MIT Sloan School of Management

MIT Sloan Working Paper 4325-03
July 2003, revised April 2004

Approximate Local Search in Combinatorial Optimization

James B. Orlin, Abraham P. Punnen, and Andreas S. Schulz

APPROXIMATE LOCAL SEARCH IN COMBINATORIAL OPTIMIZATION

JAMES B. ORLIN, ABRAHAM P. PUNNEN, AND ANDREAS S. SCHULZ

July 2003; revised April 2004

ABSTRACT. Local search algorithms for combinatorial optimization problems are in general of pseudopolynomial running time and polynomial-time algorithms are often not known for finding locally optimal solutions for NP-hard optimization problems. We introduce the concept of ε-local optimality and show that, for every $\varepsilon > 0$, an ε-local optimum can be identified in time polynomial in the problem size and $1/\varepsilon$ whenever the corresponding neighborhood can be searched in polynomial time. If the neighborhood can be searched in polynomial time for a δ-local optimum, a variation of our main algorithm produces a $(\delta + \varepsilon)$-local optimum in time polynomial in the problem size and $1/\varepsilon$. As a consequence, a combinatorial optimization problem has a fully polynomial-time approximation scheme if and only if the problem of determining a better neighbor in an exact neighborhood has a fully polynomial-time approximation scheme.

1. INTRODUCTION

A *combinatorial optimization problem* Π consists of a collection of instances $(\mathcal{F}, c)$, where the set $\mathcal{F}$ of feasible solutions is a family of subsets of a finite ground set $E = \{1, \ldots, n\}$. The objective function $c : E \to \mathbb{Q}_+$ assigns a nonnegative cost to every feasible solution $S \in \mathcal{F}$ through $c(S) := \sum_{e \in S} c_e$.[1] We assume that Π is closed under component-wise scaling of objective function coefficients; i.e., if $(\mathcal{F}, c) \in \Pi$, then $(\mathcal{F}, c') \in \Pi$ for all $c' \in \mathbb{Q}_+$. For technical reasons, let us also assume that $c(S) \neq 0$ for $S \in \mathcal{F}$. The goal is to find a globally optimal solution, i.e., a feasible solution S^* such that $c(S^*) \leq c(S)$ for all $S \in \mathcal{F}$.[2] The TRAVELING SALESPERSON PROBLEM (TSP) or the MINIMUM SPANNING TREE PROBLEM are typical examples of combinatorial optimization problems (see, e.g., Lawler 1976; Papadimitriou and Steiglitz 1982; Lawler et al. 1985; Cook et al. 1998; Korte and Vygen 2002).

Many combinatorial optimization problems are NP-hard, and one popular practical approach for attacking them is using local search strategies, which presupposes the concept of a neighborhood. A *neighborhood function* for an instance $(\mathcal{F}, c)$ of a combinatorial optimization problem Π is a mapping $N_{\mathcal{F}} : \mathcal{F} \to 2^{\mathcal{F}}$. Note that we assume that $N_{\mathcal{F}}$ does not depend on the objective function c. For convenience, we usually drop the subscript and simply write N. For a feasible solution S, $N(S)$ is called the *neighborhood* of S. We assume that $S \in N(S)$. A feasible solution $\bar{S}$ is said to be *locally optimal* with respect to N if $c(\bar{S}) \leq c(S)$ for all $S \in N(\bar{S})$. The *local search problem* is that of finding a locally optimal solution. Classic neighborhood functions include the k-opt neighborhood for the TSP (Lin 1965), the flip neighborhood for MAX CUT and MAX 2SAT (Schäffer and Yannakakis 1991), and the swap neighborhood for GRAPH PARTITIONING (Kernighan and Lin 1970). However, the class of problems that we are considering here also includes neighborhoods of exponential size, like, for the TSP, the twisted sequences neighborhood, the pyramidal tours

[1] Formally, we should point out that we focus on *linear* combinatorial optimization problems as opposed to "general" combinatorial optimization problems, in which the cost of a feasible solution is not necessarily the sum of the cost coefficients of its elements. In other words, the class of problems we are looking at here is equivalent to that of 0/1-integer linear programming problems.

[2] Although we restrict the following discourse to minimization problems, all results extend in a natural way to the case of maximization problems.

neighborhood, the permutation tree neighborhood, neighborhoods based on partial orders, and neighborhoods induced by polynomial-time solvable special cases. We refer the reader to Deineko and Woeginger (2000), Ahuja et al. (2002), and Gutin, Yeo, and Zverovitch (2002) for a detailed description of these neighborhood functions.

Roughly speaking, a local search algorithm sets out with an initial feasible solution and then repeatedly searches neighborhoods to find better and better solutions until it reaches a locally optimal solution. Figure 1 gives a generic description of the standard local search algorithm, which is sometimes also called *iterative improvement*.

Step 1: Compute a feasible starting solution $\bar{S}$;
Step 2: **while** $\bar{S}$ is not locally optimal **do**
Choose $S \in N(\bar{S})$ such that $c(S) < c(\bar{S})$;
$\bar{S} := S$;
Step 3: Output $\bar{S}$.

FIGURE 1. Algorithm **Standard Local Search**

Computational studies of local search algorithms and their variations have been extensively reported in the literature for various combinatorial optimization problems (see, e.g., Johnson et al. (1989) and Johnson and McGeoch (1997) for studies of the GRAPH PARTITIONING PROBLEM and the TSP, respectively). Empirically, local search heuristics appear to converge rather quickly, within low-order polynomial time. Compared to this wealth of information on empirical analysis, relatively little is known on theoretical properties of this class of algorithms. Of course, if one first multiplies all cost coefficients with their smallest common denominator to make them integer, the standard local search algorithm terminates in a pseudopolynomial number of iterations since it improves the objective function value by an integral amount in each iteration.[3] However, polynomial-time algorithms for computing a local optimum are in general not known. This is especially true for the above-mentioned combinatorial optimization problems and neighborhoods. On the other hand, Lawler (1976) constructed instances of the metric TSP such that the standard local search algorithm with the 2-opt neighborhood takes an exponential number of iterations under a particular pivoting rule. Chandra, Karloff, and Tovey (1999) extended this result to k-opt for all fixed $k > 2$.

