcal{N}.$$

We may interpret $\lambda_j(\alpha)$ as the total expected discounted time spent working on class-j projects under policy u.

In terms of these performance measures, the multi-armed bandit problem can be written as

$$\beta Z = \max_{u\in\mathcal{U}} \sum_{i\in\mathcal{N}} r_i \lambda_i^u(\alpha).$$

Performance region

By Theorem 10, we know that performance measure $\lambda^u(\alpha)$ satisfies generalized conservation laws, and that the performance region it spans is an extended contra-polymatroid.

Parameter computation. We shall show next how to compute the corresponding parameters $A_{i,\alpha}$ and $b_\alpha(S)$. For $S \subseteq \mathcal{N}$, let vector $t_S^S = (t_i^S)_{i\in\mathcal{N}}$ be defined as the solution of linear system

$$t_i^S = 1 + \beta \sum_{j\in S} p_{ij} t_j^S, \qquad \text{for } i \in S, \tag{4.53}$$

Let us also define

$$L_j(0) = \begin{cases} 1, & \text{if there is a project in state } j \text{ at the start;} \\ 0, & \text{otherwise.} \end{cases}$$

Proposition 7 *Matrix A_α and set function b_α are given by the following expressions:*

(a)

$$A_{i,\alpha}^S = \frac{1-\beta}{\beta}\left(1 + \beta \sum_{j\in S^c} p_{ij} t_j^{S^c}\right), \qquad \textit{for } i \in S; \tag{4.54}$$

(b)

$$b_\alpha(S) = \prod_{j\in S^c} \left(1 - (1-\beta) t_j^{S^c}\right)^{L_j(0)}, \qquad \textit{for } S \subseteq \mathcal{N} \tag{4.55}$$

Proof

We have, by equation (4.19),

$$\begin{aligned}\Phi_i(\alpha, z_1, \ldots, z_n) &= E\left[e^{-\alpha v_i} z_1^{N_{i1}} \cdots z_n^{N_{in}}\right] \\ &= e^{-\alpha} \sum_{j \in \mathcal{N}} p_{ij} z_j \\ &= \beta \sum_{j \in \mathcal{N}_k} p_{ij} z_j, \qquad \text{for } i \in \mathcal{N}_k \end{aligned} \tag{4.56}$$

and, by (4.22),

$$\begin{aligned}\Psi_i^S(\alpha) &= \Phi_i\left(\alpha, \left(\Psi_j^S(\alpha)\right)_{j \in S}, 1_{S^c}\right) \\ &= \beta\left(\sum_{j \in S} p_{ij} \Psi_j^S(\alpha) + \sum_{j \in S^c} p_{ij}\right) \\ &= \beta\left(1 - \sum_{j \in S} p_{ij}\left(1 - \Psi_j^S(\alpha)\right)\right), \qquad \text{for } i \in \mathcal{N}. \end{aligned} \tag{4.57}$$

Now, it follows from (4.57) that

$$t_i^S = \frac{1 - \Psi_i^S(\alpha)}{1 - \Psi_i(\alpha)}, \qquad \text{for } i \in S,$$

and part (a) follows by (4.57) and Proposition 6.

Moreover, since $\Psi_j^{\mathcal{N}}(\alpha) = 0$, Proposition 6 yields

$$\begin{aligned} b_\alpha(S) &= \prod_{j \in S^c} [\Psi_j^{S^c}(\alpha)]^{L_j(0)} \\ &= \prod_{j \in S^c} \left(1 - (1 - \beta) t_j^{S^c}\right)^{L_j(0)}, \end{aligned} \tag{4.58}$$

which proves (b). □

Proposition 7 together with Corollary 2 yield directly the following result.

Proposition 8 (Performance region for multi-armed bandits) *The performance region spanned by performance vectors $\lambda^u(\alpha)$ in the multi-armed bandit problem is the extended*

contra-polymatroid defined by

$$\sum_{i\in\mathcal{N}}\left(1+\beta\sum_{j\in S^c}p_{ij}t_j^{S^c}\right)\lambda_i \geq \frac{\beta}{1-\beta}\prod_{j\in S^c}\left(1-(1-\beta)t_j^{S^c}\right)^{L_j(0)}, \qquad \text{for } S\subset\mathcal{N},$$

$$\sum_{i\in\mathcal{N}}\lambda_i = \frac{\beta}{1-\beta},$$

$$\lambda_i \geq 0, \qquad \text{for } i\in\mathcal{N}.$$

Optimal solution

Gittins and Jones (1974) first showed that the optimal policy for the multi-armed bandit problem is a priority-index policy. These optimal priority indices can be computed by running adaptive greedy algorithm $\mathcal{AG}$ with input (r, A_α).

Index decomposition. Gittins and Jones (1974) further showed that the optimal indices associated with states of a project only depend on characteristics of that project (rewards and transition probabilities).

Theorem 14 (Gittins and Jones (1974)) *For each project k there exist indices $\{\gamma_i^k\}_{i\in\mathcal{N}_k}$, depending only on characteristics of project k, such that an optimal policy is to work at each time on a project with largest current index.*

This classical result follows in our framework as a consequence of Theorem 3 on the decomposition of optimal indices. In particular, the structure of matrix $P = (p_{ij})$ implies that

$$A_{j,\alpha}^{S} = A_{j,\alpha}^{S\cap\mathcal{N}_k}, \qquad \text{for } j\in S\cap\mathcal{N}_k,$$

so that Assumption 2 holds.

By the results of Section 4.1.1 we know that the allocation indices for this bandit problem are precisely the dynamic allocation indices introduced by Gittins and Jones(1974) (also called Gittins indices). Furthermore, by definition of allocation indices. weobtain a characterization of Gittins indices as sums of dual variables, a purely algebraic characterization. By Theorem 9, the Gittins indices can be computed by solving K subproblems, applying adaptive greedy algorithm $\mathcal{AG}$, presented in Chapter 2, to subproblem k, which has $|\mathcal{N}_k|$ job classes, for $k = 1,\ldots,K$.

The index-computing algorithm proposed by Varaiya, Walrand and Buyukkoc (1985) has the same time complexity as adaptive greedy algorithm $\mathcal{AG}$. In fact, both algorithms are closely related, as we will see next. Let t_i^S be as given by (4.53). Let r_i^S be given by

$$r_i^S = r_i + \beta \sum_{j \in S} p_{ij} r_j^S, \qquad \text{for } i \in S.$$

The algorithm of Varaiya, Walrand and Buyukkoc can be stated as follows:

Algorithm $\mathcal{VWB}$:

Step 0. Pick $\pi_n \in \text{argmax} \left\{ \frac{r_i^{\{i\}}}{t_i^{\{i\}}} : i \in \mathcal{N} \right\}$; let $g_{\pi_n} = \max \left\{ \frac{r_i^{\{i\}}}{t_i^{\{i\}}} : i \in \mathcal{N} \right\}$;
set $J_n = \{\pi_n\}$.

Step k. For $k = 1, \ldots, n-1$:
pick $\pi_{n-k} \in \text{argmax} \left\{ \frac{r_i^{J_{n-k} \cup \{i\}}}{t_i^{J_{n-k} \cup \{i\}}} : i \in \mathcal{N} \setminus J_{n-k} \right\}$; set $g_{\pi_{n-k}} = \left\{ \frac{r_i^{J_{n-k} \cup \{i\}}}{t_i^{J_{n-k} \cup \{i\}}} : i \in \mathcal{N} \setminus J_{n-k} \right\}$;
set $J_{n-k} = J_{n-k+1} \cup \{\pi_{n-k}\}$.

Varaiya et al. (1985) proved that $g_1, \ldots, g_n$, as given by algorithm $\mathcal{VWB}$, are the Gittins indices of the multi-armed bandit problem. Let $(\pi, \overline{y}, \nu, S)$ be an output of algorithm $\mathcal{AG}$. The following relation between algorithms $\mathcal{AG}$ and $\mathcal{VWB}$ can be easily seen to hold by induction:

Proposition 9 *The following relations hold: For* $j = 2, \ldots, n$

$$\frac{r_i^{\{\pi_j, \ldots, \pi_n\} \cup \{i\}}}{t_i^{\{\pi_j, \ldots, \pi_n\} \cup \{i\}}} - \frac{r_i - \sum_{l=j}^{n} A_{i,\alpha}^{\{\pi_1, \ldots, \pi_l\}} \nu_l}{A_{i,\alpha}^{\{\pi_1, \ldots, \pi_{j-1}\}}} \equiv \frac{r_{\pi_j}^{\{\pi_j, \ldots, \pi_n\}}}{t_{\pi_j}^{\{\pi_j, \ldots, \pi_n\}}}, \qquad \textit{for } i \in \{\pi_1, \ldots, \pi_{j-1}\},$$

and

$$\frac{r_i^{\{i\}}}{t_i^{\{i\}}} - \frac{r_i}{A_{i,\alpha}^{\mathcal{N}}} \equiv 0, \qquad \textit{for } i \in \mathcal{N},$$

and therefore, algorithms $\mathcal{AG}$ and $\mathcal{VWB}$ are equivalent.

Algorithm $\mathcal{AG}$ thus provides a new *off-line* top-down method (i.e., priorities are computed from highest to lowest) for computing Gittins indices. As shown above, it has the same computational complexity as the algorithm of Varaiya et al. (the fastest off-line algorithm for computing Gittins indices known). The algorithms presented by Chen and Katehakis (1986) and by Katehakis and Veinott (1987) are *on-line* methods (they compute the Gittins index of a given state).

A closed formula for the optimal value function; submodularity

In this section we present a closed formula for the optimal value of a multi-armed bandit problem. We apply that formula to provide a new proof that the optimal value of the problem, seen as a function of the subset of projects involved, is a submodular set function.

Given a multi-armed bandit problem with K projects, let us consider subproblem k, the one-armed bandit problem corresponding to project k, for $k = 1, \ldots, K$. Let

$$\gamma^k_{i_k}, \qquad \text{for } i_k \in \mathcal{N}_k$$

be the Gittins indices corresponding to subproblem (project) k. Let b^k be the set function corresponding to subproblem k as given by equation (4.58), i.e.,

$$b^k(S_k) = \prod_{j \in \mathcal{N}_k \setminus S_k} \left(1 - (1-\beta) t_j^{\mathcal{N}_k \setminus S_k}\right)^{L_j(0)}, \qquad \text{for } S_k \subseteq \mathcal{N}_k.$$

Proposition 10 *For each k, l:*

$$b(S_k \cup S_l) = b^k(S_k) b^l(S_l), \quad \text{for } S_k \subseteq \mathcal{N}_k, S_l \subseteq \mathcal{N}_l. \tag{4.59}$$

Proof Outline Using the fact that $t_j^S = t_j^{S \cap \mathcal{N}_k}$, if $j \in \mathcal{N}_k$ the result follows trivially. □

Now, in order to solve the maximum reward problem, we run algorithm $\mathcal{AG}$ with input (r, A_α). Let π be a permutation of $\mathcal{N}$ produced by the algorithm. If $\{\gamma_i\}_{i \in \mathcal{N}}$ are the corresponding Gittins indices, π must satisfy

$$\gamma_{\pi_n} \leq \cdots \leq \gamma_{\pi_1}.$$

Permutation π of $\mathcal{N}$ induces permutations π^k of $\mathcal{N}_k$, for $k = 1, \ldots, K$.

If $\{\gamma^k_i\}_{i \in \mathcal{N}_k}$ are the Gittins indices corresponding to subproblem k, π^k must satisfy

$$\gamma_{\pi^k_{|\mathcal{N}_k|}} \leq \cdots \leq \gamma_{\pi^k_1}.$$

Let us define independent random variables η, $\eta_k \in \mathcal{N}_k$, for $k = 1, \ldots, K$, by

$$P\{\eta \in \{\pi_1, \ldots, \pi_n\}\} = b(\{\pi_1, \ldots, \pi_n\}), \qquad \text{for } i = 1, \ldots, n, \tag{4.60}$$

and

$$P\left\{\eta_k \in \{\pi_i^k, \ldots, \pi_{|\mathcal{N}_k|}^k\}\right\} = b^k\left(\{\pi_i^k, \ldots, \pi_{|\mathcal{N}_k|}^k\}\right), \qquad \text{for } i = 1, \ldots, |\mathcal{N}_k|. \tag{4.61}$$

Given a subset of projects $H \subseteq \{1, \ldots, K\}$, let us denote $Z(H)$ the optimal reward that can be obtained in the multi-armed bandit problem when only projects in subset H are available. We have the following result:

Theorem 15 (Optimal reward of the multi-armed bandit problem) *The optimal reward $Z(H)$ can be expressed as*

$$Z(H) = E\left[\max_{k \in H} \gamma_{\eta_k}^k\right]. \tag{4.62}$$

Proof

Since projects can be aggregated, it is enough to prove the theorem for the case of two projects, i.e. $K = 2$. First, notice that by equation (2.7), the expression for the optimal objective value of a linear program over an extended contra-polymatroid, and the characterization of the performance space of the multi-armed bandit as an extended polymatroid, we obtain:

$$\begin{aligned} Z(\{1,2\}) &= (\gamma_{\pi_1}, \gamma_{\pi_2}, \ldots, \gamma_{\pi_n}) \begin{pmatrix} b(\{\pi_1, \ldots, \pi_n\}) - b(\{\pi_2, \ldots, \pi_n\}) \\ \vdots \\ b(\{\pi_{n-1}, \pi_n\}) - b(\{\pi_n\}) \\ b(\{\pi_n\}) \end{pmatrix} \\ &= E[\gamma_\eta]. \end{aligned}$$

Now, let $H = \{1, 2\}$. In order to prove that equation (4.62) holds, it is enough to show that the following two random variables have the same distribution:

$$\max(\gamma_{\eta_1}^1, \gamma_{\eta_2}^2) \stackrel{\mathrm{d}}{=} \gamma_\eta. \tag{4.63}$$

We have, for a given γ_{i^*}:

$$\begin{aligned} P\{\max(\gamma_{\eta_1}^1, \gamma_{\eta_2}^2) \leq \gamma_{i^*}\} &= P\{\gamma_{\eta_1}^1 \leq \gamma_{i^*}\} P\{\gamma_{\eta_2}^2 \leq \gamma_{i^*}\} \\ &= b^1(\{i_1 \in \mathcal{N}_1 : \gamma_{i_1}^1 \leq \gamma_{i^*}\}) b^2(\{i_2 \in \mathcal{N}_2 : \gamma_{i_2}^2 \leq \gamma_{i^*}\}) \end{aligned}$$

$$= b(\{i \in \mathcal{N} : \gamma_i \leq \gamma_{i^*}\})$$
$$= P\{\gamma_\eta \leq \gamma_{i^*}\}. \tag{4.64}$$

Hence, the equality in distribution (4.63) holds, and the result follows. □

Corollary 5 (Submodularity of the optimal reward function) *The optimal reward function $Z(H)$ of the multi-armed bandit problem is a submodular set function.*

Proof

Since the function $H \mapsto \max_{k \in H} d_k$ is submodular, for any given vector of d_k's, the result follows directly from formula (4.62). □

The fact that the optimal reward function of the multi-armed bandit problem is submodular was first shown by Weber (1992), who proved it from first principles. Tsitsiklis (1986) provided an early result in this direction.

4.2.2 Scheduling control of a multiclass queue with Bernoulli feedback

A multiclass $M/G/1$ queue with Bernouilli feedback can be described as follows: A single server provides service to jobs, which are classifed in a finite number n of classes. External arrivals of class-i jobs follow a Poisson process of rate λ_i, for $i \in \mathcal{N} = \{1, \ldots, n\}$. Service times for class-i jobs are independent and identically distributed as a random variable v_i with distribution function G_i. When the service of a class-i job is completed, the job may either join the queue of class-j jobs, with probability p_{ij} (thus becoming a class-j job) or, with probability $1 - \sum_{j \in \mathcal{N}} p_{ij}$, leave the system.

The server must select the jobs for service according to an admissible scheduling policy $u \in \mathcal{U}$, which must be nonidling, nonpreemptive and nonanticipative; the decision epochs are $t = 0$ (if there is initially some job present), epochs at which a job arrives to find the system empty, and epochs at which a job completes service.

Klimov (1974) solved, by direct methods, the associated optimal control problem over $\mathcal{U}$ with a time-average holding cost criterion. Harrison (1975a) solved, using dynamic programming, the optimal control problem over $\mathcal{U}_0$ with a discounted reward-cost criterion, in the special case that there is no feedback. Tcha and Pliska (1977) extended Harrison's results to the case with feedback. They also solved the control problem over $\mathcal{U}^p$, in the case that the service times are exponential.

Problem description: the discounted case

Let us consider the following reward-cost structure: a continuous holding cost h_i is incurred per unit time that a class class-i job stays in the system. Furthermore, an instantaneous reward r_i is earned at the service completion epoch of a class-i job. We shall also consider an instantaneous idleness reward r_0 at the end of an idle period. All costs and rewards are continuously discounted in time by a discount factor $\alpha > 0$.

The optimal control problem is to find an admissible policy to schedule the server that maximizes the total expected discounted value of the rewards earned minus the holding costs incurred over an infinite horizon. Let us denote $LP_{\mathcal{U}}$ the optimal control problem.

Modeling as a discounted branching bandit problem

We will model the problem as a discounted branching bandit problem, thus casting them into the framework presented in Section 4.1.1.

First, let us consider problem $LP_{\mathcal{U}}$. This problem can be modeled as a branching bandit problem with n job classes, as follows: The class-j descendants N_{ij} of a class-i job are composed of the internal job transfers from class i into class j, and of the external Poisson arrivals. The service times and reward/cost structure in the corresponding branching bandit model follow directly from the problem definition.

Performance measures

For a given $\alpha > 0$, we define performance measure $\lambda_j^u(\alpha)$ of class-j jobs under policy u to be total expected discounted number of class-j job service completions, i.e.,

$$\lambda_j^u(\alpha) = E_u\left[\sum_{k=1}^{\nu_j} e^{-\alpha(\tau_{jk}+v_{jk})}\right], \qquad \text{for } j \in \mathcal{N},$$

exactly as we did in the discounted branching bandit problem.

The optimal value of the problem, $Z_u^{(r,h)}(\alpha)$, can be written in terms of performance measure $\lambda^u(\alpha)$ as

$$Z_u^{(r,h)}(\alpha) = \sum_{i\in\mathcal{N}} (r_i + r_{i,\alpha})\,\lambda_i^u(\alpha) - \sum_{i\in\mathcal{N}} h_i L_i(0).$$

by (4.5), where vector $r_\alpha = (r_{i,\alpha})_{i\in\mathcal{N}}$ is given by (4.6).

Performance region

By Theorem 10, we know that performance measure $\lambda^u(\alpha)$ satisfies generalized conservation laws, and that the performance region it spans is an extended contra-polymatroid.

Parameter computation. We shall show next how to compute the corresponding parameters $A_{i,\alpha}$ and $b_\alpha(S)$. In the notation introduced in Section 4.1.1, we have that transform $\Phi_i(.)$ is given by

$$\begin{aligned}
\Phi_i(\alpha, z_1, \ldots, z_n) &= E\left[e^{-\alpha v_i} z_1^{N_{i1}} \ldots z_n^{N_{in}}\right] \\
&= E\left[\left(1 - \sum_{j\in\mathcal{N}} p_{ij}(1 - z_j)\right) e^{-v_i(\alpha + \sum_{j\in\mathcal{N}} \lambda_j(1-z_j))}\right] \\
&= \left(1 - \sum_{j\in\mathcal{N}} p_{ij}(1 - z_j)\right) \Psi_i\left(\alpha + \sum_{j\in\mathcal{N}} \lambda_j(1 - z_j)\right). \qquad (4.65)
\end{aligned}$$

Therefore, by (4.22) and (4.65) we obtain that the values of $\Psi_i^S(\alpha)$, for $i \in \mathcal{N}$, satisfy the system of equations

$$\Psi_i^S(\alpha) = \left(1 - \sum_{j\in S} p_{ij}\left(1 - \Psi_j^S(\alpha)\right)\right) \Psi_i\left(\alpha + \sum_{j\in S} \lambda_j\left(1 - \Psi_j^S(\alpha)\right)\right), \qquad \text{for } i \in \mathcal{N}.$$

Proposition 6. yields closed formulae for computing matrix A_α and set function b_α. By the results in Section 4.1.1, the performance region spanned by performance vector $\lambda^u(\alpha)$ is the extended contra-polymatroid $\mathcal{P}_c(A_\alpha, b_\alpha)$, and it shown there how to apply adaptive greedy algorithm $\mathcal{AG}$ in order to compute the priority indices corresponding to the optimal scheduling policy.

