Mathematical Programming 49 (1991) 397-411
North-Holland

On the solution of concave knapsack problems

Jorge J. Moré and Stephen A. Vavasis*

Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439, USA

Received 10 January 1989
Revised manuscript received 17 June 1989

We consider a version of the knapsack problem which gives rise to a separable concave minimization problem subject to bounds on the variables and one equality constraint. We characterize strict local miniimizers of concave minimization problems subject to linear constraints, and use this characterization to show that although the problem of determining a global minimizer of the concave knapsack problem is NP-hard, it is possible to determine a local minimizer of this problem with at most $O(n \log n)$ operations and $1+\lceil \log n \rceil$ evaluations of the function. If the function is quadratic this algorithm requires at most $O(n \log n)$ operations.

Key words: Concave functions, knapsack problems, strict minimizers, NP-hard, nonconvex, local minimizers.

1. Introduction

Certain combinatorial optimization problems can be posed as quadratic programming problems subject to special types of constraints. Consider, for example, the $0-1$ *knapsack* problem: Given n integers $a_1, a_2, \ldots, a_n$, and an integer γ, determine if there is a subset $J \subset \{1, \ldots, n\}$ such that

$$\sum_{j \in J} a_j = \gamma.$$

This problem, also known as the *subset sum* problem, was one of the first combinatorial problems to be proved NP-complete. For more information on this problem see, for example, Papadimitriou and Steiglitz [1982].

Given an instance of the knapsack problem, consider the optimization problem

$$\min\left\{ \sum_{i=1}^{n} x_i(1-x_i) : \mathbf{0} \le \boldsymbol{x} \le \boldsymbol{e}, \sum_{i=1}^{n} a_i x_i = \gamma \right\}, \tag{1.1}$$

Work supported in part by the Applied Mathematical Sciences subprogram of the Office of Energy Research of the U.S. Department of Energy under Contract W-31-109-Eng-38.

* Permanent address: Computer Science Department, Cornell University, Ithaca, New York 14853. Work supported in part by a Fannie and John Hertz Foundation graduate fellowship.

where $\boldsymbol{e}$ is the vector whose components are unity, and the relation $\boldsymbol{x} \leq \boldsymbol{y}$ between vectors is the standard componentwise relation. This is a quadratic programming problem subject to bounds on the variables and one equality constraint. Note that if the global minimium of (1.1) is zero, then the components of the corresponding minimizer satisfy $x_i \in \{0, 1\}$, and thus the minimizer is a solution of the knapsack problem. Conversely, a solution to the knapsack problem is a global minimizer of problem (1.1), and the global minimum is zero. This argument, due to Sahni [1974], shows that determining a global minimizer of problem (1.1) is an NP-hard problem.

Although global minimizers of problem (1.1) are relevant to the solution of knapsack problems, little is known about local minimizers of quadratic programming problems of the form (1.1). One of the aims of this paper is to present an O(n log n) algorithm for determining a local minimizer of quadratic programming problems of the form

$$\min\left\{\boldsymbol{x}^{\mathrm{T}}\boldsymbol{D}\boldsymbol{x}+\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}\colon \boldsymbol{l}\leq\boldsymbol{x}\leq\boldsymbol{u},\ \sum_{i=1}^{n} a_i x_i=\gamma\right\}, \tag{1.2}$$

where the vectors $\boldsymbol{l}$ and $\boldsymbol{u}$ specify bounds on the variables, and the matrix $\boldsymbol{D}$ is diagonal with negative diagonal entries. Note that problem (1.1) is a special case of (1.2) where $\boldsymbol{D}$ is the negative of the identity matrix, and that since the knapsack problem is reducible to a special case of problem (1.2), finding a global minimizer of problem (1.2) is NP-hard.

Local minimizers of quadratic programming problems of the form (1.1) do not seem to have a useful interpretation in terms of the combinatorial knapsack problem. Local minimizers provide upper bounds and can be good approximations to the solution of the global problem. They could also be useful in a branch-and-bound algorithm for the global problem, but these possibilities have not been explored.

At first sight determining a local minimizer of problem (1.1) should be possible with a standard quadratic programming algorithm. However, for the indefinite case, a standard quadratic programming algorithm is only guaranteed to find a stationary point in a finite number of operations. We also note that the results of Murty and Kabadi [1987] show that even for a relatively simple indefinite quadratic programming problem

$$\min\{\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x}+\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}\colon \boldsymbol{l}\leq\boldsymbol{x}\leq\boldsymbol{u}\}, \tag{1.3}$$

it is NP-hard to determine whether a feasible point $\boldsymbol{x}$ is a local minimizer; related results can be found in the paper of Pardalos and Schnitger [1988]. Our theorems implicitly give polynomial-time tests for a local minimizer of (1.2), and our algorithm finds a local minimizer in polynomial time.

Algorithms which guarantee a local minimizer of a quadratic programming problem in a polynomial number of operations are only known in a few cases; in all cases known to us, the quadratic is strictly convex. For example, Pang [1979] obtains a polynomial bound on the number of operations for problems of the form (1.3) provided the matrix $\boldsymbol{A}$ is positive definite with non-positive off-diagonal entries.