This discontenting situation prompted Johnson, Papadimitriou, and Yannakakis (1988) to introduce the complexity class PLS. A combinatorial optimization problem Π together with a given neighborhood function N belongs to PLS if (a) instances are polynomial-time recognizable and a feasible solution is efficiently computable, (b) the feasibility of a proposed solution can be checked in polynomial time, and (c) neighborhoods can be searched efficiently. That is, there is a polynomial-time algorithm that decides whether a given feasible solution is locally optimal and, if not, computes a better solution in its neighborhood; see Figure 2 for a detailed description of the input and output of this algorithm. Note that all common local search problems are in PLS, in particular the problems mentioned earlier. The class PLS has its own type of reduction, which gives rise to the identification of complete problems in that class. For instance, MAX CUT and MAX 2SAT with the flip neighborhood, GRAPH PARTITIONING with the swap neighborhood, and TSP with the Lin-Kernighan neighborhood are PLS-complete (Krentel 1990; Schäffer and Yannakakis 1991; Papadimitriou 1992), and so is TSP with the k-opt neighborhood for some constant k (Krentel 1989). In particular, if a local optimum can be found in polynomial time for one of these problems, then

[3]Note that the scaling of rational coefficients to integers does not cause a superpolynomial blowup. We henceforth assume w.l.o.g. that all cost coefficients c_e for $e \in E$ are integers. An algorithm is pseudopolynomial if it is polynomial in the input dimension n and in $c_{\max} := \max_{e \in E} c_e$.

Input:	Objective function $c : E \to \mathbb{N}$ and feasible solution $S \in \mathcal{F}$.
Output:	"YES", if S is locally optimal with respect to c and N.
	"NO" and S', if there exists $S' \in N(S)$ with $c(S') < c(S)$.

FIGURE 2. Specification of the subroutine (oracle) IMPROVE$_N$

a local optimum can be computed in polynomial time for all problems in PLS. Unfortunately, it is unknown whether it is hard to find a local optimum for a PLS-complete problem (but Johnson et al. (1988) pointed out that this would imply NP = co-NP) or whether this can be done in polynomial time.[4]

In light of this somewhat elusive situation, it is interesting to explore the possibility of identifying *approximately locally optimal* solutions in polynomial time. We therefore introduce the notion of an ε-locally optimal solution, which is to some extent related to the worst-case relative performance guarantee of an approximation algorithm. We say that a feasible solution S^ε to an instance of a combinatorial optimization problem Π with neighborhood function N is an *ε-local optimum* if

$$\frac{c(S^\varepsilon) - c(S)}{c(S)} \quad \leq \quad \varepsilon \qquad \text{for all } S \in N(S^\varepsilon) \ ,$$

for some $\varepsilon > 0$. Hence, while S^ε is not necessarily a local optimum, it "almost" is. A family of algorithms $(A_\varepsilon)_{\varepsilon>0}$ for Π is an *ε-local optimization scheme* if A_ε produces an ε-local optimum. If the running time of algorithm A_ε is polynomial in the input size and $1/\varepsilon$, it is called a *fully polynomial-time ε-local optimization scheme*. In this paper, we show that every combinatorial optimization problem with an efficiently searchable neighborhood has a fully polynomial-time ε-local optimization scheme. In particular, an ε-locally optimal solution can be computed in polynomial time for every problem in PLS, including the PLS-complete problems and the problems with exponentially sized neighborhoods mentioned above.

Related Work. Ausiello and Protasi (1995) introduced the class GLO (for guaranteed local optima) of optimization problems that have the property that the objective function value of each local optimum is guaranteed to be within a constant factor of that of a global optimum. Khanna et al. (1998) extended this notion to nonoblivious GLO problems, which allow for a modification of the objective function used to compute a local optimum. In either class, the underlying neighborhoods contain all solutions of bounded Hamming distance from the current solution; moreover, it is assumed that the number of distinct objective function values over the set of feasible solutions is polynomially bounded. Hence, the standard local search algorithm yields a locally optimal solution in polynomial time. In contrast, we do not make any assumption on the objective function values or neighborhoods considered; we show that an ε-local optimum can always be computed with a polynomial number of calls to the IMPROVE subroutine. On the other hand, while an ε-local optimum has nearly the properties of a local optimum, its objective function value is not guaranteed to be close to that of a global optimum. However, this is in general true for local optima as well. For instance, Papadimitriou and Steiglitz (1977) showed that no local optimum of an efficiently searchable neighborhood for the TSP can be within a constant factor of the optimal value, unless P = NP. Yet, whenever a combinatorial optimization problem has an efficiently searchable neighborhood such that the value of each local optimum is within a constant factor $\alpha \geq 1$ of that of a

[4]Note that the negative results for the TSP mentioned at the end of the previous paragraph only apply to the standard local search algorithm, as described in Figure 1. Actually, there exist instances for every PLS-complete problem for which the standard local search algorithm takes exponential time, regardless of the tie-breaking and pivoting rules used (Yannakakis 1997, Theorem 13).

global minimum, then we can compute in polynomial time an ε-local optimum of cost not worse than $\alpha + \varepsilon$ times that of a global optimum. In other words, we give an $(\alpha + \varepsilon)$-approximation algorithm, for any fixed $\varepsilon > 0$.

Klauck (1996) studied the complexity of finding a solution whose objective function value is approximately as good as that of the worst local optimum, by using a restricted form of PLS-reductions. Completeness under this reduction implies that an approximation of a local optimum cannot be achieved efficiently unless P = PLS. For instance, 0/1-Programming with the k-flip neighborhood and Tsp with the k-opt neighborhood for constant k are complete under this reduction.

A neighborhood function N of a combinatorial optimization problem Π is *exact* if every locally optimal solution with respect to N is also globally optimal. In this case, our fully polynomial-time ε-local optimization scheme actually is a fully polynomial-time approximation scheme (FPTAS). This remains true even if Improve already outputs "Yes" when the current feasible solution is a δ-local optimum (and does so in time polynomial in the input size and $1/\delta$, for any $\delta > 0$). Grötschel and Lovász (1995) and Schulz, Weismantel, and Ziegler (1995) showed that, if a combinatorial optimization problem has an exact neighborhood that can be searched efficiently (and exactly), one can actually find an exact optimal solution efficiently. Schulz and Weismantel (1999, 2002) discussed extensions of this result from 0/1-integer linear programming problems (i.e., combinatorial optimization problems) to arbitrary integer programs. However, none of the employed techniques can be extended to compute a local optimum in polynomial time, unless the neighborhood is exact. In fact, otherwise P = PLS.

Fischer (1995) examined a different question, which implies that our main result is best possible if one considers the class of algorithms that iteratively moves from one feasible solution to a feasible solution in its neighborhood: Given a feasible solution S to an instance $(\mathcal{F}, c)$ of a combinatorial optimization problem Π and a number k in unary, does there exist a local optimum within k neighborhood steps of S? She showed that this question is NP-complete for Max Cut and Max 2Sat under the flip neighborhood and for Tsp under the 2-opt neighborhood, among others.

