Integer Programming: Optimization and Evaluation Are Equivalent

Integer Programming: Optimization and Evaluation are Equivalent

James B. Orlin[1], Abraham P. Punnen[2], and Andreas S. Schulz[1]

[1] Massachusetts Institute of Technology, Cambridge, MA
[2] Simon Fraser University, Surrey, BC

Abstract We show that if one can find the optimal value of an integer linear programming problem in polynomial time, then one can find an optimal solution in polynomial time. We also present a proper generalization to (general) integer programs and to local search problems of the well-known result that optimization and augmentation are equivalent for 0/1-integer programs. Among other things, our results imply that PLS-complete problems cannot have "near-exact" neighborhoods, unless PLS = P.

1 Introduction

The following question arises naturally in the study of optimization problems (see, e.g., [12, Chap. 15.2]): Is computing the value of an optimal solution as hard as actually finding an optimal solution? Crescenzi and Silvestri [4] initiated the formal study of the relative complexity of evaluating the optimal cost of an optimization problem versus constructing an optimal solution, and they provided sufficient and necessary conditions for the existence of optimization problems for which obtaining an optimal solution is harder than computing the optimal value. Ausiello et al. [2] and Johnson [8] pointed out that evaluation is actually as hard as finding an optimal solution for all optimization problems whose associated decision problems are NP-complete. Schulz [16] studied the relative complexity of several problems related to 0/1-integer programming,[3] including augmentation, optimization and evaluation, all of which are polynomial-time equivalent. In this paper, we prove that evaluation and optimization are polynomial-time equivalent for all integer linear programming problems. That is, given a matrix $A \in \mathbb{Z}^{m \times n}$ and a vector $b \in \mathbb{Z}^m$, a polynomial-time algorithm for finding the optimal value of $\min\{cx : Ax \geq b, x \in \mathbb{Z}^n_+\}$, for any $c \in \mathbb{Z}^n$, implies the existence of such an algorithm for finding an optimal solution, $\arg\min\{cx : Ax \geq b, x \in \mathbb{Z}^n_+\}$. In fact, our result is slightly stronger than this. As long as we are given bounds on the values that individual variables may attain, the matrix A and the vector b need not be known explicitly. An evaluation oracle, which accepts as input any objective function vector c and returns the optimal objective function value, suffices. Our proof is constructive.

[3] In a 0/1-integer programming problem all variables can have values 0 or 1 only.

The proof itself gives rise to a new problem, related to questions typically brought up in postoptimality analysis of optimization problems, which we call the "unit increment problem:" Given an optimal solution x^0 with respect to an objective function vector c, find an optimal solution for $c + e_j$, where e_j is the j-th unit vector.[4] We show that an integer linear program can be solved in polynomial time if and only if its unit increment problem can be solved in polynomial time. For 0/1-integer programs, we prove that the unit increment problem is polynomial-time equivalent to the augmentation problem.[5] Hence, we have a proper generalization (to general integer programs) of a result by Grötschel and Lovász [6] and Schulz et al. [17], who showed that optimization and augmentation are polynomial-time equivalent for 0/1-integer programs. A relaxation to the augmentation problem, the ϵ-augmentation problem, can be defined as follows: Given an objective function vector c and a feasible solution x, find a feasible solution with better objective function value, or assert that x is ϵ-optimal. Here, $\epsilon > 0$, and a solution x is ϵ-optimal if $cx \leq (1 + \epsilon)cx'$ for all feasible solutions x'. The corresponding unit increment problem, the ϵ-unit increment problem, is defined as follows: Given an index j and an ϵ-optimal solution with respect to an objective function vector c, find an ϵ-optimal solution for $c+e_j$. We show that an ϵ-optimal solution can be obtained in polynomial time if and only if the ϵ-unit increment problem can be solved in polynomial time. Moreover, we show that for 0-1 integer programs, the ϵ-augmentation problem and the ϵ-unit increment problem are polynomial-time equivalent as well.

