Mathematical Programming 61 (1993) 39–52
North-Holland

A randomized scheme for speeding up algorithms for linear and convex programming problems with high constraints-to-variables ratio

Ilan Adler

Industrial Engineering and Operations Research Department, University of California, Berkeley, CA, USA

Ron Shamir

School of Mathematical Sciences, Sackler Faculty of Exact Sciences, Tel-Aviv University, Tel-Aviv, Israel

Received 12 November 1989
Revised manuscript received 26 May 1992

We extend Clarkson's randomized algorithm for linear programming to a general scheme for solving convex optimization problems. The scheme can be used to speed up existing algorithms on problems which have many more constraints than variables. In particular, we give a randomized algorithm for solving convex quadratic and linear programs, which uses that scheme together with a variant of Karmarkar's interior point method. For problems with n constraints, d variables, and input length L, if $n = \Omega(d^2)$, the expected total number of major Karmarkar's iterations is $O(d^2(\log n)L)$, compared to the best known deterministic bound of $O(\sqrt{n}\, L)$. We also present several other results which follow from the general scheme.

Key words: Linear programming, quadratic programming, convex programming, randomized algorithms, fixed dimension optimization problems, complexity.

1. Introduction

This paper deals with the *convex programming* optimization problem, that is, minimizing a convex function subject to a set of convex inequality constraints. Given an algorithm which solves such problems (or an algorithm for solving problems which are special cases of the convex programming problem), we show how to speed up the algorithm via randomization, when the number of constraints n is much larger than the number of variables d.

The convex programming problem has been the subject of numerous investigations (see e.g. [13, 15, 4]. Special cases of the problem include the important problems of linear programming, convex quadratic programming and separable convex programming, all of which have been intensively studied (see, e.g. [8, 14, 12] and the

Correspondence to: Dr. Ron Shamir, Department of Computer Science, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel-Aviv University, Ramat-Aviv, Tel-Aviv 69978, Israel.

references thereof). A major direction of investigation has been to obtain faster algorithms for the linear programming problem.

The novel algorithm introduced by Karmarkar [6] has been the origin of a plethora of interior-point algorithms for linear and convex quadratic programming problems, some of which have been shown to perform well in practice, as well as to have low polynomial complexity (see e.g. [9, 10]). Among these interior-point methods, in those with the best complexity the number of major iterations is proved to be $O(\sqrt{n}\, L)$, where each major iteration involves the inversion of a matrix. (Here L represents the binary input length, or, alternatively, the binary length of the largest subdeterminant in the constraints matrix.) A central theoretical challenge (with potential practical implications) is therefore to lower the number of major iterations as well as the overall complexity in such algorithms. One of our results substantially reduces the expected number of major iterations, when $d = O(n^{1/4}/\log^{1/2} n)$.

Our paper makes intensive use of the approach and techniques which were recently introduced by Clarkson [2] (see also [3]). Clarkson has shown how to speed-up the simplex method (or any vertex enumeration method) for linear programming via randomization, when d is small and $n \to \infty$. Specifically, he has devised a randomized algorithm for solving linear programs in an expected time complexity of

$$O(d^2 n) + \log n\, O(d)^{d/2+O(1)} + O(d^3 \sqrt{n} \log n)$$

operations, when $n \to \infty$. In this paper we extend his techniques, to give a general randomized scheme for speeding up the solution of convex optimization problems.

The scheme presented here is a "Las Vegas" scheme, i.e., the randomization is introduced by random choices within the algorithm. No probabilistic assumptions on the input distribution are required, and the results give the expected complexity for any given input. The scheme is based on randomly selecting relatively small subsets of the constraints, and solving the corresponding relaxed problems. Information from the solution of the subproblems is then used to update the probabilities in subsequent random choices, until eventually the optimal solution to the original problem is obtained. The small subproblems may be solved using any known optimization algorithm.

By using a variant of Karmarkar's interior point method [6] to solve the small subproblems, we obtain an algorithm with average number of major iterations lower than the best known bound for $d = O(n^{1/4}/\log^{1/2} n)$. In particular, we show that an optimum solution for convex quadratic problems with linear constraints can be obtained in no more than

$$O(d^2(\log n)L)$$

iterations on the average, compared to the best known deterministic bound of $O(\sqrt{n}\, L)$ iterations. Moreover, the expected total number of arithmetic operations required to solve such problems is shown to be $O(d \log n(nd + d^6 L))$. For separable convex quadratic programs and linear programs, we get an optimal solution in $O(d \log n(nd + d^4 L))$ operations on the average.

The paper is organized as follows: Section 2 introduces the problem and provides the combinatorial basis to the analysis. Section 3 describes the general randomized scheme in a generic form, together with a proof of the time complexity in terms of complexity of subroutines for solving simpler subproblems. Sections 2 and 3 follow closely the arguments and constructions as presented in Clarkson [2], generalized appropriately for the convex programming problem. Section 4 contains concrete complexity results for convex quadratic and linear programs, and discusses the amount of speedup obtained by applying the scheme to Karmarkar's algorithm and to the ellipsoid algorithm. Section 5 contains some concluding remarks.