Our Results. Apart from our main result presented above and discussed in more detail in Section 2 below, we offer various evidence in Section 3 to show that this result is indeed "best possible":

First, the fully polynomial-time ε-local optimization scheme produces an ε-locally optimal solution by proceeding from one feasible solution to a solution in its neighborhood, and so forth. In this sense, it is a typical local search algorithm. Fischer's result shows that one cannot hope to find a local optimum in polynomial time by proceeding in this manner. Yet, an ε-local optimum can be determined in polynomial time. The key is to modify the original objective function so as to make sufficient progress in a relatively small number of steps. It is worth mentioning that our algorithm follows a path of feasible solutions that is not necessarily monotone with respect to the original objective function. In particular, it differs from a standard local search algorithm, which always replaces the current solution with a neighboring solution of lower cost.

Secondly, we point out that *any* algorithm for computing a local optimum that treats the neighborhood search and the feasibility check of PLS problems as oracles must be in the worst case exponential in the input dimension. In other words, if there exists a polynomial-time algorithm to compute a local optimum for a PLS-complete problem, it must use problem-specific knowledge. In contrast, our algorithmic scheme works for any combinatorial optimization problem so long as a feasible solution can be efficiently computed; in particular, it treats the subroutine Improve as a black box.

Thirdly, we show that the existence of a family $(A_\varepsilon)_{\varepsilon>0}$ of algorithms that find ε-local optima in time polynomial in the input size and $\log 1/\varepsilon$ implies the existence of a polynomial-time algorithm for computing a local optimum, and we just argued why this is impossible in our framework. Hence, the dependence of the running time on $1/\varepsilon$ cannot be improved. Furthermore, we prove

that replacing the relative error in the definition of an ε-local optimum with the absolute error would also yield the existence of a polynomial-time algorithm for computing a local optimum.

Finally, in Section 4 we present various extensions and variations of the main result, including more general integer linear programming problems and the new characterization for when a combinatorial optimization problem has a fully polynomial-time approximation scheme, which we already described in the context of related results for exact neighborhoods. Moreover, we discuss neighborhoods of polynomial size and efficiently searchable neighborhoods with local optima guaranteed to be near-optimal.

2. A Fully Polynomial-Time ε-Local Optimization Scheme

In this section we develop a polynomial-time algorithm to compute an ε-local optimum for a given instance $(\mathcal{F}, c)$ with ground set E of a combinatorial optimization problem Π with neighborhood function N.[5] The algorithm starts with a feasible solution S^0. We then alter the element costs c_e for $e \in E$ according to a prescribed scaling rule to generate a modified instance. Using local search on this modified problem, we look for a solution with an objective function value (with respect to the original cost) that is half that of S^0. If no such solution is found we are at a local optimum for the modified problem and output this solution. Otherwise we replace S^0 by the solution of cost less than half, call the latter one S^1, and the algorithm is repeated. A formal description of the algorithm is given in Figure 3. Note that the modification of the cost coefficients in Step 2 merely

Input: Objective function $c : E \to \mathbb{N}$; subroutine Improve$_N$; initial feasible solution $S^0 \in \mathcal{F}$; accuracy $\varepsilon > 0$.

Output: Solution $S^\varepsilon \in \mathcal{F}$ that is an ε-local optimum with respect to N and c.

Step 1: Let $i := 0$;

Step 2: Let $K := c(S^i)$, $q := \dfrac{K\varepsilon}{2n(1+\varepsilon)}$, and $c'_e := \left\lceil \dfrac{c_e}{q} \right\rceil q$ for $e \in E$;

Step 3: Let $k := 0$ and $S^{i,k} := S^i$;

Step 4: **repeat**

Call Improve$_N(S^{i,k}, c')$;

if the answer is "No", **then**

Let $S^{i,k+1} \in N(S^{i,k})$ such that $c'(S^{i,k+1}) < c'(S^{i,k})$; set $k := k + 1$;

else $S^\varepsilon := S^{i,k}$; **stop**.

until $c(S^{i,k}) \le K/2$;

Step 5: Let $S^{i+1} := S^{i,k}$, set $i := i + 1$ and **goto** Step 2.

Figure 3. Algorithm **ε-Local Search**

amounts to rounding them up to the closest integer multiple of q. Let us establish correctness first.

Theorem 2.1. *Algorithm ε-Local Search produces an ε-local optimum.*

Proof. Algorithm ε-Local Search terminates, which follows from the running time analysis following this proof. Let S^ε be the solution produced by the algorithm, and let $S \in N(S^\varepsilon)$ be an arbitrary

[5] The algorithm presented here works for neighborhoods of any size, in particular exponential-sized neighborhoods. Section 4.3 features a somewhat simpler algorithm for neighborhoods of polynomial size, which are given explicitly.

solution in its neighborhood. Let K and q denote the corresponding values from the last execution of Step 2 of the algorithm. Note that

$$c(S^\varepsilon) = \sum_{e \in S^\varepsilon} c_e \leq \sum_{e \in S^\varepsilon} \left\lceil \frac{c_e}{q} \right\rceil q \leq \sum_{e \in S} \left\lceil \frac{c_e}{q} \right\rceil q \leq \sum_{e \in S} q \left(\frac{c_e}{q} + 1 \right) \leq \sum_{e \in S} c_e + nq = c(S) + nq \ ,$$

where $n = |E|$. Here, the second inequality follows from the fact that S^ε is locally optimal with respect to c'. Together with $c(S^\varepsilon) \geq K/2$, this implies

$$\frac{c(S^\varepsilon) - c(S)}{c(S)} \quad \leq \quad \frac{nq}{c(S)} \quad \leq \quad \frac{nq}{c(S^\varepsilon) - nq} \quad \leq \quad \frac{2nq}{K - 2nq} \quad = \quad \varepsilon \ .$$

□

Let us now analyze the running time of Algorithm ε-Local Search. In each improving move within the local search in Step 4 of the algorithm, the objective function value (with respect to c') is decreased by at least q units. Thus the number of calls to IMPROVE between two consecutive iterations of Step 2 is $O(n(1+\varepsilon)/\varepsilon) = O(n/\varepsilon)$. Step 2 is executed at most $\log c(S^0)$ times, where S^0 is the starting solution. Thus the total number of neighborhoods searched is $O(n\,\varepsilon^{-1} \log c(S^0))$. Therefore, if the neighborhood N can be searched in polynomial time for an improving solution, we have a fully polynomial-time ε-local optimization scheme. Note that the number of iterations includes the factor $\log c(S^0)$ and hence the bound is not strongly polynomial. However, it is possible to prove a strongly polynomial bound on the number of iterations. For this, we make use of the following lemma, which Radzik (1993) attributed to Goemans.

