DIMACS Technical Report 91–24
March, 1991

FAST APPROXIMATION SCHEMES FOR CONVEX PROGRAMS WITH MANY BLOCKS AND COUPLING CONSTRAINTS

by

Michael D. Grigoriadis[1]
Department of Computer Science
Rutgers University
New Brunswick, New Jersey 08903

and

Leonid G. Khachiyan[2]
Department of Computer Science
Rutgers University
New Brunswick, New Jersey 08903

&
Computing Center
USSR Academy of Sciences
Moscow, USSR

[1] Permanent member of DIMACS

[2] Supported in part by DIMACS

DIMACS is a cooperative project of Rutgers University, Princeton University, AT&T Bell Laboratories and Bellcore.

DIMACS is an NSF Science and Technology Center, funded under contract STC–88–09648; and also receives support from the New Jersey Commission on Science and Technology.

1. Introduction.

Large mathematical programming models which arise in practice frequently exhibit some structure in their constraints. The most common of these is the *block angular* structure. It consists of K sets of "block" constraints and a set of M "coupling" constraints, so that if the coupling constraints are sufficiently relaxed, the problem decomposes into K simpler independent problems. Among the earliest methods specifically designed to take advantage of this structure for linear problems are the Dantzig-Wolfe decomposition principle [2] and Rosen's partitioning method [12]. Currently there is a renewed interest in complexity results for solving large structured optimization problems, mainly prompted by advances in parallel computation and potential-function methods for linear and nonlinear programming. In this paper we present and analyze the complexity of a block-coordinate potential-descent method for a class of such problems.

1.1. The problem. We consider the following block-angular convex programming problem:

$$\lambda^* = \min \left\{ \lambda \;\middle|\; \begin{array}{l} \sum_{k=1}^{K} f_m^k(x^k) \le \lambda, \quad m = 1, \cdots, M \\ \qquad\qquad x^k \in B^k, \quad k = 1, \cdots, K \end{array} \right\},$$

where B^k are nonempty convex compact sets, and $f_m^k : B^k \to \mathbb{R}$ are nonnegative continuous convex functions. The sets B^k are called *blocks*, the variables $x^k \in B^k$ are called *block coordinates*, and the M inequalities are called *coupling* or *resource constraints*. We shall use the following notation: $x = (x^1, \cdots, x^K)$, $B = B^1 \times \cdots \times B^K$, $f^k = (f_1^k, \cdots, f_M^k) : B^k \to \mathbb{R}^M$, $f = \sum_{k=1}^K f^k : B \to \mathbb{R}^M$, and $e \in \mathbb{R}^M$ the vector of all ones. With this notation, the above problem can be written as

$$\lambda^* = \lambda(x^*) = \min_{x \in B} \{ \lambda(x) = \max\{ f_1(x), \cdots, f_M(x) \} \}, \tag{1.1}$$

or simply written as follows

$$(\mathcal{P}): \qquad \lambda^* = \min\{ \lambda \mid f(x) \le \lambda e, \; x \in B \}.$$

We shall assume throughout that $M \ge 2$, and that $\lambda_k^* = \min\{ \lambda \mid f^k(x^k) \le \lambda e, \; x^k \in B^k \}$ is positive for all $k = 1, \cdots, K$; that is, there are no blocks B^k that can be discarded from $(\mathcal{P})$ without affecting the value λ^*. Our main focus shall be to state an algorithm which, for a given relative accuracy $\varepsilon \in (0, 1]$, computes an *ε-approximate solution* to $(\mathcal{P})$:

$$(\mathcal{P}_\varepsilon): \qquad f(x) \le (1 + \varepsilon)\lambda^* e, \quad x \in B.$$

Such an algorithm can be used in a two-phase approximate algorithm for the general block-angular optimization problem

$$f^* = \min\{ f_0(x) \mid f(x) \leq e,\ x \in B \}, \tag{1.2}$$

with a block-separable nonnegative convex objective function $f_0 : B \to \mathbb{R}_+$. The computation of an ε-feasible solution for (1.2) in the first phase is subsumed by problem $(\mathcal{P}_\varepsilon)$. The optimization phase can be implemented by solving $(\mathcal{P}_\varepsilon)$ for a moving threshold on the objective function $O(r)$ times, so that the initial interval $[0, f_0(x_0)]$ of possible optimal values f^* is contracted by a factor of 2^{-r}. The nonnegativity assumption on the objective and constraint functions can also be relaxed, albeit at some cost, as stated in Section 9.2.

The special case of $(\mathcal{P})$ for which functions $f_m^k(x^k)$ are additively separable is of considerable practical importance. That is, $f_m^k(x^k) = \sum_{j \in J_m^k} f_m^{kj}(x^{kj})$, where f_m^{kj} are nonnegative convex functions and J_m^k is the set of indices of those elements of x^k that enter the function $f_m^k(x^k)$. We shall denote the largest number of block elements that enter any coupling constraint of $(\mathcal{P})$ by $S = \max_{k,m} |J_m^k|$, and refer to this number as the *block coupling degree* of $(\mathcal{P})$. S is the maximum dimension of the x^k in the worst case, but in many applications it is a small constant, frequently one. We shall consider several such applications in the course of this paper.

1.2. Block-coordinate descent strategy. This strategy is best understood in terms of duality as follows. Suppose that a point $x_0 \in B$ is known. Let $\mu \geq \lambda(x_0)$ be a fixed scalar. Then, by the nonnegativity of the functions f^k, any solution x to $(\mathcal{P})$ also belongs to $B_\mu = B_\mu^1 \times \cdots \times B_\mu^K$, where

$$B_\mu^k = \{ x^k \in B^k \mid f^k(x^k) \leq \mu e \},\ k = 1, \cdots, K. \tag{1.3}$$

From Von Neumann's saddlepoint theorem (see e.g. Rockafellar [15], p.393),

$$\lambda^* = \min_{x \in B_\mu} \max_{p \in P} L(x,p) = \max_{p \in P} \min_{x \in B_\mu} L(x,p),$$

where $P = \{ p \in \mathbb{R}^M \mid p \geq 0,\ \ e^T p = 1 \}$ is the M-dimensional unit simplex, and $L(x,p) = \sum_{k=1}^K L^k(x^k,p) = \sum_{k=1}^K p^T f^k(x^k) = p^T f(x)$ is the Lagrangian of $(\mathcal{P})$. An immediate consequence of this theorem is the well-known duality relation:

$(\mathcal{D})$:
$$\lambda^* = \max_{p \in P} \{ g_\mu(p) = \sum_{k=1}^K g_\mu^k(p) \mid p \in P \},$$

where

$$(\mathcal{P}^k): \qquad g_\mu^k(p) = \min\{\, L^k(x^k, p) = p^T f^k(x^k) \mid x^k \in B_\mu^k \,\}.$$

Thus, for fixed μ and *price vector* $p \in P$, the computation of $g_\mu(p)$ decomposes into K independent *block problems* $(\mathcal{P}^k)$. This observation is the basis for a class of block-coordinate descent schemes which maintain $x \in B$, $p \in P$, the scalar μ, and a subset of blocks $\mathcal{K} \subseteq \{1, \cdots, K\}$ selected for recomputation. Each iteration consists of the following two steps.

Step 1. (Block optimization.) For each $k \in \mathcal{K}$, use a "block solver" to compute an approximate solution $\hat{x}^k \in B_\mu^k$ to the block problem $\mathcal{P}^k$.

Step 2. (Coordination.) For each $k \in \mathcal{K}$ replace the current x^k by a convex combination of x^k and the new point $\hat{x}^k$. Use an appropriate method to update p, μ, and $\mathcal{K}$. ■

The terms "iteration" and "coordination step" are used interchangeably. Typically, the amount of effort expended on the block problems dominates the cost of coordination. The block problems can be solved in parallel.

This class of methods, also known as two-level price decomposition, has been studied mainly for the case of $\mu = +\infty$ (see e.g. Lasdon [9]). Among the earliest contributions is the scheme of Everett [3], an idea which has its origins in the earlier work of Brown and Robinson for polyhedral games (see [13]). In the past, straightforward implementations of these early methods were found to be practical for computing reasonable estimates of solutions, but inefficient for attaining high accuracy, especially for linear models. A lower bound on the rate of convergence for the latter case was given by Khachiyan [7].

Recently, Schultz and Meyer [17] studied the convergence of a specific block-coordinate descent for linear constraints. Their method uses a p for which $p^T f(x)$ is the linear approximation to a logarithmic barrier function at the current iterate, and performs a K-dimensional search to minimize this potential over all convex combinations of block coordinates in Step 2. By further exploiting the coarse parallelism inherent in Step 1, they were able to solve several large linear multicommodity network flow problems with $K = 11$ to an acceptable accuracy.

Shahrokhi and Matula [14] proposed a combinatorial, fully polynomial-time approximation scheme for the concurrent flow problem[1] with uniform capacities, whose running time was subsequently improved by Tardos [18], and Klein, Stein and Tardos [8]. This algorithm can also be interpreted as a block-coordinate descent strategy, with the gradient of an exponential potential function as the price vector p, and shortest-path block problems with $\mu = \infty$. It is also similar to the *flow-deviation* method of Fratta, Gerla and Kleinrock [4] which uses a rational potential function.

More recently, Leighton, Makedon, Plotkin, Stein, Tardos, and Tragoudas [10] extended [14] to handle nonuniform integer capacities, and showed that this concurrent flow problem can be solved to a relative accuracy $\varepsilon \in (0,1]$ in $O(K \ln MU \ln K + \varepsilon^{-2} K \ln(MU\varepsilon^{-1}))$ coordination steps, where M is the number of edges, U the largest edge capacity, and K the number of commodities. Each coordination step solves K minimum-cost network flow problems. Leighton et al. also gave a randomized variant, which uses an expected number of $O(K \ln MU \ln K + \varepsilon^{-3} K \ln(MU\varepsilon^{-1}))$ coordination steps with two minimum-cost flow problems per step, one of which is randomly selected. (In particular, for any s, such a Las Vegas approximation algorithm terminates in $2s$ times the expected number of coordination steps with probability at least $1 - 2^{-s}$.)