2. The problem and some preliminary results

We shall deal with the following optimization problem:

$$\begin{aligned} (\mathrm{P}) \qquad & \text{minimize} \quad f(x) \\ & \text{subject to} \quad g_i(x) \leq 0, \quad i = 1, \ldots, n, \end{aligned}$$

where $x \in \mathbb{R}^d$, $f(x)$ and $g_1(x), \ldots, g_n(x)$ are convex functions.

Let $N = \{1, \ldots, n\}$. For $Q \subset N$, $g_Q(x) \leq 0$ will denote the corresponding subset of the inequalities. P(Q) will be the following problem:

$$\begin{aligned} (\mathrm{P}(Q)) \qquad & \text{minimize} \quad f(x) \\ & \text{subject to} \quad g_Q(x) \leq 0. \end{aligned}$$

Throughout the paper, we shall refer to single constraints and to sets of constraints by their indices. E.g., "constraint i" is short for "the ith constraint", and "the set of constraints Q" will be used instead of "the set of constraints with index set Q". Also, "$\subset$" denotes weak containment.

The randomized scheme which will be described in Section 3 requires random sampling of subsets of constraints, where the probabilities of single constraints are unequal. This process is essentially equivalent to random sampling with equal probabilities from a larger sample space, in which each of the original constraints may formally appear more than once. We define the following notation to handle multiple appearance of identical constraints: Let $L = \{1, 2, \ldots, l\}$ be the set of original constraints, which are distinct. For $i = 1, 2, \ldots, l$, $N_i \subset N$ is the set of identical constraints all of which correspond to the ith original constraint. Hence $N_1, N_2, \ldots, N_l$ form a partition of N. Given $Q \subset N$ define for $i \in L$, $Q_i = N_i \cap Q$. Denote the unique optimal solution of P(Q) by $x(Q)$. For every subproblem P(Q) which has a unique optimal solution, we define a partition of the constraints in N into three sets, V_Q, I_Q and T_Q, by assigning each constraint $j = 1, \ldots, n$ to one of the sets, as follows:

$$\text{if } g_j(x(Q)) > 0 \text{ then } j \in V_Q, \tag{1}$$

$$\text{if } g_j(x(Q)) < 0 \text{ then } j \in I_Q. \tag{2}$$

If $g_j(x(Q))=0$, let $j\in N_i$; if $Q_i\neq\emptyset$ define $\mu=\mu(i,Q)=\max\{j\,|\,j\in Q_i\}$. If $Q_i=\emptyset$ define $\mu=0$. Then j is assigned as follows:

$$\text{if } g_j(x(Q))=0 \text{ and } j>\mu \text{ then } j\in V_Q, \tag{3a}$$

$$\text{if } g_j(x(Q))=0 \text{ and } j=\mu \text{ then } j\in T_Q, \tag{3b}$$

$$\text{if } g_j(x(Q))=0 \text{ and } j<\mu \text{ then } j\in I_Q. \tag{3c}$$

In other words: when all constraints are distinct, V_Q, T_Q and I_Q are the sets of constraints which are violated, tight or satisfied as a strict inequality, respectively, at the optimum of the subproblem. When there are several identical constraints which are satisfied as equality, out of each group N_i which is satisfied as equality, only that constraint which is in Q and has the *largest index* is called "tight", and we define the other constraints in N_i with smaller or larger index to be "strictly satisfied" or "violated", respectively. Throughout the paper, when we say that a constraint is tight, strictly satisfied or violated, we are using these terms in the sense defined above. Define

$$M=\{Q\subset N\,|\,\mathrm{P}(Q) \text{ has an optimal solution}\}$$

The cardinality of a set Q will be denoted by $|Q|$. We make the following assumptions:

(A0) $N\in M$.

(A1) For all $Q\in M$, $\mathrm{P}(Q)$ has a unique optimal solution, which will be denoted by $x(Q)$.

(A2) For all $Q\in M$, $|T_Q|\leq d$.

(A3) For all $Q\in M$, $T_Q\subset Q$.

The assumptions are made in order to avoid discussing infeasible, degenerate or unbounded problems. We shall show later how to satisfy these assumptions for some concrete types of problems. In particular, when the function f is linear or convex quadratic, the functions g_i are all linear, the matrix of the constraints has full rank, and the primal and dual problems are feasible and nondegenerate, then all the assumptions are satisfied. Define

$$F=\{T_Q\,|\,Q\in M\}.$$

In order to prove the main theorem of this section, we shall need several propositions. The first one implies that for every problem which has an optimum, all the constraints which are tight at the optimum are tight also at the optimum of the subproblem which consists of these constraints only:

