MATHEMATICS OF OPERATIONS RESEARCH
Vol. 27, No. 4, November 2002, pp. 681–692
Printed in U.S.A.

THE COMPLEXITY OF GENERIC PRIMAL ALGORITHMS FOR SOLVING GENERAL INTEGER PROGRAMS

ANDREAS S. SCHULZ AND ROBERT WEISMANTEL

Primal methods constitute a common approach to solving (combinatorial) optimization problems. Starting from a given feasible solution, they successively produce new feasible solutions with increasingly better objective function value until an optimal solution is reached. From an abstract point of view, an augmentation problem is solved in each iteration. That is, given a feasible point, these methods find an augmenting vector, if one exists. Usually, augmenting vectors with certain properties are sought to guarantee the polynomial running time of the overall algorithm.

In this paper, we show that one can solve every integer programming problem in polynomial time provided one can efficiently solve the directed augmentation problem. The directed augmentation problem arises from the ordinary augmentation problem by splitting each direction into its positive and its negative part and by considering linear objectives on these parts. Our main result states that in order to get a polynomial-time algorithm for optimization it is sufficient to efficiently find, for any linear objective function in the positive and negative part, an arbitrary augmenting vector.

This result also provides a general framework for the design of polynomial-time algorithms for specific combinatorial optimization problems. We demonstrate its applicability by considering the min-cost flow problem, by giving a novel algorithm for linear programming over unimodular spaces, and by providing a different proof that for $0/1$-integer programming an efficient algorithm solving the ordinary augmentation problem suffices for efficient optimization. Our main result also implies that directed augmentation is at least as hard as optimization. In other words, for an NP-hard problem already the task of finding a directed augmenting vector in polynomial time is hopeless, unless $\mathrm{P} = \mathrm{NP}$. We illustrate this kind of consequence with the help of the knapsack problem.

1. Introduction. During the last ten years, there has been a strong, renewed interest in so-called test sets for integer programming problems; see, e.g., Conti and Traverso (1991), Cornuéjols et al. (1997), Hosten and Sturmfels (1995), Sturmfels and Thomas (1997), Sturmfels et al. (1995), Thomas (1995), Thomas and Weismantel (1996), and Urbaniak et al. (1997). This research has not only led to stimulating new insights, but also it has raised new questions. Prompted by the more abstract point of view, these questions often involve natural generalizations of concepts used in the design of primal algorithms for specific combinatorial optimization problems. The questions we address in this paper are of this type. They basically have their origin in the following algorithmic problem. Given a test set for an integer program, i.e., given a set $\mathcal{T}$ of vectors with the property that any feasible solution x to the integer program is optimal if and only if there is no $z \in \mathcal{T}$ such that $x + z$ gives a better feasible solution, what is the complexity of determining an optimal solution to the integer program? We do not require the test set to be explicitly given but only assume to have implicit access by some kind of augmentation oracle. Essentially, given a feasible point to the integer program and an objective function, such an oracle asserts either that the point is optimal with respect to the objective function, or if not, it provides a better feasible point. We show that having a directed augmentation oracle at our disposal enables us to solve the corresponding integer programming problem in (oracle-) polynomial time. In the course of the proof, we utilize the fact that a certain maximum-ratio augmentation

Received June 1, 1999; revised July 11, 2001, and May 5, 2002.
MSC 2000 subject classification. Primary: 90C10, 90C27.
OR/MS subject classification. Primary: Programming/integer.
Key words. Integer programming, combinatorial optimization, computational complexity, primal algorithms, augmentation problem.

0364-765X/02/2704/0681/$05.00
1526-5471 electronic ISSN, © 2002, INFORMS

problem can be solved in oracle-polynomial time as well. The fact that the solvability of a maximum-ratio augmentation problem—the ability to efficiently find an improving direction that is maximal in some average sense—implies the polynomial-time solvability of the original maximization problem is interesting on its own. Therefore, we also elaborate on the power of different maximum mean augmentation oracles.

Throughout the paper, we consider integer programming problems of the following form. We are given an integral $m \times d$-matrix A and integral vectors w, b, and u of compatible dimension, and the goal is to find an optimal solution to

$$(1) \qquad \max\{wx\colon Ax = b, 0 \le x \le u, x \in \mathbb{Z}^d\}.$$

This class of problems hosts, as special cases, virtually every combinatorial optimization problem. Several known polynomial-time algorithms for solving specific problems are of primal nature. That is, given a feasible solution x^0, they successively produce new feasible solutions $x^1, x^2, \ldots$ with $wx^0 < wx^1 < wx^2 < \cdots$ until an optimal solution is reached. Most primal algorithms, however, need to make a special choice of the augmenting vector $x^{i+1} - x^i$ in each iteration in order to result in a polynomial number of overall iterations. We give two examples. In augmenting-path algorithms for the max flow problem, a shortest augmenting path or one with largest residual capacity helps to obtain a polynomial-time algorithm (Edmonds and Karp 1972). In cycle-canceling algorithms for the min-cost flow problem, in each iteration, flow is to be augmented along a negative cycle with, e.g., minimum (mean) cost (Goldberg and Tarjan 1989).

Main result. We present two primal algorithms for solving the integer programming problem (1). Both algorithms assume that at any feasible point x we can find a feasible direction $z = z^+ - z^-$ with $z^+, z^- \ge 0$ such that $w'z^+ + w''z^- > 0$, for any pair of vectors $w', w'' \in \mathbb{Q}^d$, if such a direction exists. If we have an oracle that solves this so-called directed augmentation problem, both algorithms have oracle-polynomial running time. We refer the reader to Grötschel, Lovász, and Schrijver (1988) for a thorough discussion of oracle-polynomial algorithms. Whenever we say that an algorithm performs L calls to an oracle, it implicitly also holds that the number of additional elementary operations is bounded by L as well. Consequently, if one has a polynomial-time algorithm that simulates the oracle, the overall algorithm runs in polynomial time.