Problem definition: the undiscounted case

Klimov (1974) first considered the problem of optimal control of a single-server multiclass queue with Bernoulli feedback, with the criterion of minimizing the time average holding cost per unit time. He proved that the optimal nonidling, nonpreemptive and nonanticipative policy is a fixed priority policy, and presented an algorithm for computing the priorities (starting with the lowest priority class and ending with the highest priority). Tsoucas (1991) modeled Klimov's problem as an optimization problem over an extended polymatroid using

as performance measures

$$Q_i^u = \text{time average length of queue } i \text{ under policy } u.$$

Algorithm $\mathcal{AG}$ applied to this problem is exactly Klimov's original algorithm. A disadvantage in this case is that Klimov's is a bottom-up algorithm: priorities are computed from lowest to highest priority. Also, Tsoucas does not obtain closed form formulae for the right hand sides of the extended polymatroid, so it is not possible to evaluate the performance of an optimal policy. Our approach provides a top-down algorithm, gives explicit formulae for all the parameters of the extended polymatroid and also explains the somewhat surprising property that the optimal priority rule does not depend in this case on the arrival rates. The key observation is that an optimal policy under the time average holding cost criterion also minimizes the expected total holding cost in each busy period (see Nain et al. (1989) for further discussion).

We shall model the first busy period of Klimov's problem as a branching bandit process with the undiscounted tax criterion, as considered in Section 4.1.2. Assuming that the system is stable, we apply the results of Section 4.1.2. By (4.49),

$$E\left[T_i^{S^c}\right] = E[v_i] + \sum_{j \in S^c} (p_{ij} + E[v_i]\lambda_j) E\left[T_j^{S^c}\right], \quad i \in \mathcal{N},$$

which in vector notation becomes

$$E\left[T_{S^c}^{S^c}\right] = E\left[v_{S^c}\right] + \left(P_{S^c S^c} + E\left[v_{S^c}\right]\lambda'_{S^c}\right) E\left[T_{S^c}^{S^c}\right],$$

i.e.

$$E\left[T_{S^c}^{S^c}\right] = \left(I_{S^c} - P_{S^c S^c} - E\left[v_{S^c}\right]\lambda'_{S^c}\right)^{-1} E[v_{S^c}],$$

and

$$E\left[T_S^{S^c}\right] = E[v_S] + \left(P_{S,S^c} + E\left[v_S\right]\lambda'_{S^c}\right) t_{S^c}^{S^c}.$$

Let us define

$$K_{S^c} = \frac{\det\left(I_{S^c} - P_{S^c S^c}\right)}{\det\left(I_{S^c} - P_{S^c S^c} - E\left[v_{S^c}\right]\lambda'_{S^c}\right)}.$$

The following algebraic invariance relations can be shown to hold:

$$E\left[T_{S^c}^{S^c}\right] = K_{S^c}\left(I_{S^c} - P_{S^cS^c}\right)^{-1} E\left[v_{S^c}\right],$$

and

$$E\left[T_S^{S^c}\right] = K_{S^c}\left(E\left[v_S\right] + P'_{S,S^c}\left(I_{S^c} - P_{S^cS^c}\right)^{-1} E\left[v_{S^c}\right]\right).$$

Therefore, by definition of A_i^S in (4.35), we have that $A_i^S = E\left[T_i^{S^c}\right]$, for $i \in S$, while $b(S)$ is given by (4.36). Now, we may define $\hat{A}_i^S = A_i^S/K_{S^c}$, and $\hat{b}(S) = b(S)/K_{S^c}$, thus eliminating the dependence on the arrival rates of matrix $\hat{A}$. As for the objective function, by (4.34), and the fact that

$$E\left[N_{ij}\right] = p_{ij} + \lambda_j E[v_i]$$

we obtain:

$$Z_u^h = \sum_{i\in\mathcal{N}}\left(h_i - \sum_{j\in\mathcal{N}} p_{ij}h_j\right) C_i^u - b(\mathcal{N}) \sum_{j\in\mathcal{N}} h_j\lambda_j E[v_j] - \sum_{i\in\mathcal{N}} h_i g_i,$$

where h and b are as given in Section 4.1.2. Therefore the control problem can be solved by running algorithm $\mathcal{AG}$ with input $(\hat{r}, \hat{A})$, where

$$\hat{r}_i = h_i - \sum_{j\in\mathcal{N}} p_{ij}h_j, \qquad \text{for } i \in \mathcal{N},$$

and since $(\hat{r}, \hat{A})$ does not depend on the arrival rates neither does the optimal policy. Notice that in contrast to Klimov's algorithm,with this method priorities are computed from highest to lowest. This top-down algorithm was proposed by Lai and Ying (1988) and by Nain et al. (1989), who proved its optimality using interchange arguments. Bhattacharya et al. (1991) provided a direct optimality proof. They proved that the resulting optimal index rule is also optimal among idling policies for general service time distributions, and among preemptive policies when the service time distributions are exponential. It is also easy to verify these result using our approach (in particular, the index of the idling state is 0, whereas all other indices are nonnegative).

Notice that by modeling the busy period of Klimov's problem as a branching bandit's tax problem, using performance measure W^u, we obtain exactly Klimov's algorithm.

Moreover, in the case that the arriving jobs are divided into K projects, where a class-k

project consists of jobs with classes in a finite set $\mathcal{N}_k$, jobs in $\mathcal{N}_k$ can only make transitions within $\mathcal{N}_k$, and $\mathcal{N}$ is partitioned as $\mathcal{N} = \cup_{k=1}^{K} \mathcal{N}_k$, then it is easy to see that the Index decomposition theorem 9 applies, and therefore we can decompose the problem into K smaller subproblems.

4.2.3 Optimal scheduling problems without job arrivals; deterministic scheduling

There is a batch of n jobs to be processed by a single server. Job i has a service requirement distributed as the random variable v_i, with Laplace transform Ψ_i. It is clear how to model this job scheduling process as a branching bandit process in which jobs have no descendants. Let us consider first the discounted case: For $\alpha > 0$ it is clear by definition of $A^S_{i,\alpha}$, in (4.8), that $A^S_{i,\alpha} \equiv \alpha$, for $i \in S$. Therefore the performance space of the vectors $\lambda^u(\alpha)$ studied in Section 4.1.1 is a polymatroid. Consider the discounted reward-tax problem discussed in Section 4.1.1, in which a instantaneous reward r_i is received at the completion of job i, and a holding tax αh_i is incurred for each unit of time that job i is in the system. Rewards and taxes are discounted in time with discount factor α. By (4.5) it follows that the allocation index for job i, in the problem of maximizing rewards minus taxes, is

$$\gamma_i(\alpha) = (r_i + h_i)\,\frac{\alpha\Psi_i(\alpha)}{1-\Psi_i(\alpha)}.$$

Let us consider now the problem of minimizing the total weighted expected completion time of the jobs, where a holding cost h_i is incurred per unit of time that a class i job is in the system. By definition of A^S_i in (4.35), $A^S_i \equiv 1$, for $i \in S$. Hence the performance region of the performance vectors $E[v_i]C^u$ studied in Section 4.1.2 is also the base of a polymatroid. Thus by equation (4.34) it follows that the allocation index for job i in the undiscounted tax problem is

$$\gamma_i = \frac{h_i}{E[v_i]}.$$

We thus obtain that for every nonanticipative, nonpreemptive and nonidling scheduling policy u,

$$\sum_{i \in S} E[v_i] W_i^u \geq \left(\sum_{i \in S} E[v_i] \right)^2 \qquad \text{for } S \subset \mathcal{N}, \tag{4.66}$$

and

$$\sum_{i \in \mathcal{N}} E[v_i] W_i^u = \left(\sum_{i \in \mathcal{N}} E[v_i] \right)^2,$$

with equality in (4.66) if policy u gives priority to S-jobs. Queyranne (1993) characterized this performance space in the case that the processing times v_i are deterministic.

In the case that there are precedence constraints among the jobs that form out-forests, i.e., each job can have at most one immediate predecessor, the problem can also be modeled as a branching bandit problem. The formulations developed in this chapter apply therefore to it (see also Horn (1972) and Glazebrook (1976)).

project consists of jobs with classes in a finite set $\mathcal{N}_k$, jobs in $\mathcal{N}_k$ can only make transitions within $\mathcal{N}_k$, and $\mathcal{N}$ is partitioned as $\mathcal{N} = \cup_{k=1}^{K} \mathcal{N}_k$, then it is easy to see that the Index decomposition theorem 9 applies, and therefore we can decompose the problem into K smaller subproblems.

4.2.3 Optimal scheduling problems without job arrivals; deterministic scheduling

There is a batch of n jobs to be processed by a single server. Job i has a service requirement distributed as the random variable v_i, with Laplace transform Ψ_i. It is clear how to model this job scheduling process as a branching bandit process in which jobs have no descendants. Let us consider first the discounted case: For $\alpha > 0$ it is clear by definition of $A^S_{i,\alpha}$, in (4.8), that $A^S_{i,\alpha} \equiv \alpha$, for $i \in S$. Therefore the performance space of the vectors $\lambda^u(\alpha)$ studied in Section 4.1.1 is a polymatroid. Consider the discounted reward-tax problem discussed in Section 4.1.1, in which a instantaneous reward r_i is received at the completion of job i, and a holding tax αh_i is incurred for each unit of time that job i is in the system. Rewards and taxes are discounted in time with discount factor α. By (4.5) it follows that the allocation index for job i, in the problem of maximizing rewards minus taxes, is

$$\gamma_i(\alpha) = (r_i + h_i)\,\frac{\alpha\Psi_i(\alpha)}{1-\Psi_i(\alpha)}.$$

Let us consider now the problem of minimizing the total weighted expected completion time of the jobs, where a holding cost h_i is incurred per unit of time that a class i job is in the system. By definition of A^S_i in (4.35), $A^S_i \equiv 1$, for $i \in S$. Hence the performance region of the performance vectors $E[v_i]C^u$ studied in Section 4.1.2 is also the base of a polymatroid. Thus by equation (4.34) it follows that the allocation index for job i in the undiscounted tax problem is

$$\gamma_i = \frac{h_i}{E[v_i]}.$$

We thus obtain that for every nonanticipative, nonpreemptive and nonidling scheduling policy u,

Chapter 5

The restless bandit problem

In this chapter we develop a mathematical programming approach, as outlined in Chapter 1, to a computationally intractable variation on the classical multi-armed bandit problem: the restless bandit problem.

Our contributions are the following:

1. We present a series of N linear programming relaxations for the restless bandit problem on N bandits (see next section). These relaxations capture increasingly higher order interactions among the bandits. These relaxations are increasingly stronger at the expense of higher computational times, the last one (Nth) being exact. These relaxations utilize the following projection representation idea nicely outlined in Lovász and Schrijver (1991):

   > It has been recognized recently that to represent a polyhedron as the projection of a higher-dimensional, but simpler, polyhedron, is a powerful tool in polyhedral combinatorics ... The idea is that a projection of a polytope may have more facets than the polytope itself. This remark suggests that even if P has exponentially many facets, we may be able to represent it as the projection of a polytope Q in higher (but still polynomial) dimension, having only a polynomial number of facets.

2. We propose a primal-dual heuristic that defines indices based on dual variables of the first order linear programming relaxation. Under a natural assumption, we interpret the heuristic as an fixed-priority index heuristic. We report computational results that suggest that the heuristic is exceptionally accurate. Primal-dual heuristics construct

a linear programming relaxation of the problem, compute optimal primal and dual solutions of the relaxed formulation, and then construct a feasible solution for the original problem using information contained in the optimal primal and dual solutions. They have been proven quite effective for solving hard discrete optimization problem (see for example Bertsimas and Teo (1994)).

The chapter is structured as follows: In Section 5.1 we introduce the restless bandit problem and review previous research efforts. In Section 5.2 we review and strengthen a classical result on the performance region of a Markov decision chain and use it to present a monotone sequence of linear programming relaxations for the problem, the last one being exact. In Section 5.3 we introduce a primal-dual heuristic for the restless bandit problem, based on the optimal solution to the first-order relaxation. In Section 5.4 we address the tightness of the relaxations and the performance of the heuristic via computational testing.

5.1 The restless bandit problem: description and background

The restless bandit problem is defined as follows: There is a collection of N projects. Project $n \in \mathcal{N} = \{1, \ldots, N\}$ can be in one of a finite number of states $i_n \in E_n$, for $n = 1, \ldots, N$. At each instant of discrete time $t = 0, 1, 2, \ldots$, exactly $M < N$ projects must be operated. If project n, in state i_n, is in operation, then an *active* reward $R^1_{i_n}$ is earned, and the project state changes into j_n with an active transition probability $p^1_{i_n j_n}$. If the project remains idle, then a *passive* reward $R^0_{i_n}$ is received, and the project state changes into j_n with a passive transition probability $p^0_{i_n j_n}$. Rewards are discounted in time by a discount factor $0 < \beta < 1$. Projects are to be selected for operation according to an *admissible* scheduling policy u: the decision as to which M projects to operate at any time t must be based only on information on the current states of the projects. Let $\mathcal{U}$ denote the class of admissible scheduling policies. The goal is to find an admissible scheduling policy that maximizes the expected present value of the rewards earned,

$$Z^* = \max_{u \in \mathcal{U}} E_u \left[\sum_{t=0}^{\infty} \left(R^{a_1(t)}_{i_1(t)} + \cdots + R^{a_N(t)}_{i_N(t)} \right) \beta^t \right], \tag{5.1}$$

where $i_n(t)$ and $a_n(t)$ denote the state and the action (active or passive), respectively, corresponding to project n at time t. We assume that the initial state of project n is i_n

with probability α_{i_n}, independently of all other projects.

The restless bandit problem was introduced by Whittle (1988), as an extension of the classical multi-armed bandit problem (see e.g. Gittins (1989)). The latter corresponds to the special case that exactly one project must be operated at any time (i.e., $M = 1$), and passive projects are frozen: they do not change state ($p^0_{i_n i_n} = 1$, $p^0_{i_n j_n} = 0$ for all $n \in N$ and $i_n \neq j_n$).

As already mentioned, in contrast to the classical multi-armed bandit problem, the restless bandit problem is computationally intractable. Papadimitriou and Tsitsiklis (1993) have proved that the problem is $PSPACE-hard$, even in the special case of deterministic transition rules and $M = 1$. In the multi-armed bandit case, Gittins and Jones (1974) first showed that the optimal scheduling policy is a *priority index* policy: to each project state is assigned an index, and the policy operates at each time a project with largest index. The optimal *Gittins indices* are computable in polynomial time.

The restless bandit problem provides a very flexible modeling framework and, as a result, a number of interesting practical problems can be modeled naturally as restless bandits. As an indication of its modeling power we include the following examples: Clinical trials (Whittle (1988)), Aircraft surveillance (Whittle (1988)), Worker scheduling (Whittle (1988)), Police control of drug markets, Control of a make-to-stock production facility (Veatch and Wein (1992)).

Whittle approached the restless bandit problem with dynamic programming methods. He presented a relaxed version of the problem, solvable in polynomial time. He then proposed a fixed-priority index heuristic based on the optimal solution of the relaxation. This index heuristic reduces to the Gittins index optimal policy when applied to the classical multi-armed bandit problem. A disadvantage is that Whittle's index heuristic only applies to a restricted class of restless bandits: those that satisfy a certain indexability property, which is difficult to check.

5.2 A sequence of relaxations for the restless bandit problem

In this section we first strengthen a classical result on the polyhedral characterization of the performance region for a finite discounted Markov decision chain (MDC) (see Heyman

and Sobel (1984)). Then we formulate the restless bandit problem as a linear program over a certain *restless bandit polytope*. By applying the previous extension on polyhedral representation of MDCs, we present a monotone sequence of approximations to the restless bandit polytope (each approximation is tighter than the previous one), that yields a corresponding sequence of polynomial-size linear programming relaxations for the problem. These relaxations provide a monotone sequence of polynomial-time bounds for the optimal value of the restless bandit problem.

5.2.1 Polyhedral representation of Markov decision chains

Markov decision processes provide a general framework to model stochastic optimization problems. In this section we strengthen a classical result on the polyhedral characterization of the performance region for a finite discounted Markov decision chain.

Let $E = \{1, \ldots, n\}$ be the finite state space. At state $i \in E$ there is a finite set A_i of actions available. Let us denote $\mathcal{C}$ the state-action space,

$$\mathcal{C} = \{(i, a) : i \in E, a \in A_i\}.$$

Let α_i be the probability that the initial state is i. If action $a \in A_i$ is taken in state i, then the chain moves to state j with probability p_{ij}^a. Let $0 < \beta < 1$ denote the discount factor. Let

$$I_j^a(t) = \begin{cases} 1, & \text{if action } a \text{ is taken at time } t \text{ in state } j; \\ 0, & \text{otherwise.} \end{cases}$$

An *admissible* policy is specified by a probability distribution on the actions A_i corresponding to every state i. If at state i action a is drawn from the corresponding distribution, then action a is taken. Let us denote $\mathcal{U}$ the class of all admissible policies. We call a policy *admissible* if the decision as to which action to take at any time t depends only on the current state. An *admissible* (i.e., nonanticipative) policy $u \in \mathcal{U}$ for selecting the actions generates a Markov chain. Let us associate with policy u the following performance measures:

$$x_j^a(u) = E_u \left[\sum_{t=0}^{\infty} I_j^a(t) \beta^t \right].$$

Notice that $x_j^a(u)$ is the total expected discounted time spent taking action a in state j under policy u.

We are interested in finding a complete description of the corresponding performance region $X = \{x^u, u \in \mathcal{U}\}$. Let us consider the polyhedron

$$P = \left\{ x \in \Re_+^n : \sum_{a \in A_j} x_j^a = \alpha_j + \beta \sum_{(i,a) \in C} p_{ij}^a x_i^a, \quad j \in E, \right\}.$$

Notice that by summing over all $j \in E$ we obtain that $\sum_{(i,a) \in C} x_i^a = \frac{1}{1-\beta}$ and therefore P is a bounded polyhedron.

It was first shown by d'Epenoux (1960) that, if all the initial probabilities $\alpha_j > 0$, then polytope P is a complete description of the performance region X (see also Heyman and Sobel (1984)). He also showed that polytope P always contains the performance region, i.e., $X \subseteq P$.

We strengthen next that classical result, by proving that polytope P is *always* a complete description of performance region X, even if the assumption that all initial probabilities α_j are positive is dropped.

Theorem 16 (Performance region of discounted MDCs) (a) $X = P$.
(b) *The vertices of polytope P are achievable by stationary deterministic policies.*

Proof In Heyman and Sobel [49] it is shown that $X \subseteq P$ always. We will thus prove only the other inclusion, i.e., $P \subseteq X$.

Since P is a bounded polyhedron, any point in P can be written as a convex combination of its extreme points. Therefore, it suffices to show that any extreme point of P is achievable by some stationary deterministic policy, since any point of P can be achieved using a policy that randomizes over deterministic policies that achieve the corresponding extreme points.

Let $\overline{x}$ be an extreme point of polytope P. By standard linear programming theory, $\overline{x}$ is the unique maximizer of a linear objective function. Let $\sum_{(i,a) \in C} R_i^a x_i^a$ be such an objective. Since $\overline{x}$ is an extreme point, it has at most n positive components.
Let us now partition the state space E into two subsets, E_1 and E_2, in the following way:

$$E_1 = \{j \in E: \ \overline{x}_j^a > 0 \quad \text{for some } a \in A_j\}, \text{ and } E_2 = \{j \in E: \ \overline{x}_j^a = 0 \quad \text{for all } a \in A_j\}.$$

Let $\overline{x}_{E_1} = \{\overline{x}_j^a,\ j \in E_1\}$. Consider now the following linear program:

$$
\begin{aligned}
(LP_1) \qquad Z_{E_1} \;=\; & \max \sum_{j \in E_1} \sum_{a \in A_j} R_j^a x_j^a \\
& \text{subject to} \\
& \sum_{a \in A_j} x_j^a - \beta \sum_{i \in E_1} \sum_{a \in A_i} p_{ij}^a x_i^a = \alpha_j, \quad j \in E_1, \\
& x_j^a \geq 0, \quad j \in E_1, a \in A_j.
\end{aligned}
$$

By construction, $\overline{x}_{E_1}$ is the unique optimal solution of linear program (LP_1), otherwise $\overline{x}$ would not be the unique maximizer. $\overline{x}_{E_1}$ is therefore an extreme point of (LP_1), and it has at most $|E_1|$ positive components. But by definition of E_1, it follows that $\overline{x}_{E_1}$ has exactly $|E_1|$ positive components, and for each state $j \in E_1$ there is exactly one action $\overline{a}_j \in A_j$ such that $\overline{x}_j^{\overline{a}} > 0$.