Strictly convex quadratic programming problems of the form (1.2) arise as subproblems in several optimization algorithms. See, for example, Cottle, Duvall, and Zikan [1986], Lin and Pang [1987], and Calamai and Moré [1987]. For the strictly convex case Helgason, Kennington, and Lall [1980] obtained an O($n \log n$) algorithm, while O(n) algorithms have been developed by Brucker [1984], and Calamai and Moré [1987]. The strictly concave case has not received attention, although as noted above, this case arises naturally in the modelling of certain discrete optimization problems. In this paper we attack the strictly concave case of problem (1.2) by developing an algorithm which determines a local minimizer of the problem

$$\min\left\{\sum_{i=1}^{n} q_i(x_i): \boldsymbol{l} \leq \boldsymbol{x} \leq \boldsymbol{u}, \sum_{i=1}^{n} a_i x_i = \gamma\right\}, \tag{1.4}$$

where the functions $q_i : R \to R$ are strictly concave differentiable functions.

Ibaraki and Katoh [1988] survey algorithms for problem (1.4) with q_i a strictly convex function. They also provide an extensive list of applications which require the solution of a problem of the form (1.4). Related work on problem (1.4) in the non-convex case includes the algorithm of Luss and Gupta [1975] for special classes of concave functions q_i, and the algorithm of Van Den Bosch and Lootsma [1987] for general functions q_i.

Problem (1.4) is a concave minimization problem. Algorithms for the determination of a global minimizer of this problem have received considerable attention. See, for example, Pardalos and Rosen [1986, 1987]. In a similar vein, we note that although much is known about global minimizers of concave minimization problem (see Section 32 of Rockafellar [1970]), little is known about the properties of local minimizers.

Section 2 contains some results on local minimizers of concave minimization problems. The main result is a characterization of the strict minimizers for the general problem

$$\min\{q(x): x \in \Omega\}$$

where Ω is polyhedral and $q : R^n \to R$ is a differentiable concave function on Ω. This result is unusual because in general it is not possible to characterize local minimizers. Also note that this result complements the analogous result for the convex case where $\boldsymbol{x}^*$ is a global minimizer if and only if $\nabla q(\boldsymbol{x}^*)^{\mathrm{T}}\boldsymbol{d} \geq 0$ for every feasible direction $\boldsymbol{d}$.

Sections 3 and 4 propose and study the CKP (Concave Knapsack Problem) algorithm for the solution of problem (1.4). The main result of Section 3 is that the CKP algorithm obtains a local minimizer in at most n iterations. In Section 4 we show that algorithm CKP can be implemented so that execution requires at most O(n^2) operations and $n+1$ evaluations of each derivative q_i'. We also show that algorithm CKP can be modified so that execution requires at most O($n \log n$) operations and $1 + \lceil \log n \rceil$ evaluations of each derivative q_i'. Hence, in the case of problem (1.2), the CKP algorithm determines a local minimizer in at most O($n \log n$) operations.

2. Characterization of strict minimizers for concave problems

The first step in the development of algorithms for problem (1.4) is to characterize minimizers of problem (1.4). We develop this characterization result by considering the problem

$$\min\{q(x): x \in \Omega\} \tag{2.1}$$

where Ω is polyhedral and $q : R^n \to R$ is a concave function on Ω. As we shall see, the development for general linearly constrained Ω does not offer any added difficulties.

A vector x^* in Ω is a minimizer of problem (2.1) if there is a neighborhood S of x^* such that $q(x^*) \leqslant q(x)$ for all x in $\Omega \cap S$. The minimizer is strict if $q(x^*) < q(x)$ for all $x \neq x^*$ in $\Omega \cap S$. The following result characterizes strict local minimizers in terms of feasible directions d at x^* in the sense that $x^* + \alpha d$ belongs to Ω for all $\alpha > 0$ sufficiently small.

Theorem 2.1. *Assume that Ω is polyhedral, that $q : R^n \to R$ is a concave function on Ω, and that q is differentiable at x^*. The vector x^* is a strict local minimizer of problem* (2.1) *if and only if the set*

$$\{d \in R^n: \nabla q(x^*)^{\mathrm{T}} d \leqslant 0,\ d \text{ a feasible direction},\ d \neq 0\} \tag{2.2}$$

is empty.

Proof. Assume that x^* is a strict minimizer of problem (2.1) and let $d \neq 0$ be a feasible direction at x^* such that $\nabla q(x^*)^{\mathrm{T}} d \leqslant 0$. Thus the concavity of q implies that

$$q(x^* + \alpha d) \leqslant q(x^*) + \alpha \nabla q(x^*)^{\mathrm{T}} d \leqslant q(x^*)$$

for all $\alpha > 0$ sufficiently small. However, this contradicts the assumption that x^* is a strict minimizer. Thus, the set (2.2) is empty.

Now assume that the set (2.2) is empty. Since Ω is polyhedral, the set of feasible directions is closed. Hence, a compactness argument shows that there is an $\varepsilon > 0$ such that

$$\nabla q(x^*)^{\mathrm{T}} d \geqslant \varepsilon \|d\|$$

for any feasible direction d. Since $d = x - x^*$ is a feasible direction if $x \in \Omega$, the differentiability of q at x^* implies that

$$|q(x) - q(x^*) - \nabla q(x^*)^{\mathrm{T}}(x - x^*)| \leqslant \tfrac{1}{2}\varepsilon \|x - x^*\|$$

for all x in a neighborhood of x^*. In particular, $q(x) > q(x^*)$ for $x \neq x^*$ and thus x^* is a strict local minimizer. □

The result that x^* is a strict local minimizer if the set (2.2) is empty is a consequence of the standard second order sufficiency conditions provided q is twice differentiable at x^*. The above proof shows that it is only necessary to assume that q is differentiable at x^*.