The concepts of unit increment and augmentation extend naturally to local search, with interesting implications. For an integer programming or combinatorial optimization problem with a neighborhood function N, the local augmentation problem, given a feasible solution x and an objective function vector c, asks for a solution in the neighborhood $N(x)$ of x of better objective function value, if one exists. The local unit increment problem is defined similarly: Given an index j and a locally optimal solution x with respect to c, find a locally optimal solution for $c + e_j$. We show that for a given neighborhood function, a locally optimal solution can be computed in polynomial time if and only if the local unit increment problem can be solved in polynomial time. However, in contrast to the cases of global optimization and ϵ-optimization, for 0/1-integer programs, the local unit increment problem and the local augmentation problem are not known to be equivalent. In fact, it follows from our results that if polynomial solvability of the local augmentation problem implies the polynomial solvability

[4] In this part of the paper we assume, for convenience, that all objective function coefficients are nonnegative. Most of our results hold true in general, if the unit increment problem is extended to finding optimal solutions for $c \pm e_j$.

[5] The augmentation problem is defined as follows: Given a feasible solution and an objective function vector, find a feasible solution of better objective function value, if one exists.

of the local unit increment problem, then all PLS-complete[6] problems can be solved in polynomial time.

A neighborhood function is said to be "exact" if every locally optimal solution is guaranteed to be globally optimal. A neighborhood function is said to be "near exact" if the objective function value of any locally optimal solution is no worse than that of all but a polynomial number of feasible solutions. Near-exact neighborhoods are related to the domination number of local search heuristics [7]. We show that, for 0/1-integer programs, polynomial solvability of the local augmentation problem implies polynomial solvability of the local optimization problem whenever the corresponding neighborhood is near exact. This implies that no PLS-complete problem can possess a near-exact neighborhood, unless PLS = P.

The rest of the paper is organized as follows. In Sect. 2 we establish that optimization and evaluation are polynomial-time equivalent for integer programming problems. In Sect. 3, we show that the unit increment problem and the optimization problem are polynomial-time equivalent. We also give a direct proof that, for 0/1-integer programming problems, augmentation and unit increment are polynomial-time equivalent. In Sect. 4 we extend these results to ϵ-optimization, ϵ-augmentation, and ϵ-unit increment. Section 5 contains our results on local search; in particular, we show that even for 0/1-integer programs, a local unit increment oracle is stronger than a local augmentation oracle, unless PLS = P.

2 Evaluation versus Optimization

It is well known (see, e.g. [15, Chap. 17.1]) that if an integer program has a finite optimum, then it has an optimal solution of size (i.e., encoding length) polynomially bounded by the size of the input. Hence, instead of considering $\min\{cx : Ax \geq b, x \in \mathbb{Z}^n_+\}$, we may restrict ourselves to solving $\min\{cx : Ax \geq b, x \leq u, x \in \mathbb{Z}^n_+\}$, for a vector $u \in \mathbb{Z}^n_+$ whose encoding length is polynomial in that of A, b, and c. From now on, we therefore consider a family $\mathcal{F}$ of integer programming problems that is described as follows. For each instance of the family we are given a vector $u \in \mathbb{Z}^n_+$ such that the set $X \subseteq \mathbb{Z}^n$ of feasible solutions is contained in $\{0, 1, \ldots, u_1\} \times \{0, 1, \ldots, u_2\} \times \cdots \times \{0, 1, \ldots, u_n\}$. We are also given an evaluation oracle that contains the only additional information that we have on X.[7] (In particular, we do not explicitly need to know a matrix A and a vector b such that $X = \{x \in \mathbb{Z}^n_+ : Ax \geq b, x \leq u\}$.) For input vector $c \in \mathbb{Z}^n$,

[6] The complexity classes PLS and PLS-complete were introduced by Johnson et al. [9] to capture the difficulty of finding local optima. Prominent PLS-complete problems include the max-cut problem with the flip neighborhood and the graph partitioning problem with the swap neighborhood [14], the traveling salesman problem with the k-exchange neighborhood (for sufficiently large, but constant k) [10], and the problem of finding pure-strategy Nash equilibria in congestion games [5].

[7] We may assume, without loss of generality, that there exists a feasible solution, i.e., $X \neq \emptyset$. Otherwise both oracles, evaluation and optimization, would have to detect infeasibility.

the oracle returns the optimal objective function value of $\min\{cx : x \in X\}$. The following is our main result.