Both algorithms provide a general design tool to reduce the solution of a combinatorial optimization problem to a sequence of directed augmentation problems. We will show that the directed augmentation problem often admits a natural interpretation as another combinatorial problem that is—in some sense—easier than the original one. This framework therefore leads to potentially novel algorithms for every polynomial-time solvable problem. Despite its universality, the running time of the faster of the two algorithms is relatively small. If we define $W := \max_j |w_j|$ and $U := \max_j u_j$, it performs $O(d \log(dWU))$ calls to the directed augmentation oracle.

Related work. The results presented here were originally motivated by questions about the complexity of the standard test set algorithm or the generic local search algorithm and our work on 0/1-integer programs (Schulz et al. 1995). Therein we show that for problems of type (1) with $U = 1$, augmentation and optimization are strongly polynomial-time equivalent. The same result was independently obtained by Grötschel and Lovász (1995). It is special to 0/1-integer programs, however, that the diameter of the skeleton of the convex hull of all feasible solutions is bounded by the dimension d (Naddef 1989). No similar result is known for general integer programs or polytopes in general. In fact, this is one of the major open problems in discrete geometry and is closely related to the equally unknown existence of a pivot rule that turns the simplex algorithm into a polynomial-time algorithm for linear programming. The results in this paper are based on significantly different techniques that identify a sufficiently short augmenting path through the interior of the convex hull of feasible solutions, instead of along its edges.

On the other hand, our work is an extension of Wallacher's work on interior-point-like algorithms to solve the min-cost flow problem (Wallacher 1992, Wallacher and Zimmermann 1992). In fact, in the actual context of the min-cost flow problem the two above-mentioned algorithms are attributable to Wallacher. Later, McCormick and Shioura (1996) extended one of Wallacher's algorithms to linear programming over unimodular spaces. Our algorithms and analyses generalize both the work of Wallacher and that of McCormick and Shioura. Subsequently, Wayne (1999) succeeded in extending Wallacher's techniques to derive the first combinatorial algorithm for the generalized min-cost flow problem. In turn, McCormick and Shioura (2000) extended Wayne's result to an oracle-polynomial time algorithm for linear programming.

2. From AUG$_{\pm}$ to OPT. In this section, we show that for every family of integer programs given by a directed augmentation oracle optimization is easy. Section 2.1 contains the precise statement of our main result. In §§2.2 and 2.3 we give two different proofs in the form of two polynomial-time algorithms for optimization, each of which only relies on the directed augmentation oracle. Whereas the first algorithm is conceptually easier, the second one achieves a better running time. Applications to a variety of specific combinatorial optimization problems are given in §2.4.

2.1. The main result. Given an instance defined by an integer matrix A, by integer right-hand side and upper bound vectors b and u, and by the feasible region $\mathcal{F} := \{x \in \mathbb{Z}^d \colon Ax = b, 0 \leq x \leq u\}$, we want to solve the following integer programming problem.

Optimization (OPT).
Given a vector $w \in \mathbb{Z}^d$ and a point $x^0 \in \mathcal{F}$,
find a point $x^* \in \mathcal{F}$ that maximizes wx on $\mathcal{F}$.

Notice that we always assume to know an initial feasible point $x^0 \in \mathcal{F}$. Of course, finding such a point is NP-hard in general, yet easy for many problems. The only information we are given besides u, w, and x^0 is an oracle that we can call to solve a directed augmentation problem.

Directed Augmentation (AUG$_{\pm}$).
Given $w', w'' \in \mathbb{Q}^d$ and $x \in \mathcal{F}$, find $z^+, z^- \in \mathbb{Z}^d$ such that

$$\begin{aligned} w'z^+ + w''z^- &> 0, & 0 &\leq z^+ \leq u - x, \\ A(z^+ - z^-) &= 0, & 0 &\leq z^- \leq x, \end{aligned}$$

or assert that no such vectors exists.

One could actually distinguish between two versions of the directed augmentation problem. In the first one, z^+ and z^- are restricted to be the positive part and the negative part of $z = z^+ - z^-$, respectively; i.e., $z^+z^- = 0$. In the second one, they are not. It turns out that our algorithms and analyses work in either case. In the former one, we may interpret the "objective function" of the directed augmentation problem as the simple nonlinear function $\sum_{j=1}^{d}(w'_j \max\{0, z_j\} + w''_j \max\{0, -z_j\})$. For ease of presentation, our exposition will stick to this version; i.e., throughout the paper we assume that $z^+z^- = 0$.

Apparently, starting out from an (AUG$_\pm$) oracle is a stronger assumption than requiring an augmenting direction with respect to a linear objective function in the original z-variables. Nevertheless, it is still a reasonable assumption that is often fulfilled, as, e.g., the min-cost flow problem shows. For the min-cost flow problem, the matrix A corresponds to the node-arc incidence matrix of the given network; u_j is the capacity of arc j. The augmentation problem in the original variables would ask for a negative-cost circulation in the original graph in which some arcs would have negative lower capacities. Splitting z into z^+ and z^- simply corresponds to introducing for every arc its reverse arc and placing lower bounds 0 on every arc. In other words, the directed augmentation problem is the same kind of problem, but in the residual graph (or augmenting or auxiliary graph as it is sometimes called). In this model, it does not matter, of course, that different costs may be assigned to an arc and its reverse. It is this interpretation of the min-cost flow (AUG$_\pm$) problem that induced us to call it the directed augmentation problem in general.

We show next that for arbitrary integer programs it is sufficient to find, for linear objective functions in (z^+, z^-), an arbitrary augmenting vector in order to design an efficient algorithm for optimization; we do not need a "best" one in any respect. In particular, for the min-cost flow problem any negative-cost dicycle in the residual graph suffices.