1.3. Results. All of the above problems are special cases of the structured convex programming problem $(\mathcal{P})$. In this paper, we shall state a block-coordinate descent method for solving $(\mathcal{P}_\varepsilon)$ for $\varepsilon \in (0,1]$ in

$$O(K(\ln M)(\varepsilon^{-2} + \ln K)) \tag{1.4}$$

coordination steps. Each step generates K parallel calls to block solvers which compute $O(\varepsilon)$-approximate solutions to problems $(\mathcal{P}^k)$, one for each block. We shall also state a randomized variant of this method which runs in an expected number (1.4) coordination steps, each of which generates on the average only two block-solver calls for randomly-selected blocks. Up to logarithmic factors, the number of coordination steps for either method is linear in the number of blocks and ε^{-2} for the structured convex programming problem $(\mathcal{P}_\varepsilon)$. Since M enters (1.4) only logarithmically, this class of algorithms may be

[1] This problem is to minimize the maximum relative arc overload in a capacitated network to accommodate a K-commodity flow with prescribed supplies and demands for each commodity.

useful for some strongly-coupled problems $(\mathcal{P})$ having a relatively large number of coupling constraints.

These bounds hold for arbitrary nonnegative convex functions $f^k(x^k)$, and for arbitrary compact sets B^k, even if x^k is of infinite dimension. But the overall running time also depends on the complexity of solving the block problems $(\mathcal{P}^k)$, that is, of minimizing $p^T f^k(x^k)$ over the sets B^k_μ. This can be harder than minimizing $p^T f^k(x^k)$ over the original B^k. Nevertheless, in many applications it is possible to simplify the sets B^k_μ by sufficiently relaxing the constraints $f(x) \leq \mu e$. We shall show that relaxing these constraints by a fixed factor can only increase the bound on the number of coordination steps by essentially the same factor. This offers the possibility of various trade-offs with respect to the overall running time. In Section 3 we shall define flexible block solvers to help address this issue. As an example of such a trade-off, consider the case of additively separable functions $f^k_m(x^k)$. For such problems, it is often easier to minimize $p^T f^k(x^k)$ over the *box-relaxed* set $BB^k_\mu = B^k \cap Box^k_\mu$ rather than over B^k_μ, where $Box^k_\mu = \{\, x^k \mid f^{kj}_m(x^{kj}) \leq \mu, \;\; j \in J^k_m, \;\; m = 1, \cdots, M \,\}$. The resulting *box-relaxed solution* $\hat{x}^k$ can then be used as an approximate solution to $(\mathcal{P}^k)$. For the deterministic and randomized variants of our algorithms using block solvers that generate $O(\varepsilon)$-approximate solutions over BB^k_μ, the bound on coordination steps, claimed in (1.4), becomes:

$$O(SK(\ln M)(\varepsilon^{-2} + \ln K + \ln S)), \tag{1.5}$$

where S is the block coupling degree of $(\mathcal{P})$. As mentioned earlier, S is often a constant. For example, $S = 1$ for all variants of the generalized multicommodity flow problem. Accordingly, our bounds are uniformly better than those given in [14] and [10]. We are not aware of any existing bounds for the nonlinear or the general polyhedral cases of $(\mathcal{P})$.

Insofar as its use of an exponential potential function, our approach generalizes the method of Leighton et al. [10], which in turn is a generalization of the method of Shahrokhi and Matula [14]. We use an exponential potential function which can be interpreted as the statistical sum for a system with a fixed "temperature" $t > 0$ and "energy levels" $-f_m(x)$, $m = 1, \cdots, M$. (Technically, it is more convenient to work with the logarithm of this function, that is with the "free energy".) Our potential function generalizes the one used in [14] for the special case of concurrent flows with uniform capacities, but it is different from the exponential potential used in [10] for general concurrent flows. Also, we use the corresponding Gibbs distribution for the price vector $p \in P$. Finally, we employ a uniform

randomization scheme for $(\mathcal{P})$, which is simpler than that used in [10] for the concurrent flow problem, but improves the bound by a factor of ε^{-1}.

We have organized this paper as follows. After this Introduction, Section 2 defines the potential function and price vector, and Section 3 describes the approximate block solver. All of these are used in subsequent sections. We present algorithms in the order of their incremental generality and of their contribution to reducing the bound on the number of iterations. We define the first one of these, Algorithm $\mathcal{A}$, in Section 4 and analyze its performance in Section 5. This algorithm, which is ultimately used by all others, solves problem $(\mathcal{P})$ to a given absolute accuracy starting from a normalized approximation x_0 to the problem. Section 6 describes a randomized variant, Algorithm $\mathcal{R}$, and proves a bound on the expected number of iterations. The next algorithm, described in Section 7, is a ternary search procedure which uses either one of the previous two algorithms as a subroutine to solve $(\mathcal{P})$ to a given relative accuracy, starting from an arbitrary $x_0 \in B$. It results in an iteration bound that depends logarithmically on the relative accuracy of x_0, a satisfactory level of performance for many practical instances. In Section 8 we invoke a greedy initialization procedure to construct a sufficiently accurate initial approximation that ensures the claimed fully-polynomial bounds (1.4)-(1.5) on the number of iterations.

The statements of these algorithms are generic, in the sense that they provide sufficient latitude to include normalization procedures for numerical stability and heuristics, line search, K-dimensional search, and different block-problem relaxation schemes. We outline some of these in Section 9.1, followed by some illustrative applications in Section 9.2. These include concurrent-flows, (gain-loss) multicommodity flows with side constraints across several arcs, uniformly sparse linear programs with generalized upper bounds, linear, quadratic, and min-max supply sharing generalized transportation networks, deterministic equivalents of large-scale two-stage stochastic programs with discrete finite distribution of input parameters, and a fast parallel approximation algorithm for matrix games, implying that for a fixed accuracy the latter are in *NC*.

2. Potential function and price vector.

In stating our algorithms and their analysis, we shall work with a (convex) potential function of the form:

$$\Phi_t(x) = \ln \sum_{m=1}^{M} \exp \frac{f_m(x)}{t}, \quad x \in B, \tag{2.1}$$

where $t > 0$ is a small positive fixed parameter to be specified later in this section. A useful property of (2.1) is that the difference of the potential at any two points $x, y \in B$ can be written as:

$$\Phi_t(y) - \Phi_t(x) = \ln \sum_{m=1}^{M} p_m \exp\left\{\frac{f_m(y) - f_m(x)}{t}\right\}$$

where

$$p_m = p_m(x,t) = \frac{\exp\{f_m(x)/t\}}{\sum_{\mu=1}^{M} \exp\{f_\mu(x)/t\}}, \; m = 1, \cdots, M. \tag{2.2}$$

Our algorithm for $(\mathcal{P})$ will use this $p \in P$ as the price vector within the context of the block-coordinate descent strategy outlined in Section 1.

Another property of $\Phi_t(x)$ is the following:

$$\frac{\lambda(x)}{t} \leq \Phi_t(x) \leq \frac{\lambda(x)}{t} + \ln M \quad \text{for all } x \in B, \tag{2.3}$$

which is a direct consequence of the definition of $\lambda(x)$ in (1.1). Therefore,

$$\frac{\lambda^*}{t} \leq \Phi_t^* \leq \frac{\lambda^*}{t} + \ln M, \tag{2.4}$$

where $\Phi_t^* = \min_{x \in B} \Phi_t(x)$ and λ^* is the unknown optimal value of $(\mathcal{P})$. These two inequalities immediately imply the following Lemma.

Lemma 1. Let $t = \sigma/(\delta + \ln M)$, where $\sigma > 0$ is a given absolute tolerance and δ is an arbitrary positive scalar. If $x \in B$ minimizes $\Phi_t(x)$ over B to an absolute accuracy δ, i.e. $\Phi_t(x) \leq \Phi_t^* + \delta$, then x minimizes $\lambda(x)$ over B to an absolute accuracy σ, i.e. $\lambda(x) \leq \lambda^* + \sigma$. ∎

We can thus replace the problem of approximately minimizing $\lambda(x)$ over $x \in B$ by the problem of approximately minimizing $\Phi_t(x)$ over $x \in B$. We shall use this observation in Section 4 to state a simple block-coordinate descent algorithm for solving $(\mathcal{P})$ to a given absolute accuracy. This algorithm requires a block solver which we define next.

3. Approximate block solvers.

In Section 1.2 we mentioned two different approximate block solvers for $(\mathcal{P}^k)$, one that minimizes the function $p^T f^k(x^k)$ over the set B^k_μ, and another that minimizes the same function over a box-relaxed set BB^k_μ. Because of the generality of the nonlinear problem $(\mathcal{P})$, it is unrealistic to expect that either block problem can be solved exactly. It is thus necessary to ascertain the quality of a solution $\hat{x}^k$ computed by a block solver. We shall address this need by defining an approximate block solver which will also allow the simultaneous treatment of the cases of B^k_μ and BB^k_μ.

The block problems $(\mathcal{P}^k)$ are stated for fixed $\mu > 0$ and $p \in P$. Let $\epsilon' > 0$ and $q > 1$ be given finite relative *block optimization*, and *block feasibility* tolerances, respectively. We define an (ϵ', q)-*good block solver* for $(\mathcal{P}^k)$ to be a finite-time procedure $\mathcal{BS}(f^k, p, \mu)$ that attempts to find a solution $\hat{x}^k$ such that

$$\hat{x}^k \in B^k_{\mu q} = \{ x^k \in B^k \mid f^k(x^k) \leq \mu q e \} \tag{3.1}$$

and

$$p^T f^k(\hat{x}^k) \leq (1+\epsilon') g^k_\mu(p) = (1+\epsilon') \min\{ L^k(x^k, p) = p^T f^k(x^k) \mid x^k \in B^k_\mu \}.$$

We assume further that $\mathcal{BS}$ has the following properties: It succeeds whenever $B^k_\mu \neq \emptyset$, it correctly reports all instances of $B^k_{\mu q} = \emptyset$, but behaves unreliably whenever $B^k_\mu = \emptyset$ and $B^k_{\mu q} \neq \emptyset$, returning any $\hat{x}^k$ or a fallacious report that $B^k_{\mu q} = \emptyset$. (The algorithms of Sections 4-7 will invoke procedure $\mathcal{BS}$ only for the case $B^k_\mu \neq \emptyset$.)