Theorem 1. *Given a family $\mathcal{F}$ of integer programming problems described by an evaluation oracle, there is an oracle-polynomial time algorithm for solving the optimization problem.*

Proof. Let $\min\{cx : x \in X\}$ be the optimization problem to be solved, given by an evaluation oracle and a vector $u \in \mathbb{Z}^n_+$ such that $X \subseteq \{0 \leq x \leq u\}$. The main idea is as follows. Among all optimal solutions, let x' be the one that is lexicographically minimal. We perturb c in such a way that x' remains optimal for the perturbed vector c' and is also optimal for the objective function vectors $c' + e_j$, for all $j = 1, 2, \ldots, n$. With $n + 1$ calls of the evaluation oracle we can then recover x' via $x'_j = (c' + e_j)x' - c'x'$, for $j = 1, 2, \ldots, n$. If the size of c' is sufficiently small, this yields an oracle-polynomial time algorithm.

Here are the details. Let $U := \max\{u_j : j = 1, 2, \ldots, n\} + 1$. We define c' as follows:

$$c'_j := U^{2n+1}c_j + U^{2(n-j)+1}, \text{ for } j = 1, 2, \ldots, n.$$

Note that the encoding length of c' is indeed polynomial in that of c and u. We first show that, (i), every solution x^* that is optimal for c' is also optimal for c. In fact, for any $x \in X$, we obtain that

$$cx^* \leq \frac{c'x^*}{U^{2n+1}} \leq \frac{c'x}{U^{2n+1}} = cx + \frac{\sum_{j=1}^n U^{2(n-j)+1}x_j}{U^{2n+1}} < cx + 1.$$

Together with the integrality of c, x^*, and x, this implies that $cx^* \leq cx$, proving (i). We now show that, (ii), if x is an optimal solution for c that is different from x', then $c'(x' - x) \leq -U$. Let i be the first index for which $x'_i < x_i$. Then,

$$\begin{aligned} c'(x' - x) &= U^{2(n-i)+1}(x'_i - x_i) + \sum_{j=i+1}^{n} U^{2(n-j)+1}(x'_j - x_j) \\ &\leq -U^{2(n-i)+1} + \sum_{j=i+1}^{n} U^{2(n-j)+2} \\ &\leq -U. \end{aligned}$$

It remains to show that x' is optimal for $c' + e_j$, for an arbitrary, but fixed index $j \in \{1, 2, \ldots, n\}$. So, let x be some feasible solution different from x'. We distinguish two cases. If x is optimal for c, then, with the help of (ii), we get

$$(c' + e_j)(x' - x) = c'(x' - x) + (x'_j - x_j) \leq -U + U = 0.$$

If x is not optimal for c, we have

$$\begin{aligned} c'(x' - x) &= U^{2n+1}c(x' - x) + \sum_{j=1}^{n} U^{2(n-j)+1}(x'_j - x_j) \\ &\leq -U^{2n+1} + \sum_{j=1}^{n} U^{2(n-j)+2} \leq -U. \end{aligned}$$

Hence, $(c' + e_j)(x' - x) \leq 0$. Thus, x' is optimal for c' and $c' + e_j$, and $x'_j = (c' + e_j)x' - c'x'$. In particular, the j-th component of x' can be computed by two calls of the evaluation oracle. □

3 The Unit Increment Problem and Global Optimization

In this section we assume that all objective function vectors are nonnegative, for convenience. All results can be extended in a straightforward way to arbitrary objective function vectors. If c is an objective function vector, let $C := \max\{c_j : j = 1, 2, \ldots, n\}$ and $\alpha := 1 + \lceil \log_2 C \rceil$. Then each c_j can be represented as a binary number using α bits. Let $b^j = (b^j_1, b^j_2, \ldots, b^j_\alpha)$ with $b^j_i \in \{0, 1\}$ be this representation. Moreover, let c^k_j be the number represented by the k leading bits of b^j. That is, $c^k_j := \sum_{i=1}^{k} 2^{k-i} b^j_i$. Thus $c^1_j \in \{0, 1\}$, $c^\alpha_j = c_j$, and $c^{k+1}_j = 2c^k_j + b^j_{k+1}$ for all $k = 1, 2, \ldots, \alpha - 1$ and $j = 1, 2, \ldots, n$.

Let $\min\{cx : x \in X\}$ with $X \subseteq \mathbb{Z}^n_+$ be an instance of the optimization problem. We assume that an oracle UNIT-INC is available which with input j, $c+e_j$, and an optimal solution x^0 with respect to c, computes an optimal solution x^* for $\min\{(c + e_j)x : x \in X\}$. We consider the following algorithm.