Theorem 2.1 is also related to results of Contesse [1980] and Mangasarian [1980] for quadratic programming problems. They show that if $q : R^n \to R$ is a quadratic and Ω is defined by linear constraints then the standard second order sufficiency conditions characterize strict minimizers of the quadratic programming problem. If $q : R^n \to R$ is a concave quadratic these results of Contesse and Mangasarian follow from Theorem 2.1.

The following result shows that a search for strict minimizers of a concave function only needs to examine the extreme points of Ω. Note that in this result Ω is a general closed convex set.

Theorem 2.2. *If Ω is a closed convex set and $q : R^n \to R$ is a concave function on Ω then any strict minimizer must be an extreme point of Ω.*

Proof. Let x^* be a strict minimizer of q and note that if x^* is not an extreme point of Ω then $x^* = \frac{1}{2}(x_1 + x_2)$ where $x_i \in \Omega$ and $x_i \neq x^*$ for $i = 1, 2$. We can also assume that x_1 and x_2 are arbitrarily close to x^*, and thus, that $q(x^*) < q(x_i)$ for $i = 1, 2$. Hence, the concavity of q implies that

$$q(x^*) \geq \tfrac{1}{2}q(x_1) + \tfrac{1}{2}q(x_2) > q(x^*).$$

This contradiction shows that x^* must be an extreme point. □

Although a strict minimizer must be an extreme point of Ω this does not hold for minimizers. For example, let Ω be the set of x such that $\|x\| \leq 1$ and define q by setting $q(x) = -\frac{1}{2}$ for $\|x\| \leq \frac{1}{2}$ and $q(x) = -\|x\|$ otherwise. In this case all vector x with $\|x\| < \frac{1}{2}$ are minimizers of problem (2.1), but they are not extreme points of Ω. On the other hand, it is known that if Ω is a closed convex set which does not contain lines and q attains the global minimum on Ω, then the global minimum is achieved at an extreme point of Ω. For a proof of this result and more information on global minimizers of concave functions, see Section 32 of Rockafellar [1970].

We now show, in particular, that if q is strictly concave on Ω then any minimizer must be an extreme point of Ω.

Theorem 2.3. *If $q : R^n \to R$ is a strictly concave function on a polyhedral Ω then any minimizer x^* is a strict minimizer of q.*

Proof. Since Ω is polyhedral, there is an $\varepsilon > 0$ such that if $x \in \Omega$ and $\|x - x^*\| < \varepsilon$, then

$$y = \frac{1}{\alpha}x + \left(1 - \frac{1}{\alpha}\right)x^* \in \Omega$$

for all α sufficiently close to 1. This claim is established by noting that Ω is the intersection of a finite number of half-spaces and choosing ε so that both x^* and x lie in the relative interior of the same half-spaces.

Given any minimizer x^* of q we can also choose $\varepsilon > 0$ such that if $x \in \Omega$ and $\|x - x^*\| < \varepsilon$ then $q(x^*) \leq q(x)$. Hence, in view of the above claim, given $x \in \Omega$ with $\|x - x^*\| < \varepsilon$, there is a $y \in \Omega$ with $\|y - x^*\| < \varepsilon$ such that

$$x = \alpha y + (1 - \alpha) x^*, \quad \alpha \in (0, 1).$$

If $x \neq x^*$ then $y \neq x^*$, and thus the strict concavity of q implies that

$$q(x) > \alpha q(y) + (1 - \alpha) q(x^*) \geq q(x^*).$$

This shows that x^* is a strict minimizer of q. □

3. Algorithms for the concave knapsack problem

Theorems 2.1 and 2.3 characterize local minimizers of problem (1.4) when the functions $q_i : R \to R$ are strictly concave differentiable functions. In this section we use these results to develop a finite algorithm which finds a strict minimizer of (1.4).

We assume that $a_i \neq 0$ for all i, because if $a_i = 0$ then x_i is decoupled from the rest of the problem, and a minimizer may be found for that variable independently. Since q_i is concave, a global minimizer occurs at either l_i or u_i.

Since $a_i \neq 0$, we can scale each variable x_i by $1/a_i$ to simplify the equality constraint. Thus, problem (1.4) is equivalent to the problem

$$\min\left\{ \sum_{i=1}^{n} q_i(x_i) : l \leq x \leq u, \sum_{i=1}^{n} x_i = \gamma \right\}, \tag{3.1}$$

where the functions $q_i : R \to R$ are strictly concave differentiable functions. Recall that a differentiable function q_i is strictly concave if and only if q_i' is strictly decreasing.

In problem (3.1) we may assume that $l_i < u_i$ for all i because of $l_i = u_i$ then x_i is uniquely determined and may be replaced by a constant. We can also assume that

$$\sum_{i=1}^{n} l_i < \gamma < \sum_{i=1}^{n} u_i, \tag{3.2}$$

because if γ does not satisfy this constraint then either problem (3.1) is infeasible or there is only one feasible point.