Proposition 2.1. *$F=\{Q\,|\,Q\in M$ and $T_Q=Q\}$.*

Proof. Clearly $\{Q\,|\,Q\in M$ and $T_Q=Q\}\subset F$. To prove containment in the other direction, we shall show that for all $Q'\in F$ it holds that $Q'\in M$ and $T_{Q'}=Q'$. Fix

$Q' \in F$, and let $Q \in M$ be such that $Q' = T_Q$. Suppose we show that $f(\bar{x}) \geq f(x(Q))$ for every $\bar{x}$ which is feasible for P(Q'). Then P(Q') is feasible and bounded, i.e., $Q' \in M$ and therefore $x(Q')$ exists by assumption (A1). But $f(x(Q')) \leq f(x(Q))$ since $Q' \subset Q$. So $f(x(Q')) = f(x(Q))$, hence $x(Q)$ and $x(Q')$ are both optimal for P(Q'), and again by assumption (A1) $x(Q') = x(Q)$, which implies $T_{Q'} = T_Q = Q'$.

It remains to show that $f(\bar{x}) \geq f(x(Q))$ for every $\bar{x}$ which is feasible for P(Q'). Assume to the contrary that there exists a point $\bar{x} \in \mathbb{R}^d$ satisfying $g_j(\bar{x}) \leq 0$ for all $j \in Q'$, and $f(\bar{x}) < f(x(Q))$. Clearly $\bar{x} \neq x(Q)$. Define $x_\lambda = (1-\lambda)x(Q) + \lambda\bar{x}$. Let $j \in Q - T_Q$.

(i) If $g_j(x(Q)) < 0$, by convexity of g_j we have $g_j(x_\lambda) \leq 0$ for sufficiently small $\lambda > 0$.

(ii) If $g_j(x(Q)) = 0$, let $j \in Q_i$. (By assumption (A3) $m_i \in T_Q$ and $j > m_i$.) Then by definition (3b) there exists $k \in T_Q$ such that $k \in Q_i$, since otherwise j would have been included in T_Q. But $g_k(\bar{x}) \leq 0$ by our assumption, since $k \in T_Q$, hence also $g_j(\bar{x}) \leq 0$. Using again the convexity of g_j we get $g_j(x_\lambda) \leq (1-\lambda)g_j(x(Q)) + \lambda g_j(\bar{x}) \leq 0$ for all $\lambda > 0$.

We conclude that for sufficiently small positive λ, $g_j(x_\lambda) \leq 0$ for every $j \in Q - T_Q$. Hence x_λ is feasible for P(Q), since by the assumption that $g_{Q'}(\bar{x}) \leq 0$ and the convexity, also $g_j(x_\lambda) \leq 0$ for every $j \in T_Q$. By the convexity of f, since $f(\bar{x}) < f(x(Q))$, for sufficiently small positive λ, $f(x_\lambda) < f(x(Q))$, contradicting the optimality of $x(Q)$ for P(Q). Hence $f(\bar{x}) \geq f(x(Q))$ for every $\bar{x}$ which is feasible for P(Q'), and the proof is complete. □

Since by the above T_Q uniquely determines the optimal solution of P(Q), we shall call it the *optimality set* for P(Q). Hence the proposition establishes that the collection of all optimality sets of subproblems of (P) identifies with the collection of all optimality sets for the (substantially fewer) subproblems in which all constraints are tight at the optimum.

The following proposition claims that an optimal solution for a subproblem P′ solves any problem containing P′ for which that solution is feasible.

Proposition 2.2. *For all $Q \subset N$, if $Q' \subset Q$, $Q' \in M$ and $Q \cap V_{Q'} = \emptyset$, then $x(Q')$ is optimal for* P(Q).

Proof. By assumption (A0) P(Q) is feasible. Since $Q' \subset Q$ and $Q' \in M$, P(Q) has a bounded solution. Hence $Q \in M$, i.e., $x(Q)$ exists. $Q' \subset Q$, hence $f(x(Q')) \leq f(x(Q))$. But $Q \cap V_{Q'} = \emptyset$ implies that $x(Q')$ satisfies all the constraints in Q, so $x(Q')$ is optimal for P(Q). □

The following proposition implies that if the optimal solution with respect to a subproblem $Q' \subset Q$ violates some constraints of Q, at least one of these constraints should be in the optimality set of P(Q):

Proposition 2.3. *Let $Q \subset N$, $Q' \in M$, $Q' \subset Q$ and $Q \cap V_{Q'} \neq \emptyset$. Then $T_Q \cap V_{Q'} \neq \emptyset$.*

Proof. The same argument as above shows that $x(Q)$ exists. Distinguish two cases:

(i) If $x(Q) = x(Q')$ then for all $j \in Q$, $g_j(x(Q')) \leq 0$. I.e., no constraint j in Q satisfies $g_j(x(Q')) > 0$, and all violated constraints must be of type (3a). Take some $j \in Q \cap V_{Q'}$, where $j \in Q_i$. Then j is a violated constraint of type (3a), i.e., $g_j(x(Q')) = 0$, and $j > \max\{k \mid k \in Q'_i\}$. By (3b), the constraint with maximum index in Q_i is tight, namely $k = \max\{j \mid j \in Q_i\} \in T_Q$. But since $k \in V_{Q'}$, we conclude that $k \in V_{Q'} \cap T_Q$.