Lemma 2.2. *Let $d = (d_1, \ldots, d_n)$ be a real vector and let $y_1, \ldots, y_p$ be vectors in $\{0,1\}^n$. If, for all $i = 1, \ldots, p-1$, $0 \leq d\,y_{i+1} \leq \frac{1}{2}\,d\,y_i$, then $p = O(n \log n)$.*

Note that the value of K at each execution of Step 2 is reduced at least by half. Further, K is a linear combination of c_e for $e \in E$ and the coefficients in this linear combination are from the set $\{0,1\}$. Lemma 2.2 implies that Step 2 of Algorithm ε-Local Search can be executed at most $O(n \log n)$ times. Thus the total number of calls of IMPROVE in Step 4 throughout the algorithm is $O(\varepsilon^{-1}\, n^2 \log n)$. If $\zeta(n, \log c_{\max})$ is the time needed to search the neighborhood N for an improving solution (i.e., the running time of IMPROVE) and $\xi(n)$ is the time needed to obtain a feasible starting solution, the complexity of Algorithm ε-Local Search is $O(\xi(n) + \zeta(n, \log c_{\max}) n\,\varepsilon^{-1} \min\{n \log n, \log K^0\})$, where $K^0 := c(S^0)$ is the objective function value of the starting solution and $c_{\max}$ is the maximal value of an objective function coefficient. The following theorem summarizes the preceding discussion.

Theorem 2.3. *Algorithm ε-Local Search correctly identifies an ε-locally optimal solution of an instance of a combinatorial optimization problem in $O(\xi(n) + \zeta(n, \log c_{\max}) n\,\varepsilon^{-1} \min\{n \log n, \log K^0\})$ time.*

Thus if $\zeta(n, \log c_{\max})$ and $\xi(n)$ are polynomial, then Algorithm ε-Local Search is a fully polynomial-time ε-local optimization scheme. Note that $\zeta(n, \log c_{\max})$ and $\xi(n)$ are indeed polynomials of the input size for all problems in PLS.

Corollary 2.4. *Every problem in PLS has a fully polynomial-time ε-local optimization scheme.*

The running time of Algorithm ε-Local Search can sometimes be improved by exploiting the special structure of the underlying neighborhood. A neighborhood function N generates a so-called k-opt neighborhood if $S_1, S_2 \in N(S)$ implies $|(S_1 \setminus S_2) \cup (S_2 \setminus S_1)| \leq k$, which is equivalent to bounding the Hamming distance between the incidence vectors of S_1 and S_2 by k. For the k-opt neighborhood, by choosing the parameter $q := \frac{K\varepsilon}{2k(1+\varepsilon)}$ in Algorithm ε-Local Search, we still get an ε-local optimum. Moreover, the number of calls of IMPROVE between two consecutive executions of Step 2 of this modified algorithm is $O(\varepsilon^{-1})$, for fixed k. This brings down the

total number of such calls to $O(\varepsilon^{-1} \min\{n \log n, \log K^0\})$ and implies a running time of $O(\xi(n) + n^k \varepsilon^{-1} \min\{n \log n, \log K^0\})$ for Algorithm ε-Local Search.

Similarly, if the considered combinatorial optimization problem possesses a 2-approximation algorithm, one can use this algorithm to compute the starting solution S^0. If such a solution is used as the starting solution, then Algorithm ε-Local Search executes Step 2 only once and hence the total number of improving moves is $O(n\,\varepsilon^{-1})$. Consequently, the overall running time of Algorithm ε-Local Search is $O(\xi(n) + \zeta(n, \log c_{\max}) n\,\varepsilon^{-1})$, where $\xi(n)$ now denotes the running time of the 2-approximation algorithm. In fact, whenever one has a feasible solution S^0 for an instance I such that $c(S^0) \leq p(\langle I \rangle) c(S^*)$ for some polynomial p of the input size $\langle I \rangle$, then one can adapt the value of q to $q := (K\varepsilon)/(n\,p(\langle I \rangle)(1+\varepsilon))$ and the stopping criterion of the while-loop accordingly, so that Algorithm ε-Local Search computes an ε-local optimum in $O(n\,p(\langle I \rangle)\varepsilon^{-1})$ iterations.

Arkin and Hassin (1998) applied a similar approach to that used in Algorithm ε-Local Search to compute a local optimum in polynomial time in the context of the weighted k-set packing problem. They considered a neighborhood for which the value of each local optimum is within a certain factor of the value of a global optimum; hence, this approach, which they attributed to Rubinstein, leads to a polynomial-time approximation algorithm. It has since been applied in related situations as well; see, e.g., Arkin et al. (2002).

3. Negative Results

In this section we present a collection of results that underscore that neither the accuracy of the solutions produced by Algorithm ε-Local Search nor its running time can be significantly improved, unless additional, problem-specific knowledge is used. Let us first argue that any algorithm to compute a local optimum for a problem in PLS has to have exponential running time in the worst case if the algorithms for checking feasibility and neighborhood search are oracles completely hiding both the set of feasible solutions and the neighborhood structure.

Theorem 3.1. *If the only available information on an instance $(\mathcal{F}, c)$ of a combinatorial optimization problem Π is the objective function vector c, a feasible solution $S^0 \in \mathcal{F}$, a membership oracle, and a neighborhood search oracle* Improve*, then any algorithm for computing a local optimum takes exponential time in the worst case.*

Proof. Let the ground set be $E = \{1, 2, \ldots, n\}$. The objective function coefficients are $c_i = 2^{i-1}$ for $i = 1, 2, \ldots, n$. Let the nonempty subsets of E be labeled $S^0, S^1, \ldots, S^{2^n-2}$ such that $c(S^0) > c(S^1) > \cdots > c(S^{2^n-2})$. Let A be an arbitrary algorithm that computes a local optimum. Note that A can either call Improve with a feasible solution and some objective function, or it can check whether a particular solution is feasible by asking the membership oracle. We show that A needs exponential time in the worst case by exhibiting an adverse strategy. In fact, an adversary adapts the set of feasible solutions and the neighborhood function to A's sequence of questions as follows. Whenever A asks the membership oracle whether a set S^i is feasible, the adversary answers "No" unless S^i has been declared feasible earlier. On the other hand, whenever A calls up Improve with a feasible solution S^i and an objective function vector c', Improve returns S^j, where $j > i$ is the smallest index for which S^j has not been labeled infeasible. If no such j exists or $c'(S^j) \geq c'(S^i)$, then S^i is locally optimal (w.r.t. c'). It is not difficult to see that A has to touch every single subset before it can identify the unique minimum. □

The importance of Theorem 3.1 relates to the fact that Algorithm ε-Local Search only requires a subset of the information stated in the assumptions of this theorem; in particular, it does not make use of the membership oracle.