The following is our main result.

THEOREM 2.1. *Let $\mathcal{F}$ be given by a feasible point $x^0 \in \mathcal{F}$, by the upper bound vector u, and by an oracle to solve the directed augmentation problem* (AUG$_\pm$). *Then, the corresponding optimization problem* (OPT) *can be solved in oracle-polynomial time.*

We prove Theorem 2.1 by presenting two explicit algorithms. The first algorithm utilizes the directed augmentation oracle to solve a maximum-ratio augmentation problem, which is, in turn, used to solve the original optimization problem. The second algorithm does not attempt to exactly solve a maximum-ratio problem in every iteration. By computing sufficiently close approximations to a maximum-ratio direction, it essentially maintains the total number of iterations, but a single iteration is significantly less expensive.

2.2. Algorithm I—Maximum-ratio augmentation. In this section, we present a first algorithm in support of Theorem 2.1. It consists of two separate steps. First, we show that the access to an oracle that returns an augmenting vector that maximizes the ratio of improvement (with respect to the original objective function) to the cost of this augmenting direction (measured in terms of the proximity of the new feasible point to the boundary of the feasible region) enables us to solve the optimization problem efficiently. Then we prove that the directed augmentation oracle can emulate the maximum-ratio augmentation oracle, thus proving Theorem 2.1. We pay for the distinction of the two steps with a higher overall running time, but we are rewarded with insight. This insight will guide the design and analysis of Algorithm II. The intermediate problem that we utilize is the following:

Maximum-Ratio Augmentation (MRA).
Given $w \in \mathbb{Z}^d$ and $x \in \mathcal{F}$, find a feasible direction $z = z^+ - z^- \in \mathbb{Z}^d$ such that $wz > 0$ and z maximizes

$$\frac{wz}{p(x)z^+ + n(x)z^-},$$

or assert that no such z exists.

Here, the vectors $p(x), n(x) \in \mathbb{Q}^d$ are defined as follows, for every $x \in \mathcal{F}$:

$$p_j(x) = \begin{cases} \dfrac{1}{u_j - x_j}, & \text{if } x_j < u_j, \\ \infty, & \text{if } x_j = u_j, \end{cases} \quad \text{and} \quad n_j(x) = \begin{cases} \dfrac{1}{x_j}, & \text{if } x_j > 0, \\ \infty, & \text{if } x_j = 0. \end{cases}$$

In other words, every component of (z^+, z^-) is weighted with the reciprocal of the corresponding "residual capacity." In particular, unlike other well-known maximum-ratio problems (see §3 below) the denominator directly depends on the current solution x. The intuition underlying this definition is to guarantee sufficiently long steps in each iteration. The closer x_j is to one of its limits, the more expensive it is to go further in this direction. It is this kind of interior point philosophy that makes the algorithm work and which, in fact, led Wallacher (1992) to introduce this ratio in the context of min-cost flow problems.

We call an augmenting vector z at a feasible point x *exhaustive* if $x+2z$ is not feasible. Notice that if one is given a direction maximizing the ratio as well as the vector u of upper bounds, an exhaustive direction that also maximizes the ratio can be computed in $O(d+\log U)$ time.

```
ALGORITHM I
(1) Let $x$ be a feasible solution.
(2) Call the (MRA) oracle with input $x$ and $w$.
(3) IF the oracle outputs "there is no feasible
    augmenting direction,"
(4) THEN STOP. The current $x$ is optimal.
(5) ELSE let $z$ be a feasible augmenting direction
    that maximizes $wz/(p(x)z^+ + n(x)z^-)$ and which
    is exhaustive. Set $x := x + z$. GOTO (2).
```

THEOREM 2.2. *Let $\mathscr{F}$ be given by a feasible point $x^0 \in \mathscr{F}$, by the upper bound vector u, and by an oracle to solve the maximum-ratio augmentation problem* (MRA). *Then, for any $w \in \mathbb{Z}^d$, Algorithm* I *solves the corresponding optimization problem* (OPT) *with $O(d\log(dWU))$ calls to the* (MRA) *oracle.*

PROOF. Let x be the feasible point at the current iteration and let z be the (exhaustive) direction returned by the (MRA) oracle. Furthermore, let $z^* = x^* - x$ be the direction to an optimal solution x^* with respect to w.

By the choice of $p(x)$ and $n(x)$, $p(x)(z^*)^+ + n(x)(z^*)^- \le d$. Since z is exhaustive, there exists a coordinate j such that either $x_j + 2z_j > u_j$, i.e., $z_j^+ > (u_j - x_j)/2$, or $x_j + 2z_j < 0$, i.e., $z_j^- > x_j/2$. These two observations together with the property that z is a solution to (MRA) implies

$$wz \ge wz^* \frac{p(x)z^+ + n(x)z^-}{p(x)(z^*)^+ + n(x)(z^*)^-} \ge \frac{wz^*}{2d}.$$

Consequently, the improvement in every step is at least a $(1/2d)$-fraction of the best possible improvement. Since the gap in the objective function value between the optimal solution x^* and the initial solution x^0 is $O(dWU)$, Algorithm I terminates with an optimal solution after $O(d\log(dWU))$ calls to the (MRA) oracle. □

To complete the first proof of Theorem 2.1, it remains to show that the maximum-ratio augmentation problem can be solved by means of the given directed augmentation oracle.