We shall similarly define an (ϵ', q)-good *box*-block solver $\mathcal{BBS}(f^k, p, \mu)$ for $(\mathcal{P}^k)$ with additively separable functions $f^k_m(x^k)$, as follows: in the above definition of $\mathcal{BS}$, replace $B^k_{\mu q}$ (but not B^k_μ) by the box-relaxed set $BB^k_{\mu q}$ specified in Section 1.2. By the nonnegativity of the functions $f^{kj}(x^{kj})$, we have $B^k_\mu \subseteq BB^k_\mu \subseteq B^k_{\mu S}$ for all $\mu > 0$, where S is the block coupling degree of $(\mathcal{P})$. This implies the following claim.

Proposition 1. Any (ϵ', q)-good box-block solver $\mathcal{BBS}(f^k, p, \mu)$ can be used as an (ϵ', qS)-good block solver $\mathcal{BS}(f^k, p, \mu)$. ■

In Sections 4-8 we shall bound the complexity of our algorithms in terms of the number of calls to an (ϵ', q)-good block solver $\mathcal{BS}$, and only ultimately translate these bounds to the number of calls to an (ϵ', q)-good box-block solver $\mathcal{BBS}$. We shall also need to call $\mathcal{BS}$ for functions f^k scaled by some positive scalar ν. Since $\mathcal{BS}(\nu f^k, p, \mu) = \mathcal{BS}(f^k, p, \mu/\nu)$, the effect of such scaling on the parameters ϵ' and q can be ignored.

4. Solving problem $(\mathcal{P})$ to a given absolute accuracy.

The algorithm, *Algorithm* $\mathcal{A}$, which we state in this section is an intrinsic component of the overall algorithms we shall develop in subsequent sections for solving $(\mathcal{P})$ to a given relative accuracy $\varepsilon > 0$ (problem $(\mathcal{P}_\varepsilon)$). Algorithm $\mathcal{A}$ uses block-coordinate descent to solve $(\mathcal{P})$ to a given *absolute* accuracy $\sigma \in (0, 1/3]$. It accepts as input an $x_0 \in B$ such that $\lambda(x_0) \leq 1$, uses a $(\sigma/7, q)$-good block solver for a given relative block feasibility tolerance $q > 1$, and produces a solution x to $\lambda(x) \leq \lambda^* + \sigma$, $x \in B$.

Algorithm $\mathcal{A}(f, x_0, \sigma, q)$:

Step 0. (Initialization.) Select $\delta = 6 \ln M$, $t = \sigma/7 \ln M$ (as defined by Lemma 1), and $\mu \in [\lambda(x_0), 1]$. Set x to x_0, and compute p from (2.2).

Step 1. (Block optimization.) For each block $k = 1, \cdots, K$, perform the following two steps:

Step 1.1. Call a $(\sigma/7, q)$-good block solver $\mathcal{BS}(f^k, p, \mu)$ to solve $(\mathcal{P}^k)$. Since $\lambda^* \leq \lambda(x_0) \leq \mu$, we have $B^k_\mu = \{ x^k \in B^k,\ f^k(x^k) \leq \mu e \} \neq \emptyset$, and the block must report an $\hat{x}^k \in B^k$ such that

$$\begin{aligned} & f^k(\hat{x}^k) \leq \mu q e, \quad \text{and} \\ & p^T f^k(\hat{x}^k) \leq \left(1 + \frac{\sigma}{7}\right) \min\{ p^T f^k(x^k) \mid x^k \in B^k_\mu \} = \left(1 + \frac{\sigma}{7}\right) g_\mu(p). \end{aligned} \tag{4.1}$$

Step 1.2. For some step length $\tau^k \in (0, 1]$ (to be specified later), compute

$$\Delta^k_\Phi(\tau^k) = \Phi_t(y(k, \tau^k)) - \Phi_t(x) = \ln \sum_{m=1}^{M} p_m \exp\left\{ \frac{f^k_m((1-\tau^k)x^k + \tau^k \hat{x}^k) - f^k_m(x^k)}{t} \right\},$$

where $y(k, \tau^k)$ is the point obtained from x by replacing its k-th block coordinate x^k by $(1 - \tau^k)x^k + \tau^k \hat{x}^k$.

Step 2. (Stopping criterion.) If the inequality $\sum_{k=1}^{K} \Delta^k_\Phi(\tau^k) \leq -\sigma^2/28\mu q$ is not satisfied, report x and stop.

Step 3. (Coordination.) Select a k that minimizes $\Delta^k_\Phi(\tau^k)$ over $k = 1, \cdots, K$. Replace x by $y(k, \tau^k)$. For this new iterate x, compute a new p by (2.2). Go to Step 1. ∎

This generic statement of Algorithm $\mathcal{A}$ can be modified and implemented in different ways. We shall introduce a randomized variant of this algorithm in Section 6, and discuss

computational issues in Section 9.1. One of these is the normalization of computations in order to avoid the appearance of large numbers in the computation of p and $\Delta_{\Phi}^{k}(\tau^k)$ above.

In the next section we shall prove the correctness of Algorithm $\mathcal{A}$ and analyze its performance.

5. Analysis of Algorithm $\mathcal{A}$.

We begin with the following technical Lemma.

Lemma 2. Let $E[f_1,\cdots,f_M] = \sum_{m=1}^{M} p_m f_m$ be the weighted average of M given numbers $f_1,\cdots,f_M$, with weights $p_m = \exp\{f_m/t\} \;/\; \sum_{i=1}^{M} \exp\{f_i/t\}$, $m = 1,\cdots,M$, for some positive t. Then,

$$\max\{\, f_1,\cdots,f_M \,\} - E[f_1,\cdots,f_M] \;\leq\; t \ln M,$$

that is, for small t the so-weighted average is close to largest of the given numbers.

Proof. Let $\max\{\, f_1,\cdots,f_M \,\} = f_1$. Then,

$$\begin{aligned}
\max\{\, f_1,\cdots,f_M \,\} - E[f_1,\cdots,f_M] &= \sum_{m=1}^{M} (f_1 - f_m) p_m \\
&= \sum_{m=1}^{M} (f_1 - f_m) \exp\left\{ -\frac{f_1 - f_m}{t} \right\} \Bigg/ \sum_{m=1}^{M} \exp\left\{ -\frac{f_1 - f_m}{t} \right\} \\
&= t \left\{ \frac{\sum_{m=2}^{M} \rho_m \exp\{-\rho_m\}}{1 + \sum_{m=2}^{M} \exp\{-\rho_m\}} \right\},
\end{aligned}$$

where $\rho_m = (f_1 - f_m)/t \geq 0$. It remains to show that the quantity in the last large curly brackets above cannot exceed $\ln M$. To this end, suppose it exceeds some positive number θ, or equivalently,

$$\sum_{m=2}^{M} (\rho_m - \theta) \exp\{-\rho_m\} > \theta. \tag{5.1}$$

Consider the inequality $(\rho_m - \theta)\exp\{-\rho_m\} \leq \exp\{-1-\theta\}$, which is valid for all $\theta > 0$ and ρ_m. For a fixed θ, its left-hand side achieves its maximum at $\rho_m = \theta + 1$. For this value of ρ_m, (5.1) implies that $\theta \exp\{1+\theta\} < M - 1$, leading to a contradiction for $\theta = \ln M$. ∎

We are now ready to prove Algorithm $\mathcal{A}$ correct and state a bound on the number of its coordination steps for the special case of step length $\tau^1 = \cdots = \tau^K$=constant.

Theorem 1. For fixed $\tau^k = \tau = \sigma t/6\mu q$, and for an initial $x_0 \in B$ such that $\lambda(x_0) \leq 1$, Algorithm $\mathcal{A}(f, x_0, \sigma, q)$ halts in

$$N = O\left(\frac{qK}{\sigma^2} \ln M \ln\left(2 + \frac{\lambda(x_0) - \lambda^*}{\sigma}\right)\right) \tag{5.2}$$

coordination steps, with an $x \in B$ such that $\lambda(x) \leq \lambda^* + \sigma$.

Proof. Let x be the current iterate, and let $k \in \{1, \cdots, K\}$ be a block selected for processing in Step 1 of the algorithm. We shall first give an upper bound on the difference of the potential function

$$\Delta_\Phi^k(\tau) = \ln \sum_{m=1}^{M} p_m \exp\left\{\frac{f_m^k((1-\tau)x^k + \tau\hat{x}^k) - f_m^k(x^k)}{t}\right\},$$

computed in Step 1.2 for the value $\tau^k = \tau$ given in the Theorem. For convenience, denote $f_m^k = f_m^k(x^k)$, $\hat{f}_m^k = f_m^k(\hat{x}^k)$. By the convexity of $f_m^k(\cdot)$ we have

$$\begin{aligned}\Delta_\Phi^k(\tau) &\leq \ln \sum_{m=1}^{M} p_m \exp\left\{-\frac{\tau}{t}(f_m^k - \hat{f}_m^k)\right\} \\ &\leq \sum_{m=1}^{M} p_m \left(\exp\left\{-\frac{\tau}{t}(f_m^k - \hat{f}_m^k)\right\} - 1\right). \end{aligned} \tag{5.3}$$

The last inequality follows from the observation that $\ln(u) \leq u - 1$ for all $u > 0$, and from the fact that $\sum_{m=1}^{M} p_m = 1$. We claim that

$$f_m^k, \hat{f}_m^k \in [0, \mu q]. \tag{5.4}$$

This is clearly the case for $\hat{f}_m^k$ by (4.1). It is also true initially for f_m^k since $\mu q \geq \mu \geq \lambda(x_0)$. At each subsequent iteration, the algorithm replaces x^k by a convex combination of x^k and $\hat{x}^k$, and the convexity of $f_m^k(\cdot)$ implies (5.4).