Algorithm UI

begin
  let x^0 be any feasible solution
  set $c^*_j := 0$ for $j = 1$ to n
  for $k = 1$ to α **do**
    for $j = 1$ to n **do** $c^*_j := 2c^*_j$
    $S := \{j \ : \ b^j_k = 1\}$
    while $S \neq \emptyset$ **do**
      choose $j \in S$
      $S := S \setminus \{j\}$
      $c^*_j := c^*_j + 1$
      If $x^0_j > 0$ **then**
        call UNIT-INC(c^*, x^0, x^*, j)
        $x^0 := x^*$
      endif
    endwhile
  endfor
  output x^0
end

Theorem 2. *Let a family of optimization problems with linear objective functions be given by a unit-increment oracle. Then algorithm UI computes an optimal solution in oracle-polynomial time.*

Proof. Note that at the end of the k-th iteration of the main loop we have $c^* = c^k$. Assume that at the beginning of the k-th iteration of the main loop, x^0

is an optimal solution to $\min\{c^{k-1}x : x \in X\}$. (For convenience, we let $c^0 = 0$.) Then x^0 continues to be an optimal solution if we change the objective function to $2c^{k-1}$. Thus at the end of the while loop, the oracle UNIT-INC guarantees that x^0 is an optimal solution to $\min\{c^k x : x \in X\}$. The correctness of the algorithm follows by induction over k. □

Hence, if one can find a feasible solution in polynomial time and if one can solve the unit increment problem in polynomial time, then one can determine an optimal solution in polynomial time. For the assignment problem on a bipartite graph on n nodes and m edges, the general algorithm described above terminates in $O(nm \log C)$ time. This is because the unit increment problem for this special case can be solved in $O(m)$ time. Although there are special-purpose algorithms with better worst case bounds to solve the assignment problem, it is interesting to note that the general algorithm UI achieves a good time bound.

Algorithm UI is a generalization of the bit scaling algorithm studied extensively in the network flow literature (see, e.g., [1]) and in the context of 0/1-integer programming (see, e.g., [16]). The new feature here is the use of the unit increment oracle. This allows us to compare the computational complexity of optimization problems and unit increment problems and also provides a framework for our study of ϵ-optimization and local optimization. Theorem 2 establishes that an optimization problem with linear objective function can be solved in polynomial time if and only if the corresponding unit increment problem can be solved in polynomial time. Alternatively, if the optimization problem is NP-hard, then the corresponding unit increment problem is also NP-hard. Thus the additional information available for the unit increment problem (i.e., an optimal solution for the original objective function) is not of much help for NP-hard problems. This provides additional evidence that postoptimality analysis is typically hard for NP-hard problems (see, e.g., [3,13,18] for related results).

We now examine the relationship between the unit increment problem and the augmentation problem.

Lemma 3. *Let x^0 be an optimal solution to $\min\{cx : x \in X\}$, and let j be a given index, $1 \le j \le n$. If $x \in X$ is a feasible solution, then $(c+e_j)(x^0 - x) \le x_j^0$.*

Proof. Since x is a feasible solution to $\min\{cx : x \in X\}$, $cx^0 \le cx$. Thus $(c + e_j)x^0 = cx^0 + x_j^0 \le cx + x_j^0 \le (c + e_j)x + x_j^0$. □

The next theorem is, in principle, a consequence of the before-mentioned equivalence between augmentation and optimization for 0/1-integer programs, and Theorem 2. However, the following proof provides a Karp reduction from the unit increment problem to the augmentation problem.

Theorem 4. *For 0/1-integer programs, the unit increment problem and the augmentation problem are polynomial-time equivalent.*

Proof. Assume that $X \subseteq \{0,1\}^n$ is given by an augmentation oracle. Consider the instance $\min\{cx : x \in X\}$ and its unit increment version $\min\{(c+e_j)x : x \in$

$X\}$ together with respective optimal solutions x^0 (given) and x^* (unknown). By Lemma 3,

$$(c + e_j)(x^0 - x^*) \leq 1. \tag{1}$$

Since $c \in \mathbb{Z}^n$, one application of the augmentation oracle starting with x^0 either declares that x^0 is optimal for $\min\{(c + e_j)x : x \in X\}$ or finds an improving solution which must be optimal for $\min\{(c + e_j)x : x \in X\}$ in view of (1). The other direction is implied by Theorem 2. □

Using Lemma 3 we have the following result for general integer programs. Let $\min\{cx : x \in X\}$ be an instance and x^0 be an optimal solution. Let $\min\{(c+e_j)x : x \in X\}$ be the corresponding j-th unit increment instance.