LEMMA 2.3. *Let $\mathscr{F}$ be given by an oracle that solves the directed augmentation problem* $(\mathrm{AUG}_\pm)$. *Then, given any feasible point $x \in \mathscr{F}$ and any objective function vector $w \in \mathbb{Z}^d$, one can determine with $O(d\log(dWU))$ calls to the* $(\mathrm{AUG}_\pm)$ *oracle a feasible augmenting direction z that maximizes the ratio $wz/(p(x)z^+ + n(x)z^-)$, if one exists. Otherwise, one call to the* $(\mathrm{AUG}_\pm)$ *oracle suffices to assert that x is optimal with respect to w. That is, the* *maximum-ratio augmentation problem* (MRA) *can be solved with* $O(d\log(dWU))$ *calls to the* ($\text{AUG}_{\pm}$) *oracle.*

PROOF. We first use the ($\text{AUG}_{\pm}$) oracle to check whether the given point x is optimal. If it is not, we solve the maximum-ratio problem

$$\max_z \frac{wz}{p(x)z^+ + n(x)\,z^-}$$

by repeated applications of the ($\text{AUG}_{\pm}$) oracle, following the standard fractional programming approach. For a given estimate μ of the maximum-ratio value, the objective function input into the directed augmentation oracle is

$$wz - \mu(p(x)z^+ + n(x)z^-). \qquad (2)$$

Depending on the output, μ is either a lower bound for the maximum ratio ("augmenting direction w.r.t. (2) found") or an upper bound ("there is no augmenting direction w.r.t. (2)"). We use binary search to find the maximum ratio μ^* and a corresponding direction. For the running time, note that WU is an upper bound on μ^* and also that the minimum difference between two distinct ratios is $\Omega(1/(d^2U^{2d}))$. $\square$

Together, Theorem 2.2 and Lemma 2.3 imply the following result if we assume that the (MRA) oracle is simulated by the ($\text{AUG}_{\pm}$) oracle in Algorithm I.

COROLLARY 2.4. *Let* $\mathcal{F}$ *be given by a feasible point* $x^0 \in \mathcal{F}$, *by the upper bound vector* u, *and by an oracle that solves the directed augmentation problem* ($\text{AUG}_{\pm}$). *Then, for any objective function* $w \in \mathbb{Z}^d$, *Algorithm* I *solves the corresponding optimization problem* (OPT) *with* $O(d^2\log^2(dWU))$ *calls to the* ($\text{AUG}_{\pm}$) *oracle.*

2.3. Algorithm II—Scaling. The second algorithm to prove Theorem 2.1 does not aim at repeatedly determining a maximum-ratio direction (which is costly) but only at determining an augmenting direction (for a properly chosen objective) while keeping essentially the same number of iterations. It is inspired by a similar variant of Wallacher's (1992) min-cost flow algorithm.

```
ALGORITHM II
(1) Let $x$ be a feasible solution.
(2) $\mu := 2WU$.
(3) Call the ($\text{AUG}_{\pm}$) oracle with input $x$ and $\omega(z) :=$
      $wz - \mu\,(p(x)\,z^+ + n(x)\,z^-)$.
(4) IF the oracle outputs "there is no feasible direction $z$
      with $\omega(z) > 0$", THEN
    (a) IF $\mu < 1/d$, THEN STOP. The current $x$ is optimal.
    (b) ELSE (i.e., $\mu \geq 1/d$), $\mu := \mu/2$. GOTO (3).
(5) ELSE let $z$ be a feasible and exhaustive direction with
      $\omega(z) > 0$. Set $x := x + z$. GOTO (3).
```

Notice first that Algorithm II outputs a correct answer when it terminates. Since $(p(x)z^+ + n(x)z^-) \leq d$ for every direction z returned by the ($\text{AUG}_{\pm}$) oracle, it follows from the integrality of data that $\omega(z) \leq 0$ implies $wz \leq 0$ if $\mu < 1/d$ (Case 4a). As for the running time, observe that the initial value of the scaling parameter μ is an upper bound for the ratio of the (MRA) problem at any feasible point x. We halve μ only when it is an overestimate of the maximum ratio at the current point. Consequently, when we determine a direction z with $\omega(z) > 0$ for the current μ, it has a ratio at most a factor of 2 smaller than the maximal one. The last observation is crucial in order to recycle the analysis given in the proof of Theorem 2.2.

THEOREM 2.5. *Let $\mathcal{F}$ be given by a feasible point $x^0 \in \mathcal{F}$, by the upper bound vector u, and by an oracle that solves the directed augmentation problem* (AUG$_\pm$). *Then, for any objective function $w \in \mathbb{Z}^d$, Algorithm* II *solves the corresponding optimization problem* (OPT) *with $O(d\log(dWU))$ calls to the* (AUG$_\pm$) *oracle.*

PROOF. We observe first that μ is halved at most $O(\log(dWU))$ times. That is, there are at most $O(\log(dWU))$ scaling phases. A scaling phase consists of all iterations of Steps (3) and (4) in which the parameter value μ remains unchanged. We aim to show that the number of calls to the (AUG$_\pm$) oracle within a scaling phase does not exceed $4d$.

Consider for each scaling phase with parameter value μ the corresponding sequence of iterates, say $x^{0,\mu}, x^{1,\mu}, \ldots$. That is, $x^{0,\mu}$ is the current solution after we had invoked Case 4b in the previous iteration of the algorithm. Therefore, we know that

$$\frac{w(x^* - x^{0,\mu})}{p(x^{0,\mu})(x^* - x^{0,\mu})^+ + n(x^{0,\mu})(x^* - x^{0,\mu})^-} \le 2\mu, \tag{3}$$

where x^* denotes an optimal solution. On the other hand, if we consider the transition from $x^{i,\mu}$ to $x^{i+1,\mu}$, it follows from $\omega(x^{i+1,\mu} - x^{i,\mu}) > 0$, from the fact that $x^{i+1,\mu} - x^{i,\mu}$ is exhaustive, and from Equation (3) that

$$\begin{aligned} w(x^{i+1,\mu} - x^{i,\mu}) &\ge \mu(p(x^{i,\mu})(x^{i+1,\mu} - x^{i,\mu})^+ + n(x^{i,\mu})(x^{i+1,\mu} - x^{i,\mu})^-) \\ &\ge \frac{\mu}{2} \\ &\ge \frac{1}{4}\frac{w(x^* - x^{0,\mu})}{p(x^{0,\mu})(x^* - x^{0,\mu})^+ + n(x^{0,\mu})(x^* - x^{0,\mu})^-} \\ &\ge \frac{1}{4d} w(x^* - x^{0,\mu}). \end{aligned}$$