(ii) If $x(Q) \neq x(Q')$ then $f(x(Q')) < f(x(Q))$, since otherwise the two distinct points $x(Q)$ and $x(Q')$ would have been optimal for $P(Q')$, in contradiction to assumption (A1). Define $x_\lambda = (1-\lambda)x(Q) + \lambda x(Q')$. Assume that $T_Q \cap V_{Q'} = \emptyset$, i.e., all the constraints out of Q which are violated at $x(Q')$ are not tight at $x(Q)$. Thus, for a sufficiently small $\lambda > 0$, by the convexity of g_i and f we have $g_Q(x_\lambda) \leq 0$ and $f(x_\lambda) < f(x(Q))$, contradicting the optimality of $x(Q)$ for P(Q), which completes the proof. □

Define now $\Lambda^i_Q = \{\bar{Q} \in F \mid \bar{Q} \subset Q \text{ and } |Q \cap V_{\bar{Q}}| = i\}$, that is, Λ^i_Q is the set of all subset of Q in F whose optimal solutions are violated by exactly i constraints from Q. The following proposition states that every subproblem has a *unique* optimality subset:

Proposition 2.4. *For every $Q \in M$, $|\Lambda^0_Q| = 1$.*

Proof. By Proposition 2.1, there exists at least one optimality set for Q, namely T_Q. Suppose both Q^1 and Q^2 are optimality sets for Q. Then by Proposition 2.2 and assumption (A1) $x(Q^1) = x(Q^2)$. But Q^1, $Q^2 \in F$ so $T_{Q^1} = Q^1$ and $T_{Q^2} = Q^2$ which implies $Q^1 = Q^2$. □

The following proposition states that for every subproblem, the number of distinct optimality subsets contained in it which violate exactly one of its constraints is at most d:

Proposition 2.5. *For every $Q \in M$, $|\Lambda^1_Q| \leq d$.*

Proof. Let $\{Q^1, Q^2, \ldots, Q^k\} = \Lambda^1_Q$ and let α_i be the index of the unique constraint of Q violated by $x(Q^i)$, i.e. $\{\alpha_i\} = Q \cap V_{Q^i}$. If $\alpha_i = \alpha_j$, $i \neq j$, then Q^i and Q^j are optimality sets for $Q - \{\alpha_i\}$, so by Proposition 2.4, $Q^i = Q^j$. Thus $\alpha_1, \alpha_2, \ldots, \alpha_k$ are all distinct. Moreover, by Proposition 2.3, the set $\{\alpha_1, \alpha_2, \ldots, \alpha_k\} \subset T_Q$. So $|T_Q| \geq k$. Since by assumption (A2) $|T_Q| \leq d$, the result follows. □

We can now prove the main theorem of this section. We assume that a set $S \in M$ is fixed, and out of the remaining constraints, a set R of size r is chosen randomly,

such that all sets of size r out of $N-S$ have equal probability to be chosen. The theorem gives an upper bound on the expected number of constraints out of N which are violated by the optimal solution to $P(R \cup S)$:

Theorem 2.6. *Given $S \in M$, assume that a set $R \subset N-S$ of r constraints is randomly chosen. Denote $Q = R \cup S$, $n' = n - |S|$. Then*

$$E(|V_Q|) \leq \frac{n'-r+1}{r-d}\, d.$$

Proof. For any $Q \in M$, denote by $v(Q)$ the number of constraints in N violated by the optimal solution corresponding to $P(Q)$, i.e., $v(Q) = |V_Q|$. Our sample space is $G = \{Q \mid Q = S \cup R, R \subset N-S, |R| = r\}$. Since $S \in M$, by assumption (A0) every $Q \in G$ satisfies $Q \in M$. There are $\binom{n'}{r}$ equiprobable ways to choose Q, and for each Q we count the number of constraints violated by the optimum $x(Q)$. Hence

$$E(v(Q)) = \frac{\sum_{Q \in G} v(Q)}{\binom{n'}{r}}.$$

By Proposition 2.1, rather than counting over G, we can count over F. But then for every member $\bar{Q}$ of F, we should multiply $v(\bar{Q})$ by the number of sets $Q \in G$ for which $T_Q = \bar{Q}$. In other words,

$$= \frac{\sum_{\bar{Q} \in F} v(\bar{Q}) \cdot |\{Q \in G \mid T_Q = \bar{Q}\}|}{\binom{n'}{r}}.$$