Now, using (5.4) and the assumptions $\tau^k = \tau = \sigma t/6\mu q$, $\sigma \leq 1/3$, we have

$$\left|-\frac{\tau}{t}(f_m^k - \hat{f}_m^k)\right| \leq \frac{\sigma}{6} \leq \frac{1}{18}, \quad m = 1, \cdots, M. \tag{5.5}$$

Since $\exp\{u\} - 1 \leq u + \frac{6}{11}u^2$ for all $|u| \leq \frac{1}{18}$, the inequality (5.3) can be further bounded as follows:

$$\Delta_\Phi^k(\tau) \leq -\frac{\tau}{t} \sum_{m=1}^{M} p_m(f_m^k - \hat{f}_m^k) + \frac{6\tau^2}{11t^2} \sum_{m=1}^{M} p_m(f_m^k - \hat{f}_m^k)^2.$$

In view of (5.4), we have $(f_m^k - \hat{f}_m^k)^2 \leq (f_m^k)^2 + (\hat{f}_m^k)^2 \leq \mu q(f_m^k + \hat{f}_m^k)$. Substituting this bound into the preceding inequality we obtain:

$$\Delta_\Phi^k(\tau) \leq -\frac{\tau}{t}\left(1 - \frac{6\mu q\tau}{11t}\right) p^T(f^k - \hat{f}^k) + \frac{12\mu q\tau^2}{11t^2} p^T \hat{f}^k. \tag{5.6}$$

Now, in order to show that the best block k selected in Step 3 of the algorithm results in a substantial decrease in the potential, we give an upper bound on the average decrease over all blocks. To this end, we sum the upper bounds (5.6) for all k to obtain

$$\sum_{k=1}^{K} \Delta_\Phi^k(\tau) \leq -\tau\left(1 - \frac{6\mu q\tau}{11t}\right) D + \frac{12\mu q\tau^2}{11t^2} C, \tag{5.7}$$

where $D = D_1 - D_2$, $D_1 = p^T f/t$, $D_2 = p^T \hat{f}/t = C/t$, and $C = p^T \hat{f}$. The quantity C can easily be bounded from above by using both (4.1) and the duality relation ($\mathcal{D}$) of Section 1, as follows:

$$C \leq \left(1 + \frac{\sigma}{7}\right) \sum_{k=1}^{K} g_\mu^k(p) \leq \left(1 + \frac{\sigma}{7}\right) \lambda^* \leq \frac{22}{21}\lambda^* \leq \frac{22}{21}\lambda(x_0) \leq \frac{22}{21}. \tag{5.8}$$

Next, we derive a lower bound on $D = D_1 - D_2$ in terms of

$$\delta(x) = \Phi_t(x) - \Phi_t^*,$$

the absolute accuracy of minimization of the potential function[2]. To bound D_1 from below note that from Lemma 2, $D_1 = E[f_1, \cdots, f_M]/t \geq \max\{f_1, \cdots, f_M\}/t - \ln M = \lambda(x)/t - \ln M$. By the second inequality of (2.3), $\lambda(x)/t \geq \Phi_t(x) - \ln M$. Hence, $D_1 \geq \Phi_t(x) - 2\ln M$. To bound $D_2 = C/t$ from above, use the second inequality of (5.8), $\lambda^* \leq 1$, property (2.4), and the parameter values of Step 0, to write:

$$D_2 \leq \frac{\lambda^*}{t} + \frac{\lambda^*\sigma}{7t} \leq \frac{\lambda^*}{t} + \frac{\sigma}{7t} \leq \Phi_t^* + \frac{\sigma}{7t} = \Phi_t^* + \ln M.$$

Thus,

$$D = D_1 - D_2 \geq \Phi_t(x) - \Phi_t^* - 3\ln M = \delta(x) - 3\ln M. \tag{5.9}$$

[2] For linear functions f_m^k and for exact block solvers, the quantity D provides the decrease in the linear approximation to the potential function at x. By the convexity of $\Phi_t(\cdot)$, this implies $D \geq \delta(x)$.

By the stopping criterion in Step 2 and the choice of k in Step 3, the algorithm generates monotonically decreasing $\delta(x)$. By Lemma 1, if an iterate x enters the region $\delta(x) \leq \delta = 6 \ln M$, then x minimizes $\lambda(x)$ over $x \in B$ to the desired absolute accuracy σ, albeit undetected when it occurs. For this reason, we consider the total number of iterations in two parts: N_1, the number of iterations needed to enter the region $\delta(x) \leq \delta$, and N_2, the number of additional iterations required to obtain the stopping criterion.

We shall first derive a bound for N_1. Suppose that $\delta(x) \geq \delta = 6 \ln M$. By (5.9), this implies that $D \geq \delta(x)/2$. Substituting this bound, along with $C \leq 22/21$ into (5.7), results in:

$$\begin{aligned}\sum_{k=1}^{K} \Delta_{\Phi}^{k}(\tau) &\leq -\frac{\tau}{2}\left(1 - \frac{6\mu q \tau}{11t}\right)\delta(x) + \frac{8\mu q \tau^2}{7t^2} \\ &\leq -\delta(x)\left\{\frac{\tau}{2}\left(1 - \frac{6\mu q \tau}{11t}\right) - \frac{4\mu q \tau^2}{21 t^2 \ln M}\right\} \\ &< -\delta(x)\left\{\frac{\tau}{2} - \frac{3\mu q \tau^2}{2\sigma t}\right\}.\end{aligned}$$

The last inequality follows from $\sigma \leq 1/3$ and from the initialization in Step 0 of the algorithm. Substituting our value of $\tau = \sigma t / 6\mu q$, which maximizes the expression in the last pair of curly brackets, we obtain

$$\sum_{k=1}^{K} \Delta_{\Phi}^{k}(\tau) \leq -\left(\frac{c\sigma^2}{\mu q \ln M}\right)\delta(x) \qquad (5.10).$$

with $c = 1/168$. In particular, for $\delta(x) \geq \delta$ we have the inequality given in Step 2, whose negation is the stopping criterion[3]. This proves the correctness of Algorithm $\mathcal{A}$. Furthermore, (5.10) and the choice of the best k in Step 3 of the algorithm yield the recurrence

$$\delta(x_l) \geq \delta \quad \Rightarrow \quad \delta(x_{l+1}) \leq \left(1 - \frac{c\sigma^2}{\mu q K \ln M}\right)\delta(x_l),$$

where x_l is the l-th iterate of the algorithm. Hence, the number of iterations N_1 of the algorithm sufficient for an x to enter the region $\delta(x) \leq \delta$ can be bounded by

$$N_1 \leq \left\lceil \frac{\mu q K \ln M}{c\sigma^2} \ln \frac{\delta(x_0)}{\delta} \right\rceil.$$

[3] For the linear case, where $D \geq \delta(x)$, the optimal choice of the step length τ is approximately twice the value used in Theorem 1. Consequently, the value of c (the guaranteed decrease in the potential in the stopping criterion of Step 2) increases and the iteration bound decreases by about a factor of four.

In view of (2.3)-(2.4),

$$\delta(x_0) \leq \frac{\lambda(x_0) - \lambda^*}{t} + \ln M,$$

which upon substitution of our values for t and δ, gives

$$\ln \frac{\delta(x_0)}{\delta} \leq \ln \left(\frac{1}{6} + \frac{7}{6} \frac{\lambda(x_0) - \lambda^*}{\sigma} \right) = O\left(\ln \left(2 + \frac{\lambda(x_0) - \lambda^*}{\sigma} \right) \right). \tag{5.11}$$

Hence, N_1 is bounded by

$$N_1 = O\left(\frac{\mu q K}{\sigma^2} \ln M \ln \left(2 + \frac{\lambda(x_0) - \lambda^*}{\sigma} \right) \right).$$

It remains to bound N_2. By the stopping criterion and the choice of k in Step 3, the potential decreases by at least $\Delta = \sigma^2 / 28 \mu q K$ at each iteration. Therefore,

$$N_2 \leq \left\lceil \frac{\delta}{\Delta} \right\rceil = O\left(\frac{\mu q K \ln M}{\sigma^2} \right).$$

Since $\mu \leq 1$, the above bounds for N_1 and N_2 prove the Theorem. ∎

6. A randomized variant of Algorithm $\mathcal{A}$.

Each iteration of Algorithm $\mathcal{A}$ solves K independent block problems and then selects the best one of these solutions to compute a new iterate x and the corresponding price vector p. A natural question is whether solving fewer block problems per step might provide comparable performance. In this section, we shall state and analyze a *randomized variant* of this algorithm, *Algorithm* $\mathcal{R}$, which solves two block problems per step, on the average. As we shall see, the expected number of iterations of this variant is within a constant factor (of about two) of the bound given by Theorem 1 for the deterministic Algorithm $\mathcal{A}$. Thus, the randomized algorithm is advantageous for problems with a relatively large number of blocks.

Algorithm $\mathcal{R}(f, x_0, \sigma, q)$:

Step R0. (Initialization.) Same as Step 0 of Algorithm $\mathcal{A}$.

Step R1. (Randomization.) Flip a $K+1$-sided fair coin giving the outcome $k \in \{0, 1, \cdots, K\}$. If $k = 0$, go to Step R3.

Step R2. (Process random block k.) Perform Steps 1.1 - 1.2 of Algorithm $\mathcal{A}$ for the k-th block, which compute $\Delta_\Phi^k(\tau^k)$. If $\Delta_\Phi^k(\tau^k) < 0$, replace x by $y(k, \tau^k)$ (this guarantees the monotonicity of potential function values), and compute a new p by (2.2). Go to Step R1.

Step R3. (Process all blocks.) Perform one complete iteration (Steps 1-3) of Algorithm $\mathcal{A}$. If Algorithm $\mathcal{A}$ does not halt, go to Step R1. ∎

Our analysis of Algorithm $\mathcal{R}$ uses the same uniform step size as Theorem 1.

Theorem 2. For fixed $\tau^k = \tau = \sigma t/6\mu q$, and for an initial $x_0 \in B$ such that $\lambda(x_0) \leq 1$, Algorithm $\mathcal{R}(f, x_0, \sigma, q)$ halts in expected

$$E[N] = O\left(\frac{qK}{\sigma^2} \ln M \ln\left(2 + \frac{\lambda(x_0) - \lambda^*}{\sigma}\right)\right)$$

coordination steps, with an $x \in B$ such that $\lambda(x) \leq \lambda^* + \sigma$.

Proof. Algorithm $\mathcal{R}$ halts only if $k = 0$ and the stopping criterion of Algorithm $\mathcal{A}$ is satisfied. That is, $\mathcal{R}$ halts if the current outcome in Step R1 is $k = 0$ and $\sum_{k=1}^{K} \Delta_\Phi^k(\tau^k) > -c_1$, where $c_1 = \sigma^2/28\mu q$. In particular, $\mathcal{R}$ halts if $k = 0$ and $\sum_{k=1}^{K} \min\{0, \Delta_\Phi^k(\tau^k)\} > -c_1$. To simplify the proof, we shall replace the stopping criterion in Step 2 of Algorithm $\mathcal{A}$ by this stronger condition. This can only increase the expected number of iterations.