Hence, there can be at most $4d$ iterations per scaling phase. Therefore, Algorithm II terminates with an optimal solution after at most $O(d\log(dWU))$ calls to the (AUG$_\pm$) oracle. □

Note that Theorem 2.2 and Theorem 2.5 only imply that both Algorithm I and Algorithm II have weakly polynomial running time, respectively. In fact, McCormick and Shioura (1996, 2000) proved that Algorithm I is not a strongly polynomial-time algorithm, even for the min-cost flow problem.

2.4. Examples.

In this section, we aim to briefly illustrate the potential of the algorithmic framework provided by Theorem 2.1 (and especially by Algorithm II) with a few selected examples. We show that the directed augmentation problem often has a natural combinatorial interpretation and that this holds even when the formulation contains slack or surplus variables.

The min-cost flow problem. Our discussion of the min-cost flow problem was already started in §2.1. Recall that solving the directed augmentation problem simply amounts to finding a negative-cost dicycle in a given arc-weighted digraph. Since one can detect a negative dicycle in $O(\min\{nm, \sqrt{n}m\log(nW)\})$ time (Karp 1978, Orlin and Ahuja 1992), Algorithm II solves the min-cost flow problem with integral data in time $O(m\log(nUW)\min\{nm, \sqrt{n}m\log(nW)\})$. Here, n and m denote the number of nodes and arcs of the given network, respectively. Interestingly, among cycle-canceling algorithms the running time of the generic Algorithm II is only beaten by Goldberg and Tarjan's (1989) cancel-and-tighten algorithm, which has running time $O(mn\log n\log(nW))$, and by the $O(m(m+n\log n)\log(nU))$ capacity-scaling algorithm of Sokkalingam et al. (2000).

Goldberg and Tarjan's bound is within log factors of the fastest running time of any known min-cost flow algorithm. We refer the reader to Shigeno et al. (2000) and Sokkalingam et al. (2000) for surveys of cycle-canceling algorithms and to Ahuja et al. (1993) for min-cost flow algorithms in general.

Both Algorithms I and II are known in the min-cost flow context; see Wallacher (1992).

Linear programming in unimodular spaces. McCormick and Shioura (1996) extended Wallacher's min-cost flow algorithm (version I) to the problem of $\max\{wx\colon Ax = b, 0 \le x \le u\}$ where A is totally unimodular. For their algorithm, they obtain the same running time as stated in Theorem 2.2. However, they solve every (MRA) problem occurring within Algorithm I by solving a sequence of $O(\rho \min\{\phi, \eta\})$ linear programs where ρ is the rank of A, ϕ is the maximal number of nonzero components of nonnegative circuits of A, and η is the number of these circuits.

One may use Algorithm II instead. Then, Theorem 2.5 together with the observation that ($\text{AUG}_\pm$) amounts to solving one simple linear program gives a different algorithm for linear programming in unimodular spaces. It solves $O(d \log(dUW))$ simple linear programs in total. Depending on the size of U and W, this algorithm can be faster than the algorithm proposed in McCormick and Shioura (1996). In a subsequent paper, McCormick and Shioura (2000) also propose the use of Algorithm II.

The knapsack problem. So far, we have discussed problems that are naturally formulated as integer programs with equality constraints, i.e., as programs of the form (1). Although it is well known that any integer program can be brought into this form by the introduction of auxiliary variables, we want to exemplify with the help of the NP-hard knapsack problem that the directed augmentation problem may still have a nice combinatorial interpretation. While the standard knapsack problem is $\max\{wx\colon ax \le b, 0 \le x \le u, x \in \mathbb{Z}^d\}$, the version that fits our framework is $\max\{wx\colon ax + s = b, 0 \le x \le u, 0 \le s \le b, x \in \mathbb{Z}^d, s \in \mathbb{Z}\}$. As for the directed augmentation problem, s is regarded as an additional item, and we have a "+" and a "−" copy of every item. The "+" copy and the "−" copy of an item j usually have a different value but the same weight a_j. The problem is to select "+" and "−" items of positive total value such that the weight of items chosen from each side is equal. Multiple selection of the same item j is permitted as long as its supply ($u_j - x_j$ for a "+" copy, x_j for a "−" copy) is not exceeded. Notice that in this case Theorem 2.1 implies that this problem is NP-hard.

0/1-integer programming. It is obvious that for 0/1-integer programming, i.e., $\mathcal{F} \subseteq \{0, 1\}^d$, directed augmentation and ordinary augmentation in the original variables are equivalent. Consequently, Algorithms I and II constitute new proofs of the equivalence of optimization and augmentation. The previous algorithm was based on bit scaling and needed $O(d \log W)$ calls to the augmentation oracle (Grötschel and Lovász 1995, Schulz et al. 1995).

3. Maximum mean augmentation.

The driving force behind Theorem 2.1 is the ability to (approximately) solve the maximum-ratio problem (MRA) in polynomial time. Different ratios have been considered in the literature; see, e.g., Shigeno et al. (2000) and Sokkalingam et al. (2000) for an overview of corresponding algorithms for the min-cost flow problem. In this section we discuss the power of two (other) maximum mean augmentation oracles for general integer programming. The first maximizes the improvement per element in the support of the augmenting vector, the second averages over its l_1-norm. Notice that in the case of 0/1-integer programming problems, all three maximum ratio augmentation problems coincide.