Theorems 2.2 and 2.3 show that any minimizer of problem (1.4) must be an extreme point. Hence, x_i^* lies in the open interval (l_i, u_i) for at most one index i. Assumption (3.2) shows that we must have $x_i^* \neq l_i$ for some index i. This implies that there is an index set F and an index $m \notin F$ such that $x_i^* = l_i$ for $i \in F$ and $x_i^* = u_i$ for $i \notin F \cup \{m\}$. Moreover, $x_m^* > l_m$. In the remainder of this section we show that the following algorithm determines an appropriate set F.

Algorithm CKP. Initialize F by setting $F = \emptyset$, and update F according to the following steps until the halting condition is satisfied.

1. For each $i \notin F$ define r_i by

$$r_i = \gamma - \sum_{j \in F} l_j - \sum_{j \notin F} u_j + u_i,$$

and let G be the set of indices

$$G := \{i \notin F: r_i > l_i\}.$$

2. Halt the algorithm if $G \neq \emptyset$ and

$$w < q_i'(l_i), \quad i \in F \quad \text{and} \quad w \geq q_i'(l_i), \quad i \notin (F \cup G),$$

where

$$w := \max\{q_i'(r_i): i \in G\}.$$

3. Update F by setting $F^+ := F \cup \{k\}$ where $q_k'(l_k)$ is the largest element of

$$V := \{q_i'(l_i): i \notin F, r_i \leq l_i\}.$$

We can provide some motivation for the strategy used to update F in Algorithm CKP. We claim that if we defined a set F^* by letting $x_i^* = l_i$ for $i \in F^*$ and $x_m^* = r_m > l_m$, then

$$q_m'(r_m) < q_i'(l_i), \quad i \in F^*. \tag{3.3}$$

This follows from Theorem 2.1 because it guarantees that $\nabla q(\boldsymbol{x}^*)^{\mathrm{T}} \boldsymbol{d} > 0$ for any non-zero feasible direction $\boldsymbol{d}$; we only need to choose $\boldsymbol{d}$ as the vector whose non-zero components are $d_m = -1$ and $d_i = 1$. Now note that inequality (3.3) suggests that the set F should be updated by adding the index i such that $q_i'(l_i)$ is largest. We only consider those indices with $r_i \leq l_i$ because if $r_i > l_i$ then r_i is a possible choice for r_m in (3.3). This is the strategy followed by Algorithm CKP.

The analysis of Algorithm CKP is mainly concerned with the behavior of the sets F and G. Note that the updating of F in the third step of Algorithm CKP is defined provided the sets F and G do not satisfy

$$F \cup G = \{1, 2, \ldots, n\}, \tag{3.4}$$

and that since the size of the set F increases by one at each step of Algorithm CKP, there are at most n steps. One of our first tasks is to prove that the halting condition of Algorithm CKP is satisfied in at most n steps.

In our analysis of Algorithm CKP we do not make explicit the dependence of the algorithm on the iteration. This should not cause any confusion because the discussion always centers around a given iteration. References to the previous or the next iteration are indicated by superscripts. For example, the notation r_i^+ refers to r_i for the new set F^+, and r_i^- denotes r_i for the previous set F^-.

An important observation is that the indices examined during the third step of Algorithm CKP are precisely those indices $i \notin (F \cup G)$. An index k is chosen during this step and added to F; other indices $i \notin (F \cup G)$ either enter G or remain outside $F \cup G$. We will need the following result to prove that if an index enters G then it remains in the updated G, that is, $G \subset G^+$.

Lemma 3.1. (1) *If $i \notin F$ then $r_i < r_i^+$ and $r_i \leq u_i$.*

(2) *If $r_i = u_i$ for some $i \notin F$ then $r_j = u_j$ for all $j \notin F$.*

Proof. Note that $r_i^+ - r_i = u_k - l_k$ where k is the index such that $F^+ = F \cup \{k\}$. Since $u_k > l_k$ we obtain that $r_i^+ > r_i$. The proof that $r_i \leq u_i$ is similar. Since $r_i^+ - r_k = u_i - l_k$, and since $r_k \leq l_k$ by the choice of k, we obtain that $r_i^+ \leq u_i$. Moreover, if $F = \emptyset$ then the definition of r_i implies that

$$r_i = \gamma - \sum_{j=1}^{n} u_j + u_i \leq u_i.$$

This establishes the first claim of this result; the second claim is established by noting that $r_i - r_j = u_i - u_j$ for all i and j not in F. Hence, if $r_i = u_i$ for some $i \notin F$ then $r_j = u_j$ for all $j \notin F$. □

Lemma 3.1 provides some of the basic properties of the set F. The following result contains the basic properties of the set G; in particular, that $G \subset G^+$.

Lemma 3.2. (1) *$G \subset G^+$.*

(2) *If (3.4) holds then $G \neq \emptyset$.*

Proof. We first prove that $G \subset G^+$. If $i \in G$ then $i \notin F$ and $r_i > l_i$. Hence, $i \neq k$ where $F^+ = F \cup \{k\}$, and thus $i \notin F^+$. Moreover, Lemma 3.1 shows that $r_i^+ > l_i$, and thus $i \in G^+$.