We can now define a stationary deterministic policy $\overline{u}$ that achieves $\overline{x}$: For each state $j \in E_2$ pick an arbitrary action $\overline{a}_j \in A_j$. Now, policy $\overline{u}$ deterministically takes action $\overline{a}_j$ in state j. Clearly, this policy achieves the vector $\overline{x}$, which completes the proof of (a) and (b). □

5.2.2 The restless bandit polytope

In order to formulate the restless bandit problem as a linear program we define decision variables and characterize the corresponding feasible region. We introduce the indicators

$$
I_{i_n}^1(t) = \begin{cases} 1, & \text{if project } n \text{ is in state } i_n \text{ and active at time } t; \\ 0, & \text{otherwise,} \end{cases}
$$

and

$$
I_{i_n}^0(t) = \begin{cases} 1, & \text{if project } n \text{ is in state } i_n \text{ and passive at time } t; \\ 0, & \text{otherwise.} \end{cases}
$$

Given an admissible scheduling policy $u \in \mathcal{U}$ let us define performance measures

$$
x_{i_n}^1(u) = E_u \left[\sum_{t=0}^{\infty} I_{i_n}^1(t) \beta^t \right],
$$

and

$$
x_{i_n}^0(u) = E_u \left[\sum_{t=0}^{\infty} I_{i_n}^0(t) \beta^t \right].
$$

Notice that performance measure $x_{i_n}^1(u)$ (resp. $x_{i_n}^0(u)$) represents the total expected discounted time that project n is in state i_n and active (resp. passive) under scheduling policy u. Let us denote P the corresponding performance region,

$$P = \left\{ \mathbf{x} = \left(x_{i_n}^{a_n}(u) \right)_{i_n \in E_n, a_n \in \{0,1\}, n \in \mathcal{N}} \mid \quad u \in \mathcal{U} \right\}.$$

It is clear that performance region P is a polytope. This follows from the fact that the restless bandit problem can be viewed as a discounted MDC (in the state space $E_1 \times \cdots \times E_N$), and the performance region of the latter is a polytope from Theorem 16.

We will refer to P in what follows as the *restless bandit polytope*. The restless bandit problem can thus be formulated as the linear program

$$(LP) \qquad Z^* \;=\; \max_{\mathbf{x} \in P} \sum_{n \in \mathcal{N}} \sum_{i_n \in E_n} \sum_{a_n \in \{0,1\}} R_{i_n}^{a_n} x_{i_n}^{a_n}.$$

In chapter 4 we showed how to characterize polyhedron P in the special case of the classical multi-armed bandit problem as a polytope with special structure (an *extended polymatroid*). This characterization leads to strong structural properties of the optimal scheduling policy (Gittins priority index policy). For general restless bandits, however, it is highly unlikely that a complete description of polytope P can be found, since as mentioned above the problem is $PSPACE - hard$.

Our approach will be to construct approximations of polytope P that yield polynomial-size relaxations of the linear program (LP). We will represent these approximations $\hat{P} \supseteq P$ as projections of higher dimensional polytopes $\hat{Q}$. An advantage of pursuing this *projection representation* approach is that we will be able to represent approximations $\hat{P}$ of P with exponentially many facets as projections of polytopes $\hat{Q}$ with a polynomial number of facets, thus providing polynomial-time bounds on the optimal value Z^*. The approximations we develop are based on exploiting the special structure of the restless bandit problem as an MDC, and on applying Theorem 16.

5.2.3 A first-order linear programming relaxation

Whittle (1988) introduced a relaxed version of the restless bandit problem, solvable in polynomial time. The original requirement that exactly M projects must be active at any time is relaxed to an averaged version: the total expected discounted number of active

projects over the infinite horizon must be $M/(1-\beta)$. Whittle showed that this relaxed version can be interpreted as the problem of controlling optimally N independent MDCs (one corresponding to each project), subject to one binding constraint on the discounted average number of active projects. In this section we formulate Whittle's relaxation as a polynomial-size linear program.

The restless bandit problem induces a *first-order MDC* over each project n in a natural way: The state space of this MDC is E_n, its action space is $\mathcal{A}^1 = \{0,1\}$, and the reward received when action a_n is taken in state i_n is $R_{i_n}^{a_n}$. Rewards are discounted in time by discount factor β. The transition probability from state i_n into state j_n, given action a_n, is $p_{i_n j_n}^{a_n}$. The initial state is i_n with probability α_{i_n}.
Let

$$Q_n^1 = \left\{ \mathbf{x}_n = \left(x_{i_n}^{a_n}(u) \right)_{i_n \in E_n, a_n \in \mathcal{A}^1} \mid u \in \mathcal{U} \right\}.$$

From a polyhedral point of view, Q_n^1 is the projection of the restless bandit polytope P over the space of the variables $x_{i_n}^{a_n}$ for project n. From a probabilistic point of view, Q_n^1 is the performance region of the first-order MDC corresponding to project n. In order to see this, we observe that as policies u for the restless bandit problem range over all admissible policies $\mathcal{U}$, they induce policies u_n for the first-order MDC corresponding to project n that range over all admissible policies for that MDC. Applying Theorem 16 we obtain:

Proposition 11 *A complete polyhedral description of Q_n^1 is given by*

$$Q_n^1 = \left\{ \mathbf{x}_n \geq \mathbf{0} \mid x_{j_n}^0 + x_{j_n}^1 = \alpha_{j_n} + \beta \sum_{i_n \in E_n} \sum_{a_n \in \{0,1\}} p_{i_n j_n}^{a_n} x_{i_n}^{a_n}, \quad j_n \in E_n \right\}. \tag{5.2}$$

Remark: A consequence of Proposition 11 is that the general restless bandit problem, with active and passive rewards, can be reduced to the case with active rewards only. This follows since by (5.2) the passive performance vector $\mathbf{x}_n^0(u)$ is a linear transformation of the active one, $\mathbf{x}_n^1(u)$.

Now, Whittle's condition on the discounted average number of active projects can be written as

$$\sum_{n \in \mathcal{N}} \sum_{i_n \in E_n} x_{i_n}^1(u) \;=\; \sum_{t=0}^{\infty} E_u \left[\sum_{n \in \mathcal{N}} \sum_{i_n \in E_n} I_{i_n}^1(t) \right] \beta^t$$

$$= \sum_{t=0}^{\infty} M\beta^t$$
$$= \frac{M}{1-\beta} \tag{5.3}$$

Therefore, the first-order relaxation can be formulated as the linear program

$$(LP^1) \quad Z^1 = \max \sum_{n\in\mathcal{N}} \sum_{i_n\in E_n} \sum_{a_n\in\{0,1\}} R_{i_n}^{a_n} x_{i_n}^{a_n}$$
$$\text{subject to}$$
$$\mathbf{x}_n \in Q_n^1, \qquad n \in \mathcal{N},$$
$$\sum_{n\in\mathcal{N}} \sum_{i_n\in E_n} x_{i_n}^1 = \frac{M}{1-\beta}.$$

We will refer to the feasible region of linear program (LP^1) as the first-order approximation of the restless bandit polytope, and will denote it as P^1. Notice that linear program (LP^1) has $O(N|E_{\max}|)$ variables and constraints, where $|E_{\max}| = \max_{n\in\mathcal{N}} |E_n|$.

5.2.4 A second-order linear programming relaxation

In this section we present a second-order polynomial-size linear programming relaxation for the restless bandit problem, which is represented as the projection of a higher-dimensional polytope (introducing new variables). The new decision variables we introduce correspond to second-order performance measures for the restless bandit problem, associated with pairs of projects. Given a pair of projects, $n_1 < n_2$, the valid actions that can be taken over each pair of states, $(i_1, i_2) \in E_{n_1} \times E_{n_2}$, range over

$$\mathcal{A}^2 = \left\{ (a_1, a_2) \in \{0,1\}^2 \mid |a_1 + a_2 \leq M \right\}.$$

Given an admissible scheduling policy u, let us define the second-order performance measures by

$$x_{i_1 i_2}^{a_1 a_2}(u) = E_u \left[\sum_{t=0}^{\infty} I_{i_1}^{a_1}(t) I_{i_2}^{a_2}(t) \beta^t \right].$$

Similarly as in the first-order case, the restless bandit problem induces a *second-order MDC* over each pair of projects $n_1 < n_2$, in a natural way: The state space of the MDC is $E_{n_1} \times E_{n_2}$, the action space is $\mathcal{A}^2$, and the reward corresponding to state (i_{n_1}, i_{n_2}) and

action (a_{n_1}, a_{n_2}) is $R_{i_{n_1}}^{a_{n_1}} + R_{i_{n_2}}^{a_{n_2}}$. Rewards are discounted in time by discount factor β. The transition probability from state (i_{n_1}, i_{n_2}) into state (j_{n_1}, j_{n_2}), given action (a_{n_1}, a_{n_2}), is $p_{i_{n_1} j_{n_1}}^{a_{n_1}} p_{i_{n_2} j_{n_2}}^{a_{n_2}}$. The initial state is (i_{n_1}, i_{n_2}) with probability $\alpha_{i_{n_1}} \alpha_{i_{n_2}}$. Let

$$Q_{n_1,n_2}^2 = \left\{ \mathbf{x}_{n_1,n_2} = \left(x_{i_1 i_2}^{a_1 a_2}(u) \right)_{i_1 \in E_{n_1}, i_2 \in E_{n_2}, (a_1,a_2) \in \mathcal{A}^2} \mid | u \in \mathcal{U} \right\},$$

be the projection of P over the space of the variables $(x_{i_1 i_2}^{a_1 a_2})_{i_1 \in E_{n_1}, i_2 \in E_{n_2}, (a_1,a_2) \in \mathcal{A}^2}$. An admissible scheduling policy u for the restless bandit problem induces an admissible scheduling policy u_{n_1,n_2} for the MDC corresponding to projects n_1 and n_2. It is easy to see that as u ranges over all admissible scheduling policies for the restless bandit problem, the corresponding induced policy u_{n_1,n_2} ranges over all admissible policies for the MDC. Therefore, the projection Q_{n_1,n_2}^2 is the performance region of the discounted MDC corresponding to the pair of projects (n_1, n_2), $(n_1 < n_2)$. Therefore, from Theorem 16 we obtain:

Proposition 12 *A complete polyhedral description of $Q_{n_1\, n_2}^2$ is given by*

$$\sum_{(a_1,a_2) \in \mathcal{A}^2} x_{j_1 j_2}^{a_1 a_2} = \alpha_{j_1} \alpha_{j_2} + \beta \sum_{\substack{i_1 \in E_{n_1}, i_2 \in E_{n_2} \\ (a_1,a_2) \in \mathcal{A}^2}} p_{i_1 j_1}^{a_1} p_{i_2 j_2}^{a_2} x_{i_1 i_2}^{a_1 a_2}, \quad (j_1, j_2) \in E_1 \times E_2, \tag{5.4}$$

$$x_{i_1 i_2}^{a_1 a_2} \geq 0, \qquad (i_1, i_2) \in E_{n_1} \times E_{n_2},\ (a_1, a_2) \in \mathcal{A}^2. \tag{5.5}$$

We can show some other second-order conservation laws to hold, based on combinatorial arguments. For all admissible scheduling policies u, we have, if $N \geq M + 2$,

$$\sum_{1 \leq n_1 < n_2 \leq N} \sum_{i_1 \in E_{n_1}} \sum_{i_2 \in E_{n_2}} x_{i_1 i_2}^{00}(u) = \frac{\binom{N-M}{2}}{1-\beta}, \tag{5.6}$$

since the $N - M$ passive projects required at any time correspond to $\binom{N-M}{2}$ passive-passive project pairs. Moreover,

$$\sum_{1 \leq n_1 < n_2 \leq N} \sum_{i_1 \in E_{n_1}} \sum_{i_2 \in E_{n_2}} (x_{i_1 i_2}^{10}(u) + x_{i_1 i_2}^{01}(u)) = \frac{M(N-M)}{1-\beta}, \tag{5.7}$$

since at any time the M active and $N - M$ passive required projects give rise to $M(N - M)$ active-passive project pairs.

Furthermore, in the case that $M \geq 2$, we have

$$\sum_{1 \leq n_1 < n_2 \leq N} \sum_{i_1 \in E_{n_1}} \sum_{i_2 \in E_{n_2}} x^{11}_{i_1 i_2}(u) = \frac{\binom{M}{2}}{1-\beta}, \tag{5.8}$$

since at any time the $M \geq 2$ active projects give rise to $\binom{M}{2}$ active-active project pairs.

In order to lift the first-order approximation to the restless bandit polytope into a higher dimensional space with variables $x^{a_1 a_2}_{i_1 i_2}$ we need to relate the first and second-order performance measures. It is easy to see that, for any admissible policy u,

$$x^{a_1}_{i_1}(u) = \sum_{\substack{i_2 \in E_{n_2} \\ a_2:(a_1,a_2)\in\mathcal{A}^2}} x^{a_1 a_2}_{i_1 i_2}(u), \quad i_1 \in E_{n_1}, a_1 \in \{0,1\}, 1 \leq n_1 < n_2 \leq N, \tag{5.9}$$

and

$$x^{a_2}_{i_2}(u) = \sum_{\substack{i_1 \in E_{n_1} \\ a_1:(a_1,a_2)\in\mathcal{A}^2}} x^{a_1 a_2}_{i_1 i_2}(u), \quad i_2 \in E_{n_2}, a_2 \in \{0,1\}, 1 \leq n_1 < n_2 \leq N. \tag{5.10}$$

We define now the second-order relaxation, based on the above identities, as the linear program

$$\begin{aligned}
(LP^2) \quad Z^2 \;=\; & \max \sum_{n \in \mathcal{N}} \sum_{i_n \in E_n} \sum_{a_n \in \{0,1\}} R^{a_n}_{i_n} x^{a_n}_{i_n} \\
& \text{subject to} \\
& \mathbf{x}_{n_1,n_2} \in Q^2_{n_1,n_2}, \quad 1 \leq n_1 < n_2 \leq N, \\
& \sum_{1 \leq n_1 < n_2 \leq N} \sum_{i_1 \in E_{n_1}} \sum_{i_2 \in E_{n_2}} x^{00}_{i_1 i_2} = \frac{\binom{N-M}{2}}{1-\beta}, \\
& \sum_{1 \leq n_1 < n_2 \leq N} \sum_{i_1 \in E_{n_1}} \sum_{i_2 \in E_{n_2}} (x^{10}_{i_1 i_2} + x^{01}_{i_1 i_2}) = \frac{M(N-M)}{1-\beta}, \\
& \sum_{1 \leq n_1 < n_2 \leq N} \sum_{i_1 \in E_{n_1}} \sum_{i_2 \in E_{n_2}} x^{11}_{i_1 i_2} = \frac{\binom{M}{2}}{1-\beta}, \\
& x^{a_1}_{i_1} = \sum_{i_2 \in E_{n_2}} \sum_{a_2:(a_1,a_2)\in\mathcal{A}^2} x^{a_1 a_2}_{i_1 i_2}, \quad i_1 \in E_{n_1}, a_1 \in \{0,1\}, 1 \leq n_1 < n_2 \leq N,
\end{aligned}$$

$$x_{i_2}^{a_2} = \sum_{i_1 \in E_{n_1}} \sum_{a_1:(a_1,a_2)\in\mathcal{A}^2} x_{i_1 i_2}^{a_1 a_2}, \quad i_2 \in E_{n_2}, a_2 \in \{0,1\}, 1 \leq n_1 < n_2 \leq N,$$

$$\sum_{n\in\mathcal{N}} \sum_{i_n \in E_n} \sum_{a_n \in \{0,1\}} x_{i_n}^1 = \frac{M}{1-\beta},$$

$$x_i^a \geq 0.$$

We define the second-order approximation to the restless bandit polytope P as the projection of the feasible region of linear program (LP^2) into the space of the first-order variables, x_i^a, and will denote it as P^2.

Notice that the second-order relaxation (LP^2) has $O(N^2|E_{\max}|^2)$ variables and constraints, (recall $|E_{\max}| = \max_{n\in\mathcal{N}} |E_n|$.)

5.2.5 A kth-order linear programming relaxation

In this section we generalize the results of the previous sections to present a kth-order linear programming relaxation for the restless bandit problem, corresponding to a kth-order approximation for the restless bandit polytope, for any $k = 1, \ldots, N$. This kth-order approximation corresponds again to lifting the first-order approximation polytope into a higher dimensional space, and then projecting back into the original first-order space.

In the kth-order case, we introduce new decision variables corresponding to performance measures associated with kth-order project interactions. For each k-tuple of projects $1 \leq n_1 < \cdots < n_k \leq N$, the admissible actions that can be taken at a corresponding k-tuple of states $(i_1, \ldots, i_k)$ range over

$$\mathcal{A}^k = \left\{ (a_1, \ldots, a_k) \in \{0,1\}^k \mid |a_1 + \cdots + a_k \leq M \right\}.$$

Given an admissible scheduling policy u for the restless bandit problem, we define kth-order performance measures, for each k-tuple $1 \leq n_1 < \cdots < n_k \leq N$ of projects, by

$$x_{j_1 \ldots j_k}^{a_1 \ldots a_k}(u) = E_u \left[\sum_{t=0}^{\infty} I_{j_1}^{a_1}(t) \cdots I_{j_k}^{a_k}(t) \beta^t \right], \quad j_1 \in E_{n_1}, \ldots, j_k \in E_{n_k}. \tag{5.11}$$

Analogously as in the first and second-order cases, the restless bandit problem induces a *kth-order MDC* over each k-tuple of projects $n_1 < \cdots < n_k$ in a natural way: The state space of the MDC is $E_{n_1} \times \cdots \times E_{n_k}$, the action space is $\mathcal{A}^k$, and the reward corresponding

to state $(i_{n_1},\ldots,i_{n_k})$ and action $(a_{n_1},\ldots,a_{n_k})$ is $R_{i_{n_1}}^{a_{n_1}}+\cdots+R_{i_{n_k}}^{a_{n_k}}$. Rewards are discounted in time by discount factor β. The transition probability from state $(i_{n_1},\ldots,i_{n_k})$ into state $(j_{n_1},\ldots,j_{n_k})$, given action $(a_{n_1},\ldots,a_{n_k})$, is $p_{i_{n_1}j_{n_1}}^{a_{n_1}}\cdots p_{i_{n_k}j_{n_k}}^{a_{n_k}}$. The initial state is $(i_{n_1},\ldots,i_{n_k})$ with probability $\alpha_{i_{n_1}}\cdots\alpha_{i_{n_k}}$. Introducing the projection

$$Q_{n_1\ldots n_k}^k=\left\{\mathbf{x}_{n_1\ldots n_k}=\left(x_{i_1\ldots i_k}^{a_1\ldots a_k}(u)\right)_{i_1\in E_{n_1},\ldots,i_k\in E_{n_k},(a_1,\ldots,a_k)\in\mathcal{A}^k}\mid u\in\mathcal{U}\right\}.$$

and arguing as before, we conclude that the projection $Q_{n_1\ldots n_k}^k$ is the performance region of the discounted MDC corresponding to the k-tuple of projects $n_1<\cdots<n_k$.

Proposition 13 *A complete polyhedral description of $Q_{n_1\ldots n_k}^k$ is given by the following system of equations: For $j_1\in E_{n_1},\ldots,j_k\in E_{n_k}$,*

$$\sum_{(a_1,\ldots,a_k)\in\mathcal{A}^k}x_{j_1\ldots j_k}^{a_1\ldots a_k}=\alpha_{j_1}\cdots\alpha_{j_k}+\beta\sum_{\substack{i_1\in E_{n_1},\ldots,i_k\in E_{n_k}\\(a_1,\ldots,a_k)\in\mathcal{A}^k}}p_{i_1j_1}^{a_1}\cdots p_{i_kj_k}^{a_k}\,x_{i_1\ldots i_k}^{a_1\ldots a_k},\tag{5.12}$$

and

$$x_{i_1\ldots i_k}^{a_1\ldots a_k}\geq 0,\quad i_1\in E_{n_1},\ldots,i_k\in E_{n_k},(a_1,\ldots a_k)\in\mathcal{A}^k.\tag{5.13}$$

Similarly as in the second-order case, we show that some additional kth-order conservation laws hold, by using combinatorial arguments. If u is an admissible scheduling policy, then

$$\sum_{1\leq n_1<\cdots<n_k\leq N}\ \sum_{i_1\in E_{n_1},\ldots,i_k\in E_{n_k}}\ \sum_{\substack{(a_1,\ldots,a_k)\in\mathcal{A}^k:\\a_1+\cdots+a_k=r}}x_{i_1\ldots i_k}^{a_1\ldots a_k}(u)=\frac{\binom{M}{r}\binom{N-M}{k-r}}{1-\beta},\tag{5.14}$$

for

$$\max(0,k-(N-M))\leq r\leq\min(k,M).\tag{5.15}$$

Conservation law (5.14) follows since at each time the number of k-tuples of projects that contain exactly r active projects is $\binom{M}{r}\binom{N-M}{k-r}$, for r in the range given by (5.15).