The support $\mathrm{supp}(z)$ of a vector $z \in \mathbb{R}^d$ is defined as $\mathrm{supp}(z) := \{j \in \{1, \ldots, d\}\colon z_j \neq 0\}$. If the denominator of the maximum-ratio oracle contains the cardinality of the support, it is relatively easy to show that, given such a maximum mean augmentation oracle, one can optimize in oracle-polynomial time. Even better, we do not even need to assume that the integer program is given in equation form.

THEOREM 3.1. *Let $\mathcal{F} \subseteq \{x \in \mathbb{Z}^d\colon 0 \leq x \leq u\}$ be given by a feasible point $x^0 \in \mathcal{F}$ and by an oracle that solves the following problem*:

> *Given $w \in \mathbb{Z}^d$ and a point $x \in \mathcal{F}$, find a feasible direction $z \in \mathbb{Z}^d$ such that $wz > 0$ and z maximizes*
> $$\frac{wz}{|\mathrm{supp}(z)|},$$
> *or assert that no such z exists.*

Then, there exists an algorithm that solves $\max\{wx\colon x \in \mathcal{F}\}$ *with $O(d\log(dWU))$ calls to the oracle, for any objective function $w \in \mathbb{Z}^d$.*

PROOF. The algorithm is the obvious one. We simply invoke the oracle iteratively with the solution it produced in the previous step until we reach an optimum. We only have to show that the number of iterations (and thus calls to the oracle) is sufficiently small.

Let $w \in \mathbb{Z}^d$ be an arbitrary objective function and let x^* be an optimal solution to the problem $\max\{wx\colon x \in \mathcal{F}\}$. Consider an arbitrary iteration of the algorithm with current (nonoptimal) solution x. Let z be the output of the oracle and $z^* = x^* - x$. By the properties of the oracle, we know that

$$\frac{wz}{|\mathrm{supp}(z)|} \geq \frac{wz^*}{|\mathrm{supp}(z^*)|}.$$

Since the cardinality of the support of any augmenting vector is at least one, but at most d, it follows that $wz \geq wz^*/d$. Consequently, the algorithm terminates with an optimal solution after $O(d\log(dWU))$ iterations. □

The situation is less obvious if the denominator is formed by the l_1-norm of the augmenting vector. Nevertheless, efficient optimization is still possible if we additionally assume that the circuits of A are reasonably small. A vector $z \in \mathbb{Z}^d \backslash \{0\}$ satisfying $Az = 0$ is a circuit of A if the components of z are relatively prime and the support of z is minimal with respect to inclusion. The input size $\langle A \rangle$ of a matrix A is the sum of the encoding lengths of its entries, in binary representation. An improving direction z at point x is called reducible if there exist two other directions, say, $z_1 \neq 0$ and $z_2 \neq 0$ such that $x + z_1$ and $x + z_2$ are also feasible, $z = z_1 + z_2$, and $z_1^+ \leq z^+$ as well as $z_1^- \leq z^-$. If z is not reducible, we say it is irreducible.

THEOREM 3.2. *Let $\mathcal{F} = \{x \in \mathbb{Z}^d\colon Ax = b, 0 \leq x \leq u\}$ be given by a feasible point x^0, by the upper bound vector u, and by an oracle that solves the following problem*:

> *Given $w \in \mathbb{Z}^d$ and $x \in \mathcal{F}$, find a feasible direction $z \in \mathbb{Z}^d$ such that $wz > 0$, z is irreducible, and z maximizes*
> $$\frac{wz}{\|z\|_1},$$
> *or assert that no such z exists.*

If the l_1-norm of the circuits of A is bounded from above by a polynomial in d and $\langle A \rangle$, then there exists an algorithm that solves the optimization problem $\max\{wx\colon x \in \mathcal{F}\}$ *in oracle-polynomial time, for any objective function $w \in \mathbb{Z}^d$.*

Before the proof of Theorem 3.2 we mention without proof that the maximum mean augmentation oracle needed in Theorem 3.2 can be simulated by the directed augmentation oracle ($\mathrm{AUG}_{\pm}$).

The proof of Theorem 3.2 requires some preparation. Again, the overall plan is to show that the key measure of proximity to optimality, i.e., the considered ratio, cannot start out too large, does not need to end up too small, and decreases at a geometric rate. Not surprisingly, this outline reveals similarities, e.g., to proving that minimum mean cycle-canceling is a polynomial-time procedure for min-cost flow problems (see Goldberg and Tarjan 1989 or Ahuja et al. 1993, 376–380). Notice, however, that their proof uses explicit dual information, which is not available in the general case. Dual information is also used by McCormick et al. (1994) to extend the arguments of Goldberg and Tarjan in order to obtain the analog to Theorem 3.2 for general linear programs.

Once again, the strategy is to repeatedly call the oracle starting from the initial feasible solution x^0. Let $z^1, z^2, \ldots$ be the directions returned by the oracle. It is easy to determine positive integer scalars $\lambda_1, \lambda_2, \ldots$ such that $\lambda_1 z^1, \lambda_2 z^2, \ldots$ are still feasible, but also exhaustive. These are the directions we use. Observe that the ratio of improvement in the original objective function to the l_1-norm is the same for z^i and $\lambda_i z^i$, for all i.