Fix $\bar{Q} \in F$ and denote $\bar{Q}_S = \bar{Q} \cap S$, $\bar{Q}_{N-S} = \bar{Q} - \bar{Q}_S$, and $i_{\bar{Q}} = |\bar{Q}_{N-S}|$. We wish to count the number of sets $Q \in G$ which satisfy $T_Q = \bar{Q}$. Every such Q should contain the set $\bar{Q}_{N-S}$, and the remaining $r - i_{\bar{Q}}$ constraints should be selected out of $I_{\bar{Q}} - S$, i.e., out of $n' - i_{\bar{Q}} - v(\bar{Q})$ constraints. (Note that $x(\bar{Q})$ violates no constraints from S, since $x(\bar{Q})$ is optimal for Q, which alway contains S). Hence,

$$= \frac{\sum_{\bar{Q} \in F} v(\bar{Q}) \binom{n' - i_{\bar{Q}} - v(\bar{Q})}{r - i_{\bar{Q}}}}{\binom{n'}{r}}$$

$$= \sum_{\bar{Q} \in F} v(\bar{Q}) \left(\frac{n'-r-v(\bar{Q})+1}{r-i_{\bar{Q}}}\right) \frac{\binom{r'-i_{\bar{Q}}-v(\bar{Q})}{r-i_{\bar{Q}}-1}}{\binom{n'}{r}}.$$

Since $v(\bar{Q}) \geq 0$ and $i_{\bar{Q}} \leq |\bar{Q}| \leq d$ (by assumption (A2)) we have

$$\leq \left(\frac{n'-r+1}{r-d}\right) \frac{\sum_{\bar{Q}\in F} \nu(\bar{Q}) \binom{n'-i_{\bar{Q}}-\nu(\bar{Q})}{r-i_{\bar{Q}}-1}}{\binom{n'}{r}}.$$

For $\bar{Q} \in F$,

$$v(\bar{Q}) \binom{n'-i_{\bar{Q}}-n(\bar{Q})}{r-i_{\bar{Q}}-1}$$

is precisely the number of distinct $Q \in G$ for which $\bar{Q}$ belongs to Λ_Q^1. This follows since for fixed $\bar{Q}$, in order for $x(\bar{Q})$ to violate exactly one constraint of Q, Q should contain the set $\bar{Q}_{N-S}$, one violated constraint out of the set $V_{\bar{Q}}$ (whose size is $v(\bar{Q})$) and the remaining $r-i_{\bar{Q}}-1$ constraints should be selected out of the non-violated constraints. Summation over all $\bar{Q} \in F$ will therefore yield at most $\sum_{Q\in G} |\Lambda_Q^1|$. Thus,

$$E(|V_Q|) \leq \left(\frac{n'-r+1}{r-d}\right) \frac{\sum_{Q\in G} |\Lambda_Q^1|}{\binom{n'}{r}} \leq \left(\frac{n'-r+1}{r-d}\right) d$$

where the last inequality follows from Proposition 2.5. □

3. The randomized scheme

We now describe a randomized scheme for solving convex optimization problems. It is based on Clarkson's scheme for linear programming [2]. Since the scheme is designed to speed up solution of problems with large n/d ratio, we assume throughout the rest of the paper that $n = \Omega(d^2)$. We solve a sequence of randomly chosen subproblems, each with a relatively small number of constraints. The random choice is done according to integer weights attached to the constraints. The weights are initially all equal, and are modified during the execution of the algorithm. In each iteration, the relative weight of each constraint reflects the posterior probability that we attach at that stage to the event that this constraint is in the optimality set for the original problem.

Each subproblem is solved by standard techniques, and the set of constraints violated at its optimum point are identified. Theorem 2.6 guarantees that with high probability, the number of violated constraints is small. By Proposition 2.3, at least one of these violated constraints is in the optimality set of the original problem. If indeed the number of violated constraints is small, the algorithm increases the weights of these constraints before the next iteration. When the next subproblem is randomly chosen, the probability of including in it more constraints from the optimally set is thus increased, until eventually it includes all the optimally set and the process terminates.

The formal description of the algorithm now follows. The algorithm uses a generic subroutine called SOLVE to solve the small subproblems. The input to SOLVE is a subproblem P(Q), and it outputs the optimal solution $x(Q)$. Given the problem (P), we assume that a set $S \subset N$ is known such that $|S| \leq d+1$ and P(S) has a bounded solution. The set S will participate in every subproblem solved by the algorithm, thereby guaranteeing that for every $R \subset N-S$, P($R \cup S$) has a bounded solution. We shall show how to achieve that situation later, for particular types of problems. The integer weight attached to constraint i is denoted by w_i. Also, for a set Q of constraints, $W(Q) = \sum_{i \in Q} w_i$ will denote its total weight.

Algorithm RANDOPT.

Input: Problem (P) in d variables, with a set N of n constraints.

Output: An optimal solution $x(N)$ for the problem.

Step 1. Set $\alpha_d \leftarrow 4d^2+d$; $\beta_d \leftarrow 1/(2d)$; for $i=1, \ldots, n$ set $w_i \leftarrow 1$.

Step 2. Choose $R \subset N-S$ with $|R| = \alpha_d$ at random, according to the weights w_i. Set $Q \leftarrow R \cup S$.