We next note that finding an ε-local optimum of *additive error* ε with respect to a given neighborhood structure is as hard as finding a local optimum with respect to the same neighborhood structure. While its proof relies on a standard argument, the result is still worth recording.

Observation 3.2. *If there is an algorithm that for every instance $(\mathcal{F}, c)$ of a combinatorial optimization problem Π with neighborhood N finds in polynomial time a feasible solution S_ε such that $c(S_\varepsilon) \leq c(S) + \varepsilon$ for all $S \in N(S_\varepsilon)$, for some fixed $\varepsilon > 0$, then there is a polynomial-time algorithm to find a local optimum.*

Proof. Let $(\mathcal{F}, c)$ be an instance of Π, where w.l.o.g. c is an integer-valued function. Create a new instance $(\mathcal{F}, c')$ by setting $c'_e := (1 + \varepsilon)c_e$ for all elements e of the ground set. Apply the given algorithm to the new instance and let S' be the resulting solution. Then, $c'(S') - c'(S) \leq \varepsilon$ for all $S \in N(S')$. Thus, $c(S') - c(S) \leq \varepsilon/(\varepsilon + 1) < 1$ for all $S \in N(S')$. Since c is integer-valued, it follows that S' is a local optimum for the original instance. □

The next result is somewhat similar to (Garey and Johnson 1979, Theorem 6.8) except that we are discussing it in the context of local optimality.

Observation 3.3. *If a combinatorial optimization problem Π has a fully polynomial-time ε-local optimization scheme $(A_\varepsilon)_{\varepsilon>0}$ such that the actual running time of A_ε is polynomial in the input size and $\log 1/\varepsilon$, then there is a polynomial-time algorithm that computes a local optimum.*

Proof. Let $(\mathcal{F}, c)$ be an instance of Π, where w.l.o.g. c is an integer-valued function. Choose $\varepsilon := 1/(n\, c_{\max} + 1)$ and apply A_ε. Note that its running time is polynomial in the input size of the instance. If S^ε is the solution returned by this algorithm, then $c(S^\varepsilon) \leq (1 + \varepsilon)c(S) < c(S) + 1$ for all $S \in N(S^\varepsilon)$. Hence, S^ε is a local optimum. □

4. Extensions and Variants

We now discuss some extensions and variations of the results of Section 2 that range from approximation guarantees in case of exact neighborhoods, over simplifications of Algorithm ε-Local Search for explicitly given neighborhoods of polynomial size, to replacing Improve with weaker oracles and bounded integer programming problems.

4.1. Exact Neighborhoods.

Recall that a neighborhood function N for a combinatorial optimization problem Π is exact if every local optimum is already globally optimal. In view of this, one may be tempted to conjecture that the objective function value of an ε-local optimum with respect to an exact neighborhood is also within a factor of $(1+\varepsilon)$ of the value of a global optimum. However, this is not true as shown by the following example.

Let $G = (V, E)$ be a connected graph with edge weights c_e for $e \in E$. Let $\mathcal{F}$ be the family of all spanning trees of G. Hence, we are considering the Minimum Spanning Tree Problem. For any tree $T \in \mathcal{F}$, consider the neighborhood $N(T)$ that consists of those spanning trees obtained from T by adding an edge $e \in E \setminus T$ to T and removing an edge $f \in T$ from the induced elementary cycle. This is the 2-opt neighborhood, which is known to be exact. Now choose G as a wheel (see Figure 4) with node set $\{0, 1, \ldots, n\}$. For each edge $(0, i)$, $i = 1, 2, \ldots, n$, of this wheel, assign a cost of 1 and for each edge $(i, i + 1)$, $i = 1, 2, \ldots, n$ (where node $n + 1$ is identified with node 1) assign a cost of zero. The spanning tree T^ε, which is a star rooted at node 0, is a $1/(n - 1)$-local optimum for any $n \geq 3$. However, the Hamiltonian path $T^* = (0, 1, \ldots, n)$ is a minimum spanning tree and $c(T^\varepsilon) - c(T^*) = (n-1)c(T^*)$. Thus, T^ε is not a $(1+\varepsilon)$-approximation for any $\varepsilon < n-1$.

Still, for exact neighborhoods our fully polynomial-time ε-local optimization scheme actually is a fully polynomial-time approximation scheme (FPTAS), as the following theorem shows.

Theorem 4.1. *If the neighborhood N of a combinatorial optimization problem Π is exact, then the objective function value of the solution produced by Algorithm ε-Local Search is within a factor of $(1 + \varepsilon)$ of that of a global minimum.*

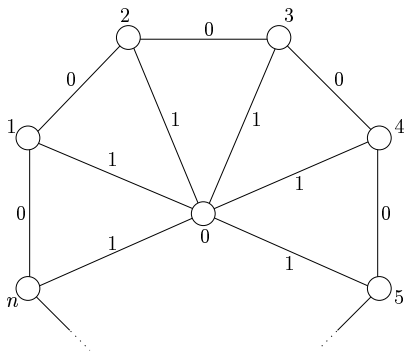

FIGURE 4. A wheel on $n+1$ nodes

Proof. Let $(\mathcal{F}, c)$ be a given instance of Π. Let S^* be an optimal solution and S^ε be the solution produced by the algorithm. Let K, q, and c' denote the corresponding values from the last execution of Step 2 of Algorithm ε-Local Search. Since S^ε is locally optimal with respect to c' and the neighborhood is exact, S^ε is an optimal solution for $(\mathcal{F}, c')$. Thus,

$$c(S^\varepsilon) \leq \sum_{e \in S^\varepsilon} \left\lceil \frac{c_e}{q} \right\rceil q \leq \sum_{e \in S^*} \left\lceil \frac{c_e}{q} \right\rceil q \leq \sum_{e \in S^*} q \left(\frac{c_e}{q} + 1 \right) \leq c(S^*) + nq \leq c(S^*) + \frac{\varepsilon}{1+\varepsilon}\, c(S^\varepsilon) \ ,$$

where the last inequality follows from the definiton of q and the fact that $c(S^\varepsilon) \geq K/2$. The result follows. □

Theorem 4.1 implies that whenever a combinatorial optimization problem has an exact neighborhood and an initial feasible solution is readily available, then Algorithm ε-Local Search is a $(1+\varepsilon)$-approximation algorithm that calls IMPROVE a polynomial number of times (for fixed $\varepsilon > 0$). However, Grötschel and Lovász (1995) and Schulz et al. (1995) showed that under these circumstances, one can actually compute an optimal solution with a polynomial number of calls of IMPROVE. Yet, one still obtains an FPTAS even if the exact neighborhood can only be searched approximately, as we are about to see next.