In order to lift the first-order approximation to the restless bandit polytope into a higher dimensional space with variables $x_{i_1\ldots i_k}^{a_1\ldots a_k}$ we need to relate the first and kth-order

performance measures. It is easy to see that if u is an admissible scheduling policy, then for $n \in \mathcal{N}$, $i \in E_n$, $a \in \{0,1\}$, $1 \leq r \leq k$, $n_r = n$, and $n_1 < \ldots < n_k$, we have

$$x_i^a(u) = \sum_{\substack{(a_1,\ldots,a_k)\in\mathcal{A}^k \\ i_1\in E_{n_1},\ldots,i_k\in E_{n_k}\,: \\ i_r=i,a_r=a}} x_{i_1\ldots i_k}^{a_1\ldots a_k}(u). \tag{5.16}$$

We now define the kth-order relaxation of the restless bandit problem as the linear program

$$\begin{aligned}
(LP^k) \quad Z^k \;=\; & \max \sum_{n\in\mathcal{N}} \sum_{i_n\in E_n} \sum_{a_n\in\{0,1\}} R_{i_n}^{a_n} x_{i_n}^{a_n} \\
& \text{subject to} \\
& \mathbf{x}_{n_1\ldots n_k} \in Q_{n_1\ldots n_k}^k, \\
& \sum_{1\leq n_1<\cdots<n_k\leq N} \sum_{i_1\in E_{n_1},\ldots,i_k\in E_{n_k}} \sum_{\substack{(a_1,\ldots,a_k)\in\mathcal{A}^k\,: \\ a_1+\cdots+a_k=r}} x_{i_1\ldots i_k}^{a_1\ldots a_k} = \frac{\binom{M}{r}\binom{N-M}{k-r}}{1-\beta}, \\
& \max(0, k-(N-M)) \leq r \leq \min(k, M), \\
& x_i^a = \sum_{\substack{(a_1,\ldots,a_k)\in\mathcal{A}^k \\ i_1\in E_{n_1},\ldots,i_k\in E_{n_k}\,: \\ i_r=i,a_r=a}} x_{i_1\ldots i_k}^{a_1\ldots a_k}, \\
& x_i^a \geq 0.
\end{aligned}$$

We define the kth-order approximation to the restless bandit polytope P as the projection of the feasible region of linear program (LP^k) into the space of the first-order variables, x_i^a, and denote it P^k. It is easy to see that the sequence of approximations is monotone, in the sense that

$$P^1 \supseteq P^2 \supseteq \cdots \supseteq P^N = P.$$

Notice that the kth-order relaxation (LP^k) has $O(N^k|E_{\max}|^k)$ variables and constraints, for k fixed. Therefore, the kth-order relaxation has polynomial size, for k fixed.

The last relaxation of the sequence, (LP^N), is exact (i.e., $Z^N = Z^*$), since it corresponds to the linear programming formulation of the restless bandit problem modeled as a MDC in the standard way.

5.3 A primal-dual heuristic for the restless bandit problem

In this section we present a heuristic for the restless bandit problem, which uses information contained in an optimal primal and dual solution of the first-order relaxation, (LP^1). Under some mixing assumptions on the active and passive transition probabilities, we can interpret the primal-dual heuristic as an index heuristic. The dual of the linear program (LP^1) is

$$
\begin{aligned}
(D^1) \qquad Z^1 \;=\; & \min \sum_{n\in\mathcal{N}} \sum_{j_n\in E_n} \alpha_{j_n}\lambda_{j_n} + \frac{M}{1-\beta}\lambda \\
& \text{subject to} \\
& \lambda_{i_n} - \beta \sum_{j_n\in E_n} p^0_{i_n j_n}\lambda_{j_n} \geq R^0_{i_n}, \quad i_n \in E_n, \quad n\in\mathcal{N}, \\
& \lambda_{i_n} - \beta \sum_{j_n\in E_n} p^1_{i_n j_n}\lambda_{j_n} + \lambda \geq R^1_{i_n}, \quad i_n \in E_n, \quad n\in\mathcal{N}, \\
& \lambda \geq 0. \qquad (5.17)
\end{aligned}
$$

Let $\{\overline{x}^{a_n}_{i_n}\}$, $\{\overline{\lambda}_{i_n}, \overline{\lambda}\}$, $i_n \in E_n$, $n \in \mathcal{N}$ be an optimal primal and dual solution to the first-order relaxation (LP^1) and its dual (D^1). Let $\{\overline{\gamma}^{a_n}_{i_n}\}$ be the corresponding optimal reduced cost coefficients, i.e.,

$$\overline{\gamma}^0_{i_n} = \overline{\lambda}_{i_n} - \beta \sum_{j_n\in E_n} p^0_{i_n j_n}\overline{\lambda}_{j_n} - R^0_{i_n},$$

$$\overline{\gamma}^1_{i_n} = \overline{\lambda}_{i_n} - \beta \sum_{j_n\in E_n} p^1_{i_n j_n}\overline{\lambda}_{j_n} + \overline{\lambda} - R^1_{i_n},$$

which are nonnegative. It is well known (cf. Murty (1983), pp. 64-65), that the optimal reduced costs have the following interpretation:

$\overline{\gamma}^1_{i_n}$ is the *rate of decrease* in the objective-value of linear program (LP^1) *per unit increase* in the value of the variable $x^1_{i_n}$.

$\overline{\gamma}^0_{i_n}$ is the rate of decrease in the objective-value of linear program (LP^1) per unit increase in the value of the variable $x^0_{i_n}$.

The proposed heuristic takes as input the vector of current states of the projects, $(i_1, \ldots, i_N)$, an optimal primal solution to (LP^1), $\{\overline{x}^{a_n}_{j_n}\}$, and the corresponding optimal reduced costs, $\{\overline{\gamma}^{a_n}_{j_n}\}$, and produces as output a vector with the actions to take on each project, $(a^*(i_1), \ldots, a^*(i_N))$. An informal description of the heuristic, with the motivation that inspired it, is as follows:

The heuristic is structured in a primal and a dual stage. In the primal stage, projects n whose corresponding active primal variable $\overline{x}^1_{i_n}$ is strictly positive are considered as candidates for active selection. The intuition is that we give preference for active selection to projects with positive $\overline{x}^1_{i_n}$ with respect to those with $\overline{x}^1_{i_n} = 0$, which seems natural given the interpretation of performance measure $x^1_{i_n}(\cdot)$ as the total expected discounted time spent selecting project n in state i_n as active. Let p represent the number of such projects. In the case that $p = M$, then all p candidate projects are set active and the heuristic stops. If $p < M$, then all p candidate projects are set active and the heuristic proceeds to the dual stage that selects the remaining $M - p$ projects. If $p > M$ none of them is set active at this stage and the heuristic proceeds to the dual stage that finalizes the selection.

In the dual stage, in the case that $p < M$, then $M - p$ additional projects, each with current active primal variable zero ($\overline{x}^1_{i_n} = 0$), must be selected for active operation among the $N - p$ projects, whose actions have not yet been fixed. As a heuristic index of the undesirability of setting project n in state i_n active, we take the active reduced cost $\overline{\gamma}^1_{i_n}$. This choice is motivated by the interpretation of $\overline{\gamma}^1_{i_n}$ stated above: the larger the *active index* $\gamma^1_{i_n}$ is, the larger is the rate of decrease of the objective-value of (LP^1) per unit increase in the active variable $x^1_{i_n}$. Therefore, in the heuristic we select for active operation the $M - p$ additional projects with smallest active reduced costs.

In the case that $p > M$, then M projects must be selected for active operation, among the p projects with $\overline{x}^1_{i_n} > 0$. Recall that by complementary slackness, $\overline{\gamma}^1_{i_n} = 0$ if $\overline{x}^1_{i_n} > 0$. As a heuristic index of the desirability of setting project n in state i_n active we take the passive reduced cost $\overline{\gamma}^0_{i_n}$. The motivation is given by the interpretation of $\overline{\gamma}^0_{i_n}$ stated above: the larger the *passive index* $\gamma^0_{i_n}$ is, the larger is the rate of decrease in the objective-value of (LP^1) per unit increase in the value of the passive variable $x^0_{i_n}$. Therefore, in the heuristic we select for active operation the M projects with largest passive reduced costs. The heuristic is described formally in 5.1.

An index interpretation of the primal-dual heuristic

Input:

- $(i_1, \ldots, i_N)$ { *current states of the N projects* }
- $\{\overline{x}_{j_n}^{a_n}\}$ { *optimal primal solution to first-order relaxation* (LP^1) }
- $\{\overline{\gamma}_{j_n}^{a_n}\}$ { *optimal reduced costs for first-order relaxation* (LP^1) }

Output:

- $(a^*(i_1), \ldots, a^*(i_N))$ { *actions to take at the projects* }

{ *Initialization:* }
set $S := \emptyset$; {*S: set of projects whose actions have been set*}

set $a^*(i_n) := 0$, for $n \in \mathcal{N}$; {*actions are initialized as passive*}

{ *Primal Stage:* }

set $p := |\{\overline{x}_{i_n}^1 : \overline{x}_{i_n}^1 > 0, n \in \mathcal{N}\}|$ { *p: number of projects with positive active primals* }

if $p \leq M$ **then** { *set active the projects with positive active primals, if no more than M* }
  for $n \in \mathcal{N}$ **do**
    if $\overline{x}_{i_n}^1 > 0$ **then**
    begin
    set $a^*(i_n) := 1$;
    set $S := S \cup \{n\}$
    end

{ *Dual Stage:* }

if $p < M$ **then** { *set active the* $M - p$ *additional projects with smallest active reduced costs* }
  until $|S| = M$ **do**
  begin
  select $\overline{n} \in \text{argmin}\{\overline{\gamma}_{i_n}^1 : n \in \mathcal{N} \setminus S\}$
  set $a^*(i_{\overline{n}}) := 1$;
  set $S := S \cup \{\overline{n}\}$
  end

if $p > M$ **then** { *set active the M projects with largest passive reduced costs* }
  until $|S| = M$ **do**
  begin
  select $\overline{n} \in \text{argmax}\{\overline{\gamma}_{i_n}^0 : n \in \mathcal{N} \setminus S\}$
  set $a^*(i_{\overline{n}}) := 1$;
  set $S := S \cup \{\overline{n}\}$
  end

Table 5.1: Primal-dual heuristic for the restless bandit problem.

We next observe that under natural mixing conditions, the primal-dual heuristic reduces to **an indexing rule**. For each project $n \in \mathcal{N}$ we consider a directed graph that is defined from the passive and active transition probabilities respectively as follows: $G_n = (E_n, A_n)$, where $A_n = \{(i_n, j_n) | \; p^0_{i_n j_n} > 0$, and $p^1_{i_n j_n} > 0 \; i_n, j_n \in E_n\}$. We shall assume that ,for every $n \in \mathcal{N}$, at least one of the following two conditions is satisfied:

a) $\alpha_{j_n} > 0$ for all $j_n \in E_n$; b) the directed graph G_n is connected.

Given that the polytope P^1 has independent constraints for every $n \in \mathcal{N}$ and only one global constraint, elementary linear programming theory establishes that

Proposition 14 *Under assumption A, every optimal extreme point $\overline{x}$ solution of the polytope P^1 has the following properties: a) There is at most one project k and at most one state $i_k \in E_k$, for which $\overline{x}^1_{i_k} > 0$ and $\overline{x}^0_{i_k} > 0$.*

b) For all other projects n and all other states either $\overline{x}^1_{i_n} > 0$ or $\overline{x}^0_{i_n} > 0$.

Therefore, starting with an optimal extreme point solution $\overline{x}$ and a complementary dual optimal solution, with corresponding reduced costs $\overline{\gamma}$, let us consider the following index rule:

Index heuristic:

1. Given the current states $(i_1, \ldots, i_N)$ of the N projects, compute the indices

$$\delta_{i_n} = \overline{\gamma}^1_{i_n} - \overline{\gamma}^0_{i_n}.$$

2. Set active the projects that have the M smallest indices. In case of ties, set active projects with $\overline{x}^1_{i_n} > 0$.

We next remark that under Assumption A, the primal-dual and the index heuristics are identical. In order to see this we consider first the case $p \leq M$. The primal-dual heuristic, would set active first the projects that have $\overline{x}^1_{i_n} > 0$. From complementarity, these projects have $\overline{\gamma}^1_{i_n} = 0$ and therefore, $\delta_{i_n} \leq 0$. Then, the primal-dual heuristic sets active the remaining $M - p$ projects with the smallest $\overline{\gamma}^1_{i_n}$. Since for these projects $\overline{x}^1_{i_n} = 0$ and therefore, $\overline{x}^0_{i_n} > 0$, i.e., $\overline{\gamma}^0_{i_n} = 0$, we obtain that $\delta_{i_n} = \overline{\gamma}^1_{i_n} \geq 0$. Therefore, the choices of the two heuristics are indeed identical.

If $p > M$, the primal-dual heuristic sets active the projects that have the largest values of $\overline{\gamma}^0_{i_n}$. For these projects $\overline{\gamma}^1_{i_n} = 0$, and therefore, $\delta_{i_n} = -\overline{\gamma}^0_{i_n} \leq 0$. Since the remaining

projects have $\delta_{i_n} = \overline{\gamma}^1_{i_n} \geq 0$, the choices of the two heuristics are identical in this case as well.

In contrast with the Gittins indices for usual bandits, notice that the indices δ_{i_n} for a particular project depend on characteristics of all other projects.

5.4 Computational experiments

In this section we address the tightness of the relaxations and the performance of the primal-dual heuristic introduced previously.

In order to address the tightness of the relaxations and the heuristic for restless bandit problems we performed a series of computational experiments. For each test problem we computed the following measures:

Z_{Greedy}: Estimated (through simulation) expected value of the greedy heuristic (at each time M projects with largest active reward are operated). We simulate a run using the heuristic policy and we obtain a value for the reward for the particular run. In order to obtain the value for a particular run, we truncated the infinite summation in (5.1) ignoring terms after time t, such that $\beta^t > 10^{-10}$. Even if we used a smaller tolerance, the results did not change. The stopping criterion for the simulation was that the difference between the average from the first $l+1$ runs and the average from the first l runs is less than 10^{-5} (using a smaller tolerance, did not change the results in this case as well).

Z_{PDH}: Estimated expected value of primal-dual heuristic. The estimation was achieved through simulation as before.

Z^*: Optimal value, which is equal to Z^N (due to the size of the formulation, this value was calculated only for small instances).

Z^2: Optimal value of the second-order relaxation (LP^2).

Z^1: Optimal value of the first-order relaxation (LP^1).

The heuristics and the simulation experiments were implemented in C. The linear programming formulations were implemented using GAMS and solved using CPLEX. All the

experiments were performed in a SUN 10 workstation. In order to test the proposed approach we generated the following 7 problem instances.

Problems 1 and 2 involve 10 projects with 7 states each, with $M = 1$, and their data (the reward vectors and the passive and active transition probabilities) was randomly generated. For these problems we were not able to compute the optimal solution because of the large size of the instance. Since these instances were randomly generated we expected that the greedy heuristic would perform very close to the optimal solution. To test the last statement we generated Problem 3 that has 5 projects with 3 states each, for which the data was also randomly generated and $M = 1$.

Problem 4 has 5 projects with 3 states each, with $M = 1$. The data was designed so that the greedy algorithm would not perform optimally. Problems 5 through 7 have the same data as problem 4, except that the number of active projects ranges from $M = 2$ through $M = 4$, respectively. The data sets are available upon request from the authors.

In Table 5.2 we report the results of our experiments for various values of the discount factor β. Some observations on the results, shown in Table 5.2, are:

1. The primal-dual heuristic performed exceptionally well. It was essentially optimal in Problems 3-7 and it was slightly better than the greedy heuristic in Problems 1 and 2. Given that we expect that the greedy heuristic is near optimal for randomly generated instances (as a verification Problem 3 had also randomly generated data and the greedy heuristic was extremely close to the optimal solution), we believe that the heuristic is extremely close to the optimal solution for Problems 1 and 2 as well. For this reason, we did not experiment with other heuristics, as we feel that the quality of solutions produced by the primal-dual heuristic is adequate for solving realistic size problems.

2. Regarding the performance of the relaxations, the bounds from the second-order relaxation improve over the first-order ones, and in most instances the bound was very close to the exact optimal value. In Problem 1 there is a wider gap between the value of the primal-dual heuristic and the value of the second-order relaxation. The closeness of the value of the heuristic with the value of the greedy solution (which is expected to be near optimal in this case), suggests that the main source of this gap is the inaccuracy of the second-order bound.

Problem instance	β	Z_{Greedy}	Z_{PDH}	Z^*	Z^2	Z^1
Problem 1	0.20	59.9	59.9		70.62	74.67
	0.50	124.2	124.3		162.05	166.35
	0.90	814.4	819.1		898.99	913.40
Problem 2	0.20	117.1	117.3		117.92	118.46
	0.50	180.2	180.2		183.89	186.10
	0.90	863.1	863.4		894.18	915.44
Problem 3	0.50	14.7	14.7	14.72	15.33	16.10
	0.90	81.3	81.5	81.55	84.54	85.29
	0.95	164.5	164.9	164.98	169.60	171.07
Problem 4	0.50	10.8	11.4	11.40	11.65	11.92
	0.90	65.1	75.1	75.15	75.99	78.36
	0.95	135.5	156.0	156.09	157.81	162.12
Problem 5	0.50	19.2	21.5	21.63	21.93	21.93
	0.90	122.5	144.5	144.73	146.50	147.29
	0.95	257.2	300.1	300.35	303.73	305.68
Problem 6	0.50	28.0	30.8	30.95	31.33	31.53
	0.90	167.5	209.5	209.56	209.56	209.56
	0.95	346.2	434.7	434.74	434.74	434.74
Problem 7	0.50	10.7	10.9	10.93	10.93	10.93
	0.90	58.0	74.3	74.35	74.37	74.55
	0.95	119.2	154.0	154.09	154.09	154.42

Table 5.2: Numerical experiments.

3. As expected, the performance of the greedy heuristic deteriorates as the discount factor approaches 1, since in that case the long-term impact of current decisions is more heavily weighted. The primal-dual heuristic outperforms the greedy heuristic over the sample problems (it performs significantly better in instances with higher discount factors, and never worse, even for $\beta = 0.2$). Notice that in the randomly generated instances both heuristics yield very close rewards.

Chapter 6

Optimal scheduling of multiclass queueing networks: from flow conservation laws to linear programming formulations

Designers of a wide variety of real-world systems are faced with the task of managing a flow of diverse jobs with limited service resources in order to achieve desired performance characteristics. Consider, for example, a manufacturing plant that produces several product parts. As parts flow through the plant floor for completing their processing requirements, they compete for the service capacity of the available workstations. A scheduling policy is thus needed that decides dynamically how to allocate workstations to parts demanding simultaneously their attention. Economic criteria for comparing the performance characteristics of different policies typically involve factors such as the value of the parts produced and/or the cost of the inventories held. Similar scheduling problems arise in other important application areas such as computer and communication networks.

Two fundamental problems faced by designers of real-world scheduling systems are:

1. The performance evaluation problem: given a proposed scheduling policy, evaluate its performance characteristics.

2. The performance optimization problem: Given an economic performance objective,

design a scheduling policy with optimal (or close to optimal) performance characteristics.

An analytical approach to these problems represents the real-world system by an appropriate queueing network model (see e.g. Walrand (1988)). Solutions to the corresponding problems in the queueing model would thus provide guidelines for evaluating and designing scheduling policies in the real system.

The application of this approach has been hindered by the fact that only a relatively small range of relevant queueing network models seem to yield exact analytical results. These include mainly models whose equilibrium distribution admits a product-form solution. Starting with the pioneering works of R. R. P. Jackson (1954) and of J. R. Jackson (1963), researchers have significantly extended the range of queueing network models known to possess that property (see e.g. Kelly (1979) and Walrand (1988)). For these models one can evaluate, for certain simple scheduling policies, the distribution of performance measures of interest, thus solving the performance evaluation problem. Most relevant models, however, are known not to admit a product-form equilibrium solution. That is the case, for example, with such a simple model as a multiclass $M/M/1$ queue in which each class has its own rate of service, under a first-come first-serve scheduling policy. It is not surprising, therefore, that the most widespread approach to performance evaluation among practitioners is based on computer simulation.