Algorithm $\mathcal{R}$ corresponds to a stochastic process which can be modeled as a *random descent* P in an infinite $(K+1)$-ary rooted tree (arborescence) G as follows. The root of G corresponds to the initial iterate $x_0 \in B$, and the $(K+1)^l$ vertices in the l-th level of G correspond to all possible l-th iterates $x_l \in B$ of Algorithm $\mathcal{R}$. The adjacency $A(x_l)$ consists of $K+1$ arcs from x_l to its $K+1$ children, each of which represents one of the $K+1$ outcomes of the coin flip in Step R1. For each vertex x_l, the leftmost arc $e_{x_l} \in A(x_l)$ models the execution of one iteration of Algorithm $\mathcal{A}$, i.e. the outcome $k = 0$ in Step R1.

Assign the nonnegative weight $\delta(x_l) = \Phi_t(x_l) - \Phi^*$ to each vertex x_l of G. Since Algorithm $\mathcal{R}$ generates monotonically decreasing values of the potential, the sequence of vertex weights encountered by a path P from the root to any vertex of G is nonincreasing. Also assign the nonnegative weight $w(x_l, x_{l+1,k}) = \delta(x_l) - \delta(x_{l+1,k}) = \Phi_t(x_l) - \Phi_t(x_{l+1,k})$ to each arc $(x_l, x_{l+1,k})$, $k = 0, 1, \cdots, K$, of G. We shall call the leftmost arc in the $A(x_l)$ of any vertex x_l a *stopping arc* if the total weight of the remaining K arcs, $w(A(x_l)\backslash\{e_{x_l}\})$, is less than c_1. This corresponds to the modified stopping criterion stated in the first paragraph of this proof.

Imagine a stroller who starts at the root of G with an amount of cash $\delta(x_0)$ and who descends the tree along a path P, randomly selecting an arc $e \in A(x_l)$ to traverse at each

vertex x_l he visits. For every arc e the stroller traverses, he pays a "toll" $w(e)$. Since for any N, the weight $w_N(P)$ of an arbitrary descent P with N arcs from the root cannot exceed $\delta(x_0)$, the stroller can never go into debt and his walk will not terminate. Now, modify the weight of all stopping arcs to be any large number exceeding $\delta(x_0)$. Then, the first time the stroller traverses a stopping arc he goes into debt, and conversely. This means that in this modified tree model of Algorithm $\mathcal{R}$, the expected number of iterations of $\mathcal{R}$ is equal to the expected number of steps $E[N]$ needed to bankrupt the stroller.

Represent the random variable N as the sum of two random variables: N_1, the time (number of arcs traversed) when the stroller reaches the "level of poverty" $2\delta = 12 \ln M$, and N_2, the additional time to bankrupt the stroller. In order to bound the expectations $E[N_1]$ and $E[N_2]$, observe that the (redefined) arc-weight function w has the following two properties:

1) $w(A(x_l)) > c_1 = \sigma^2/28\mu q$ for any vertex x_l of G, and

2) $\delta(x_l) \geq \delta = 6 \ln M \;\Rightarrow\; w(A(x_l)) \geq c_2 \delta(x_l)$, where $c_2 = \sigma^2/168\mu q \ln M$. This inequality corresponds to (5.10), and it can be shown exactly as in the proof of Theorem 1. This is a necessary exercise because the sequence of $\lambda(x)$ is not monotonic in general, and thus the inequality $\lambda(x_l) \leq 1$ is not assured for any iteration of Algorithm $\mathcal{R}$. The proof of (5.10) in Theorem 1 is valid because it uses the assumption $\lambda(x) \leq 1$ only for the very first iterate x_0.

To proceed we need the following Lemma, whose proof we shall postpone.

Lemma 3. Let G be a d-ary tree with a nonnegative arc-weight function u such that $u(A(v)) \geq C$ for each vertex of $v \in G$, where $C > 0$ is a constant. Let $U \geq 0$ be a given threshold weight. For a random descent P in G, let $T = \min\{\, t \mid u_t(P) \geq U \,\}$, the first instant when the weight $u_t(P)$ of the first t arcs in P reaches U. Then, $E[T] \leq d(1+U/C)$.

Using this Lemma with $d = K+1$, $U = 2\delta$, and $C = c_1 = \sigma^2/28\mu q$ from property (1) and the fact that $\mu \leq 1$, we immediately obtain the bound

$$E[N_2] \leq (K+1)\left(1 + \frac{2\delta}{c_1}\right) = O\left(\frac{qK \ln M}{\sigma^2}\right).$$

To bound $E[N_1]$, suppose that at some instant we have $\delta(x_l) \geq 2\delta$. Then, by property (2) and by Lemma 3 with $d = K+1$, $U = \delta(x_l)/2$, and $C = c_2\delta(x_l)/2$, the expected number of iterations needed to halve $\delta(x_l)$ does not exceed

$$(K+1)\left(1+\frac{1}{c_2}\right) = O\left(\frac{qK\ln M}{\sigma^2}\right),$$

which results in the bound:

$$E[N_1] = O\left(\frac{qK\ln M}{\sigma^2}\left\lceil \ln\frac{\delta(x_0)}{\delta}\right\rceil\right).$$

The bounds for $E[N_1]$ and $E[N_2]$, along with (5.11), prove the Theorem. ∎

We now give a simple inductive proof of Lemma 3. Samorodnitsky [16] gave an earlier proof of this Lemma using the optional stopping theorem for martingales.

Proof of Lemma 3. Without loss of generality we can assume that $u(A(v)) = C$ for each vertex of $v \in G$. We first prove the Lemma for integer-valued u, U and C. Let v_0 be the root of G, and denote the weights $u(A(v_0))$ by $u_1, \cdots, u_d$. The expectation $E[T]$, viewed as a function $E_G(U)$ of G and U, satisfies the recurrent inequality $E_G(U) \leq 1 + (1/d)\sum_{k=1}^{d} E_{G_k}([U-u_k]^+)$, where G_k denotes the subtree of G rooted at the k-th child of the root, and $[x]^+ = \max\{0, x\}$. Taking suprema over the right-hand and then of the left-hand side of this inequality yields the recurrence:

$$E(U) \leq \max\left\{1 + \frac{1}{d}\sum_{k=1}^{d} E([U-u_k]^+)\right\} \tag{6.1}$$

where $E(U)$ is the supremum of $E_G(U)$ over all G satisfying the assumption $u(A(v)) = C$ for each $v \in G$, and the maximum is taken over all d-tuples of integers $\{u_1, \cdots, u_d\} \geq 0$ that sum up to C. Observe that $E(0) = 0$, and that $E(U)$ is a finite nondecreasing function of $U = 0, 1, \cdots$. Now use (6.1) to prove by induction on U that $E(U) \leq d(1+U/C)$, which coincides with the claim of the Lemma for the integer case.

For a given $U = 1, 2, \cdots$, the maximum in (6.1) is attained for some d-tuple $\{u_1, \cdots, u_d\}$. If $u_k \geq U$ for some $k \in \{1, \cdots, d\}$, then one of summands in (6.1) is zero. Since $E([U-u_k]^+) \leq E(U)$, we have $E(U) \leq 1 + 0 + \frac{d-1}{d}E(U)$, or $E(U) \leq d$. Suppose now that $u_k < U$ for all $1 \leq k \leq d$, and that there are $0 \leq p < d$ zero u_k's, say, $u_1 = \cdots = u_p = 0$. Rewriting (6.1) as $\left(1-\frac{p}{d}\right)E(U) \leq 1 + \frac{1}{d}\sum_{k=p+1}^{d} E(U-u_k)$ and using the induction hypothesis

$E(U-u_k) \le d(1+(U-u_k)/C)$ and $\sum_{k=p+1}^{d} u_k = C$, completes the proof for the integer case, and consequently for rational-valued u, C and U with common denominator $Q > 0$. The real-valued case follows by approximating U from above and u at each arc from below by nonnegative rational numbers with denominator $Q \to \infty$. ∎

7. A ternary search procedure.

The algorithms presented in Sections 4 and 6 solve problem $(\mathcal{P})$ to an absolute accuracy σ. In this section we shall state a *ternary search* procedure $\mathcal{TS}$ which computes the optimal value λ^* to our original problem $(\mathcal{P}_\varepsilon)$ for a given relative accuracy $\varepsilon > 0$, assuming that an initial approximation $x_0 \in B$ is given. We shall no longer assume that $\lambda(x_0) \le 1$.

A deterministic implementation of this procedure requires an initial point $x_0 \in B$, it uses Algorithm $\mathcal{A}$ as a subroutine, and maintains a lower bound $\underline{\lambda}$ and an upper bound $\bar{\lambda}$ to the optimal value λ^*. Its randomized variant uses Algorithm $\mathcal{R}$ instead of Algorithm $\mathcal{A}$.

Algorithm $\mathcal{TS}(x_0, \varepsilon, q)$:

Step 0. Set $\underline{\lambda} = 0$, $x = x_0$, and $\bar{\lambda} = \lambda(x)$. Then repeat the following three steps until the stopping criterion $\lambda(x) = \bar{\lambda} \le (1+\varepsilon)\underline{\lambda}$ is satisfied:

Step 1. Set

$$\nu = \frac{1}{\bar{\lambda} + 2\underline{\lambda}}, \quad \sigma = \frac{1}{3}\left(\frac{\bar{\lambda} - \underline{\lambda}}{\bar{\lambda} + 2\underline{\lambda}}\right) \tag{7.1}$$

and call Algorithm $\mathcal{A}(\nu f, x, \sigma, q)$. (Theorem 1 can be applied here since $\sigma \le 1/3$ and $\nu\lambda(x) = \nu\bar{\lambda} \le 1$.) Algorithm $\mathcal{A}$ provides a point $y \in B$ such that $\nu\lambda(y) \le \nu\lambda^* + \sigma$, i.e.

$$\lambda(y) \le \lambda^* + (\bar{\lambda} - \underline{\lambda})/3. \tag{7.2}$$

Step 2. If $\lambda(y) \le \lambda_2 = (1/3)\underline{\lambda} + (2/3)\bar{\lambda}$, replace x by y and the upper bound $\bar{\lambda}$ by $\lambda(y)$.