4.2. **Approximate Version of Neighborhood Search.** Very large-scale neighborhood (VLSN) search algorithms are local search algorithms using neighborhoods of very large size; see Ahuja et al. (2002) for a survey. For many very large-scale neighborhoods of NP-hard combinatorial optimization problems, the problem of finding an improving solution is itself NP-hard. On the other hand, solutions produced by local search algorithms using such huge neighborhoods could be of very high quality. To keep the complexity of a VLSN algorithm manageable, approximation algorithms are often employed to search an underlying neighborhood such that if the approximation algorithm fails to find an improving move, the algorithm terminates leaving an "approximate" local solution. More precisely, instead of IMPROVE, we may only have at our command a subroutine δ-IMPROVE, which solves the following problem:

> Given an objective function vector c and a solution $S \in \mathcal{F}$, find $S' \in N(S)$ such that $c(S') < c(S)$, or assert that S is a δ-local optimum. (4.1)

Naturally, finding a δ-local optimum efficiently, given an algorithm δ-IMPROVE, faces similar difficulties as the problem of finding a local optimum when an algorithm IMPROVE is provided. However, one can easily modify Algorithm ε-Local Search so that it computes a $(\delta+\varepsilon)$-local optimum in polynomial time. In fact, if one uses δ-IMPROVE in lieu of IMPROVE and selects the scaling parameter q to be $q := \frac{K\varepsilon}{2n(1+\delta)(1+\delta+\varepsilon)}$, the resulting algorithm produces a $(\delta+\varepsilon)$-local optimum in

time $O(\xi(n) + \psi(n, \log c_{\max})\, n\, \varepsilon^{-1} \min\{n \log n, \log K^0\})$, where $\psi(n, \log c_{\max})$ is the running time of δ-IMPROVE. Hence, a (strongly) polynomial-time algorithm for solving (4.1) implies a (strongly) polynomial-time algorithm to compute a $(\delta + \varepsilon)$-local optimum, for every fixed $\varepsilon > 0$. In view of the results discussed in Section 4.1, it is particularly interesting to note that in case of an exact neighborhood, the existence of a polynomial-time algorithm δ-IMPROVE for all $\delta > 0$ implies that the combinatorial optimization problem possesses a fully polynomial-time approximation scheme. Indeed, if we call (4.1) the *augmentation problem* if N is exact (e.g., $N(S) = \mathcal{F}$ for all $S \in \mathcal{F}$) and $\delta = 0$, and a family of algorithms δ-IMPROVE (with running time polynomial in the input size and $1/\delta$ for all $\delta > 0$) a *fully polynomial-time approximation scheme for the augmentation problem*, we can state the following result.

Theorem 4.2. *A combinatorial optimization problem has a fully (strongly) polynomial-time approximation scheme if and only if its corresponding augmentation problem has a fully (strongly) polynomial-time approximation scheme.*

The proof of Theorem 4.2 is similar to that of Theorem 4.1.

Sometimes the objective function value of each local optimum is within a constant factor of that of a global optimum. The class of problems with this property contains the class GLO (Ausiello and Protasi 1995). For instance, each local optimum of the flip neighborhood for the MAX CUT PROBLEM has value at least half that of an optimal cut (Sahni and Gonzalez 1976). The following theorem can be proved in a similar manner to Theorem 4.1 by setting $q := \frac{K\varepsilon}{2n\alpha(1+\varepsilon)}$ in Algorithm ε-Local Search.

Theorem 4.3. *Let Π be a combinatorial optimization problem with efficiently searchable neighborhood function N such that every local optimum is within a constant factor $\alpha \geq 1$ of the optimal value. Then there is an algorithm that computes a feasible solution of cost at most $\alpha + \varepsilon$ times that of an optimal solution in time polynomial in the input size and $1/\varepsilon$, for any $\varepsilon > 0$.*

4.3. **Polynomial-Sized Neighborhoods.** Algorithm ε-Local Search is designed to work for any local search problem for which an IMPROVE (or δ-IMPROVE) oracle is available, regardless of the size or structure of the neighborhood and the implementation of IMPROVE. In particular, Algorithm ε-Local Search identifies for the TSP and each of the following neighborhoods an ε-local optimum in polynomial time: the twisted sequences neighborhood, the pyramidal tours neighborhood, the permutation tree neighborhood, neighborhoods based on partial orders, as well as neighborhoods induced by polynomial-time solvable special cases. While all these exponential-sized neighborhoods can be searched efficiently (i.e., they have polynomial-time improvement oracles), it is not known for any of them how to find a local optimum in polynomial time (Deineko and Woeginger 2000; Ahuja et al. 2002; Gutin et al. 2002).

Nonetheless, neighborhoods frequently are of polynomial size and explicitly given, like the k-opt neighborhood for the TSP (for fixed k), the flip neighborhood for MAX CUT and MAX 2SAT, or the swap neighborhood for GRAPH PARTITIONING. In this case, one can give a simpler algorithm for computing an ε-local optimum, see Figure 5. Note that Step 1 can be realized by an exhaustive search of the neighborhood of S. Obviously, the running time of this algorithm is polynomial in the input size and $1/\varepsilon$. By using an appropriately modified version of Lemma 2.2, one can actually show that the running time only depends on the input dimension n and $1/\varepsilon$. However, this simpler algorithm has some drawbacks compared to Algorithm ε-Local Search, even if one limits this comparision to problems with explicitly given neighborhoods of polynomial size. In particular, neither Theorem 4.1 nor Theorem 4.2 can be proved with its help. In fact, for the minimum spanning tree example described in Figure 4, the simpler algorithm with accuracy $\varepsilon \geq 1/(n-1)$ would terminate with the initially given solution T^ε although the cost of this solution is n times that of an optimal solution. In contrast, Algorithm ε-Local Search would return an optimal solution in this case (and a $(1+\varepsilon)$-approximate solution in general).