As for the performance optimization problem, it can be formulated in principle in the framework of dynamic programming. The key difficulty with this approach is the large — or even infinite — size of the resulting formulations, which hinders their application. Researchers have overcome this difficulty in several important problems whose optimal policies have a simple structure, as was shown in Chapter 4. For most important models, though, it has not been possible to characterize the structure of optimal scheduling policies nor, as a result, to provide efficient solution schemes. A theoretical explanation of this fact has been recently provided by Papadimitriou and Tsitsiklis (1994), who have established that the problem of scheduling optimally a queueing network is $EXPTIME - complete$: its solution requires an amount of time which is at least exponential in the size of the input data, independently of the $P = NP$ conjecture.

Given that situation, practitioners rely on heuristic considerations for designing scheduling policies. The performances of alternative heuristic policies — estimated by simulation

— are often compared only among themselves, instead of with the optimal performance.

The purpose of this chapter is to develop the mathematical programming approach to scheduling problems, outlined in Chapter 1, in the domain of multiclass queueing networks. The approach is based on constructing compact polyhedral relaxations of the performance region corresponding to a queueing network model. These relaxations lead to approximate linear programming formulations of the associated scheduling problem. Their solution provides an estimate of the optimal performance vector, as well as polynomial-time bounds on the optimum performance value, which can be used as a benchmark for assessing the degree of suboptimality of a proposed policy.

As has been shown in the previous chapters, the key for constructing strong formulations for the problems investigated seems to lie in identifying a "complete" set of conservation laws that, when translated into corresponding constraints characterize the performance region. Thus the set of work conservation laws led to complete polyhedral characterizations of the performance region in indexable scheduling problems, as was shown in Chapters 3 and 4.

That result leads naturally to the question: What are the laws of multiclass queueing networks, that allow to characterize their performance region? We provide in this chapter a partial answer to that question, by identifying a set of *flow conservation laws*, and applying them to construct corresponding linear programming formulations.

Higher-order flow conservation laws, that account for higher-order interactions in the system, lead to additional constraints, and therefore to stronger formulations and performance bounds. A sequence of formulations may thus be obtained, which correspond geometrically to a lift-and-project approach: a new formulation in the sequence is obtained by lifting the feasible region of the previous one into a higher-dimensional space — introducing new variables and constraints — and then projecting back into the original performance space. This sequence of formulations was first obtained by Bertsimas, Paschalidis and Tsitsiklis (1994) using potential function methods, and independently by Kumar and Kumar (1994).

The chapter is structured as follows: Section 6.1 reviews a classical flow conservation law of queueing theory, that is the basis for the formulations to be developed in the chapter. Section 6.2 presents an exact compact reformulation of the undiscounted branching bandit problem. Section 6.3 develops a sequence of relaxations for dynamic scheduling problems in Markovian multiclass queueing networks. The formulations are derived in a unifying way, by identifying and applying flow conservation laws satisfied by the underlying queueing

systems.

6.1 Stochastic flow conservation

In this section we review a classical flow conservation law of queueing systems. The law roughly states that, under mild conditions, the state of a queueing system as found by arriving jobs is equal in distribution to the state left behind by departing jobs.

We shall present two version of the flow conservation law of a queueing system: a busy period sample-path version, that asserts total flow conservation over a busy period of the system, and a stationary version, that may be interprested as stating long-run flow conservation over the system in equilibrium. In the applications of the law to be presented later, we will show that the first version is useful for analyzing systems with a single busy period, such as branching bandits, whereas the second one is appropriate for studying systems that reach stochastic equilibrium, such as Markovian queueing networks.

Consider a generic queueing system where jobs arrive with random demands for service, wait until a server is assigned to them, and leave after their service is completed. The key assumption for the flow conservation law to be presented below to hold is that jobs arrive to and leave the system one at a time.

6.1.1 The flow conservation law $L^- = L^+$: busy period version

We next describe the flow conservation law over the first busy period of such a queueing system. Let us denote the length of the first busy period T, which we assume to be finite. Assuming that the system starts servicing jobs at time $t = 0$, the busy period is thus the interval $[0, T)$.

Let $L(t)$ denote the number of jobs in the system (waiting or in service) at time t. We assume that process $\{L(t)\}_{t\in[0,T)}$ has right-continuous sample paths. We denote $L(t-)$ the left limit of the process at time t. The corresponding right limit is $L(t+) = L(t)$, because of the right-continuity.

Let ν be the total number of jobs serviced during the busy period. Let us denote $\{\tau_k^a\}_{k=1}^{\nu}$ and $\{\tau_k^d\}_{k=1}^{\nu}$ the sequences of job arrival and departure epochs, respectively. Let $L(\tau_k^a-)$ be the number of jobs in the system that the kth arriving job sees just before his arrival.

Clearly,

$$L(\tau_k^a-) = L(\tau_k^a) - 1.$$

Similarly, $L(\tau_k^d)$ may be interpreted as the number of jobs in the system that the kth departing job sees just after his departure. Let us define

$$L^- \triangleq \sum_{k=1}^{\nu} L(\tau_k^a-),$$

and

$$L^+ \triangleq \sum_{k=1}^{\nu} L(\tau_k^d).$$

Notice that L^- represents the total number of jobs in the system seen by arriving jobs, while L^+ is the total number of jobs in the system seen by departing jobs. The flow conservation law states the equality of L^- and L^+.

Theorem 17 (Flow conservation law: busy period version) *If jobs enter and leave the system one at a time, then the following sample-path identity holds:*

$$L^- = L^+.$$

Proof

Consider a given realization of the busy period (see Figure 6-1). Since the system becomes empty at the end of the busy period, and arrivals and departures occur one at a time, it follows that for each time that there is an upward transition for the number in the system from i to $i+1$, there is a corresponding downward transition for the number in the system from $i+1$ to i. That is, every $L(\tau_k^a-)$ is equal to a distinct $L(\tau_k^d)$, thus proving the result. □

6.1.2 The flow conservation law $L^- = L^+$: stationary version

Consider now a stationary queueing system with number in system process $\{L(t)\}_{t\in\Re}$. Analogously as in the busy period version case, we assume that process $\{L(t)\}_{t\in\Re}$ has right-continuous sample paths. We denote $L(t-)$ the left limit of the process at time t. The corresponding right limit is $L(t+) = L(t)$, because of the right-continuity.

Let $A = \{\tau_k^a\}$ and $D = \{\tau_k^d\}$ denote the sequences of arrival and departure epochs of

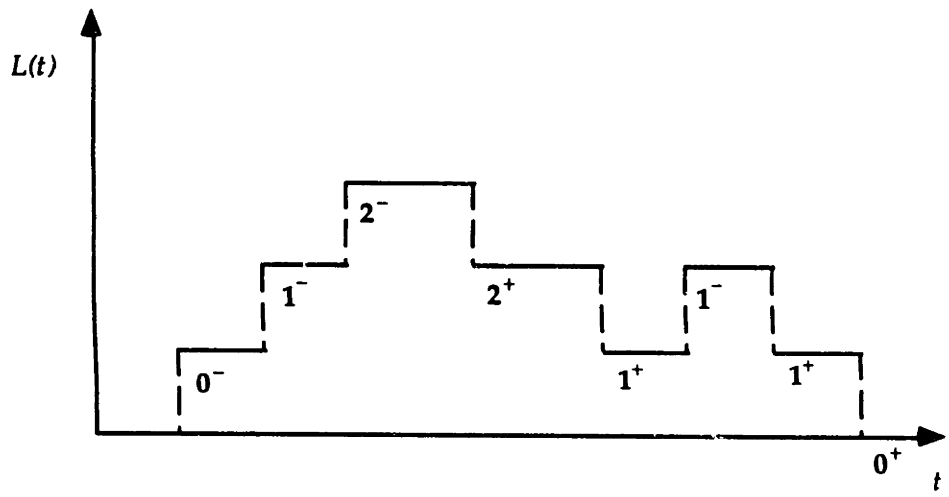

Figure 6-1: The flow conservation law $L^- = L^+$ over a busy period.

jobs, respectively. Let us define

$$L^- \triangleq L(\tau_0^a -),$$

and

$$L^+ \triangleq L(\tau_0^d).$$

Since we assumed the system to be in equilibrium, L^- may be interpreted as the number of jobs in the queue seen by a typical arriving job, while L^+ corresponds to the number in the system seen by a typical departig job.

Theorem 18 (Flow conservation law: stationary version) *If jobs enter and leave the system one at a time, then the identity*

$$L^- \stackrel{\mathrm{d}}{=} L^+$$

holds in distribution.

6.2 A compact linear programming formulation for branching bandit problems

In Chapter 3 we derived an exponential-size linear programming formulation of the undiscounted branching bandit problem, by identifying work conservation laws satisfied by the system and expressing them as linear constraints on performance vectors. In this section we

construct a compact linear programming formulation of the problem based on identifying and applying stochastic flow conservation laws. In contrast to the previous formulation, the resulting reformulation has polynomial-size which has practical implications for solving variations of the problem involving, e.g., a nonlinear objective function, or additional side constraints. This reformulation was first obtained by Bertsimas, Paschalidis and Tsitsiklis (1994b) using potential function methods.

Let us review the description of the *branching bandit model*, which was presented in Chapter 4. A single server must complete the service requirements of a stream of arriving jobs, which come in a finite set of classes $\mathcal{N} = \{1, \ldots, n\}$. Jobs in the same class have the same stochastic characteristics: a class-i job has a service time v_i and, just before completing its service, a number N_{ij} of new class-j *descendant* jobs enter the system, for $j \in \mathcal{N}$. The random vector of service time and descendants of a job is drawn, independently of other jobs, from an arbitrary joint distribution that only depends on its class. Jobs are to be selected for service under a *scheduling policy*, which must be *nonanticipative* (decisions may only use past or current information on the evolution of the system), *nonidling* (the server is busy while there are jobs in the system), and *nonpreemptive* (the service of a job, once started, proceeds until its completion). We shall refer to scheduling policies with these three properties as the class $\mathcal{U}$ of *admissible* policies.

We consider now economic criteria for comparing the system performance characteristics under different scheduling policies, and define a corresponding optimization problem. A holding cost h_j is incurred per unit time that a class-j job is in the system (waiting or in service). The *undiscounted branching bandit problem* may be now stated as follows: find an admissible scheduling policy under which the total expected holding cost is minimized.

As seen in Chapter 3, this optimization problem is well defined only when the total expected number of jobs that complete service is finite or, equivalently, when the length of the busy period is finite. We will assume in what follows that this is the case.

In order to analyze the sample-path of a branching bandit process, it will be useful to introduce the following variables:

$T =$ length of the busy period;

$v_{ik} =$ service time corresponding to the kth service of a class-i job; notice that the distribution of v_{ik} is independent of k (v_i).

τ_{ik} = starting time of the kth service of a class i job;

ν_i = total number of class-i jobs served (may be infinite);

$I_i(t)$ = 1 if a class i job is in service at time t; 0 otherwise;

$L_i(t)$ = number of class-i jobs in the system (waiting or in service) at time t. We denote $L(t) = (L_i(t))_{i \in \mathcal{N}}$

6.2.1 Performance measures

Our approach to the undiscounted branching bandit problem is based on formulating the problem as a linear program over a suitably defined performance region. For that purpose, we introduce the following performance measures: given an admissible scheduling policy $u \in \mathcal{U}$, let us define $W_j(u)$ as the total system time of class-j jobs, i.e.,

$$W_j(u) = E_u \left[\int_0^T L_j(t)\, dt \right], \tag{6.1}$$

and $x_{ij}(u)$ as the total system time of class-j jobs while class-i jobs are in service, scaled by the expected class-i jobs service time, i.e.,

$$x_{ij}(u) = \frac{1}{E[v_i]} E_u \left[\int_0^T I_i(t) L_j(t)\, dt \right]. \tag{6.2}$$

Let us also write $W(u) = (W_j(u))_{j \in \mathcal{N}}$ and $X(u) = (x_{ij}(u))_{i,j \in \mathcal{N}}$.

The branching bandit problem we introduced above can now be stated, in terms of these performance measures, as the optimal control problem

$$(OCBB) \qquad Z = \min \{\, h'W(u) : u \in \mathcal{U} \,\}.$$

In Chapter 3 we used work conservation laws to characterize the performance region spanned by performance vectors $W(u)$ as a special kind of polyhedron defined by an exponential number of inequality constraints (an extended polymatroid), thus reducing control problem $(OCBB)$ to a linear program with special structure. In this section we shall see that the same performance region can be represented as the projection of a higher-dimensional polyhedron with a polynomial number of constraints, by identifying certain strong flow conservation laws and expressing them in terms of performance measures W^u and X^u.

6.2.2 Strong flow conservation laws

The conservation laws we shall use correspond to applying the stochastic flow conservation law $L^- = L^+$ (busy period version) described in Section 6.1 to certain subsets of job classes. Consider first the subsystem consisting only of class-j jobs, for a given $j \in \mathcal{N}$. This subsystem can be seen as a queueing system in its own right. Let $L_{\{j\}}(t) = L_j(t)$ denote the corresponding number in system at time t. Consistently with the definition of L^- in Section 6.1, we define $L^-_{\{j\}}$ as the total number of class-j jobs seen by arriving class-j jobs, and $L^+_{\{j\}}$ as the total number of class-j jobs seen by departing class-j jobs. Similarly, we may consider the the subsystem that consists only of jobs of classes i and j, for given $i, j \in \mathcal{N}$, with $i < j$. Let $L_{\{i,j\}}(t) = L_i(t) + L_j(t)$ denote the corresponding number in system at time t. Analogously as above, we define $L^-_{\{i,j\}}$ (resp. $L^+_{\{i,j\}}$ as the total number of jobs of classes i or j seen by arriving (resp. departing) jobs of classes i or j. Direct application of the law $L^- = L^+$, in its busy period version, to these subsystems yields the following result.

Proposition 15 (Stochastic flow conservation laws for branching bandits) *The following sample-path flow conservation laws hold, under any admissible scheduling policy,*

(a) $L^-_{\{j\}} = L^+_{\{j\}}$, *for* $j \in \mathcal{N}$.

(b) $L^-_{\{i,j\}} = L^+_{\{i,j\}}$, *for* $i, j \in \mathcal{N}$, *with* $i < j$.

We shall express expectation versions of the flow conservation laws in Proposition 15 as linear equations on performance measures W^u and X^u. In order to achieve that goal, we shall first show how to express these performance measures in terms of the sample-path variables of the process.

Proposition 16 *Under any admissible scheduling policy u, the following relations hold:*

(a)

$$x^u_{ij} = E_u\left[\sum_{k=1}^{\nu_i} L_j(\tau_{ik})\right], \qquad \text{for } i, j \in \mathcal{N}.$$

(b)

$$W^u_j = \sum_{i \in \mathcal{N}} E[v_i] x^u_{ij}, \qquad \text{for } j \in \mathcal{N}.$$

Proof

(a) We can write

$$x^u_{ij} = E_u\left[\int_0^T I_i(t) L_j(t)\, dt\right]$$

$$
\begin{aligned}
&= E_u\left[\sum_{k=1}^{\nu_i}\int_{\tau_{ik}}^{\tau_{ik}+v_{ik}} L_j(t)\,dt\right] \\
&= E_u\left[\sum_{k=1}^{\nu_i} L_j(\tau_{ik})v_{ik}\right] \\
&= E[v_i]E_u\left[\sum_{k=1}^{\nu_i} L_j(\tau_{ik})\right].
\end{aligned}
$$

(b) Since u is nonidling, $\sum_{i\in\mathcal{N}} I_i(t) = 1$, for $t \in [0,T)$. Therefore,

$$
\begin{aligned}
W_j^u &= E_u\left[\int_0^T L_j(t)\,dt\right] \\
&= E_u\left[\sum_{k=1}^{\nu_i} I_i(t)\,L_j(t)\,dt\right] \\
&= \sum_{i\in\mathcal{N}} E[v_i]x_{ij}^u.
\end{aligned}
$$

□

The next result expresses the expected values of $L^-_{\{j\}}$, $L^+_{\{j\}}$, $L^-_{\{i,j\}}$ and $L^+_{\{i,j\}}$ as linear functions of performance measures W^u and X^u.

Proposition 17 *Under every admissible scheduling policy u, the following relations hold:*
(a)

$$
E_u\left[L^-_{\{j\}}\right] = \sum_{r\in calN} E[N_{rj}]\,x_{rj}^u + \frac{1}{2}E\left[\,L_j(0)\,(L_j(0)-1)\,\right] + \frac{1}{2}\sum_{r\in\mathcal{N}} E[\nu_r]\,E\left[N_{rj}(N_{rj}-1)\right];
$$

(b)

$$
E_u\left[L^+_{\{j\}}\right] = x_{jj}^u + E[\nu_j]E[N_{jj}-1];
$$

(c)

$$
\begin{aligned}
E_u\left[L^-_{\{i,j\}}\right] &= E_u\left[L^-_{\{i\}}\right] + E_u\left[L^-_{\{j\}}\right] + \sum_{r\in\mathcal{N}} E[N_{rj}]\,x_{ri}^u + \\
&\quad \sum_{r\in\mathcal{N}} E[N_{ri}]\,x_{rj}^u + E_u\left[L_i(0)\,L_j(0)\right] + E[\nu_r]\,E[N_{ri}\,N_{rj}];
\end{aligned}
$$

(d)

$$
E_u\left[L^+_{\{i,j\}}\right] = E_u\left[L^+_{\{i\}}\right] + E_u\left[L^+_{\{j\}}\right] + x_{ij}^u + x_{ji}^u + E[N_{ij}]\,E[\nu_i] + E[N_{ji}]\,E[\nu_j].
$$

Proof

(a) We can decompose $L^-_{\{j\}}$ into two terms:

1. The total number of class-j jobs seen by class-j jobs present at the start (we assume they all arrive at time $t = 0$, but in an arbitrary order).

2. The total number of class-j jobs seen by class-j jobs that arrive later.

As for the first term, there are initially $L_j(0)$ class-j jobs in the system. Assuming that they arrive in an arbitrary order, the first one to arrive will see 0 class-j jobs in the system, the second one will see 1, and so on. In this way, the $L_j(0)$th class-j job to arrive will see $L_j(0) - 1$ class j jobs in the system. The total for this first term is therefore

$$0 + 1 + \cdots + (L_j(0) - 1) = \frac{1}{2} L_j(0)(L_j(0) - 1).$$

As for the second term, notice that for every class $r \in \mathcal{N}$, the kth job of class r that completes service generates N^k_{rj} descendant jobs of class j. By definition of the branching bandit process, these descendants are assumed to arrive immediately *before* the departing class-r job leaves the system. Again, ordering these class j arrivals in an arbitrary order, we obtain that the nth of them to arrive sees $L_j(\tau_{rk}) + n - 1$ class-j jobs in the system, for $n = 1, \ldots, N^k_{rj}$. The total for the second term is thus

$$(L_j(\tau_{rk}) + \frac{N^k_{rj} - 1}{2}) N^k_{rj}.$$

Therefore, the sample-path identity

$$L^-_{\{j\}} = \frac{1}{2} L_j(0)(L_j(0) - 1) + \sum_{r \in \mathcal{N}} \sum_{k=1}^{\nu_r} (L_j(\tau_{rk}) + \frac{N^k_{rj} - 1}{2}) N^k_{rj} \tag{6.3}$$

holds. Now, taking expectations in both sides of equation (6.3) under an admissible policy u, and simplifying the resulting expressions using Proposition 16(a), we obtain (a).

(b) Notice that the kth job of class-j that completes service leaves behind on departure $L_j(\tau_{jk}) - 1 + N^k_{jj}$ class-j jobs. Therefore,

$$L^+_{\{j\}} = \sum_{k=1}^{\nu_j} (L_j(\tau_{jk}) - 1 + N^k_{jj}). \tag{6.4}$$

Taking expectations in equation (6.4) with respect to scheduling policy u, we obtain

$$E\left[L^+_{\{j\}}\right] = E_u\left[\sum_{k=1}^{\nu_j} L_j(\tau_{jk})\right] - E[\nu_j](1 - E[N_{jj}])$$

which, when simplified using Proposition 16(a) yields (b).