Step 3. If $\lambda_2 < \lambda(y)$, then $\lambda^* > \lambda_1 = (2/3)\underline{\lambda} + (1/3)\bar{\lambda}$ by (7.2). Replace the lower bound $\underline{\lambda}$ by λ_1. ∎

Procedure $\mathcal{TS}$ is correct: it generates a sequence of intervals $[\underline{\lambda}, \lambda(x)]$, the optimal value λ^* is contained in each such interval, and each interval shrinks by at least a factor of 2/3 from one iteration of $\mathcal{TS}$ to the next.

In order to bound the complexity of $\mathcal{TS}$, observe that for each iteration,

$$\frac{\nu\lambda(x) - \nu\lambda^*}{\sigma} = \frac{\nu\bar{\lambda} - \nu\lambda^*}{\sigma} \leq \frac{\nu\bar{\lambda} - \nu\underline{\lambda}}{\sigma} = 3.$$

Thus, by (5.2), the number N of coordination steps of Algorithm $\mathcal{A}$ for each iteration of $\mathcal{TS}$ is bounded by

$$N = O\left(\frac{qK \ln M}{\sigma^2}\right) = O\left(qK \ln M \left(\frac{\bar{\lambda} + 2\underline{\lambda}}{\bar{\lambda} - \underline{\lambda}}\right)^2\right),$$

where $\underline{\lambda}$ and $\bar{\lambda}$ are the bounds for the current iteration. We claim that

$$\sum \left(\frac{\bar{\lambda} + 2\underline{\lambda}}{\bar{\lambda} - \underline{\lambda}}\right)^2 = O\left(\varepsilon^{-2} + \ln \frac{\lambda(x_0)}{\lambda^*}\right), \tag{7.3}$$

with the sum taken over all iterations of $\mathcal{TS}$. To see this, partition the iterations performed by $\mathcal{TS}$ into two parts: iterations for which $\underline{\lambda} = 0$, and the subsequent iterations for which $\underline{\lambda} > 0$. Each iteration in the first part causes the upper bound $\bar{\lambda} = \lambda(x)$ to decrease by at least a factor of $2/3$. Hence,

$$\sum_{\underline{\lambda}=0} \left(\frac{\bar{\lambda} + 2\underline{\lambda}}{\bar{\lambda} - \underline{\lambda}}\right)^2 = O\left(\ln \frac{\lambda(x_0)}{\lambda^*}\right).$$

For each iteration in the second part, the positivity of $\underline{\lambda}$ implies that $\bar{\lambda} \leq 3\underline{\lambda}$. Consequently,

$$\frac{1}{3}\left(\frac{\underline{\lambda}}{\bar{\lambda} - \underline{\lambda}}\right) \leq \frac{1}{5}\left(\frac{\bar{\lambda} + 2\underline{\lambda}}{\bar{\lambda} - \underline{\lambda}}\right) \leq \frac{\underline{\lambda}}{\bar{\lambda} - \underline{\lambda}} \leq \frac{\lambda^*}{\bar{\lambda} - \underline{\lambda}} \leq \frac{3\underline{\lambda}}{\bar{\lambda} - \underline{\lambda}}. \tag{7.4}$$

that is, $(\bar{\lambda} + 2\underline{\lambda})/(\bar{\lambda} - \underline{\lambda})$ and $\underline{\lambda}/(\bar{\lambda} - \underline{\lambda})$ are of the same order as $\lambda^*/(\bar{\lambda} - \underline{\lambda})$, the reciprocal of the current relative error.

Since at each iteration the gap $\bar{\lambda} - \underline{\lambda}$ decreases by at least a factor of $2/3$, by the stopping criterion of $\mathcal{TS}$ we have

$$\sum_{\underline{\lambda}>0} \left(\frac{\bar{\lambda} + 2\underline{\lambda}}{\bar{\lambda} - \underline{\lambda}}\right)^2 = O\left(\varepsilon^{-2}\left[1 + \left(\frac{2}{3}\right)^2 + \left(\frac{2}{3}\right)^4 + \cdots\right]\right) = O(\varepsilon^{-2}),$$

where ε is the desired (final) relative accuracy of the approximation. This proves (7.3).

We have thus shown that the total number of coordination steps $N(\varepsilon)$ required to solve $x \in B$, $\lambda(x) \leq (1 + \varepsilon)\lambda^*$ is bounded by

$$N(\varepsilon) = O\left(qK \ln M \left(\varepsilon^{-2} + \ln \frac{\lambda(x_0)}{\lambda^*}\right)\right). \tag{7.5}$$

Clearly, this bound also holds for the expected number of iterations for the randomized variant of $\mathcal{TS}$.

Note that by the choice of σ in (7.1) and the inequalities (7.4), Algorithm $\mathcal{TS}(x_0, \varepsilon, q)$ calls an (ε', q)-good block solver via Algorithm $\mathcal{A}$, where $\varepsilon' = \varepsilon/(7 \cdot 3 \cdot 5) \approx 0.01\varepsilon$. That is, the relative optimization accuracy demanded of the block solver differs only by a constant factor from the relative tolerance prescribed for $(\mathcal{P}_e)$. This is true for the randomized variant of $\mathcal{TS}$ as well.

8. The overall approximation scheme and its complexity.

The ternary search procedure $\mathcal{TS}$ of the previous section solves the original problem $(\mathcal{P}_e)$ provided that an arbitrary $x_0 \in B$ is known. The number of block-solver calls $\mathcal{TS}$ generates depends only logarithmically on the relative accuracy of the available approximation. Although this is not an overly restrictive assumption in practice, it can be easily removed by using a simple *greedy initialization procedure* $\mathcal{GP}$ which we shall describe below. Thus, our overall scheme for $(\mathcal{P}_e)$ will consist of $\mathcal{GP}$ followed by $\mathcal{TS}$. We shall denote the resulting deterministic and randomized variants of this scheme by $\mathcal{DS}$ and $\mathcal{RS}$, respectively.

The greedy initialization procedure $\mathcal{GP}$ operates separately on each block k as follows. Initially, set $p = e/M \in P$ and $\mu = \infty$. Then, call an (ε, q)-good block solver to compute a solution $\hat{x}^k$ to problem $(\mathcal{P}^k)$:

$$p^T f^k(\hat{x}^k) \leq (1+\varepsilon) \min\{ p^T f^k(x^k) \mid x^k \in B^k_\mu \}. \tag{8.1}$$

(The block solver must provide such a solution since initially $\mu = \infty$.) Clearly, the above minimum does not exceed $\lambda^*_k = \min\{ \lambda \mid f^k(x^k) \leq \lambda e, \quad x^k \in B^k \}$. This fact, and the nonnegativity of the $f^k(\cdot)$, imply that $f^k(\hat{x}^k) \leq \mu e$ with $\mu \leq (1+\varepsilon) M \lambda^*_k$. Now, halve μ and repeat the above step for any p until the block solver reports (possibly incorrectly) that $B^k_{\mu q} = \emptyset$ for some value of $\mu = \mu_1$. Recalling our assumptions on the performance of the block solver in Section 3, we conclude that for $\mu_2 = 2q\mu_1$, the set $B^k_{\mu_2} \neq \emptyset$. Now, use the block solver for $\mu = \mu_2$ to compute an $\hat{x}^k$ to (8.1) and save this solution as x^k_0. It is easy to see that $f^k(x^k_0) \leq 2q^2 \lambda^*_k e$.

The total number of block-solver calls issued by $\mathcal{GP}$ is $O(\ln M + \ln(1+\varepsilon) + \ln q)$ per block. Since $f^k(x^k_0) \leq 2q^2 \lambda^*_k e$ for each x^k_0 computed by $\mathcal{GP}$, we have $f(x_0) \leq 2q^2 \sum_{k=1}^K \lambda^*_k e \leq$

$2Kq^2\lambda^* e$. Thus $\lambda(x_0)/\lambda^* \leq 2Kq^2$. Incorporating these bounds into (7.5) of Section 7, we obtain our final bounds for the overall algorithm.

Theorem 3. The deterministic approximation scheme $\mathcal{DS}$ solves problem $(\mathcal{P})$ with nonnegative-valued block-separable convex coupling functions $f(x)$ to a given relative accuracy $\varepsilon \in (0,1]$ in

$$N(\varepsilon) = O\left(qK \ln M \left(\varepsilon^{-2} + \ln qK\right)\right)$$

coordination steps, each generating K parallel calls to $(\text{const} \cdot \varepsilon, q)$-good block solvers. This expression also bounds the expected number of calls to such block solvers for the randomized variant $\mathcal{RS}$. ∎

Recalling Proposition 1 on the equivalence of block and box-block solvers discussed in Section 3, Theorem 3 implies the following Corollary.

Corollary 1. If in addition to the assumptions of Theorem 3, the coupling functions are additively separable, the bounds of Theorem 3 are

$$N(\varepsilon) = O\left(qSK \ln M \left(\varepsilon^{-2} + \ln qSK\right)\right)$$

coordination steps using $(\text{const} \cdot \varepsilon, q)$-good box-block solvers. ∎

Finally, for $q = 1 + O(\varepsilon)$, or even for $q = O(1)$, Theorem 3 and Corollary 1 provide the bounds (1.4) and (1.5) claimed in the Introduction.

9. Remarks.

We have shown how to render the block-coordinate descent strategy efficient for computing an approximate solution to the nonlinear programming problem $(\mathcal{P})$. We have developed a deterministic and a randomized algorithm, both of which are fully polynomial in the number of blocks they solve and in ε^{-1}. These algorithms result in fully polynomial time approximation schemes for some special cases of linear programming, such as ordinary and generalized flow networks with additional constraints, and generalized multicommodity concurrent flows. Our analysis suggests that the algorithms presented here, as well as in [14] and [10], can have a combinatorial flavor only to the extent that some applications result in block problems amenable to efficient combinatorial methods. We shall illustrate this point with several applications later in this section. Since the number of coupling

constraints M enters the complexity bounds only logarithmically, we expect that additional applications will be found.

9.1. Implementation issues. As we have seen, Algorithms $\mathcal{A}$ and $\mathcal{R}$ of Sections 4 and 6 are the key ingredients of the overall method for problem $(\mathcal{P})$. The generic description of these algorithms suggests a number of straightforward enhancements. We enumerate some of these below.