We now show that if (3.4) holds then $G \neq \emptyset$. Consider F^- and let k be such that $F = F^- \cup \{k\}$. Then

$$r_k^- = \gamma - \sum_{j=1}^{n} l_j + l_k > l_k$$

because of assumption (3.2) on γ. Thus $k \in G^-$, and since in the first part of this result we established that $G^- \subset G$, this implies that $k \in G$. In particular, $G \neq \emptyset$. □

Lemma 3.2 shows that the set G becomes non-empty at some iteration k_0, and that it remains non-empty for all iterations $k \geq k_0$. The proof that the other two requirements of the halting condition are satisfied in at most n steps requires the results formulated in the following three lemmas.

Lemma 3.3. *If $j \notin (F \cup G)$ then $q_j'(l_j) \leq q_i'(l_i)$ for $i \in F^+$.*

Proof. This result is established by noting that the assumption that $j \notin (F \cup G)$ implies that the index j was available for selection at the third step during all previous iterations. Since the index j was not selected, we must have $q_j'(l_j) \leq q_i'(l_i)$ for $i \in F$. Moreover, the choice of k on the third step guarantees that $q_j'(l_j) \leq q_k'(l_k)$, and thus the result holds. □

The results that we have established so far do not depend on the concavity of the functions q_i. All other results in this section, however, depend on this assumption.

The proof that the halting condition is eventually satisfied requires two preliminary results. We first show that when the set G becomes non-empty, the first part of the halting condition is satisfied.

Lemma 3.4. *If G^- is empty but $G \neq \emptyset$ then $w < q_i'(l_i)$ for $i \in F$.*

Proof. If $m \in G$ satisfies $w = q_m'(r_m)$ then $r_m > l_m$, and since q_m' is strictly decreasing,

$$w = q_m'(r_m) < q_m'(l_m).$$

Note that $m \notin (F^- \cup G^-)$ because G^- is empty and $m \notin F$. Hence, Lemma 3.3 shows that

$$q_m'(l_m) \leq q_i'(l_i), \quad i \in F.$$

The last two inequalities yield the result. □

We next show that if the first part of the halting condition is satisfied on a given iteration, then either the halting condition is satisfied or the first part of the halting condition is again satisfied on the next iteration.

Lemma 3.5. *If $G \neq \emptyset$ and $w < q_i'(l_i)$ for $i \in F$ then either*
(a) *$w^+ < q_i'(l_i)$ for $i \in F^+$, or*
(b) *the halting condition of Algorithm* CKP *is satisfied.*

Proof. Let $w^+ = q_m'(r_m^+)$ for some $m \in G^+$. We now show that if $m \notin (F \cup G)$ then $w^+ < q_i'(l_i)$ for $i \in F^+$. First note that $r_m^+ > l_m$ because $m \in G^+$, and thus the strict concavity of q_m implies that

$$w^+ = q_m'(r_m^+) < q_m'(l_m).$$

Moreover, since $m \notin (F \cup G)$, Lemma 3.3 shows that

$$q_m'(l_m) \leq q_i'(l_i), \quad i \in F^+.$$

Hence, the last two inequalities prove that $w^+ < q_i'(l_i)$ for $i \in F^+$ as desired.

Now assume that $m \in (F \cup G)$ and that the algorithm does not halt. Since $m \in G^+$, we must have $m \notin F$, and thus the assumption that $m \in (F \cup G)$ implies that $m \in G$. Thus, $q_m'(r_m) \leq w$ by the definition of w, and $r_m^+ > r_m$ by Lemma 3.1. Hence, the concavity of q_m implies that

$$w^+ = q_m'(r_m^+) \leq q_m'(r_m) \leq w.$$

The assumption that the algorithm does not halt guarantees that $w < q_j'(l_j)$ for some $j \notin (F \cup G)$, and thus Lemma 3.3 shows that

$$w < q_j'(l_j) \leq q_i'(l_i), \quad i \in F^+.$$

The last two inequalities prove that $w^+ < q_i'(l_i)$ for $i \in F^+$ as desired. □

Lemmas 3.2 and 3.4 show that there is an iteration k_0 such that G is non-empty and $w < q_i'(l_i)$ for $i \in F$. Lemma 3.5 shows that if the halting condition is not satisfied on any iteration $k \geq k_0$ then $w < q_i'(l_i)$ for $i \in F$ on all iterations $k \geq k_0$. The halting condition must hold in at most n steps because (3.4) holds in at most n steps, and then $w \geq q_i'(l_i)$ for $i \notin (F \cup G)$ holds trivially. Thus argument yields the following result.

Theorem 3.6. *The halting condition of Algorithm* CKP *is satisfied in at most n steps.* □

If the halting condition of Algorithm CKP is satisfied for a given set F and m is the index chosen so that $w = q_m'(r_m)$, then we show that the vector x^* defined by setting

$$x_i^* = \begin{cases} l_i, & i \in F, \\ r_m, & i = m, \\ u_i, & i \notin F \cup \{m\}, \end{cases} \tag{3.5}$$

is a strict minimizer of (3.1). It is important to note that at this point that this result does not depend on the fact that m is the first index at which the halting condition is satisfied. In other words, the following results show that x^* is a local minimizer of (3.1) even if we forget to check the halting condition on some iterations. This is necessary because the $O(n \log n)$ implementation of the algorithm is not guaranteed to find the first index m for which the halting condition is satisfied.