Input: Objective function $c : E \to \mathbb{N}$; neighborhood function $N : \mathcal{F} \to 2^{\mathcal{F}}$; initial feasible solution $S \in \mathcal{F}$; accuracy $\varepsilon > 0$.
Output: Solution $S^\varepsilon \in \mathcal{F}$ that is an ε-local optimum with respect to N and c.

Step 1: **while** S is not ε-locally optimal **do**
Choose $S' \in N(S)$ satisfying $c(S') < c(S)/(1+\varepsilon)$;
$S := S'$;
Step 2: **return** $S^\varepsilon := S$.

FIGURE 5. ε-local search algorithm for neighborhoods of polynomial size

4.4. **Weaker Version of Neighborhood Search.** In the context of global optimization, Schulz et al. (1996) pointed out that the requirements on the improvement oracle can be somewhat weakened. Interestingly, this extension also works in the context of local optimization, as we will show now. For that, we replace IMPROVE with another subroutine, which we call TEST. TEST accepts the same input as IMPROVE, namely a current feasible solution S together with an objective function vector c. It also answers "YES" or "NO" depending on whether S is locally optimal with respect to c or not, but in contrast to IMPROVE it does not provide a solution $S' \in N(S)$ of lower cost if S is not locally optimal. It just answers "YES" or "NO".

Lemma 4.4. TEST *can emulate* IMPROVE *in polynomial time; i.e., whenever the input solution S is not locally optimal, a polynomial number of calls to* TEST *suffices to create a solution $S' \in N(S)$ of lower cost.*[6]

Proof. Let $(\mathcal{F}, c)$ be an instance and S a feasible solution that is not locally optimal with respect to c under the given neighborhood N. W.l.o.g., we may assume that $S = E$.[7] Here, $E = \{1, 2, \ldots, n\}$ is the ground set. The algorithm that we are about to explain proceeds by considering one coordinate after the other. In particular, it will call TEST n times. The first call $\text{TEST}_N(S, c^1)$ is made with the objective function

$$c^1_e := \begin{cases} c_1 - M & \text{if } e = 1, \\ c_e & \text{otherwise,} \end{cases} \qquad \text{for } e \in E \ ,$$

where $M := n\, c_{\max} + 1$. If TEST responds with "YES", then we can infer that all solutions S' in $N(S)$ of lower cost than S with respect to c satisfy $1 \notin S'$. On the other hand, if the reply is "NO", then there is at least one solution $S' \in N(S)$ such that $c(S') < c(S)$ and $1 \in S'$.

In general, assume that we already know a subset $R \subseteq \{1, 2, \ldots, k\}$, for some $k \in \{1, 2, \ldots, n-1\}$, with the following two properties:

(i) there exists a solution $S' \in N(S)$ with $c(S') < c(S)$ such that $R = S' \cap \{1, 2, \ldots, k\}$;
(ii) if $j \notin R$ for $1 \le j \le k$, then all solutions $S'' \in N(S)$ with $c(S'') < c(S)$ satisfy $R \cap \{1, 2, \ldots, j\} = S'' \cap \{1, 2, \ldots, j\}$.

We then call $\text{TEST}_N(S, c^{k+1})$ with

$$c^{k+1}_e := \begin{cases} c_e - M & \text{if } e \in R, \\ c_{k+1} - M & \text{if } e = k+1, \\ c_e & \text{otherwise,} \end{cases} \qquad \text{for } e \in E \ .$$

[6] In contrast to all other results presented in this paper, here we need to assume that TEST accepts arbitrary, not just nonnegative cost coefficients. In particular, we set $c_{\max} := \max_{e \in E} |c_e|$.

[7] Otherwise one can transform the given instance into an equivalent one for which this is the case.

If the result is "YES", then we can infer that all solutions $S' \in N(S)$ with $c(S') < c(S)$ and $S' \cap \{1, 2, \ldots, k\} = R$ satisfy $k+1 \notin S'$. However, if the reply is "NO", then there must be a solution $S' \in N(S)$ with $c(S') < c(S)$ and $S' \cap \{1, 2, \ldots, k\} = R$ such that $k+1 \in S'$. We leave R unchanged in the former case, and set $R := R \cup \{k+1\}$ in the latter case.

Consequently, after n steps we have identified a set $S' = R \in N(S)$ such that $c(S') < c(S)$. □

Lemma 4.4 and Theorem 2.3 imply the following result.

Corollary 4.5. *An ε-local optimum of an instance $(\mathcal{F}, c)$ of a combinatorial optimization problem Π with neighborhood function N that is given via a* TEST *oracle of running time $\zeta(n, \log c_{\max})$, can be computed in time $O(\xi(n) + \zeta(n, \log c_{\max}) n^2 \varepsilon^{-1} \min\{n \log n, \log K^0\})$.*

4.5. **Bounded Integer Linear Programming Problems.** Let us finally discuss some generalizations to the case of integer linear programs with bounded variables. In our discussions so far, we considered combinatorial optimization problems, which in fact are 0/1-integer linear programs, i.e., problems of the form $\min\{c\,x : x \in \mathcal{F}\}$ with $\mathcal{F} \subseteq \{0,1\}^n$. Interestingly, most of our results extend directly to the case of integer linear programs with bounded variables, which can be described as follows: $\min\{c\,x : x \in \mathcal{F}\}$ with $\mathcal{F} \subseteq \{0, 1, \ldots, u\}^n$ for some nonnegative integer u. In this context, a neighborhood assigns to each feasible solution $x \in \mathcal{F}$ a set of feasible points in $\{0, 1, \ldots, u\}^n$. If an initial feasible solution and an algorithm IMPROVE are available, Algorithm ε-Local Search can easily be modified to compute an ε-local optimum in this setting as well. In fact, by choosing the scaling parameter $q := \frac{K\varepsilon}{2n(1+\varepsilon)u}$, its number of iterations (and therefore the number of calls of IMPROVE) is $O(n\,\varepsilon^{-1} u \log K^0)$. Thus, if an initial feasible solution can be identified in polynomial time, if IMPROVE can be implemented in polynomial time, and if u is bounded by a polynomial in n and $\log c_{\max}$, Algorithm ε-Local Search runs in polynomial time.

ACKNOWLEDGMENTS

The authors are grateful to an anonymous referee for several insightful suggestions, which helped to improve the presentation of this paper. In particular, he or she brought the algorithm described in Figure 5 to their attention. They also thank Rafi Hassin for pointing them to the papers by Arkin and Hassin (1998) and Arkin et al. (2002) after reading an earlier version of this paper. Jim Orlin was partially supported through NSF contract DMI-0217123; Abraham Punnen was partially supported by NSERC grant OPG 0170381; Jim Orlin and Andreas Schulz were partially supported by ONR contract N00014-98-1-0317. An extended abstract of this paper appeared in the Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA '04), New Orleans, LA, 2004, pp. 580–589.