Stopping criteria. There are at least two additional stopping criteria that can be incorporated into Algorithm $\mathcal{A}$ and into the ternary search procedure $\mathcal{TS}$ to affect an earlier termination. First, by (5.9), Algorithm $\mathcal{A}$ can be terminated whenever $D \leq 3 \ln M$ or equivalently whenever the approximate duality gap $p^T(f(x) - f(\hat{x})) \leq \frac{3}{7}\sigma$. Second, since Algorithm $\mathcal{A}$ is used by $\mathcal{TS}$ as a subroutine, its branching rule $\lambda(y) \leq \lambda_2 = \frac{1}{3}\underline{\lambda} + \frac{2}{3}\bar{\lambda}$ can also be used to terminate $\mathcal{A}$. The same criteria also benefit Algorithm $\mathcal{R}$ since it uses $\mathcal{A}$ as a subroutine.

Normalized computations. At each iteration of Algorithms $\mathcal{A}$ and $\mathcal{R}$ it is necessary to recompute the price vector p by (2.2), which can be written in the normalized form [14]:

$$p_m = \frac{\exp\{(f_m(x) - \lambda(x))/t\}}{\sum_{\mu=1}^{M} \exp\{(f_\mu(x) - \lambda(x))/t\}},$$

where the exponents are nonpositive. Furthermore, the exponents in the expression for $\Delta_\Phi^k(\tau^k)$ used in Step 1.2 of the generic form of Algorithm $\mathcal{A}$ have been shown to be small by the inequalities (5.3) and (5.5) in the proof of Theorem 1. Hence, the generic form of Algorithm $\mathcal{A}$ can be implemented in a way that avoids having to deal with large numbers.

Line search. Suppose that instead of the quantity $\Delta_\Phi^k(\tau)$ defined in Step 1.2 of Algorithm $\mathcal{A}$, we use throughout the quantity

$$\Delta^k(\frac{\tau^k}{t}) = \ln \sum_{m=1}^{M} p_m \exp\left\{-\frac{\tau^k}{t}(f_m^k - \hat{f}_m^k)\right\},$$

which is an upper bound on $\Delta_\Phi^k(\tau)$ as stated in (5.3). (These two quantities coincide when $f^k(\cdot)$ are linear.) Theorems 1, 2 and 3 remain valid for this change because our entire analysis uses this approximation instead of $\Delta_\Phi^k(\tau)$. They also remain valid if the step length is selected independently for each block by performing a line search for $\Delta^k(\tau^k/t)$, instead of using the same constant step length for all blocks as in our previous analysis.

This may improve performance in practice, particularly when used with standard heuristics for coordinate descent methods.

K-search. Potentially larger improvements can be attained by replacing the K independent line searches for Algorithm $\mathcal{A}$ by a single K-dimensional approximate minimization of the function

$$\Delta(\frac{\tau^1}{t},\cdots,\frac{\tau^K}{t}) = \ln \sum_{m=1}^{M} p_m \exp\left\{ -\sum_{k=1}^{K} \frac{\tau^k}{t}(f_m^k - \hat{f}_m^k) \right\},$$

which majorizes the decrease

$$\Phi_t(x^1(1-\tau^1)+\tau^1\hat{x}^1,\cdots,x^K(1-\tau^K)+\tau^K\hat{x}^K) - \Phi_t(x^1,\cdots,x^K)$$

in the potential function (see (5.3).) In order to avoid large exponents, a normalization similar to the one described for p above can be used for this multidimensional search in the variables $\tau^1/t,\cdots,\tau^K/t$. In fact, the minimization of Δ is a simple geometric programming problem with box constraints on the variables, which can be solved efficiently in practice to a fixed accuracy of order $\sigma^2/\mu q$ for moderate values of K. This idea, albeit for a different kind of potential function, was proposed by Schultz and Meyer [17] and it was used to solve large multicommodity flow problems with $K = 11$. Multidimensional search can be performed for the randomized algorithm $\mathcal{R}$ as well, by randomly selecting a small subset of blocks $\mathcal{K} \subseteq \{1,\cdots,K\}$.

Relaxation of the threshold q. The bounds of Theorem 3 suggest that there is a certain amount of trade-off between the number of coordination steps and the relative feasibility tolerance q prescribed to the block solvers. The larger the value of q, the easier the task of the block solver but the larger the bound on the number of coordination steps. Perhaps the simplest example of such a trade-off is the device of box-relaxed block problems we used for separable constraint functions in Section 3. Another illustration of this trade-off is the concurrent flow problem with uniform capacities discussed in the Section 9.2 below.

Constraint relaxation for block solvers. Another way to improve the efficiency of block solvers that operate on the sets B_μ^k is to omit those constraints $f_m^k(x^k) \le \mu$ which either correspond to relatively small values p_m or remain inactive for a block problem over a number of coordination steps. It is possible to devise adaptive strategies which periodically reconstitute the set of constraints $f_m^k(x^k) \le \mu$ included in B_μ^k, similar to those described by Grigoriadis and White [6]. Such heuristic strategies were found to be very effective for linear multicommodity problems with weighted coupling constraints on arc flows [6].

9.2. Applications. We conclude with a few applications that illustrate the use the algorithms presented in this paper.

Concurrent flows with uniform capacities [14]. Consider a communication network having n-vertices and M-arcs representing channels of uniform capacity λ. There are $K \leq n$ commodities which will share this network. A commodity k is associated with a vertex v having a supply $d(v) > 0$ and possibly $n-1$ demands distributed over the rest of the vertices to be satisfied exactly. The problem is to assign the minimum capacity λ to each arc of the network so that all commodity flows can be accommodated. This is a linear problem $(\mathcal{P})$ with M resource constraints of coupling degree $S = 1$, and K blocks B^k each of which consists of flow conservation and nonnegativity constraints. The maximum vertex degree d in communication networks is typically small. For instance, $d = \log n$ in a hypercube.

The solution of this problem can be approached in many ways. On one hand, if $q = O(1)$, the solution to a relative accuracy ε can be obtained in expected $K \ln n(\varepsilon^{-2} + \ln K)$ uniformly capacitated minimum-cost flow computations, which can be reduced to assignment problems in $O(M)$ vertices and edges. The algorithm can be implemented to run in expected time $O(KM^{3/2} \ln n \ln(n\varepsilon^{-1})(\varepsilon^{-2} + \ln K))$, which is better than the bound given in [10], but coincides with the latter for a fixed ε. On the other hand, setting $q = d$ effectively removes the capacity constraints in $B^k_{\mu q}$. This simplifies the assignment problems mentioned above to single-source shortest-path problems. Thus, the problem can also be solved in expected $O(dK \ln n(\varepsilon^{-2} + \ln dK)(M + n \ln n))$ time. For a fixed ε and for K of order n, the ratio of the latter time to the former is $d/\sqrt{M}$, which shows that this trade-off is advantageous for small d. These two extreme cases can be combined to create data-dependent hybrid algorithms which adaptively select $q \in [1, d]$ and solve some block problems as assignment and some others as shortest-path problems. The solution of such block problems can be further parallelized. For example, for a fixed ε and up to logarithmic factors, the second of the above bounds can be reduced to $O(dK)$ parallel time.

Generalized concurrent flow problems with side constraints. We define this problem as follows. There are K commodities, each with given demands to be satisfied. There are also M coupling constraints bounding from above weighted sums of edge flows over sets of edges within the same commodity or across several commodities, provided that there are at most S positive entries in each partition of a coupling row corresponding to any

block. That is, there are at most S edges that enter in any one such generalized bounding constraint. With no loss of generality each arc in the network is assumed to have capacity λ. It is required to find a flow that minimizes λ. Our algorithm reduces this problem to expected $O(SK \ln M(\epsilon^{-2} + \ln SK))$ single-commodity minimum-cost flow computations. The case $S = 1$ includes the concurrent-flow problem considered in [10], but still results in a better bound. If in addition, the individual commodity blocks are generalized flow (gain-loss) networks, this is also the expected number of minimum-cost generalized network flow problems to be solved.

Generalized supply sharing transportation network. Consider the following resource sharing problem for a capacitated generalized (gain-loss) bipartite transportation network with m arcs and $n_1, n_2 \geq 2$ sources and sinks, respectively. There are given demands $b_j > 0$ for each sink j, and a source utilization factor $a_i > 0$, for each source i. It is required to meet the demands by shipping the least amount of total supply from the sources in the given proportions a_i. This is a linear programming problem $(\mathcal{P})$ with $S = 1$, $M = n_1$, and $K = n_2$, also known as the *machine loading* problem. By Theorem 3, the expected number of block problems to be solved is $O(n_2 \ln n_1(\epsilon^{-2} + \ln n_2))$. Each block problem is a capacitated minimum-cost star-tree problem of one sink j and d_j sources, where d_j is the degree of sink j. This block problem can be trivially solved in $O(d_j \ln d_j)$ sequential time, but it can also be solved in $O(d_j)$ time (Brucker [1]). Averaging this time over the sinks for the randomized variant of our algorithm, we obtain an amortized time of $O(d)$, where d is the average degree of a sink in the network. This sharing problem can thus be solved in expected $O(m \ln n_1(\epsilon^{-2} + \ln n_2))$ sequential time. If the block problems are solved on n_1 parallel processors, the resulting expected time is $O(\epsilon^{-2} n_2)$ up to logarithmic factors in n_1 and n_2. This problem is a generalization for the bipartite case of the minimax maximum-flow sharing problem described by Gallo, Grigoriadis and Tarjan [5].

Generalized quadratic transportation problems. The feasibility form of such problems is similar to the previous model. It calls for computing a flow that satisfies given supplies and demands to a given relative accuracy, and such that a given convex separable quadratic function of the arc flows (possibly including linear terms) does not exceed a given threshold. This is a nonlinear problem $(\mathcal{P})$ with $S = n_1 \geq 2$, the number of sources, $K = n_2 \geq 2$, the number of sinks, and $M = n_1 + 1$. Each block problem is now a capacitated quadratic minimum-cost star-tree problem, which again can be solved in $O(d_j)$ time using

a parametric method with median search [1]. Thus we obtain the same bounds as those in the previous paragraph multiplied by a factor of n_1. In particular, the problem can be solved on n_1 processors in $O(\varepsilon^{-2}n_1n_2)$ expected time, up to logarithmic factors in n_1 and n_2.