Theorem 3.7. *If the halting condition of Algorithm* CKP *is satisfied for a given set F and m is the index chosen so that $w = q_m'(r_m)$, then the vector x^* defined by* (3.5) *is a strict minimizer of problem* (3.1).

Proof. We first show that x^* satisfies all the constraints. Lemma 3.1 shows that $r_m \leq u_m$, and the choice of r_m guarantees that $r_m > l_m$. Hence, x^* satisfies the inequality constraints. Moreover, the definition of r_i shows that x^* also satisfies the equality constraint. We next show that

$$q_i'(l_i) \geq q_m'(r_m), \quad i \in F, \tag{3.6}$$

$$q_i'(u_i) \leq q_m'(r_m), \quad i \notin F. \tag{3.7}$$

If $i \in F$ then the halting condition shows that (3.6) is strictly satisfied. For (3.7) we have two cases: either $i \notin (F \cup G)$ or $i \in G$. If $i \notin (F \cup G)$ then the halting condition implies that $q_m'(r_m) \geq q_i'(l_i)$. In addition, $q_i'(l_i) > q_i'(u_i)$ by strict concavity. Thus, (3.7) is satisfied strictly in this case. If $i \in G$ then the definition of w implies that $q_m'(r_m) \geq q_i'(r_i)$. Moreover, $q_i'(r_i) \geq q_i'(u_i)$ because Lemma 3.1 guarantees that $r_i \leq u_i$. Thus, (3.7) is also satisfied strictly unless $r_j = u_j$ for some $j \in G$.

Now consider a feasible direction $\boldsymbol{d}$ at $\boldsymbol{x}^*$ such that $\nabla q(\boldsymbol{x}^*)^{\mathrm{T}}\boldsymbol{d} \leq 0$ where $q : R^n \to R$ is the function defined by

$$q(\boldsymbol{x}) = \sum_{i=1}^{n} q_i(x_i).$$

We now show that $\boldsymbol{d} = \boldsymbol{0}$ and thus Theorem 2.1 yields that $\boldsymbol{x}^*$ is a strict minimizer.

Since $\boldsymbol{d}$ is a feasible direction at $\boldsymbol{x}^*$, we have that $d_j \geq 0$ for $j \in F$, that $d_j \leq 0$ if $j \notin F \cup \{m\}$, and that $\boldsymbol{d}^{\mathrm{T}}\boldsymbol{e} = 0$. Hence, (3.6) and (3.7) imply that

$$\begin{aligned} 0 &\geq \nabla q(\boldsymbol{x}^*)^{\mathrm{T}}\boldsymbol{d} \\ &= \sum_{j \in F} q_j'(l_j) d_j + q_m'(r_m) d_m + \sum_{j \notin F \cup \{m\}} q_j'(u_j) d_j \\ &\geq q_m'(r_m) \sum_{j=1}^{n} d_j = 0. \end{aligned}$$

In particular, since we have shown that (3.6) holds strictly, $d_j = 0$ for $j \in F$. If (3.7) also holds strictly then $d_j = 0$ for $j \notin F$, and thus $\boldsymbol{d} = \boldsymbol{0}$ as desired. On the other hand, if (3.7) fails to hold strictly then have shown above that $r_j = u_j$ for some $j \notin F$. However, in this case Lemma 3.1 implies that $r_m = u_m$. The definition of $\boldsymbol{d}$ as a feasible direction at $\boldsymbol{x}^*$ then shows that $d_m \leq 0$, and thus $d_j \leq 0$ for $j \notin F$. We have now established that $d_j = 0$ for $j \in F$ and that $d_j \leq 0$ for $j \notin F$. Since $\boldsymbol{d}^{\mathrm{T}}\boldsymbol{e} = 0$, we must have $\boldsymbol{d} = \boldsymbol{0}$ as desired. □

4. Implementation

In this section we first propose an implementation of Algorithm CKP which requires at most $\mathrm{O}(n^2)$ operations and $n+1$ evaluations of each derivative q_i'. A more sophisticated algorithm requires at most $\mathrm{O}(n \log n)$ operations and $1 + \lceil \log n \rceil$ evaluations of each derivative q_i'. If q_i' can be evaluated at any x in $[l_i, u_i]$ with a bounded number (independent of n) arithmetic operations, then the execution times of these algorithms are $\mathrm{O}(n^2)$ and $\mathrm{O}(n \log n)$, respectively. This assumption on the evaluation time of q_i holds if q_i is a quadratic, and is a reasonable assumption for general q_i.

Each step of Algorithm CKP can be implemented in $\mathrm{O}(n)$ operations provided the r_i are updated at each step. This can be done by noting that if

$$\sigma_l = \sum_{j \in F} l_j, \qquad \sigma_u = \sum_{j \notin F} u_j, \tag{4.1}$$

then $r_i = \gamma - \sigma_l - \sigma_u + u_i$. If F is updated by adding the index k, then σ_l and σ_u can be updated by adding l_k to σ_l, and subtracting u_k from σ_u. Since each step requires $\mathrm{O}(n)$ operations, Algorithm CKP can be implemented with $\mathrm{O}(n^2)$ operations. This establishes the following result.