(c) Applying a similar counting argument as in part (a), we obtain

$$\begin{aligned} L^-_{\{i,j\}} &= \frac{1}{2}(L_i(0) + L_j(0))(L_i(0) + L_j(0) - 1) + \\ &\quad \sum_{r\in\mathcal{N}}\sum_{k=1}^{\nu_r}(N^k_{ri} + N^k_{rj})(L_i(\tau_{rk}) + L_j(\tau_{rk}) + \frac{N^k_{ri} + N^k_{rj} - 1}{2}) \\ &= L^-_i + L^-_j + L_i(0)L_j(0) + \sum_{r\in\mathcal{N}}(\sum_{k=1}^{\nu_r} L_i(\tau_{rk})N^k_{rj} + L_j(\tau_{rk})N^k_{ri} + N^k_{ri}N^k_{rj}), \quad (6.5) \end{aligned}$$

from which (c) follows by taking expectations with respect to u and simplifying as in parts (a) and (b).

(d) Applying a similar counting argument as in part (b), we obtain

$$\begin{aligned} L^+_{\{i,j\}} &= \sum_{k=1}^{\nu_i}(L_i(\tau_{ik}) + L_j(\tau_{ik}) - 1 + N^k_{ii} + N^k_{ij}) + \\ &\quad \sum_{k=1}^{\nu_j}(L_i(\tau_{jk}) + L_j(\tau_{jk}) - 1 + N^k_{ji} + N^k_{jj}) \\ &= L^+_i + L^+_j + \sum_{k=1}^{\nu_i}(L_j(\tau_{ik}) + N^k_{ij}) + \sum_{k=1}^{\nu_j}(L_i(\tau_{jk}) + N^k_{ji}), \quad (6.6) \end{aligned}$$

from which (d) follows by taking expectations with respect to u and simplifying as in parts (a) and (b). □

We shall next show that performance measures W^u, X^u for branching bandits satisfy strong flow conservation laws, as defined in Section 3.2.2. Let $\Lambda = \mathrm{Diag}(E[\nu_i])$.

Theorem 19 (Strong flow conservation laws for branching bandits) *Performance measures W^u and X^u for branching bandits satisfy the following flow conservation laws: for any admissible scheduling policy u,*

(a)

$$(I - E[N])'X^u + X^{u\prime}(I - E[N]) = E\left[L(0)L(0)'\right] + E\left[(I-N)'\Lambda(I-N)\right], \qquad (6.7)$$

and

$$W^{u\prime} = E[v]'X^u. \qquad (6.8)$$

(b) *For any pair of job classes $i, j \in \mathcal{N}$, and any admissible scheduling policy u that gives priority to class-j over class-i jobs,*

$$x^u_{ij} = 0$$

.

Proof

(a) Matrix equation (6.7) follows by taking expectations with respect to u in the stochastic flow conservation laws stated in Proposition 15, and simplifying the resulting expressions using the relations in Proposition 17.

In this way, the identity $E_u\left[L^-_{\{j\}}\right] = E_u\left[L^+_{\{j\}}\right]$ in Proposition 15(a) may be written equivalently as

$$\begin{aligned} x^u_{jj} - \sum_{r\in\mathcal{N}} E[N_{rj}]x^u_{rj} \;&=\; \frac{1}{2}E\left[L_j(0)^2\right] + \frac{1}{2}E[\nu_j] - \\ &\qquad E[\nu_j]\,E[N_{jj}] + \frac{1}{2}\sum_{r\in\mathcal{N}} E[\nu_r]E\left[N^2_{rj}\right], \end{aligned} \qquad (6.9)$$

by using Proposition 17(a),(b) and simplifying the resulting equation using the identity

$$E[\nu_j] = L_j(0) + \sum_{r\in\mathcal{N}} E[\nu_r]E[N_{rj}], \qquad \text{for } j \in \mathcal{N}. \qquad (6.10)$$

Equation (6.9) expresses the equality between the jth diagonal elements in matrix equation (6.7).

Moreover, the identity $E_u\left[L^-_{\{i,j\}}\right] = E_u\left[L^+_{\{i,j\}}\right]$ in Proposition 15(b) may be written equivalently as

$$\begin{aligned} x^u_{ij} + x^u_{ji} - \sum_{r\in\mathcal{N}} E[N_{rj}]\,x^u_{ri} - \sum_{r\in\mathcal{N}} E[N_{ri}]\,x^u_{rj} \;&=\; L_i(0)L_j(0) - E[\nu_i]\,E[N_{ij}] - \\ &\qquad E[\nu_j]\,E[N_{ji}] + \end{aligned}$$

$$\sum_{r \in \mathcal{N}} E[\nu_r] E[N_{ri} N_{rj}], \tag{6.11}$$

by using Proposition 17(c),(d). Equation (6.11) expresses the equality between the elements in position (i, j) in matrix equation (6.7).

Furthermore, relation (6.8) follows from (16).

Part (b) follows directly from the definition of x_{ij}^u. □

6.2.3 Linear programming formulation

As a direct consequence of Theorem 7, on the polyhedral characterization of the performance region in a system that satisfies strong flow conservation laws, the following result follows:

Theorem 20 *The performance region of branching bandits corresponding to performance measures W^u is the projection over the space of the variables W_j's of the polyhedron*

$$(I - E[N])'X + X'(I - E[N]) = E\left[L(0)L(0)'\right] + E\left[(I - N)'\Lambda(I - N)\right], \tag{6.12}$$

$$W' = E[v]'X, \tag{6.13}$$

$$X \geq 0. \tag{6.14}$$

As a consequence of Theorem 20, branching bandit problem ($OCBB$) may be formulated as a linear program over the feasible region defined by constraints (6.12), (6.13) and (6.14), having thus $O(n^2)$ variables and constraints.

The polynomial-size projection representation of the performance region for branching bandits presented in Theorem 20 was first obtained by Bertsimas, Paschalidis and Tsitsiklis (1994b) using potential function methods.

6.3 Linear programming formulations for scheduling problems in Markovian multiclass queueing networks

In this section we apply the stationary version of the flow conservation law $L^- = L^+$ to construct a sequence of increasingly stronger compact linear programming relaxations for dynamic scheduling problems in open Markovian multiclass queueing networks.

A Markovian open multiclass queueing network can be described as follows: A finite set of single-server stations $\mathcal{M} = \{1, \ldots, M\}$ provides service to jobs which belong in a finite set of classes $\mathcal{N} = \{1, \ldots, N\}$. We shall refer to the set of job classes served at station m as its *constituency*, and shall denote it $\mathcal{C}_m$. Jobs of class $i \in \mathcal{N}$ enter the system according to a Poisson process of rate λ_{i0}. Class i jobs demand service at server $m(i) \in \mathcal{M}$, and their service time is exponentially distributed with rate μ_i. Just after completing its service, a class i job transfers to class j with probability p_{ij}, or leaves the system with probabiliy p_{i0}. Service times for different jobs are independent, and they are also independent of the arrival streams.

Servers are to be allocated to jobs at arrival and departure epochs (internal as well as external), according to a scheduling policy which must satisfy two requirements:

1. It must be *nonanticipative*, i.e., scheduling decisions may be based only on past or current information on the evolution of the process.

2. It must be *stable*, i.e., the queueing network must admit an equilibrium distribution under the scheduling policy.

We shall refer to the class $\mathcal{U}$ of scheduling policies satisfying these requirements as the class of *admissible* policies. A sufficient condition for the existence of a stable policy is that the system of traffic equations

$$\lambda_j = \lambda_{0j} + \sum_{r \in \mathcal{N}} p_{rj} \lambda_r, \qquad \text{for } j \in \mathcal{N}, \tag{6.15}$$

has a unique positive solution $(\lambda_1, \ldots, \lambda_n)$ that satisfies

$$\rho(m) < 1, \qquad \text{for } m \in \mathcal{M}, \tag{6.16}$$

where

$$\rho(m) = \sum_{i \in \mathcal{C}_m} \frac{\lambda_i}{\mu_i}, \qquad \text{for } m \in \mathcal{M}.$$

In this case, λ_j represents the total arrival rate (internal plus external) of class-j jobs (see e.g. Walrand (1988)).

We consider now economic criteria for comparing the system performance characteristics under different scheduling policies, and define a corresponding optimization problem. A

holding cost h_j is incurred per unit time that a class-j job is in the system (waiting or in service). The *open multiclass queueing network scheduling problem* may be now stated as follows: find an admissible scheduling policy under which the expected holding cost in equilibrium is minimized.

Let us denote $\{L(t)\}_{t\in\Re}$ the number in system process, which we assume to be in equilibrium, where $L_i(t)$ is the number of class-i jobs in the system at time t.

We shall show next how the stationary version of the flow conservation law can be used to obtain partial polyhedral characterizations of the perfomance region for open multiclass queueing networks. The approach can be summarized as follows: We apply the flow conservation law $L^- = L^+$ to subsets of job classes; we then express the corresponding identities in terms of job averages, applying Theorem 23 on the superposition of point processes; finally, we translate the resulting relations into equations involving time averages, using Papangelou's Theorem 24, which yields corresponding equality constraints on achievable performance vectors.

In order to analyze the sample-path of a multiclass queueing network, it will be useful to introduce the following variables: Let

$$B_j(t) = \begin{cases} 1 & \text{if a class-}j\text{ job is in service at time } t; \\ 0 & \text{otherwise,} \end{cases}$$

and

$$B^m(t) = \begin{cases} 1 & \text{if service node } m \text{ is busy at time } t; \\ 0 & \text{otherwise.} \end{cases}$$

Our approach to the multiclass queueing network scheduling problem is based on formulating a sequence of linear programming relaxations. The feasible regions of these linear programs contain a suitably defined performance region of the problem. For that purpose, we introduce the following Kth-order performance measures, for a given $K \geq 1$: given an admissible scheduling policy $u \in \mathcal{U}$, let us define $x_u(i_1, \ldots, i_K)$ as the expected value of the product of the number in system corresponding to classes $i_1, \ldots, i_K$ at a typical time (such as $t = 0$), i.e.,

$$x_u(i_1, \ldots, i_K) \triangleq E_u\left[L_{i_1}(0) \cdots L_{i_K}(0)\right];$$

let $x_u(i_1, \ldots, i_K \mid i)$ denote the same expected product, but conditional on the event that

there is a class-i job in service. i.e., and

$$x_u(i_1, \ldots, i_K \mid i) \triangleq E_u[L_{i_1}(0) \cdots L_{i_K}(0) \mid B_i(0) = 1];$$

finally, let $x_u^m(i_1, \ldots, i_K \mid 0)$ denote the same expected p.oduct, but conditional on the event that station m is idle, i.e.,

$$x_u^m(i_1, \ldots, i_K \mid 0) \triangleq E_u[L_{i_1}(0) \cdots L_{i_K}(0) \mid B^m(0) = 0].$$

The open multiclass queueing network scheduling problem we introduced above can now be stated, in terms of these performance measures, as the optimal control problem

$$(OCON) \qquad Z = \min \left\{ \sum_{i \in \mathcal{N}} h_i \, x_u(i) : u \in \mathcal{U} \right\}.$$

We shall show in this section that the performance region corresponding to performance measures $x_u(i)$'s may be approximated as the projection of higher-dimensional polyhedra, by identifying certain flow conservation laws and expressing them in terms of the performance measures introduced above.

The conservation laws we shall use correspond to applying the stochastic flow conservation law $L^- = L^+$ described in Section 6.1 to certain subsets of job classes. Let $S \subset \mathcal{N}$ be a subset of job classes. Consider first the subsystem consisting only of jobs whose classes belong in S, which we shall refer to as S-jobs. This subsystem can be seen as a queueing system in its own right. Let $L_S(t) = \sum_{i \in S} L_i(t)$ denote the corresponding number in system at time t. Consistently with the definition of L^- in Section 6.1, we define L_S^- as the total number of S-jobs seen by arriving S-jobs, and L_S^+ as the total number of S-jobs seen by departing S-jobs. Direct application of the law $L^- = L^+$, in its stationary version, to this subsystem yields the following result.

Proposition 18 (Stochastic flow conservation laws for queueing networks) *The following distributional flow conservation laws hold, under any admissible scheduling policy,*

$$L_S^- = L_S^+, \qquad \text{for } S \subset \mathcal{N}.$$

We shall express expectation versions of the flow conservation laws in Proposition 18 as

linear equations on performance measures $x_u(i_1,\ldots,i_K)$, $x_u(i_1,\ldots,i_K \mid i)$ and $x_u^m(i_1,\ldots,i_K \mid 0)$ introduced above. Let us denote

$$p(i,S) = \sum_{j\in S} p_{ij},$$

$$p(i,S\cup\{0\}) = \sum_{j\in S\cup\{0\}} p_{ij},$$

and

$$\lambda^0(S) = \sum_{i\in S} \lambda_i^0.$$

Let us denote $i^K = (i_1,\ldots,i_K)$, and let S^k be the set of all k-tuples $i^K = (i_1,\ldots,i_K)$, with components in S.

Theorem 21 (Kth-order flow conservation laws) *The following relations hold, under any admissible scheduling policy u:*

(a) *For a subset S of job classes,*

$$\lambda^0(S)\sum_{i^K\in S^K} x_u(i^K) + \sum_{i\in S^c}\sum_{i^K\in S^K} \lambda_i p(i,S)\, x_u(i^K \mid i) = \sum_{i\in S}\sum_{k=1}^{K}\sum_{i^k\in S^k} (-1)^{K-k}\binom{K}{k}\lambda_i p(i,S^c\cup\{0\})\, x_u(i^k \mid i) + (-1)^K \sum_{i\in S}\lambda_i p(i,S^c\cup\{0\}). \tag{6.17}$$

(b)

$$x_u(i^K) = \sum_{i\in\mathcal{C}_m} \frac{\lambda_i}{\mu_i} x_u(i^K \mid i) + (1-\rho(m)) x_u^m(i^K \mid 0). \tag{6.18}$$

Proof

The evolution of the state vector is driven by events corresponding to external job arrivals, internal job transfers, and external job departures. We shall relate expectations with respect to these event epochs with expectations in equilibrium using the theory of Palm calculus, reviewed in Appendix A. We denote the point process of external class i job arrivals A_i^0. Its intensity is λ_i^0, which coincides with its stochastic intensity at any time, i.e., $\lambda_i(t) = \lambda_i^0$. The point process corresponding to internal job transfers from class i to class j is denoted T_{ij}. Its intensity is easily seen to be $\lambda_{ij} = \lambda_i p_{ij}$, whereas its stochastic intensity is $\lambda_{ij}(t) = \mu_i p_{ij} 1_{\{B_i(t)=1\}}$. The process of external job departures is denoted D_j^0, and it has intensity

$\mu_j^0 = \lambda_j p_{j0}$ and stochastic intensity $\mu_j^0(t) = \mu_j p_{j0} 1_{\{B_j(t)=1\}}$. Let $P_{A_i^0}$, $P_{T_{ij}}$ and $P_{D_j^0}$ denote the Palm probability distributions associated to the corresponding point processes.

We next apply the flow conservation law $L^- = L^+$. Let $S \subset \mathcal{N}$ be a subset of job classes. As mentioned before, the queues corresponding to S-jobs may be considered as constituting a queueing system in their own right, which we shall refer to as $\mathcal{Q}_S$. The number of jobs in this subsystem at time t, which we denote $L_S(t)$, is given by

$$L_S(t) = \sum_{i \in S} L_i(t).$$

Let A_S be the point process corresponding to arrivals to system $\mathcal{Q}_S$, and let D_S be the corresponding point process for departures. These arrival and departure processes can be expressed as the superposition of elementary processes as follows:

$$A_S = \sum_{j \in S} A_j^0 + \sum_{i \in S^c} \sum_{j \in S} T_{ij},$$

and

$$D_S = \sum_{j \in S} D_j^0 + \sum_{i \in S} \sum_{j \in S^c} T_{ij}.$$

This is because arrivals to queueing system $\mathcal{Q}_S$ correspond in the original system either to external arrivals, or to internal job transfers from classes in S^c to classes in S. Similarly, departures from queueing system $\mathcal{Q}_S$ correspond in the original system either to external departures, or to internal job transfers from classes in S to classes in S^c.

On the other hand, we may apply the flow conservation law law $L^- = L^+$ to subsystem $\mathcal{Q}_S$. By the equality of distribution between L_S^- and L_S^+ it follows that we may equate the correspoding Kth-order moments, for any K, which yields

$$E_{A_S}\left[L_S(0-)^K\right] = E_{D_S}\left[L_S(0)^K\right].$$

Direct application of the superposition theorem of point processes (Theorem 23) allows us to express the Kth-order flow conservation identity $E_{A_S}[L_S(0-)^K] = E_{D_S}[L_S(0)^K]$ as

$$\sum_{i \in S} \frac{\lambda_i^0}{\lambda_S} E_{A_i^0}\left[L_S(0-)^K\right] + \sum_{i \in S^c} \sum_{j \in S} \frac{\lambda_i p_{ij}}{\lambda_S} E_{T_{ij}}\left[L_S(0-)^K\right] = \sum_{i \in S} \frac{\lambda_i p_{i0}}{\lambda_S} E_{D_i^0}\left[L_S(0)^K\right] +$$

$$\sum_{i \in S} \sum_{j \in S^c} \frac{\lambda_i p_{ij}}{\lambda_S} \left[L_S(0)^K \right]. \tag{6.19}$$

Our next goal is to express the Palm expectations that appear in equation (6.19) in terms of stationary expectations (and hence of performance measures). Papangelou's theorem (see Appendix A) allows us to relate the distribution of the state of the system observed at the elementary event epochs, which is easy to characterize, with the state observed in equilibrium, which is not immediately evident. Let e_i denote the ith unit coordinate vector in $\Re^n$. Straightforward application of Papangelou's theorem yields the following relations between Palm probabilities and stationary probabilities:

$$\begin{aligned} P_{A_i^0}\{L(0) = l + e_i\} &= P_{A_i^0}\{L(0-) = l\} \\ &= P\{L(0) = l\}, \end{aligned} \tag{6.20}$$

$$\begin{aligned} P_{T_{ij}}\{L(0) = l - e_i + e_j\} &= P_{T_{ij}}\{L(0-) = l\} \\ &= P\{L(0) = l \mid B_i(0) = 1\}, \end{aligned} \tag{6.21}$$

and

$$\begin{aligned} P_{D_j^0}\{L(0) = l - e_j\} &= P_{D_j^0}\{L(0-) = l\} \\ &= P\{L(0) = l \mid B_j(0) = l\}. \end{aligned} \tag{6.22}$$

Identities (6.20), (6.21) and (6.22) are obtained by applying Papangelou's theorem (Theorem 24) with

$$f(L(t)) = \begin{cases} 1, & \text{if } L(t-) = l; \\ 0, & \text{otherwise.} \end{cases}$$

Now, using the formulae (6.20), (6.21) and (6.22), that express the relation between Palm and stationary probabilities for point processes A_i^0, T_{ij} and D_j^0, we obtain corresponding relation between Palm and stationary moments. For external arrival process A_j^0, we thus have

$$E_{A_j^0}\left[L_S(0-)^K \right] = E\left[L_S(0)^K \right];$$

for transfer process T_{ij}, with $i \in S^c$ and $j \in S$,

$$E_{T_{ij}}\left[L_S(0-)^K\right] = E\left[L_S(0)^K \mid B_i(0) = 1\right];$$

for transfer process T_{ij}, with $i \in S$ and $j \in S^c$,

$$\begin{aligned} E_{T_{ij}}\left[L_S(0)^K\right] &= E_{T_{ij}}\left[(L_S(0-) - 1)^K\right] \\ &= \sum_{k=0}^{K}(-1)^{K-k}\binom{K}{k}E_{T_{ij}}\left[L_S(0-)^k\right] \\ &= \sum_{k=0}^{K}(-1)^{K-k}\binom{K}{k}E\left[L_S(0)^k \mid B_i(0) = 1\right], \end{aligned}$$

and for external departure process D_j^0,

$$\begin{aligned} E_{D_j^0}\left[L_S(0)^K\right] &= E_{D_j^0}\left[(L_S(0-) - 1)^K\right] \\ &= \sum_{k=0}^{K}(-1)^{K-k}\binom{K}{k}E_{D_j^0}\left[L_S(0-)^k\right] \\ &= \sum_{k=0}^{K}(-1)^{K-k}\binom{K}{k}E\left[L_S(0)^k \mid B_j(0) = 1\right] \end{aligned}$$

Now, substituting in identity (6.19), and simplifying, we obtain:

$$\begin{aligned} &\lambda^0(S)E_u\left[L_S(0)^K\right] + \\ \sum_{i\in S^c}\lambda_i p(i,S)E_u\left[L_S(0)^K \mid B_i = 1\right] &= \sum_{i\in S}\sum_{k=1}^{K}(-1)^{K-k}\binom{K}{k}\lambda_i p(i, S^c \cup \{0\})E_u\left[L_S(0)^k \mid B_i = 1\right] + \\ &\quad (-1)^K \sum_{i\in S} p(i, S^c \cup \{0\})\,\lambda_i, \end{aligned}$$

which proves (a).