Objective and constraint functions of arbitrary sign. Suppose that the objective and constraint functions of the nonlinear optimization problem (1.2) of Section 1.1 are not necessarily nonnegative, but that an a priori bound $|f_m^k(x^k)| \leq V_m^k$, $x^k \in B^k$ is known for each block k and each $m = 0, 1, \cdots, M$, so that $|f_m(x)| \leq V_m = \sum_{k=1}^K V_m^k$ for all $x \in B$. Define an *ε-approximate solution* to (1.2) to be an arbitrary point $x \in B$ that satisfies $f_0(x) \leq f^* + V_0\varepsilon$, $f_m(x) \leq b_m + V_m\varepsilon$ for all m. By replacing $f_m^k(x^k)$ by $f_m^k(x^k) + 2V_m^k$, the problem of computing such a solution can be reduced to the original problem $(\mathcal{P}_{\varepsilon/3})$. Furthermore, letting $q = 3$ effectively removes the constraints $f_m^k(x^k) + 2V_m^k \leq \mu qe$ in the definition (3.1) of $B_{\mu q}^k$ since $\mu \geq V_m \geq V_m^k$.

Thus, the $(\varepsilon', 3)$-good block solver can operate on the original sets B^k to minimize $p^T f^k(x^k)$ to an *absolute* accuracy $\delta = \varepsilon' p^T V^k$ or $\delta = O(\varepsilon p^T V^k)$ for $\varepsilon' = O(\varepsilon)$ as we have previously done for the nonnegative case. Also, in this case it suffices to call the block solver to find an arbitrary point $x_0 \in B$ instead of using the greedy initialization procedure $\mathcal{GP}$ of Section 8. Since for the redefined positive functions $\lambda(x_0)/\lambda^* \leq 3$, the bound (7.5) on the number of coordination steps needed to solve $(\mathcal{P}_{\varepsilon/3})$ becomes $O(\varepsilon^{-2}K \ln M)$. For the nonlinear optimization problem (1.2) we obtain the bound $O(\varepsilon^{-2}K \ln M \ln \varepsilon^{-1})$. As before, each such step generates K block-solver calls for the original sets B^k in the deterministic method, but only two block-solver calls on the average for the randomized variant.

Deterministic equivalents of stochastic programs. The above reduction expands the range of applications of our analysis to many instances of the block-angular structure. Here, we mention the approximate solution of linear and nonlinear deterministic equivalents of two-stage large-scale stochastic programs with discrete finite distributions of the problem data (see Wets [19]; Nielsen and Zenios [11] have recently reported extensive computational results for a mortgage-backed securities investment problem of this type, where uncertainties are modeled by a large set of K "scenarios" based on forecasted interest-rate paths.) In general, such problems can be stated as follows:

$$F^* = \min \left\{ f(x) + \sum_{k=1}^{K} \pi^k g^k(y^k) \,|\, x \in A,\ (x, y^k) \in S^k \right\},$$

where x is the first-stage decision vector, A is a convex compact set in $\mathbb{R}^n$, y^k is the second-stage decision vector corresponding to the k-the scenario, and each scenario is given by its postulated probability π^k and a convex compact set S^k as its feasibility domain. The convex functions $f(x)$ and $g^k(y^k)$ are given first and second-stage objectives. The overall objective is the sum of the deterministic first-stage term plus the expectation of the stochastic objective terms (e.g. quadratic, entropy) over K scenarios. We assume that $A \subseteq [0,1]^n$, possibly after scaling x. We also assume that we are given an a priori bound V^k on the range of possible values of $f(x)+g^k(y^k)$ over $\{(x,y^k)\,|x \in A,\ (x,y^k) \in S^k\}$. Thus, V^k is an a priori bound on the possible "loss" in the k-th scenario. Let $V = \sum_{k=1}^K \pi^k V^k$. We shall say that $z = (x,y^1,\cdots,y^k)$ is an *ε-approximate solution* if $f(x)+\sum_{k=1}^K \pi^k g^k(y^k) \leq F^*+V\varepsilon$, $x \in A$, and each pair (x,y^k) is *ε-feasible* for the k-th scenario in the sense that there is a perturbation vector δ^k such that $\|\delta^k\|_\infty \leq \varepsilon$ and $(x+\delta^k,y^k) \in S^k$. In practical terms, the latter condition corresponds to requiring that the postulated scenarios are sufficiently robust with respect to the first-stage decision. Since our problem can be rewritten as:

$$F^* = \min\left\{\sum_{k=1}^K \pi^k(f(x^k)+g^k(y^k))\,|x^1 = x^k,\ \text{and } (x^k,y^k) \in B^k,\ k=1,\cdots,K\right\},$$

where $B^k = \{(x^k,y^k) \mid x^k \in A,\ (x^k,y^k) \in S^k\}$, it can be considered as a block-angular problem with blocks B^k and with $2(K-1)n$ linear coupling functions $\pm(x_j^1 - x_j^k)$.

The algorithm of the previous application computes K pairs $(x^k,y^k) \in B^k$ such that $\sum_{k=1}^K \pi^k(f(x^k)+g^k(y^k)) \leq F^* + V\varepsilon$ and $\|x^1 - x^k\|_\infty \leq \varepsilon$, $k = 1,\cdots,K$, in expected $O(\varepsilon^{-2}K\ln Kn\ln\varepsilon^{-1})$ iterations. At each iteration, the algorithm minimizes the sum of $f(x^k)+g(y^k)$ plus a linear form in x^k, over $(x^k,y^k) \in B^k$ for a randomly selected scenario k. Since f is convex, $z = (\sum_{k=1}^K \pi^k x^k, y^1,\cdots,y^K)$ provides an ε-approximate solution to our original problem.

Fast parallel approximation algorithm for matrix games. Let A be an $N \times M$ matrix game whose entries belong to an interval $[\alpha,\beta]$ and let v^* be the value of the game. Consider the problem $VAL(\varepsilon)$ of computing an approximation v to v^* to a prescribed relative accuracy $\varepsilon \in (0,1)$, that is $|v - v^*| \leq \varepsilon(\beta-\alpha)$, and of computing the corresponding pair of mixed ε-optimal strategies.

Assume without loss of generality that $[\alpha,\beta] = [1,2]$. Applying Theorem 3 for the case $K = 1$, $B = P$, and $q = 2$, along with some additional bounds on the required accuracy of arithmetic operations which are omitted here, one can prove the following result.

There is a parallel algorithm for $VAL(\varepsilon)$ whose execution uses NM processors and has depth of $O(\varepsilon^{-2} \ln N \cdot \ln M)$ arithmetic operations on $O(\ln(NM\varepsilon^{-1}))$-digit numbers.

10. Acknowledgments.

We thank Gennady Samorodnitsky for an earlier proof of Lemma 3 mentioned in Section 6, and Éva Tardos for providing us with the manuscript [10].

11. References.

[1] P. Brucker, "An $O(n)$ algorithm for quadratic knapsack problems", *Operations Research Letters* 3 (1984), pp. 163-166.

[2] G. B. Dantzig and P. Wolfe, "The decomposition algorithm for linear programming", *Econometrica*, 29 (1961), pp. 767-778.

[3] Hugh Everett III, "Generalized Lagrange resources", *Operations Research*, 11 (1963), pp. 399-417.

[4] L. Fratta, M. Gerla, and L. Kleinrock, "The flow-deviation method: an approach to store-and-forward communication network design", *Networks*, 3 (1973), pp. 97-133.

[5] G. Gallo, M. D. Grigoriadis and R. E. Tarjan, "A fast parametric maximum flow algorithm and applications", *SIAM Journal on Computing*, 18 (1989), pp. 30-55.

[6] M. D. Grigoriadis and W. W. White, "A partitioning algorithm for the multicommodity network flow problem", *Mathematical Programming*, 3 (1972), pp. 157-177.

[7] L. G. Khachiyan, "Convergence rate of the game processes for solving matrix games", *Zh. Vȳchisl. Mat. & Mat. Fiz.*, 17-6 (1977) 1421-1431. English translation in *U.S.S.R. Comput. Math. & Math. Phys.*, 17 (1978), pp. 78-88.

[8] P. Klein, C. Stein and E. Tardos, "Leighton-Rao might be practical: faster approximation algorithms for concurrent flow with uniform capacities", *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, IEEE (May 1990), pp. 310-321.

[9] L. S. Lasdon, *Optimization Theory for Large Systems*, The MacMillan Company, New York, NY, 1970.

[10] T. Leighton, F. Makedon, S. Plotkin, C. Stein, E. Tardos, and S. Tragoudas, "Fast approximation algorithms for multicommodity flow problems", (manuscript), 11p., November 1990. To appear in *Proceedings of the 31st Annual ACM Symposium on Theory of Computing* (1991).

[11] S. S. Nielsen and S. A. Zenios, "A massively parallel algorithm for nonlinear stochastic network problems", Report 90-09-08, The Wharton School, University of Pennsylvania, Phialdelphia, PA.

[12] J. B. Rosen, "Primal partition programming for block diagonal matrices", *Numerische Mathematik*, 6 (1964), pp. 250-260.

[13] J. Robinson, "An iterative method of solving a game", *Annals of Math.* 54 (1951), pp. 196-301.

[14] F. Shahrokhi and D. W. Matula, "The maximum concurrent flow problem", Technical Report 87-CSE-4, Department of Computer Science and Engineering, Southern Methodist University, Dallas, TX 75275, February 1987. Appeared in *Journal of the ACM*, 37 (1990) pp. 318-334.

[15] T. R. Rockafellar, *Convex Analysis*, Princeton University Press, Princeton, NJ, 1970.

[16] G.P. Samorodnitsky, *Personal communication* (1990).

[17] G.L. Schultz and R. R. Meyer, "A structured interior point method", Computer Sciences Technical Report 934, Computer Science Department, University of Wisconsin-Madison, Madison, WI, May 1990, 25p.

[18] E. Tardos, "Improved approximation algorithm for concurrent multicommodity flows", Technical Report No. 872, School of OR/IE, Cornell University, Ithaca, NY, October 1989, 9p.

[19] R. Wets, *Numerical Techniques for Stochastic Optimization*, Springer-Verlag, 1988.