Step 3. Use SOLVE on $P(Q)$ to obtain its optimal solution, $x(Q)$.

Step 4. Find $V = V_Q$.

Step 5. If $V = \emptyset$ then output $x(Q)$ and terminate.

Step 6. Else if $W(V) \leq \beta_d \cdot W(N)$ then for all $i \in V$ set $w_i \leftarrow 2w_i$. Go to Step 2.

The random choice in Step 2 is done as follows: r constraints are chosen sequentially from the set of original constraints $N-S$. Initially, constraint i has relative probability $w_i / W(N)$ to be chosen, for all i. If constraint j has been chosen, then we set $w_j \leftarrow w_j - 1$ and repeat the process. Note that the same constraint may be chosen several times.

We say that a new iteration has started whenever the algorithm executes step 2. Hence, the number of iterations performed is equal to the number of executions of SOLVE. We say that an iteration is successful if the weights are updated in Step 6, namely, if the total weight of the set of violated constraints has not exceeded the limit.

Theorem 3.1. *The expected number of iterations required by the algorithm is* $O(d \log n)$.

Proof. We first show that the expected number of iterations between successful iterations is at most two. For the sake of the proof, we use the following equivalent representation of the random choice: We select at random a subset of size r from the multiset $M = M(w)$ in which for each $i \in N-S$, constraint i appears w_i times. That is,

$$M(w) = \{c(i,j) \mid c(i,j) = i, j = 1, \ldots, w_i, i = 1, \ldots, n\}.$$

All subsets of size r in M are equiprobable. Clearly, $|M| = W(N)$. Denote by $\bar{V} \subset M$

the multiset of constraints which are violated at $x(Q)$. Note that since $\bar{V}$ is a multiset, violation here is in the sense defined in Section 2, so in particular, if $g_{c(i,j)}(x(Q))=0$ and $j>\max\{k \mid c(i,k)\in Q\}$ then $c(i,j)\in V_Q$ is a violated constraint of type (3a), but it does not contribute to the doubling of weights in Step 6 of the algorithm. In other words, $W(V)\leq|\bar{V}|$. By Theorem 2.6, $E(|\bar{V}|)$ is not more than $d(W(N)-r+1)/(r-d)$. Taking $\alpha_d=4d^2+d$ and $\beta_d=1/(2d)$, we get

$$E(W(V))\leq E(|\bar{V}|)\leq(W(N)-4d^2-d+1)/(4d)<W(N)/(4d)$$

$$=\tfrac{1}{2}\beta_d W(N).$$

By Markov's inequality, the probability that $W(V)\geq\beta_d W(N)$ is thus at most $\frac{1}{2}$, hence it takes on the average no more than two iterations until the condition in Step 6 is satisfied. Consequently, the expected total number of iterations is at most twice the expected total number of successful iterations.

Let $Z=T_N-S$ and $d'=|Z|$. By assumption (A2), $|T_N|\leq d$, so $d'\leq d$. We assume that $d'>0$, since if $d'=0$ then $T_N\subset S$, and then the first iteration immediately provides the optimal solution. To bound the expected number of successful iterations we use again the multiset interpretation: If the algorithm generates $V\neq\emptyset$, then clearly $\bar{V}\neq\emptyset$. Moreover, if $V\neq\emptyset$ Proposition 2.3 implies that V contains at least one member of Z. Hence in each successful iteration, the weight of at least one constraint of Z is doubled. This implies that $W(Z)$ is at least doubled after every d' successful iterations. Initially $W(Z)=d'$, so after kd' successful iterations $W(Z)\geq d'2^k$. Also, initially $W(N)=n$, and Step 6 in the algorithm guarantees that $W(N)$ increases by no more than a factor of $1+\beta_d$ upon each successful iteration. Hence after kd' successful iterations, $W(N)\leq n(1+\beta_d)^{kd'}\leq n\,\mathrm{e}^{\beta_d kd'}$. $W(Z)$ increases much faster than $W(N)$, and the algorithm should terminate before $W(Z)\geq W(N)$. Hence termination happens before k satisfies

$$W(Z)\geq d'2^k\geq n\,\mathrm{e}^{\beta_d kd'}\geq W(N),$$

i.e., if $k\ (\ln 2-\beta_d d')\geq\ln(n/d')$. In summary, we get that the algorithm requires $O(d\log n)$ iterations on the average. □

We can now state the complexity of the scheme. It will be expressed in terms of the complexity of the subroutine for solving small problems, SOLVE, and of the effort required to check whether a point satisfies a constraint. For fixed d, denote by $C_{\mathrm{SOLVE}}(m)$ the time complexity of SOLVE when solving a problem with m constraints. Denote by C_{CHECK} the time complexity of an oracle which checks whether a point $x\in\mathbb{R}^d$ satisfies a constraint of the problem. (The dependence on the dimension and the input size L is suppressed here; it will be discussed for some concrete classes in the next section). All complexity results are stated according to the model of computation of uniform-cost sequential random access machine [1].