As for (b), it follows from the relation

$$\begin{aligned} E_u[L_{i_1}(0)\cdots L_{i_K}(0)] &= \sum_{i\in C_m}\frac{\lambda_i}{\mu_i}E_u[L_{i_1}(0)\cdots L_{i_K}(0) \mid B_i(0) = 1] + \\ &\quad (1-\rho(m))E[L_{i_1}\cdots L_{i_K}(0) \mid B^m(0) = 0]. \end{aligned}$$

□

Solving the linear programming problem of minimizing objective $\sum_{i\in\mathcal{N}} h_i x(i)$ subject to the constraints induced by conservation relations (6.17) and (6.18) yields a lower bound on the optimal value of scheduling problem $(OCON)$.

By varying the subset S of job classes, and the order K of the interactions considered, the constraints induced by conservation relations (6.17) and (6.18) yield an infinite family of linear constraints on performance measures. If ordered according to the order K of the interactions taken into account, this family can be viewed as a sequence of increasingly stronger relaxations.

This sequence of relaxations may be interpreted geometrically as corresponding to a lift-and-project approach. The exact performance region spanned by performance measures $x_u(i)$, for $i \in \mathcal{N}$, is contained in the projection of higher-dimensional polyhedra.

6.3.1 Second-order relaxation for multiclass queueing networks

Let us denote $x_u = (x_u(n))_{n\in\mathcal{N}}$, $X_u = (x_u(n_1,n_2))_{n_1,n_2\in\mathcal{N}}$, and $X_u^0 = (x_u^m(n))_{m\in\mathcal{M},n\in\mathcal{N}}$. Also, let us define the *constituency matrix* $C = (c_{mn})_{m\in\mathcal{M},n\in\mathcal{N}}$ by

$$c_{mn} = \begin{cases} 1, & \text{if job class } n \text{ is served at service node } m; \\ 0, & \text{otherwise,} \end{cases}.$$

Let $\lambda^0 = (\lambda_n^0)_{n\in\mathcal{N}}$ be the vector of external arrival rates, and let $\Lambda = \mathrm{Diag}(\lambda)$, where vector λ of total arrival rates is given by traffic equations (6.15).

Direct application of Theorem 21, in the case of second-order interactions, yields the following result, first obtained by Bertsimas, Paschalidis and Tsitsiklis (1994a) using potential function methods. We use the notation $1_{\mathcal{M}}$ to denote a vector of ones with components indexed by set $\mathcal{M} = \{1,\ldots,M\}$.

Proposition 19 (Second-order flow conservation laws) *For any admissible scheduling policy $u \in \mathcal{N}$, performance measures x_u, X_u and X_u^0 satisfy the following equations:*

$$\lambda^0 x' + x\lambda^{0\prime} + (I-P)'X + X'(I-P) = (I-P)'\Lambda + \Lambda'(I-P), \tag{6.23}$$

$$1_{\mathcal{M}} x' = \begin{pmatrix} x' \\ \vdots \\ x' \end{pmatrix} = CX + X^0. \tag{6.24}$$

6.3.2 Projection of the second-order relaxation

By exploiting the structure of the second-order polyhedron defined by equations (6.23) and (6.24) it is possible to obtain an explicit partial characterization of the performance region spanned by vectors x_u, thus projecting out other variables from the formulation.

We shall define next a parametric family of polyhedra, that generalize those arising as the second-order relaxation of the performance region of open multiclass queueing networks. We shall present a partial characterization of the projection of such polyhedra over the space of certain variables. This result will later be applied to the specific case of polyhedra arising in open multiclass queueing networks.

Let $\mathcal{N} = \{1, \ldots, N\}$, $\mathcal{M} = \{1, \ldots, M\}$. Let $a_0 = (a_{0n})_{n \in \mathcal{N}} \geq 0$, $Q = (q_{ij})_{i,j \in \mathcal{N}} \geq 0$, $B = (b_{ij})_{i,j \in \mathcal{N}} \geq 0$. Let matrix C be an $M \times N$ $0-1$ matrix with the following structure: there is a partition of set $\mathcal{N} = \{1, \ldots, N\}$ into subsets $\mathcal{C}_1, \ldots, \mathcal{C}_M$, in such a way that

$$c_{mn} = \begin{cases} 1, & \text{if job class } n \text{ is served at service node } m; \\ 0, & \text{otherwise,} \end{cases}.$$

When matrix C arises from a queueing network model, $\mathcal{C}_m$ is the constituency of service node m. Let $1_{\mathcal{M}}$ denote a vector of ones with components indexed by set $\mathcal{M}$.

Consider polyhedron $\mathcal{P}$ defined by the equations

$$a_0 x' + x a_0' + (I - Q)'X + X'(I - Q) = B,$$

$$1_{\mathcal{M}} x' = \begin{pmatrix} x' \\ \vdots \\ x' \end{pmatrix} = CX + X^0,$$

$$x, X, X^0 \geq 0.$$

Let us define, for $S \subseteq \mathcal{N}$, and $m \in \mathcal{M} = \{1, \ldots, M\}$,

$$d_m^S = \begin{cases} 1, & \text{if } \mathcal{C}_m \cap S \neq \emptyset; \\ 0, & \text{otherwise,} \end{cases}.$$

For a given vector $p = (p_i)_{i \in \mathcal{N}} > 0$, let us define vector $F_{\mathcal{N}}^S = (F_n^S)_{n \in \mathcal{N}}$ as the solution of the system of equations

$$F_S^S = p_S + Q_{SS} F_S^S.$$

Proposition 20 *The projection of polyhedron $\mathcal{P}$ over the space of the x-variables is contained in the polyhedron defined by the following equations:*

$$\sum_{j \in S} F_j^S x_j \geq \frac{1}{2} \frac{F_S^{S\prime} B_{SS} F_S^S}{1\prime d^S - a_{0S}^{\prime} F_S^S}, \qquad \text{for } S \subseteq \mathcal{N}$$

Proof

We have, for

$$f = \begin{pmatrix} F_S^S \\ 0 \end{pmatrix},$$

$$\begin{aligned}
\frac{1}{2} f' B f &= \frac{1}{2} f' \{(I-Q)'X + X'(I-Q) - a_0 x' - x a_0'\} f \\
&= f' \{(I-Q)'X - a_0 x'\} f \\
&= \{(I-Q)f\}' X f - (f' a_0)(f' x) \\
&= \left\{ \begin{pmatrix} I_S - Q_{SS} & -Q_{SS^c} \\ -Q_{S^cS} & I_{S^c} - Q_{S^cS^c} \end{pmatrix} \begin{pmatrix} F_S^S \\ 0 \end{pmatrix} \right\}' X \begin{pmatrix} F_S^S \\ 0 \end{pmatrix} - (f' a_0)(f' x) \\
&= (p_S', -(F_S^S)'(Q_{S^cS})') \begin{pmatrix} X_{SS} & X_{SS^c} \\ X_{S^cS} & X_{S^cS^c} \end{pmatrix} \begin{pmatrix} F_S^S \\ 0 \end{pmatrix} - (f' a_0)(f' x) \\
&= \begin{pmatrix} p_S' X_{SS} - (F_S^S)'(Q_{S^cS})' X_{S^cS} & p_S' X_{SS^c} - (F_S^S)' Q_{S^cS}' X_{S^cS^c} \end{pmatrix} \begin{pmatrix} F_S^S \\ 0 \end{pmatrix} - \\
&\quad (f' a_0)(f' x).
\end{aligned}$$

Notice also that, for any $m = 1, \ldots, M$,

$$x_n = \sum_{i \in S \cap \mathcal{C}_m} x_{in} + \sum_{i \in S^c \cap \mathcal{C}_m} x_{in} + x_{mn}^0.$$

Hence, we obtain

$$\left(\sum_{m=1}^{M} d_m^S \right) x_n = \sum_{i \in S} x_{in} + \sum_{m=1}^{M} d_m^S \sum_{i \in S^c \cap \mathcal{C}_m} x_{in} + \sum_{m=1}^{M} d_m^S x_{mn}^0.$$

Therefore,

$$\begin{aligned}
\frac{1}{2}(F_S^S)'B_{SS}F_S^S &= \sum_{j\in S} Q_j^S(\sum_{i\in S} p_i x_{ij}) - \sum_{i\in S^c}\sum_{j\in S} Q_i^{S^c} Q_j^S p_i x_{ij} - (\sum_{i\in S} a_{0i}F_i^S)(\sum_{j\in S} F_j^S x_j) \\
&= \sum_{j\in S} F_j^S \left((\sum_{m=1}^{M} d_m^S)x_j - \sum_{m=1}^{M} d_m^S \sum_{i\in S^c\cap C_m} x_{ij} - \sum_{m=1}^{M} d_m^S x_{mj}^0 \right) - \\
&\quad (\sum_{i\in S} a_{0i}F_i^S)(\sum_{j\in S} F_j^S x_j) \\
&\leq (\sum_{m=1}^{M} d_m^S - \sum_{i\in S} a_{0i}F_i^S) \sum_{j\in S} F_j^S x_j,
\end{aligned}$$

whence the result follows. □

Specializing Proposition 20 to the second-order polyhedra arising from open queueing networks (see Proposition 19) we obtain the following result. Let $\Lambda = \mathrm{diag}(\lambda_i)$, and let $\rho = (\rho_i)_{i\in\mathcal{N}}$, where $\rho_i = \lambda_i/\mu_i$ is the traffic intensity of class-i jobs.

Theorem 22 (Workload relaxation for open networks) *For any admissible scheduling policy u, the first-order performance vector x_u of an open multiclass queueing network is contained in the polyhedron defined by the equations*

$$F_S^{S'}x_S \geq \frac{F_S^{S'}\rho_S}{1'd^S + \lambda'_{S^c}P_{S^cS}F_S^S - 1'_S\rho_S}, \qquad \text{for } S \subseteq \mathcal{N}. \tag{6.25}$$

Proof

We apply Proposition 20 to the second-order flow polyhedron describefd in Proposition 19. Let $Q = P$, $B = (I-P)'\Lambda + \Lambda'(I-P)$, $a_0 = \lambda_0$, and $p = v$, where $v = (v_i)_{i\in\mathcal{N}}$, and $v_i = 1/\mu_i$.

As for the numerator, we have

$$\begin{aligned}
\frac{1}{2}F_S^{S'}B_{SS}F_S^S &= \frac{1}{2}\begin{pmatrix} F_S^{S'} & 0 \end{pmatrix}\{(I-P)'\Lambda + \Lambda'(I-P)\}\begin{pmatrix} F_S^S \\ 0 \end{pmatrix} \\
&= \begin{pmatrix} F_S^{S'} & 0 \end{pmatrix}(I-P)'\Lambda\begin{pmatrix} F_S^S \\ 0 \end{pmatrix} \\
&= \left\{\begin{pmatrix} I_S - P_{SS} & -P_{SS^c} \\ -P_{S^cS} & I_{S^c} - P_{S^cS^c} \end{pmatrix}\begin{pmatrix} F_S^S \\ 0 \end{pmatrix}\right\}'\Lambda\begin{pmatrix} F_S^S \\ 0 \end{pmatrix}
\end{aligned}$$

$$
\begin{aligned}
&= \left(v'_S \quad v'_{S^c} - F_{S^c}^{S\,\prime} \right) \Lambda \begin{pmatrix} F_S^S \\ 0 \end{pmatrix} \\
&= F_S^{S\prime} \rho_S .
\end{aligned}
$$

As for the denominator, we have (using $\lambda_0 = (I - P)'\lambda$),

$$
\begin{aligned}
\lambda'_{0S} F_S^S &= \lambda'_0 \begin{pmatrix} F_S^S \\ 0 \end{pmatrix} \\
&= \lambda'(I - P) \begin{pmatrix} F_S^S \\ 0 \end{pmatrix} \\
&= \lambda' \begin{pmatrix} (I_S - P_{SS}) F_S^S \\ -P_{S^cS} F_S^S \end{pmatrix} \\
&= \lambda' \begin{pmatrix} v_S \\ v_{S^c} - F_{S^c}^S \end{pmatrix} \\
&= \lambda'_{S^c} P_{S^cS} F_S^S - 1_S \rho_S
\end{aligned}
$$

□

The inequalities (6.25) appearing in Theorem 22 were first obtained by Bertsimas, Paschalidis and Tsitsiklis (1994a). The contribution of Theorem 22 is that it simplifies the formulae derived by Bertsimas, Paschalidis and Tsitsiklis. It reveals the simple structure of the right-hand side in inequality (6.25), which extends similar formulae for the single queue ($M/M/1$) case (see e.g. Gelenbe and Mitrani (1980)).

Chapter 7

Summary, conclusions and directions for future research

This thesis has developed a mathematical programming approach to optimal dynamic and stochastic resource allocation problems. The goal in such problems is to obtain an optimal resource allocation policy, as well as to compute the optimal value of the problem.

Given a resource allocation problem, the proposed approach can be articulated as follows:

1. Define suitable performance measures. The objective to be optimized must be a function of the corresponding performance vector.

2. Identify physical laws (conservation laws) satisfied by the system. These conservation laws must be expressed as linear equalities or inequalities satisfied by performance vectors.

3. Formulate the problem as a mathematical program. The feasible region of this program is a polyhedron defined by the set of linear constraints given by the conservation laws. This polyhedron contains or, in some cases, coincides with, the region spanned by performance vectors achievable under admissible resource allocation policies (performance region).

4. Translate a solution to the mathematical programming formulation into a corresponding solution to the resource allocation problem. The formulation should yield bounds

on the optimal problem value, as well as optimal or heuristic resource allocation policies.

The approach was tested in three problem areas in the field of stochastic scheduling, with an increasing level of complexity: scheduling problems solved by priority-index rules (indexable problems), the restless bandit problem, and problems of dynamic scheduling control in Markovian multiclass queueing networks.

In the first area of indexable scheduling problems, the approach yielded a unified framework for constructing and solving exact linear programming formulations. The framework was grounded on identifying a set of conservation laws that have enough power to characterize completely the performance region of such problems. Optimal scheduling policies as well as the optimal value of the problem are readily obtained from such formulations. The approach provides further new results (such as closed formulae for the optimal value of the problem and the representation of Gittins indices as sums of dual variables) and insights.

In the second area, that deals with the computationally intractable restless bandit problem, the approach yielded a sequence of increasingly stronger linear programming relaxations of the problem, that give correspondingly tighter bounds on the optimal value. These relaxations are based on exploiting certain Markov decision chain conservation laws. A primal-dual index heuristic was designed, based on the optimal solution of the first-order relaxation. Limited computational experience suggested that the heuristic seems to generate a near optimal policy.

In the third area of scheduling problems in Markovian multiclass queueing networks, we identified a set of flow conservation laws that can be translated into a sequence of increasingly stronger linear programming relaxations for the problem, with correspondingly tighter bounds on the optimal value.

Consequently, the approach has proved to be fruitful for addressing the problems investigated. In each problem area, we were able to identify a set of conservation laws that were translated into corresponding linear programming formulations.

The results obtained paralled those observed in the application of mathematical programming in the field of combinatorial optimization. In particular, they confirm the insight that our ability to solve a problem seems to be related to our ability to construct strong formulations for it. Researchers in combinatorial optimization have thus been able to construct exact formulations for well-solved problems, such as that of computing a minimum

spanning tree in a graph (see e.g. Nemhauser and Wolsey (1988)). On the other hand, only relaxed formulations have been developed for computationally hard problems, such as that of finding a minimum length hamiltonian tour in a graph.

There are several directions in which the approach developed in this dissertation can be further extended:

- Strengthening the formulations with nonlinear positive semidefinite constraints. Since the variables that appear in the formulations we presented represent moments of random variables, further nonlinear positive semidefinite constraints can be added to them. See Bertsimas, Paschalidis and Tsitsiklis (1994a).

- Heuristics from formulations. An important open problem is how to design good heuristic policies in a unifying way from the solution to the relaxed formulations we presented.

- Extensions of the approach to other dynamic and stochastic optimization problems. Finally, it would be interesting to extend the mathematical programming approach to other classes of dynamic and stochastic optimization problems, such as inventory control problems.

Appendix A

A review of the Palm calculus of point processes

In this appendix we review some basic concepts and results from Palm calculus (see e.g. Baccelli and Brémaud (1994)).

Given a system whose evolution is driven by certain events (a discrete-event system), Palm calculus is concerned with the relation between the distribution of the system state at a typical time and that at a typical event epoch. For example, in the context of a queueing system, it may be of interest to find the relation between the mean number of job in the system, when averaged over time, and when averaged over embedded job arrival epochs. This is because performance measures of queueing systems are usually defined in terms of time-averages (e.g. time-average number of jobs in the system). Moreover, certain conservation relations are naturally expressed in terms of job-averages (e.g. average number in the system seen by arriving jobs = average number in the system seen by departing jobs). Palm calculus thus provides a tool for bridging the gap between the two kinds of averages and, as a result, it is the key for translating well-known job-average conservation laws into corresponding laws that involve only time-averages. The latter laws, in turn, are the key for obtaining the linear programming formulations for queueing network scheduling problems developed in Chapter 6.

Let $Z = \{Z(t)\}$ be a continuous-time stationary stochastic process (i.e., the distribution of $Z(t)$ is the same at every time t). Let $N = \{T_n\}$ be a point process (see Baccelli and

Brémaud (1994)) in the real line. We shall normalize the point process N so that

$$\cdots < T_{-1} < T_0 = 0 < T_1 < \cdots,$$

i.e. the 0th point occurs at time $t = 0$. The number of points that lie in a subset B of the real line is denoted $N(B)$. In applications to discrete-event systems, $Z(t)$ represents the state of the system at time t, while T_n represents the epoch at which the nth event takes place. The notion of stationary point process captures mathematically the intuitive idea that the sequence of points $\{T_n\}$ corresponds to a time-homogeneous stream of events.

We assume that processes Z and N are defined in a common underlying probability space $(\Omega, \mathcal{F}, P)$. Furthermore, both processes are *adapted* to a common *history* $\{\mathcal{F}_t\}$ (which is a family of sub-σ-fields of $\mathcal{F}$ such that $\mathcal{F}_s \subseteq \mathcal{F}_t$ whenever $s < t$). That is, for any time $t \in \Re$, the random variable $Z(t)$ is $\mathcal{F}_t$-measurable, and the event $\{T_n \leq t\} \in \mathcal{F}_t$. Intuitively, $\mathcal{F}_t$ represents the information available up to and including time t. The value of $Z(t)$, and the event that the nth point T_n occurs before or at time t, can thus only depend on events that happen up to time t.

Given a real-value function $f(\cdot)$ defined on the state-space of process Z, we may consider the expected value of $f(Z)$ at a typical time, such as $t = 0$,

$$E[f(Z(0))] = E[f(Z(t))], \qquad \text{for any } t \in \Re.$$

The Palm probability distribution of process Z with respect to point process N may be described intuitively as the distribution of process Z at point/event epochs (see e.g. Baccelli and Brémaud (1994) for a formal definition). We denote $E_N[\cdot]$ the expectation operator with respect to the Palm probability associated to point process N. We may thus consider the expected value of process $f(Z)$ at a typical point/event epoch, such as $T_0 = 0$,

$$E_N[f(Z(0))] = E_N[f(Z(T_n))], \qquad \text{for any } n.$$

In both cases we assume that the corresponding expectations exist.

Intensity. A basic quantity associated with a stationary point processes is its *intensity*. It may be interpreted as a global measure of the density of points/frequency of events in

the process.

Definition 5 (Intensity) Given a stationary point process, the expected number of points that lie in a unit length interval,

$$\lambda = E\left[N\left([0,1)\right)\right],$$

is called the *intensity* of the process.

Stochastic intensity. In some applications, such as queueing systems, the frequency at which events take place may vary locally, depending on the current state of the system. For example, in an $M/M/2$ queue, events happen "more often" around a given time when the two servers are busy than when they are both idle. This intuitive notion of local density of points/frequency of events in a point process is captured by the concept of *stochastic intensity.*

Let N be a point process, not necessarily stationary, let $\{\mathcal{F}_t\}$ be a history of N, and let $\{\lambda(t)\}$ be a nonnegative process adapted to the history.

Definition 6 (Stochastic Intensity) The process $\{\lambda(t)\}$ is called a $(P, \mathcal{F}_t)$ *stochastic intensity* of N if

(i) it is locally integrable; that is, $\int_B \lambda(s)ds < \infty$ for all bounded Borel sets B; and

(ii) For all $a < b$,

$$E[N(a,b] \mid \mathcal{F}_a] = E\left[\int_a^b \lambda(s)\, ds \mid \mathcal{F}_a\right].$$

The value $\lambda(t)$ may be interpreted as the instantaneous intensity of points at time t.