LEMMA 3.3. *Let x be a feasible point, let z be a feasible augmenting vector in x that maximizes $\mu(x) := wz/\|z\|_1$, and let z' be a feasible augmenting vector in $x+z$ that maximizes $\mu(x+z)$. Assume that $\|z+z'\|_1 \le \|z\|_1 + \alpha\|z'\|_1$ for some scalar $0 < \alpha \le 1$. Then, $\alpha\mu(x) \ge \mu(x+z)$.*

PROOF. Since $z+z'$ is a feasible augmenting vector in x, and z is one maximizing the ratio, we obtain

$$\frac{wz}{\|z\|_1} \ge \frac{w(z+z')}{\|z+z'\|_1} \ge \frac{wz+wz'}{\|z\|_1+\alpha\|z'\|_1}, \quad \text{so that } \alpha\frac{wz}{\|z\|_1} \ge \frac{wz'}{\|z'\|_1}. \quad \square$$

In particular, Lemma 3.3 implies that the sequence $\mu(x^0)$, $\mu(x^1)$, $\mu(x^2), \ldots$ of maximum ratios is nonincreasing. In addition, if two consecutive directions of improvement differ in sign in at least one coordinate, then the ratio corresponding to the second direction is significantly smaller than the ratio corresponding to the first direction. First, however, we need the following observation and lemma.

OBSERVATION 3.4. Let z and z' be two integral vectors such that there exists a coordinate j with $z_j \cdot z'_j < 0$. Then, $\|z+z'\|_1 \le \|z\|_1 + \|z'\|_1 - 2$.

We next show that there cannot be too many consecutive improving vectors with no coordinate pointing into opposite directions.

LEMMA 3.5. *Let $\lambda_i z^i, \lambda_{i+1} z^{i+1}, \ldots, \lambda_q z^q$ be a subsequence of exhaustive augmenting vectors such that no two of them point into opposite directions with respect to some coordinate. Then, the number $q-i+1$ of vectors in this sequence is $O(d \log U)$.*

PROOF. Since the vectors $\lambda_i z^i, \lambda_{i+1} z^{i+1}, \ldots, \lambda_q z^q$ are exhaustive, in each step l there is a coordinate, say j, such that either $u_j - x_j^l < (u_j - x_j^{l-1})/2$ or $x_j^l < x_j^{l-1}/2$. Here, x^l is the feasible point determined in iteration l. That is, $x^l = x^{l-1} + \lambda_l z^l$. The claim follows. $\square$

We are now ready to prove Theorem 3.2.

PROOF OF THEOREM 3.2. Lemma 3.5 implies that after a subsequence of at most $O(d \log U)$ iterations, say, from i to q, there is an augmenting vector z^{q+1} such that $z_j^l \cdot z_j^{q+1} < 0$ for some coordinate $j \in \{1, 2, \ldots, d\}$ and some iterate $l \in \{i, i+1, \ldots, q\}$ (if we have not yet arrived at an optimal solution). We may assume that l is the last index in this sequence with this property. Therefore, we may apply z^{q+1} at x^l; i.e., $x^l + z^{q+1}$ is feasible. Since z^{q+1} is irreducible, z^{q+1} belongs to the Hilbert basis of the pointed polyhedral cone generated by the circuits of A that lie in the same orthant as z^{q+1}. Thus, its

l_1-norm is polynomially bounded in d and the maximal l_1-norm of a circuit of A. We therefore obtain $\|z^{q+1}\|_1 \leq p(d, \langle A\rangle)$ for some polynomial p. Then, Observation 3.4 implies that $\|z^l + z^{q+1}\|_1 \leq \|z^l\|_1 + (1 - 1/p(d, \langle A\rangle))\|z^{q+1}\|_1$. Consequently, by Lemma 3.3 we have that $\mu(x^q) \leq (1 - 1/p(d, \langle A\rangle))\mu(x^{l-1})$.

Hence, we observe a significant improvement in the maximum mean ratio after every sequence of at most $O(d \log U)$ consecutive iterations, in which the maximum mean ratio does not increase. It therefore follows that, after at most $O(p(d, \langle A\rangle) d \log U)$ iterations, we halve the maximum mean ratio. Since it is bounded from above by W and from below by $1/p(d, \langle A\rangle)$, it follows that the overall number of calls to the oracle is bounded by a polynomial in the input size. $\square$

4. Concluding remarks.

The main purpose of this paper is to show that if one can solve the directed augmentation problem in polynomial time, it immediately follows that there is a polynomial-time algorithm for optimization. This result partially answers a question raised in Schulz et al. (1995).

The generic algorithms and their analyses that underly this implication provide a general framework for the design and analysis of algorithms for special integer programming and combinatorial optimization problems. We have pointed out that Algorithms I and II generalize some known algorithms for special combinatorial problems, and are new ones for some others.

On the other hand, we cannot expect to solve ($\text{AUG}_\pm$) in polynomial time if the corresponding optimization problem is NP-hard, unless $\text{P} = \text{NP}$. We may hope, however, to be able to solve ($\text{AUG}_\pm$) efficiently if we restrict ourselves to special augmenting vectors. Then, of course, by the algorithms described above we will not necessarily obtain an optimal solution but maybe a good one.

It seems to be an interesting question to what extent the results of this paper can be extended if one only has access to an augmentation oracle in the original variables; i.e., the directed version is not available. Either answer to this question is of importance. A positive answer would imply that the diameter of the skeleton of the convex hull of $\{x \in \mathbb{Z}^d\colon Ax = b, 0 \leq x \leq u\}$ is polynomially bounded in $\langle (A, b)\rangle$ and $\log U$, as we could assume to walk along the edges. Still, no polynomial bound on the diameter of polytopes is known in general.

Acknowledgments. The authors are grateful to Tom McCormick for pointing them to McCormick and Shioura (1996) and Wallacher (1992) after he attended a talk given by the first author on the results presented in §2.2 and §3 at the 2nd Workshop on Combinatorial Optimization in Aussois, France (February 1–7, 1998); to Tom McCormick and Maurice Queyranne for helpful comments on a preliminary version of this paper; and to two anonymous referees for their constructive comments. An extended abstract of this work appeared in the *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms* (Schulz and Weismantel 1999).

References

Ahuja, R. K., T. L. Magnanti, J. B. Orlin. 1993. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, Englewood Cliffs, NJ.