Theorem 3.2. *The above scheme solves* (P) *in expected time complexity*:

$$O(d \log n[n \cdot C_{\text{CHECK}} + C_{\text{SOLVE}}(4d^2+2d+1)]).$$

Proof. By Theorem 3.1, the expected number of iterations is $O(d \log n)$. In each iteration we use SOLVE on a subproblem with at most $4d^2+2d+1$ constraints, and then check which of the n constraints of the original problem are violated at the optimum point. The complexity of the random choice of R can be performed in $O(n)$ steps (see [2] and the references thereof), so it is dominated by the latter test. □

4. Complexity results for specific problem classes

In this section we apply the scheme to several problems, including linear and convex quadratic programming. The algorithms we shall use as subroutines in the scheme have complexity bounds which depend polynomially on the size of the input. Therefore, the resulting complexities will be expressed as functions of n, d and the binary input length L.

Theorem 4.1. *Given a subroutine for solving a convex programming problem with d variables, n constraints and input length L in time* $T(n, d, L)$, *and another subroutine for checking if a constraint is satisfied by a given point of complexity* C_{CHECK}, *the expected time complexity of algorithm RANDOPT is*

$$O(d \log n[n \cdot C_{\text{CHECK}} + T(4d^2+2d+1, d, L)]).$$

Proof. Follows immediately from Theorem 3.2. □

Note that the theorem applies only to those convex programming problems for which the input length is well defined. For general convex programming problems, this its not always the case.

Corollary 4.2. *A problem with convex quadratic objective and linear constraints is solvable in expected time* $O(d \log n(dn + d^6 L))$.

Proof. For convex quadratic programming problems, an algorithm based on Karmarkar's interior path method gives a solution in $O(n^3 L)$ time (see, e.g., [10]). The time required to check whether a point satisfies a linear constraint is obviously $O(d)$. Hence the result follows from Theorem 4.1. □

Corollary 4.3. *A problem with separable convex quadratic objective and linear constraints is solvable in expected time* $O(d \log n(nd + d^4 L))$.

Proof. These problems are solvable by the same algorithm as in the previous case, in time $O((nd^2 + n^{3/2}d)L)$ (see [10]). □

Corollary 4.4. *The linear programming problem is solvable in expected time*

$$O(d \log n(nd + d^4 L)).$$

Proof. This is a special case of Corollary 4.3. □

Theorem 4.5. *Using interior path following algorithms, convex quadratic programming problems and linear programming problems can be solved within* $O(d^2(\log n)L)$ *major iterations on the average.*

Proof. In proving Theorem 3.1, we have shown that overall the scheme solves on the average at most $O(d \log n)$ problems each of which has at most $4d^2 + 2d + 1$ constraints, and size at most L. Since it is possible to solve convex quadratic problems with n constraints in $O(n^{1/2}L)$ major iterations of interior path following algorithm (see, e.g., [10]), the expected total number of such iterations is no more than that stated in the theorem. □

We now demonstrate the speedup (in terms of complexity) obtained by the scheme for two well known polynomial algorithms for the linear programming problem. Given an algorithm ALG for solving linear programs, denote by ρ_{ALG} the ratio of the complexity of ALG to the expected complexity of RANDOPT which uses ALG as a subroutine to solve small subproblems. By Corollary 4.4, for Karmarkar's interior point algorithm (using, for example, the variant studied in [10]), one gets $\rho_{\mathrm{KAR}} = \Omega(\sqrt{n}\, L/(d \log n))$ when $L = O(n/d^3)$, and $\rho_{\mathrm{KAR}} = \Omega(n^{3/2}/(d^4 \log n))$ otherwise. For the ellipsoid algorithm [7] which requires $O(nd^3 L)$ operations (see, e.g., [5]), $\rho_{\mathrm{ELL}} = \Omega(dL/\log n)$ if $L = O(n/d^4)$ and $\rho_{\mathrm{ELL}} = \Omega(n/(d^3 \log n))$ otherwise. Note that if L represents the size of the input, $L \geq n$, so only the second case applies for both algorithms, with a guaranteed speedup for $n \to \infty$ and n/d sufficiently large.

In all the above expressions L may be replaced by Δ, the logarithm of the largest absolute value of a subdeterminant of the problem data (see, e.g., [14]), if that value is known in advance. In that case ρ depends on the relation between Δ and n. However, if we assume that there are no identical constraints in the input, then we can determine a uniform lower bound on the speedup, which is independent on Δ, by noting that n cannot increase too much without affecting Δ: If the largest absolute value of an input number in the integer data is k, then $n \leq (2k+1)^d$, which implies

$\log k = \Omega((\log n)/d)$, so $\Delta = \Omega((\log n)/d)$. This implies $\rho_{\mathrm{KAR}} = \Omega(\sqrt{n}/d^2)$ and $\rho_{\mathrm{ELL}} = \Omega(1)$. Hence even with Δ as small as possible, a speedup is guaranteed for the interior point algorithm when $n = \Omega(d^4)$, and for the ellipsoid the randomization is always competitive. Note that for fixed d and $n \to \infty$ all the above results become even stronger.