Superposition of point processes. In some applications we are interested in evaluating the average value of some quantity over certain event epochs, which can be expressed as the superposition of several simpler kinds of events. Thus in a queueing system the events can be classified into external arrivals at each of queue, service completions at each server, etc. It is often easier to evaluate averages over each of the separate kinds of events than over their superposition. Fortunately, expectations with respect to the superposition can be easily expressed in terms of expectations with respect to each kind of event. This is a well-known result from the theory of point processes (see e.g. Baccelli and Brémaud (1994), and Miyazawa (1983)), that we present next.

Let $N_1, \ldots, N_K$ be stationary point proceses, defined in a common probability space. Let $\lambda_1, \ldots, \lambda_K$ be their respective intensities, which we assume to be finite and positive. Consider the point process N defined by the superposition of $N_1, \ldots, N_K$: Process N has a point at time t if any of the processes $N_1, \ldots, N_K$ has a point at that time. Let us denote $N = N_1 + \cdots + N_K$. The intensity of N can be shown to be $\lambda = \lambda_1 + \cdots + \lambda_K$. Let $Z = \{Z(t)\}$ be a stationary process, and let $f(\cdot)$ be a function on the state space of Z such that the expectation $E_N[f(Z(0))]$ exists.

Theorem 23 (Superposition of point processes) *The following relation holds:*

$$E_N[f(Z(0))] = \sum_{k=1}^{K} \frac{\lambda_k}{\lambda} E_{N_k}[f(Z(0))].$$

Papangelou's formula It is often of interest to find relations between time and event averages of a given process. As we mentioned above, some conservation laws of queueing systems are naturally expressed in terms of job averages. Such relations may be useful for translating those conservation laws into laws involving only time averages, which are more often used as performance measures.

Under certain assumptions, there is an easy relation between time and event averages. For example, a classical result of queueing theory, known as PASTA (see Wolff (1982), asserts that "Poisson arrivals see time averages." This fact, and extensions of it such as ESTA (Events See Time Averages) (see e.g. Walrand (1988) and conditional PASTA (see König and Schmidt (1990)) correspond to special cases of Papangelou's formula.

Papangelou's formula (see Papangelou 1972 and Baccelli and Brémaud (1994)) represents the link between stationary probability, Palm probability and stochastic intensity. Let N is a stationary point process with intensity $0 < \lambda < \infty$, adapted to the history $\{\mathcal{F}_t\}$, $\{Z(t)\}$ is a process also adapted to $\{\mathcal{F}_t\}$.

Theorem 24 (Papangelou (1972)) *If N admits a $(P, \mathcal{F}_t)$ stochastic intensity $\{\lambda(t)\}$, and $Z(0)$ is $\mathcal{F}_{0-}$-measurable, then*

$$E[f(Z(0))\lambda(0)] = \lambda E_N[f(Z(0))].$$

In words, Papangelou's theorem asserts that, under certain conditions, the value of function $f(Z(t))$ times the stochastic intensity $\lambda(t)$ observed at a typical time coincides

with the process intensity λ times the value of process $f(Z(t))$ observed at a typical event epoch.

Bibliography

[1] Baccelli, F. and Brémaud, P. (1994). *Elements of Queueing Theory: Palm-Martingale Calculus and Stochastic Recurrences.* Springer-Verlag, Berlin.

[2] Bazaraa, M. S. and Shetty, C. M. (1979). *Nonlinear Programming: Theory and Algorithms.* Wiley, New York.

[3] Bell, C. (1971). Characterization and computation of optimal policies for operating an $M/G/1$ queue with removable server. *Oper. Res.* **19** 208-218.

[4] Bellman, R. (1957). *Dynamic Programming.* Princeton University Press, Princeton, NJ.

[5] Berry, D. A. and Fristedt, B. (1985). *Bandit Problems.* Chapman and Hall, London.

[6] Bertsekas, D. P. (1987). *Dynamic Programming: Deterministic and Stochastic Models.* Prentice-Hall, Englewood Cliffs, NJ.

[7] Bertsimas, D. (1994). A mathematical programming approach to stochastic and dynamic optimization problems. Working paper, Operations Research Center, MIT.

[8] Bertsimas, D. and Niño-Mora, J. (1993a). Conservation laws, extended polymatroids and multi-armed bandit problems; a unified approach to indexable systems. Working paper OR 277-93, Operations Research Center, MIT. To appear in *Math. Oper. Res.*

[9] Bertsimas, D. and Niño-Mora, J. (1993b). Conservation laws, extended polymatroids and multi-armed bandit problems; a unified approach to indexable systems. In *Proceedings of Third International Conference on Integer Programming and Combinatorial Optimization*, G. Rinaldi and L. Wolsey, eds., Mathematical Programming Society, Erice, Italy, 355-384.

[10] Bertsimas, D. and Niño-Mora, J. (1994). Restless bandits, linear programming relaxations and a primal-dual heuristic. Working paper 3727-94 MSA, Sloan School of Management, MIT.

[11] Bertsimas, D., Paschalidis, I. and Tsitsiklis, J. (1994a). Optimization of multiclass queueing networks: Polyhedral and nonlinear characterizations of achievable performance. *Ann. Appl. Probab.* **4** 43-75.

[12] Bertsimas, D., Paschalidis, I. and Tsitsiklis, J. (1994b). Branching bandits and Klimov's problem: Achievable region and side constraints. Working paper, Operations Research Center, MIT.

[13] Bertsimas D. and Teo, C. (1994). From valid inequalities to heuristics: A unified view of primal-dual approximation algorithms in covering problems. Working paper, Operations Research Center, MIT.

[14] Bhattacharya, P. P., Georgiadis, L. and Tsoucas, P. (1991). Problems of adaptive optimization in multiclass $M/GI/1$ queues with Bernoulli feedback. Paper presented in part at the ORSA/TIMS *Conference on Applied Probability in the Engineering, Information and Natural Sciences*, January 9-11, 1991, Monterey, California.

[15] Bhattacharya, P. P., Georgiadis, L. and Tsoucas, P. (1992). Extended polymatroids: Properties and optimization. In *Proceedings of Second International Conference on Integer Programming and Combinatorial Optimization*, Mathematical Programming Society, Carnegie Mellon University, 298-315.

[16] Blazewicz, J., Dror, M. and Weglarz, J. (1991). Mathematical programming formulations for machine scheduling: A survey. *European J. Oper. Res.* **51** 283-300.

[17] Brandt, A., Franken, P. and Lisek, B. (1990). *Stationary Stochastic Models.* Wiley, New York.

[18] Burke, P. J. (1956). The output of a queueing system. *Oper. Res.* **4** 699-704.

[19] Caulkins, J. P. (1993). Local drug market's response to focused police enforcement. *Oper. Res.* **41** 848-863.

[20] Chen, Y. R. and Katehakis, M. N. (1986). Linear programming for finite state multi-armed bandit problems. *Math. Oper. Res.* **11** 180-183.

[21] Chevalier, P. B. and Wein, L. M. (1993). Scheduling networks of queues: Heavy traffic analysis of a multi-station closed network. *Oper. Res.* **41** 743-757.

[22] Cobham, A. (1954). Priority assignment in waiting line problems. *Oper. Res.* **2** 70-76.

[23] Coffman, E. G., Jr., Hofri, M. and Weiss, G. (1989). Scheduling stochastic jobs with a two point distribution on two parallel machines. *Probab. Engrg. Inform. Sci.* **3** 89-116.

[24] Coffman, E. G., Jr. and Mitrani, I. (1980). A characterization of waiting time performance realizable by single server queues. *Oper. Res.* **28** 810-821.

[25] Cox, D. R. and Smith, W. L. (1961). *Queues.* Methuen, London.

[26] d'Epenoux, F. (1960). Sur un problème de production et de stockage dans l'aléatoire. *RAIRO Rech. Opér.* **14** 3-16.

[27] Dunstan, F. D. J. and Welsh, D. J. A. (1973). A greedy algorithm for solving a certain class of linear programmes. *Math. Programming* **5** 338-353.

[28] Edmonds, J. (1970). Submodular functions, matroids and certain polyhedra. In *Proceedings of Calgary International Conference on Combinatorial Structures and Their Aplications*, R. Guy, H. Hanani, N. Sauer and J. Schönheim, eds., Gordon and Breach, New York, 69-87.

[29] Federgruen, A. and Groenevelt, H. (1988a). Characterization and optimization of achievable performance in general queueing systems. *Oper. Res.* **36** 733-741.

[30] Federgruen, A. and Groenevelt, H. (1988b). $M/G/c$ queueing systems with multiple customer classes: Characterization and control of achievable performance under non-preemptive priority rules. *Management Sci.* **34** 1121-1138.

[31] Feller, W. (1966). *An Introduction to Probability Theory and its Applications, Vol. 2.* Wiley, New York.

[32] Finch, P. D. (1959). On the distribution of queue size in queueing problems. *Acta Math. Hungar.* **10** 327-336.

[33] Franken, P., König, D., Arndt, U. and Schmidt, V. (1982). *Queues and Point Processes.* Wiley, Chichester.

[34] Fujishige, S. (1991). Submodular Functions and Optimization. *Ann. Discrete Math.* **41**. North-Holland, Amsterdam.

[35] Gelenbe, E. and Mitrani, I. (1980). *Analysis and Synthesis of Computer Systems.* Academic Press, London.

[36] Gelenbe, E. and Pujolle, G. (1987). *Introduction to Queueing Networks.* Wiley, Chichester.

[37] Gittins, J. C. and Jones, D. M. (1974). A dynamic allocation index for the sequential design of experiments. In *Progress in Statistics: European Meeting of Statisticians, Budapest, 1972*, J. Gani, K. Sarkadi and I. Vince, eds., North-Holland, Amsterdam, 241-266.

[38] Gittins, J. C. (1979). Bandit processes and dynamic allocation indices. *J. Roy. Statist. Soc. Ser. B* **14** 148-177.

[39] Gittins, J. C.(1989). *Multi-armed Bandit Allocation Indices.* Wiley, Chichester.

[40] Glazebrook, K. D. (1976). Stochastic scheduling with order constraints. *Internat. J. Systems Sci.* **7** 657-666.

[41] Glazebrook, K. D. (1987). Sensitivity analysis for stochastic scheduling problems. *Math. Oper. Res.* **12** 205-223.

[42] Glazebrook, K. D. (1994). Reflections on a new approach to Gittins indexation. Working paper, Dept. of Mathematics and Statistics, Newcastle University, UK.

[43] Gordon, W. I. and Newell, G. F. (1967). Closed queueing systems with exponential servers. *Oper. Res.* **15** 154-165.

[44] Grötschel, M., Lovász, L. and Schrijver, A. (1988). *Geometric Algorithms and Combinatorial Optimization*, Springer-Verlag, Berlin.

[45] Harrison, J. M. (1975a). A priority queue with discounted linear costs. *Oper. Res.* **23** 260-269.

[46] Harrison, J. M. (1975b). Dynamic scheduling of a multiclass queue: Discount optimality. *Oper. Res.* **23** 270-282.

[47] Harrison, J. M. and Wein, L. M. (1990). Scheduling networks of queues: Heavy traffic analysis of a two-station closed network. *Oper. Res.* **38** 1052-1064.

[48] Hebuterne, G. (1988). Relation between states observed by arriving and departing customers in bulk systems. *Stochastic Process. Appl.* **27** 279-288.

[49] Heyman, D. P. and Sobel, M. J. (1984). *Stochastic Models in Operations Research, vol. II: Stochastic Optimization.* McGraw-Hill, New York.

[50] Horn, W. A. (1972). Single-machine job sequencing with treelike precedence ordering and linear delay penalties. *SIAM J. Appl. Math.* **23** 189-202.

[51] Ibaraki, T. and Katoh, N. (1988). *Resource Allocation Problems: Algorithmic Approaches.* MIT Press, Cambridge, MA.

[52] Ishikida and Varaiya (1994). The multi-armed bandit problem revisited. *J. Optim. Theory Appl.* **83** 113-

[53] Jackson, J. R. (1963). Jobshop-like queueing systems. *Management Sci.* **10** 131-142.

[54] Jackson, R. R. P. (1954). Queueing systems with phase-type service. *Oper. Res. Quart.* **5** 109-120.

[55] Katehakis, M. N. and Veinott, A. F. (1987). The multiarmed bandit problem: Decomposition and computation. *Math. Oper. Res.* **12** 262-268.

[56] Kelly, F. P. (1979). *Reversibility and Stochastic Networks*, Wiley, New York.

[57] König, D. and Schmidt, V. (1990). Extended and conditional versions of the PASTA property. *Adv. Appl. Probab.* **22** 510-512.

[58] Kleinrock, L. (1976). *Queueing Systems, Vol. II: Computer Applications.* Wiley, New York.

[59] Klimov, G. P. (1974). Time sharing service systems I. *Theory Probab. Appl.* **19** 532-551.

[60] Kumar, S. and Kumar, P. R. (1994). Performance bounds for queueing networks and scheduling policies. *IEEE Trans. Autom. Control* **39** 1600-1611.

[61] Lai, T. L. and Ying, Z. (1988). Open bandit processes and optimal scheduling of queueing networks. *Adv. Appl. Probab.* **20** 447-472.

[62] Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G. and Shmoys, D. B. (1989). Sequencing and scheduling; algorithms and complexity. Report BS-R8909, Centre for Mathematics and Computer Science, Amsterdam.

[63] Lovász, L. and Schrijver, A. (1991). Cones of matrices and set-functions and $0-1$ optimization. *SIAM J. Optim.* **1** 166-190.

[64] Makowski, A. M. and Shwartz, A. (1993). On constrained optimization of the Klimov network and related Markov decision processes. *IEEE Trans. Automat. Control* **38 (2)** 354-359.

[65] Manne, A. (1960). Linear programming and sequential decisions. *Management Sci.* **6** 259-267.

[66] Miyazawa, M. (1983). The derivation of invariance relations in complex queueing systems with stationary inputs. *Adv. Appl. Probab.* **15** 874-885.

[67] Meilijson, I. and Weiss, G. (1977). Multiple feedback at a single-server station. *Stochastic Process. Appl.* **5** 195-205.

[68] Mitrani, I. (1987). *Modelling of Computer and Communication Systems.* Cambridge University Press, Cambridge, UK.

[69] Murty, K. G. (1983). *Linear Programming.* Wiley, New York.

[70] Nain, P. and Ross, K. W. (1986). Optimal priority assignment with hard constraints. *IEEE Trans. Automat. Control* **31** 883-888.

[71] Nain, P., Tsoucas, P. and Walrand, J. (1989). Interchange arguments in stochastic scheduling. *J. Appl. Probab.* **27** 815-826.

[72] Nemhauser, G. L. and Wolsey, L. A. (1988). *Integer and Combinatorial Optimization.* Wiley, New York.

[73] Nesterov, Y. and Nevirovskii, A. (1994). *Interior-Point Polynomial Algorithms in Convex Programming.* SIAM, Philadelphia.

[74] Ou, J. and Wein, L. M. (1992). Performance bounds for scheduling queueing networks. *Ann. Appl. Probab.* **2** 460-480.

[75] Palm, C. (1943). Intensitätsschwankungen im Fernsprechverkehr. *Ericsson Techniks* **44** 1-189.

[76] Papaconstantinou, X. and Bertsimas, D. (1990). Relations between the prearrival and postdeparture state probabilities and the FCFS waiting time distribution in the $E_k/G/s$ queue. *Naval Res. Logist. Quart.* **37** 135-149.

[77] Papadimitriou, C. H. and Tsitsiklis, J. N. (1993). The complexity of optimal queueing network control. Working Paper, MIT.

[78] Papadimitriou, C. H. (1994). *Computational Complexity.* Addison-Wesley, Reading, MA.

[79] Papangelou, F. (1972). Integrability of expected increments and a related random change of time scale. *Trans. Amer. Math. Soc.* **165** 483-506.

[80] Pinedo, M. (1983). Stochastic scheduling with release dates and due dates. *Oper. Res.* **31** 559-572.

[81] Queyranne, M. (1993). Structure of a simple scheduling polyhedron. *Math. Programming* **58** 263-285.

[82] Queyranne, M. and Wang. Y. (1991). Single-machine scheduling polyhedra with precedence constraints. *Math. Oper. Res.* **16** 1-20.

[83] Ross, K. W. and Yao, D. D. (1989). Optimal dynamic scheduling in Jackson networks. *IEEE Trans. Automat. Control* **34** 47-53.

[84] Rothkopf, M. H. (1966a). Scheduling independent tasks on parallel processors. *Management Sci.* **12** 437-447.

[85] Rothkopf, M. H. (1966b). Scheduling with random service times. *Management Sci.* **12** 707-713.

[86] Schrijver, A. (1986). *Theory of Linear and Integer Programming.* Wiley, Chichester.

[87] Sevcik, K. C. (1974). A proof of the optimality of smallest rank scheduling. *J. Assoc. Comput. Mach.* **21**

[88] Shanthikumar, J. G. and Yao, D. D. (1992). Multiclass queueing systems: Polymatroidal structure and optimal scheduling control. *Oper. Res.* **40** S293-299.

[89] Shapley, L. S. (1971). Cores of convex games. *Internat. J. Game Theory* **1** 11-26.

[90] Smith, W. E. (1956). Various optimizers for single-stage production. *Naval Res. Logist. Quart.* **3** 59-66.

[91] Stidham, S., Jr. (1972). $L = \lambda W$: A discounted analog and a new proof. *Oper. Res.* **20** 1115-1126.

[92] Tcha, D. W. and Pliska, S. R. (1977). Optimal control of single-server queueing networks and multi-class $M/G/1$ queues with feedback. *Oper. Res.* **25** 248-258.

[93] Tsitsiklis, J. N. (1986). A lemma on the multi-armed bandit problem. *IEEE Trans. Automat. Control* **31** 576-577.

[94] Tsitsiklis, J. N. (1993). A short proof of the Gittins index theorem. *Ann. Appl. Probab.* **4** 194-199.

[95] Tsoucas, P. (1991). The region of achievable performance in a model of Klimov. Research Report RC16543, IBM T. J. Watson Research Center, Yorktown Heights, NY.

[96] Varaiya, P. P., Walrand, J. C. and Buyukkoc, C. (1985). Extensions of the multiarmed bandit problem: The discounted case. *IEEE Trans. Automat. Control* **30** 426-439.

[97] Veatch, M. and Wein, L. M. (1992). Scheduling a make-to-stock queue: Index policies and hedging points. Working paper OR 266-92, Operations Research Center, MIT.

[98] Walrand, J. (1988). *An Introduction to Queueing Networks.* Prentice Hall, Englewood Cliffs, NJ.

[99] Weber, R. (1992). On the Gittins index for multiarmed bandits. *Ann. Appl. Probab.* **2** 1024-1033.

[100] Weber, R. R. and Weiss, G. (1990). On an index policy for restless bandits. . *J. Appl. Probab.* **27** 637-648.

[101] Weber, R. R. and Weiss, G. (1991). Addendum to "On an index policy for restless bandits." *Adv. in Appl. Probab.* **23** 429-430.

[102] Weiss, G. (1988). Branching bandit processes. *Probab. Engrg. Inform. Sci.* **2** 269-278.

[103] Whittle, P. (1980). Multi-armed bandits and the Gittins index. *J. Roy. Statist. Soc. Ser. B* **42** 143-149.

[104] Whittle, P. (1981). Arm acquiring bandits. *Ann. Probab.* **9** 284-292.

[105] Whittle, P. (1982). *Optimization Over Time: Dynamic Programming and Stochastic Control, Vols. I, II.* Wiley, Chichester.

[106] Whittle, P. (1988). Restless bandits: Activity allocation in a changing world. In *A Celebration of Applied Probability*, J. Gani, ed., *J. Appl. Probab.* **25A** 287-298.

[107] Wolff, R. W. (1982). Poisson Arrivals See Time Averages. *Oper. Res.* **30** 223-231.

[108] Wolff, R. W. (1989). *Stochastic Modeling and the Theory of Queues.* Prentice-Hall, Englewood Cliffs, NJ.

[109] Wortman, M. A. and Disney, R. L. (1992). On the relationship between stationary and Palm moments of backlog in the $G/G/1$ priority queue. In *Queueing and Related Models*, U. N. Bhat and I. V. Basawa, eds., Oxford University Press, 161-174.

[110] Yao, D. D. and Shanthikumar, J. G. (1990). Optimal scheduling control of a flexible machine. *IEEE Trans. Robotics and Automation* **6** 706-712.