Theorem 4.1. *Algorithm* CKP *determines a local minimizer of problem* (3.1) *with at most* $O(n^2)$ *operations and* $n+1$ *evaluations of each derivative* q_i'. □

An algorithm for solving problem (3.1) in $O(n \log n)$ operations is obtained by first generating the same sequence of sets F as in algorithm CKP, and then doing a binary search in this sequence of sets to find a set F which satisfies the halting conditions. We elaborate on these remarks in the remainder of this section.

Consider algorithm CKP, but assume that the halting conditions are not checked. In this case algorithm CKP generates a sequence $F_1, F_2, \ldots, F_t$ with $F_1 = \emptyset$, and terminates when

$$F_t \cup G_t = \{1, 2, \ldots, n\}.$$

We can generate this sequence in $O(n \log n)$ operations by first noting that it is not necessary to compute all the r_i for $i \notin F$; we only need to compute those r_i which contribute to the computation of the largest element of V. Another important observation is that the search for the largest element of V is simplified if the set

$$W := \{q_i'(l_i) \colon i \notin F\}$$

is sorted in decreasing order. In this case it is only necessary to search W until we find an element with $r_k \leq l_k$. These two observations show that the following algorithm generates the same sequence of sets F as Algorithm CKP.

Algorithm FGEN. Initialize F by setting $F = \emptyset$. Sort $W = \{q_i'(l_i) \colon 1 \leq i \leq n\}$ in decreasing order. Initialize σ_l and σ_u.

```
For $i = 1, \ldots, n$:
1. Let $k_i$ be the index of the $i$-th largest element of $W$.
2. Compute $r_{k_i}$ by setting $r_{k_i} = \gamma - \sigma_l - \sigma_u + u_{k_i}$.
3. If $r_{k_i} \leq l_{k_i}$ set $F^+ := F \cup \{k_i\}$ and update $\sigma_l$ and $\sigma_u$.
```

As noted above, Algorithm FGEN generates the same sequence $F_1, F_2, \ldots, F_t$ of sets as Algorithm CKP provided the halting conditions are not checked. Algorithm FGEN requires $O(n \log n)$ operations to sort the array W initially. The body of Algorithm FGEN only requires $O(n)$ operations.

We now use a binary search on the sequence $F_1, F_2, \ldots, F_t$ of sets generated by Algorithm FGEN to find a set which satisfies the halting conditions. Note that if the halting conditions are not satisfied with F_t then

$$w \geq q_i'(l_i) \quad \text{for some } i \in F_t. \tag{4.2}$$

Assume that we have an integer s with $s \leq t$ such that G_s is not empty and

$$w < q_i'(l_i), \quad i \in F_s. \tag{4.3}$$

The following result shows that an integer s which satisfies $G_s \neq \emptyset$ and (4.3) can be obtained from Algorithm FGEN.

Lemma 4.2. *If k_s is the first index generated by Algorithm* FGEN *such that $r_{k_s} > l_{k_s}$ then $G_s \neq \emptyset$ and* (4.3) *holds.*

Proof. Since k_s is the first index generated by Algorithm FGEN such that $r_{k_s} > l_{k_s}$,

$$F_s = \{k_1, \ldots, k_{s-1}\},$$

and thus $k_s \in G_s$. Now let $k_m \in G_s$ satisfy $w = q'_{k_m}(r_{k_m})$. Then $m \geqslant s$ because $k_m \notin F_s$, and since $\{q'_{k_i}(l_{k_i})\}$ is sorted in decreasing order,

$$q'_{k_m}(l_{k_m}) \leqslant q'_i(l_i), \quad i \in F_s.$$

Now note that since $r_{k_m} > l_{k_m}$, the strict concavity of q'_{k_m} implies that

$$q'_{k_m}(r_{k_m}) < q'_{k_m}(l_{k_m}).$$

The last two inequalities show that (4.3) holds. □

We have shown how to obtain integers s and t which satisfy (4.2) and (4.3) with $G_s \neq \emptyset$. Given an integer k between s and t, Lemma 3.2 shows that $G_k \neq \emptyset$. The following algorithm updates s or t so that (4.2) and (4.3) hold.

Algorithm CKP*. Use Algorithm FGEN to determine integers s and t which satisfy (4.2) and (4.3) with $G_s \neq \emptyset$. If neither s nor t satisfy the halting conditions, update s and t according to the following steps until the halting condition is satisfied.

```
1. Let $k = \lfloor \frac{1}{2}(s+t) \rfloor$ and determine $F_k$.
2. Test for the halting condition at $F_k$.
3. If $w < q'_i(l_i)$ for $i \in F_k$ set $s = k$; otherwise $t = k$.
```

We claim that Algorithm CKP* requires at most $O(n \log n)$ operations to determine a set F_k which satisfies the halting conditions. The halting conditions are satisfied by Algorithm CKP* when $t = s+1$ because Lemma 3.5 shows that if (4.3) is satisfied, then either the halting conditions hold with F_s, or (4.2) does not hold for $t = s+1$. Since Algorithm CKP* needs at most $\lceil \log n \rceil$ steps to update s and t so that $t = s+1$, and since checking the halting conditions can be done with $O(n)$ operations, Algorithm CKP* requires at most $O(n \log n)$ operations.