In all the cases discussed in Theorems 4.2–4.5 above, the constraints are linear and the objective function is linear or convex quadratic. Such problems can be presented as linear complementarity problems, i.e., find $z \geq 0$ and $w \geq 0$ such that $w - Mz = q$ and $z^T w = 0$, where M is an $n \times n$ matrix and q is an n-vector (for the reductions see, e.g., [11]). Note that feasibility of the linear complementarity problem guarantees both boundedness and feasibility of the original problems. In these cases, in order to satisfy the assumptions of Section 2, the following can be done: One can construct an equivalent linear complementarity problem with additional constraint set S and additional variables, which is feasible and in which n, d and L were only slightly increased without affecting the complexity (see, e.g., [10] for details). This guarantees property (A0). By including the set S in all solved subproblems, we guarantee their boundedness which is required in (A1). Furthermore, it is possible to perturb q so that every basic feasible solution of the above problem is non-degenerate. This will satisfy the rest of the assumptions. The perturbation of q can be done formally without affecting L, as is done, for example, in [11]. Since all our assumptions can be satisfied for linear complementarity problems with positive semi-definite matrix M, the randomized scheme can be applied to any algorithm which solves such problems. For other convex problems, the existence of a solution subroutine as well as the validity of assumptions (A0)–(A3) should be established in order to guarantee the proved convergence bounds of the randomized scheme.

5. Concluding remarks

1. There is a certain resemblance between the randomized algorithm and the dual presentation of "partial pricing" method for solving linear programs. In both cases, optimality of a subproblem is reached before considering the rest of the constraints. For the general convex programming case, the randomized scheme fits into the framework of general relaxation techniques. Since those schemes are commonly used in practice, the randomized scheme might be of interest for practical implementation.

2. Algorithms with complexities which are independent of the sizes of the input numbers (e.g., the simplex method or strongly polynomial algorithms) may also be accelerated by the above scheme. For linear programs, Clarkson [2] described how to use the simplex method (or any vertex enumeration method) in SOLVE in order to improve the time complexity which depends on d and n only. Note that this case is covered by Theorem 4.1.

Acknowledgement

This research was supported by the United States Navy Office of Naval Research under contract N00014-87-0202. Part of the work was performed when the second author was visiting DIMACS and RUTCOR in Rutgers University, and supported by NSF grant NSF-STC88-09648, and Air Force grants AFOSR-89-0512 and AFOSR-90-0008. Their financial support is gratefully acknowledged.

References

[1] A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The Design and Analysis of Computer Algorithms* (Addison-Wesley, Reading, MA, 1974).

[2] K.L. Clarkson, "A Las Vegas algorithm for linear programming when the dimension is small," *Proceedings of the 29th IEEE Symposium on the Foundations of Computer Science* (1988). [Revised version: AT&T Bell Laboratories, 1991.]

[3] M.E. Dyer and A.M. Frieze, "A randomized algorithm for fixed-dimensional linear programming," *Mathematical Programming* 44 (1989) 203–212.

[4] P.E. Gill, W. Murray and M.H. Wright, *Practical Optimization* (Academic Press, New York, 1981).

[5] M. Grötschel, L. Lovász and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization* (Springer, Berlin, 1988).

[6] N. Karmarkar, "A new polynomial time algorithm for linear programming," *Combinatorica* 4 (1984) 373–395.

[7] L.G. Khachian, "Polynomial algorithms in linear programming," *Zhurnal Vychislitelnoi Matematiki i Matematicheskoi Fiziki* 20 (1980) 51–68. [English translation in: *U.S.S.R. Computational Mathematics and Mathematical Physics* 20 (1980) 53–72.]

[8] D.G. Luenberger, *Linear and Nonlinear Programming* (Addison-Wesley, Reading, MA, 1984, 2nd ed.).

[9] N. Megiddo, ed., *Algorithmica* 1, *Special issue: Progress in Mathematical Programming* (1986).

[10] R.C. Monteiro and I. Adler, "Interior path following primal-dual algorithms, Part II: Convex quadratic programming," *Mathematical Programming* 44 (1989) pp. 43–66.

[11] K.G. Murty, *Linear Complementarity, Linear and Nonlinear Programming* (Heldermann, Berlin, 1988).

[12] P.M. Pardalos and J.B. Rosen, *Constrained Global Optimization: Algorithms and Applications. Lecture Notes in Computer Science No. 268* (Springer, Berlin, 1987).

[13] R.T. Rockafellar, *Convex Analysis* (Princeton University Press, Princeton, NJ, 1970).

[14] A. Schrijver, *Theory of Integer and Linear Programming* (Wiley, New York, 1986).

[15] J. Stoer and C. Witzgall, *Convexity and Optimization in Finite Dimensions* (Springer, Berlin, 1970).