REFERENCES

Ahuja, R. K., Ö. Ergun, J. B. Orlin, and A. P. Punnen (2002). A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics 123*, 75–102.

Arkin, E. M. and R. Hassin (1998). On local search for weighted k-set packing. *Mathematics of Operations Research 23*, 640–648.

Arkin, E. M., R. Hassin, S. Rubinstein, and M. Sviridenko (2002). Approximations for maximum transportation problem with permutable supply vector and other capacitated star packing problems. In M. Penttonen and E. Meineche Schmidt (Eds.), *Algorithm Theory – SWAT 2002*, Volume 2368 of *Lecture Notes in Computer Science*, pp. 280–287. Springer. Proceedings of the 8th Scandinavian Workshop on Algorithm Theory.

Ausiello, G. and M. Protasi (1995). Local search, reducibility and approximability of NP-optimization problems. *Information Processing Letters 54*, 73–79.

Chandra, B., H. J. Karloff, and C. A. Tovey (1999). New results on the old k-opt algorithm for the traveling salesman problem. *SIAM Journal on Computing 28*, 1998–2029.

Cook, W. J., W. H. Cunningham, W. R. Pulleyblank, and A. Schrijver (1998). *Combinatorial Optimization*. John Wiley & Sons.

Deineko, V. and G. Woeginger (2000). A study of exponential neighborhoods for the traveling salesman problem and the quadratic assignment problem. *Mathematical Programming 87*, 519–542.

Fischer, S. T. (1995). A note on the complexity of local search problems. *Information Processing Letters 53*, 69–75.

Garey, M. R. and D. S. Johnson (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman.

Grötschel, M. and L. Lovász (1995). Combinatorial optimization. In R. Graham, M. Grötschel, and L. Lovász (Eds.), *Handbook of Combinatorics*, Volume II, Chapter 28, pp. 1541–1597. North-Holland.

Gutin, G., A. Yeo, and A. Zverovitch (2002). Exponential neighborhoods and domination analysis for the TSP. In G. Gutin and A. P. Punnen (Eds.), *The Traveling Salesman Problem and its Variations*, Chapter 6, pp. 223–256. Kluwer Academic.

Johnson, D. S., C. R. Aragon, L. A. McGeoch, and C. Schevon (1989). Optimization by simulated annealing: an experimental evaluation, part I (graph partitioning). *Operations Research 37*, 865–892.

Johnson, D. S. and L. A. McGeoch (1997). The traveling salesman problem: a case study. In E. Aarts and J. K. Lenstra (Eds.), *Local Search in Combinatorial Optimization*, Chapter 8, pp. 215–310. John Wiley & Sons.

Johnson, D. S., C. H. Papadimitriou, and M. Yannakakis (1988). How easy is local search? *Journal of Computer and System Sciences 37*, 79–100.

Kernighan, B. W. and S. Lin (1970). An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal 49*, 291–307.

Khanna, S., R. Motwani, M. Sudan, and U. Vazirani (1998). On syntactic versus computational views of approximability. *SIAM Journal on Computing 28*, 164–191.

Klauck, H. (1996). On the hardness of global and local approximation. In R. Karlsson (Ed.), *Algorithm Theory – SWAT '96*, Volume 1097 of *Lecture Notes in Computer Science*, pp. 88–99. Springer. Proceedings of the 5th Scandinavian Workshop on Algorithm Theory.

Korte, B. and J. Vygen (2002). *Combinatorial Optimization: Theory and Algorithms* (2nd ed.). Springer.

Krentel, M. W. (1989). Structure in locally optimal solutions. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, Los Alamitos, CA, pp. 216–221. IEEE.

Krentel, M. W. (1990). On finding and verifying locally optimal solutions. *SIAM Journal on Computing 19*, 742–749.

Lawler, E. (1976). *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehart and Winston.

Lawler, E. L., J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys (Eds.) (1985). *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. John Wiley & Sons.

Lin, S. (1965). Computer solutions of the traveling salesman problem. *Bell System Technical Journal 44*, 2245–2269.

Papadimitriou, C. H. (1992). The complexity of the Lin-Kernighan heuristic for the traveling salesman problem. *SIAM Journal on Computing 21*, 450–465.

Papadimitriou, C. H. and K. Steiglitz (1977). On the complexity of local search for the traveling salesman problem. *SIAM Journal on Computing 6*, 76–83.

Papadimitriou, C. H. and K. Steiglitz (1982). *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall.

Radzik, T. (1993). Parametric flows, weighted means of cuts, and fractional combinatorial optimization. In P. Pardalos (Ed.), *Complexity in Numerical Optimization*, pp. 351–386. World Scientific.

Sahni, S. and T. Gonzalez (1976). P-complete approximation problems. *Journal of the ACM 23*, 555–565.

Schäffer, A. A. and M. Yannakakis (1991). Simple local search problems that are hard to solve. *SIAM Journal on Computing 20*, 56–87.

Schulz, A. S. and R. Weismantel (1999). An oracle-polynomial time augmentation algorithm for integer programming. In *Proceedings of the 10th Annual Symposium on Discrete Algorithms*, Baltimore, MD, pp. 967–968. ACM/SIAM.

Schulz, A. S. and R. Weismantel (2002). The complexity of generic primal algorithms for solving general integer programs. *Mathematics of Operations Research 27*, 681–692.

Schulz, A. S., R. Weismantel, and G. M. Ziegler (1995). 0/1-integer programming: optimization and augmentation are equivalent. In P. Spirakis (Ed.), *Algorithms – ESA '95*, Volume 979 of *Lecture Notes in Computer Science*, pp. 473–483. Springer. Proceedings of the 3rd Annual European Symposium on Algorithms.

Schulz, A. S., R. Weismantel, and G. M. Ziegler (1996). Unpublished Manuscript.

Yannakakis, M. (1997). Computational complexity. In E. Aarts and J. K. Lenstra (Eds.), *Local Search in Combinatorial Optimization*, Chapter 2, pp. 19–55. John Wiley & Sons.

James B. Orlin, Operations Research Center, Office E40-137, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139-4307, USA; Email: jorlin@mit.edu

Abraham P. Punnen, Department of Mathematical Sciences, University of New Brunswick, PO Box 5050, Saint John, New Brunswick, Canada E2L 4L5, Email: punnen@unbsj.ca

Andreas S. Schulz, Sloan School of Management, Office E53-361, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA 02139-4307, USA, Email: schulz@mit.edu