Theorem 5. *Given $x^0 \in \arg\min\{cx : x \in X\}$ and an augmentation oracle, $\min\{(c + e_j)x : x \in X\}$ can be solved by $O(x^0_j)$ calls of the augmentation oracle.*

Theorems 2 and 5 show that for an integer linear program for which $x \leq u$ for all $x \in X$ and where the components of u are bounded above by a polynomial of the remaining input data, the optimization problem can be solved in polynomial time whenever the augmentation problem can be solved in polynomial time.

4 The Unit Increment Problem and ϵ-Optimization

In this section we explore the complexity of finding near-optimal solutions if ϵ-augmentation or ϵ-unit increment oracles are available. We need the assumption that $c_j \geq 0$ for all $j = 1, 2, \ldots, n$. We also fix $\epsilon > 0$. Let UI(ϵ) denote the variation of the algorithm UI where the oracle Unit-Inc is replaced by ϵ-Unit-Inc which takes as input $c + e_j$, an ϵ-optimal solution x^0 of $\min\{cx : x \in X\}$, an index j, and computes an ϵ-optimal solution x^* of $\min\{(c + e_j)x : x \in X\}$. Using arguments similar to that in the proof of Theorem 2 one can show the following result.

Theorem 6. *Given an ϵ-unit increment oracle and an initial feasible solution, algorithm UI(ϵ) computes an ϵ-optimal solution to $\min\{cx : x \in X\}$ in oracle-polynomial time.*

Thus if a feasible solution can be computed in polynomial time and the ϵ-unit increment problem can be solved in polynomial time, then an ϵ-optimal solution can be obtained in polynomial time. Alternatively, an optimization problem is not approximable if and only if the corresponding unit increment problem is not approximable. This result is interesting in several ways. For example, even if we have an ϵ-optimal solution to the traveling salesman problem, if one of the edge weights is increased by one, then getting an ϵ-optimal solution is still NP-hard. Also, there exists a (fully) polynomial-time approximation scheme for an optimization problem if and only if there is a (fully) polynomial-time approximation scheme for the unit increment problem.

Interestingly, we can show that the ϵ-augmentation problem and the ϵ-unit increment problem are equivalent for 0/1-integer programs.

Theorem 7. *For 0/1-integer programs, the ϵ-augmentation problem and the ϵ-unit increment problem are polynomial-time equivalent.*

Proof. Assume first that an ϵ-augmentation oracle and an ϵ-optimal solution x^1 to $\min\{cx : x \in X\}$ are given. Consider an instance $\min\{cx : x \in X\}$ with $X \subseteq \{0,1\}^n$ and its j-th unit increment instance $\min\{(c+e_j)x : x \in X\}$. Let x^0 and x^* be (unknown) optimal solutions of the former problem and the latter problem, respectively.

If the ϵ-augmentation oracle declares that x^1 is an ϵ-optimal solution to $\min\{(c+e_j)x : x \in X\}$, we are done. Thus suppose that starting with the solution x^1 to $\min\{(c+e_j)x : x \in X\}$ the ϵ-augmentation oracle produces an improved solution, say x^2. We will show that x^2 is an ϵ-approximate solution to $\min\{(c+e_j)x : x \in X\}$. By Lemma 3 we have

$$(c+e_j)x^* = cx^0 \text{ or } (c+e_j)x^* = cx^0 + 1. \tag{2}$$

Since x^1 is ϵ-optimal for $\min\{cx : x \in X\}$,

$$\frac{cx^1 - cx^0}{cx^0} \leq \epsilon.$$

Case 1: $x^1_j = 1$**.** In this case $(c+e_j)x^1 = cx^1 + 1$. Since x^2 is an improved solution for $\min\{(c+e_j)x : x \in X\}$ obtained from x^1, $(c+e_j)x^2 < (c+e_j)x^1$ and hence

$$(c+e_j)x^2 \leq cx^1.$$

From (2) we have $(c+e_j)x^* = cx^0$ or $(c+e_j)x^* = cx^0 + 1$. If $(c+e_j)x^* = cx^0$, then

$$\frac{(c+e_j)x^2 - (c+e_j)x^*}{(c+e_j)x^*} \leq \frac{cx^1 - cx^0}{cx^0} \leq \epsilon.$$