Conti, P., C. Traverso. 1991. Buchberger algorithm and integer programming. H. F. Mattson, T. Mora, T. R. N. Rao, eds. Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, No. 539. *Lecture Notes in Computer Science, Proc. of the 9th Internat. AAECC Sympos.* Springer, Berlin, Germany, 130–139.

Cornuéjols, G., R. Urbaniak, R. Weismantel, L. A. Wolsey. 1997. Decomposition of integer programs and of generating sets. R. Burkard, G. Woeginger, eds. Algorithms—ESA'97, No. 1284. *Lecture Notes in Computer Science, Proc. of the 5th Annual Eur. Sympos. on Algorithms*. Springer, Berlin, Germany, 92–103.

Edmonds, J., R. M. Karp. 1972. Theoretical improvements in algorithmic efficiency for network flow problems. *J. Assoc. Comput. Mach.* **19** 248–264.

Goldberg, A. V., R. E. Tarjan. 1989. Finding minimum-cost circulations by cancelling negative cycles. *J. Assoc. Comput. Mach.* **36** 873–886.

Grötschel, M., L. Lovász. 1995. Combinatorial optimization. R. Graham, M. Grötschel, L. Lovász, eds. Vol. II, Chap. 28. *Handbook of Combinatorics*, North-Holland, Amsterdam, The Netherlands, 1541–1597.

———, ———, A. Schrijver. 1988. Geometric Algorithms and Combinatorial Optimization, vol. 2. *Algorithms and Combinatorics*. Springer, Berlin, Germany.

Hosten, S., B. Sturmfels. 1995. GRIN: An implementation of Gröbner bases for integer programming. E. Balas, J. Clausen, eds. *Integer Programming and Combinatorial Optimization*, No. 920. *Lecture Notes in Computer Science*, Proc. of the 4th Internat. IPCO Conference. Springer, Berlin, Germany, 267–276.

Karp, R. M. 1978. A characterization of the minimum cycle mean in a digraph. *Discrete Math.* **23** 309–311.

McCormick, S. T., T. R. Ervolina, B. Zhou. 1994. Mean canceling algorithms for general linear programs, and why they (probably) don't work for submodular flow. Working Paper 94-MSC-011, University of British Columbia, Faculty of Commerce, Vancouver, Canada.

———, A. Shioura. 1996. A minimum ratio cycle canceling algorithm for linear programming problems with application to network optimization problems. Manuscript.

———, ———. 2000. Minimum ratio canceling is oracle polynomial for linear programming, but not strongly polynomial, even for networks. *Oper. Res. Lett.* **27** 199–207.

Naddef, D. 1989. The Hirsch conjecture is true for $(0, 1)$-polytopes. *Math. Programming* **45** 109–110.

Orlin, J. B., R. K. Ahuja. 1992. New scaling algorithms for the assignment and minimum cycle mean problems. *Math. Programming* **54** 41–56.

Schulz, A. S., R. Weismantel. 1999. An oracle-polynomial time augmentation algorithm for integer programming. *Proc. of the 10th Annual ACM–SIAM Sympos. on Discrete Algorithms*. Association for Computing Machinery (ACM) and Society for Industrial and Applied Mathematics (SIAM), Baltimore, MD, 967–968.

———, ———, G. M. Ziegler. 1995. $0/1$-integer programming: Optimization and augmentation are equivalent. P. Spirakis, ed. *Algorithms—ESA '95*, No. 979. *Lecture Notes in Computer Science, Proc. of the 3rd Annual Eur. Sympos. on Algorithms*. Springer, Berlin, Germany, 473–483.

Shigeno, M., S. Iwata, S. T. McCormick. 2000. Relaxed most negative cycle and most positive cut canceling algorithms for minimum cost flow. *Math. Oper. Res.* **25** 76–104.

Sokkalingam, P. T., R. K. Ahuja, J. B. Orlin. 2000. New polynomial-time cycle-canceling algorithms for minimum-cost flows. *Networks* **36** 53–63.

Sturmfels, B., R. R. Thomas. 1997. Variation of cost functions in integer programming. *Math. Programming* **77** 357–388.

———, R. Weismantel, G. M. Ziegler. 1995. Gröbner bases of lattices, corner polyhedra, and integer programming. *Contributions Algebra Geometry* **36** 281–298.

Thomas, R. R. 1995. A geometric Buchberger algorithm for integer programming. *Math. Oper. Res.* **20** 864–884.

———, R. Weismantel. 1996. Test sets and inequalities for integer programs. W. H. Cunningham, S. T. McCormick, M. Queyranne, eds. *Integer Programming and Combinatorial Optimization*, No. 1084. *Lecture Notes in Computer Science, Proc. of the 5th Internat. IPCO Conf.* Springer, Berlin, Germany, 16–30.

Urbaniak, R., R. Weismantel, G. M. Ziegler. 1997. A variant of the Buchberger algorithm for integer programming. *SIAM J. Discrete Math.* **10** 96–108.

Wallacher, C. 1992. Kombinatorische Algorithmen für Flußprobleme und submodulare Flußprobleme. Ph.D. Thesis, Technische Universität zu Braunschweig, Braunschweig, Germany.

———, U. Zimmermann. 1992. A combinatorial interior point method for network flow problems. *Math. Programming* **56** 321–335.

Wayne, K. D. 1999. A polynomial combinatorial algorithm for generalized minimum cost flow. *Proc. of the 31st Annual ACM Sympos. on Theory of Comput.* 11–18.

A. S. Schulz: Massachusetts Institute of Technology, Sloan School of Management, E53-361, 77 Massachusetts Avenue, Cambridge, Massachusetts 02139-4307; e-mail: schulz@mit.edu

R. Weismantel: Otto-von-Guericke-Universität Magdeburg, Fakultät für Mathematik, Universitätsplatz 2, D-39106 Magdeburg, Germany; e-mail: weismantel@imo.math.uni-magdeburg.de