Theorem 4.3. *Algorithm* CKP* *determines a local minimizer of problem* (3.1) *with at most* $O(n \log n)$ *operations and* $1 + \lceil \log n \rceil$ *evaluations of each derivative* q'_i. □

5. Numerical results

In this section we present numerical results obtained while testing the $O(n^2)$ and $O(n \log n)$ algorithms on concave knapsack problems of the form

$$\min\left\{ \sum_{i=1}^{n} x_i(a_i - x_i) \colon 0 \leqslant x_i \leqslant a_i,\ \sum_{i=1}^{n} x_i = \gamma \right\}. \tag{5.1}$$

The numerical results were done in double precision (about 16 decimal places) on the Alliant FX/8 at the Advanced Computing Research Facility of Argonne National Laboratory.

For each value of n we generated the data values a_i in problem (5.1) by a uniform distribution on the interval (0, 1); the constant γ was chosen so that

$$\sum_{j\in J} a_j = \gamma$$

for some index set J of size near $\frac{1}{2}n$. These choices lead to a problem (5.1) with a zero optimal value. Since we were interested in the ability of the algorithms to determine local minimizers with low function values, for each value of n we generated 30 problems, and for each problem we computed the value of the quadratic

$$q(x) = \sum_{i=1}^{n} x_i(a_i - x_i)$$

at the local minimizer x^*. In Tables 1 and 2 the quantities 'q_{val}' and '$q_{\max}$' represent, respectively, the average and the maximum values of $q(x^*)$ over these 30 problems, while 'time' represents the average execution time in seconds.

Table 1 presents the numerical results for the $O(n^2)$ algorithm. The execution time of this algorithm is not proportional to n^2 because for these problems the number of iterations required for convergence decreases with increasing values of n. The decline in execution time, however, is not sufficiently large to make the $O(n^2)$ algorithm acceptable for large values of n.

Also note that the values of q_{val} and $q_{\max}$ decrease with increasing values of n. This behavior was not expected. It is reassuring that the behavior of the algorithm does not deteriorate with large values of n.

Table 1
Numerical results for $O(n^2)$ algorithm

n	q_{val}	$q_{\max}$	time
10	0.572D−02	0.395D−01	0.278E−02
100	0.665D−03	0.454D−02	0.361E−01
1000	0.923D−04	0.512D−03	0.224E+01

Table 2
Numerical results for $O(n \log n)$ algorithm

n	q_{val}	$q_{\max}$	time
10	0.572D−02	0.395D−01	0.222E−02
100	0.665D−03	0.454D−02	0.194E−01
1000	0.923D−04	0.512D−03	0.202E+00
10 000	0.863D−05	0.606D−04	0.224E+01

Table 2 presents the numerical results for the O(n log n) algorithm. Although we expected the execution time to be proportional to n log n, these results show that the execution time is essentially linear. Also note that the values of q_{val} and $q_{\max}$ are identical in both tables. This is a consequence of the observation that for these problems the values of s and t obtained from algorithm FGEN are near; on most problems $|s-t| \leq 3$.

Acknowledgement

We are grateful to Richard Cottle for his help with previous work in this area.

References

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

P.H. Calamai and J.J. Moré, "Quasi-Newton updates with bounds," *SIAM Journal on Numerical Analysis* 24 (1987) 1434-1441.

L. Contesse, "Une caractérisation complète des minima locaux en programmation quadratique," *Numerische Mathematik* 34 (1980) 315-332.

R.W. Cottle, S.G. Duvall and K. Zikan, "A Lagrangean relaxation algorithm for the constrained matrix problem," *Naval Research Logistic Quarterly* 33 (1986) 55-76.

T. Ibaraki and N. Katoh, *Resource Allocation Problems* (The MIT Press, Cambridge, MA, 1988).

R. Helgason, J. Kennington and H. Lall, "A polynomially bounded algorithm for a singly constrained quadratic program," *Mathematical Programming* 18 (1980) 338-343.

Y.Y. Lin and J.S. Pang, "Iterative methods for large convex quadratic programs: A survey," *SIAM Journal on Control and Optimization* 25 (1987) 383-411.

H. Luss and S.K. Gupta, "Allocation of effort resources among competing activities," *Operations Research* 23 (1975) 360-366.

O.L. Mangasarian, "Locally unique solutions of quadratic programs, linear and nonlinear complementarity problems," *Mathematical Programming* 19 (1980) 200-212.

K.G. Murty and S.N. Kabadi, "Some *NP*-complete problems in quadratic and nonlinear programming," *Mathematical Programming* 39 (1987) 117-129.

J.S. Pang, "On a class of least-element complementarity problems," *Mathematical Programming* 16 (1979) 111-126.

C.H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization*, (Prentice-Hall, Englewood Cliffs, NJ, 1982).

P.M. Pardalos and J.B. Rosen, "Methods for global concave minimization: A bibliographic survey," *SIAM Review* 28 (1986) 367-379.

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

P.M. Pardalos and G. Schnitger, "Checking local optimality in constrained quadratic programming is *NP*-hard," *Operations Research Letters* 7 (1988) 33-35.

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

S. Sahni, "Computationally related problems," *SIAM Journal on Computing* 3 (1974) 262-279.

P.P.J. Van Den Bosch and F.A. Lootsma, "Scheduling of power generation via large-scale nonlinear optimization," *Journal on Optimization Theory and Applications* 55 (1987) 313-326.