If $(c+e_j)x^* = cx^0 + 1$, then

$$\frac{(c+e_j)x^2 - (c+e_j)x^*}{(c+e_j)x^*} \leq \frac{cx^1 - cx^0 - 1}{cx^0 + 1} \leq \frac{cx^1 - cx^0}{cx^0} \leq \epsilon.$$

Case 2: $x^1_j = 0$**.** In this case $(c+e_j)x^1 = cx^1$. We will show that x^1 is an ϵ-optimal solution to $\min\{(c+e_j)x : x \in X\}$. If $(c+e_j)x^* = cx^0$, then

$$\frac{(c+e_j)x^1 - (c+e_j)x^*}{(c+e_j)x^*} = \frac{cx^1 - cx^0}{cx^0} \leq \epsilon.$$

If $(c+e_j)x^* = cx^0 + 1$ then,

$$\frac{(c+e_j)x^1 - (c+e_j)x^*}{(c+e_j)x^*} = \frac{cx^1 - cx^0 - 1}{cx^0 + 1} \leq \frac{cx^1 - cx^0}{cx^0} \leq \epsilon.$$

Thus if the ϵ-augmentation oracle does not declare x^1 as ϵ-optimal, the improved solution x^2 is guaranteed to be ϵ-optimal for $\min\{(c+e_j)x : x \in X\}$.

The converse of the theorem follows from Theorem 6. □

One consequence of the above theorem is that a 0/1-integer program has a (fully) polynomial-time approximation scheme if and only if the corresponding augmentation problem has a (fully) polynomial-time approximation scheme. The same result was obtained by Orlin et al. using different arguments [11].

5 The Unit Increment Problem and Local Optimization

In this section we consider the complexity of computing a locally optimal solution with respect to a given neighborhood function N. Recall that the *local* augmentation problem has as input a feasible solution x and an objective function vector c, and it outputs a solution $y \in N(x)$ with $cy < cx$, unless x is already a local optimum. The *local* unit increment problem accepts as input an index j and a locally optimal solution x^0 with respect to c, and it returns a locally optimal solution x^* with respect to $c + e_j$.

As in the case of (global) optimization and ϵ-optimization, we first observe that if a feasible solution can be obtained in polynomial time and the local unit increment problem can be solved in polynomial time, then a local optimum can be computed in polynomial time. To establish this, we simply modify algorithm UI by replacing the unit increment oracle, Unit-Inc, with a local unit increment oracle, Local-Unit-Inc. We call the resulting algorithm LUI.

Theorem 8. *Given a* Local-Unit-Inc *oracle, algorithm LUI computes a locally optimal solution in oracle-polynomial time.*

The proof of Theorem 8 is similar to that of Theorem 2. Theorem 8 establishes that the complexity of finding a local optimum is captured by that of the local unit increment problem.

Unlike the case of optimization and ϵ-optimization, we are not able to establish the equivalence of the local unit increment problem and the local augmentation problem for 0/1-integer programs. In fact, if they are equivalent, then, by Theorem 8, there is a polynomial-time algorithm for finding a local optimum for any problem in PLS, including PLS-complete problems. In other words, this would imply PLS = P. However, the two problems are equivalent if the neighborhood is exact. This follows from Theorem 4. Interestingly, we can show that this is also true for near-exact neighborhoods. Recall from the introduction that a neighborhood is called near exact if the objective function value of any local optimum is worse than that of at most a polynomial number of other feasible solutions.

Theorem 9. *For* 0/1*-integer programs with near-exact neighborhoods, a polynomial-time algorithm for local augmentation implies a polynomial-time algorithm for the local unit increment problem.*

Proof. Let x^0 be a locally optimal solution with respect to the near-exact neighborhood N and the objective function vector c. As usual, X denotes the set of feasible solutions.

Since N is near exact, there exists $X^* \subseteq X$ such that $cx^0 \leq cx$ for all $x \in X^*$ and $|X \setminus X^*| \leq f(|I|)$ for some polynomial f. Here, $|I|$ is the size of the input. By Lemma 3, $(c+e_j)(x^0-x) \leq 1$ for all $x \in X^*$. Thus in one augmentation step, starting from x^0, we get a solution that is no worse than any solution in X^* w.r.t $(c + e_j)$. This solution may or may not be a local optimum with respect to N. But outside X^* there are only $f(|I|)$ solutions and hence the local augmentation oracle cannot be called more than $f(|I|)$ additional times before reaching a local optimum. □

Corollary 10. *If there exists a near-exact neighborhood for a* PLS*-complete optimization problem with linear objective function, then there is a polynomial-time algorithm that finds a local optimum for all problems in* PLS*. That is,* PLS = P*.*

However, near-exact neighborhoods are unlikely to exist, at least for the TSP [7].

Acknowledgements

This work was supported in part by ONR grant N00014-01208-1-0029.

References

1. Ahuja, R.K., T.L. Magnanti, and J.B. Orlin, Network Flows: Theory, Algorithms, and Applications, Prentice Hall, 1993.
2. Ausiello, G., P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, and M. Protasi, Complexity and Approximation, Springer, 1999.
3. Chakravarti, N. and A.P.M. Wagelmans, Calculation of stability radii for combinatorial optimization problems, Operations Research Letters 23 (1998), 1–7.
4. Crescenzi, P. and R. Silvestri, Relative complexity of evaluating the optimum cost and constructing the optimum for maximization problems, Information Processing Letters 33 (1990), 221–226.
5. Fabrikant, A., C.H. Papadimitriou, and K. Talwar, The complexity of pure Nash equilibria, Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, 2004, pp. 604–612.
6. Grötschel, M. and L. Lovász, Combinatorial optimization, Chapter 28 in R.L. Graham, M. Grötschel, and L. Lovász (eds.): Handbook of Combinatorics, volume 2, Elsevier, 1995, pp. 1541–1597.
7. Gutin, G., A. Yeo, and A. Zverovitch, Exponential neighborhoods and domination analysis for the TSP, Chapter 6 in G. Gutin and A.P. Punnen (eds.), The Traveling Salesman Problem and Its Variations, Kluwer, 2002, pp. 223–256.
8. Johnson, D.S., The NP-completeness column: Finding needles in haystacks, ACM Transactions on Algorithms 3 (2007).
9. Johnson, D.S., C.H. Papadimitriou, and M. Yannakakis, How easy is local search?, Journal of Computer and System Sciences 37 (1988), 79–100.
10. Krentel, M.W., Structure in locally optimal solutions, in Proceedings of the 30th Annual Symposium on Foundations of Computer Science, Research Triangle Park, NC, 1989, 216–221.
11. Orlin, J.B., A.P. Punnen, and A.S. Schulz, Approximate local search in combinatorial optimization, SIAM Journal on Computing 33 (2004), 1201–1214.

12. Papadimitriou, C.H. and K. Steiglitz, Combinatorial Optimization: Algorithms and Complexity, Prentice-Hall, 1982.
13. Ramaswamy, R. and N. Chakravarti, Complexity of determining exact tolerances for min-sum and min-max combinatorial optimization problems, Working Paper WPS-247/95, Indian Institute of Management, Calcutta, India, 1995.
14. Schäffer, A.A. and M. Yannakakis, Simple local search problems that are hard to solve, SIAM Journal on Computing 20 (1991), 56–87.
15. Schrijver, A., Theory of Linear and Integer Programming, Wiley, 1986.
16. Schulz, A.S., On the relative complexity of 15 problems related to 0/1-integer programming, Chapter 19 in W.J. Cook, L. Lovász, J. Vygen (eds.): Research Trends in Combinatorial Optimization, Springer, Berlin, 2009, pp. 399–428.
17. Schulz, A.S., R. Weismantel, and G.M. Ziegler, 0/1-integer programming: Optimization and augmentation are equivalent, Lecture Notes in Computer Science 979 (1995), 473–483.
18. van Hoesel, S. and A.P.M. Wagelmans, On the complexity of postoptimality analysis of 0/1 programs, Discrete Applied Mathematics 91 (1999), 251–263.