Mathematical Programming 18 (1980) 338–343.
North-Holland Publishing Company

A POLYNOMIALLY BOUNDED ALGORITHM FOR A SINGLY CONSTRAINED QUADRATIC PROGRAM

R. HELGASON, J. KENNINGTON and H. LALL

Southern Methodist University, Dallas, TX U.S.A.

Received 23 March 1978
Revised manuscript received 13 June 1979

This paper presents a characterization of the solutions of a singly constrained quadratic program. This characterization is then used in the development of a polynomially bounded algorithm for this class of problems.

Key words: Nonlinear Programming, Convex Programming, Quadratic Programming, Singly Constrained Quadratic Program.

1. Introduction

Consider the following quadratic program,

$$\min \quad \tfrac{1}{2}x'Dx - a'x, \tag{1}$$

$$\text{s.t.} \quad \sum_j x_j = c, \tag{2}$$

$$0 \le x \le b \tag{3}$$

where D is a positive diagonal matrix and b is a nonnegative vector. Quadratic programs of this type arise when one applies a resource-directive decomposition procedure using subgradient optimization to solve multicommodity network flow problems (see [1, 4, 5]). The basic approach is to distribute the arc capacity among the commodities and then solve a set of single commodity problems. This solution is feasible for the multicommodity problem. If in addition this solution is within ϵ% (where the user selects ϵ) of a lower bound, then the procedure terminates. Otherwise, a subgradient is used to determine a new proposed allocation. Generally the proposed allocation is not feasible (i.e. exceeds the original arc capacity). When this occurs, one must project this proposed allocation back onto the set of feasible allocations. This projection involves solving a quadratic program of the form (1)–(3) for each arc whose proposed allocation exceeds the arc capacity. Since the subgradient procedure has been found to be at least twice as fast as competing algorithms for this class of problems (see [1]) there is great motivation for developing a fast algorithm for (1)–(3).

Held et al. [4] present an algorithm for solving (1)–(3) when D is an identity matrix and (3) is replaced by $x \geq 0$. The objectives of this exposition are (i) to extend the theory of [4] to the more general program (1)–(3) and (ii) to present a polynomially bounded algorithm for (1)–(3) based on this theory.

Related work involving the minimization of separable convex functions subject to (2) and (3) may be found in papers by Charnes and Cooper [3], Srikantan [10], Sanathanan [9], Luss and Gupta [7], and Bitran and Hax [2]. A heuristic algorithm for solving the integer version of (1)–(3) has been developed by McCallum [8].

2. Characterization of solutions

Let us rewrite (1)–(3) in a slightly different form as follows:

$$\min \quad \tfrac{1}{2}x'Dx - a'x, \tag{4}$$

$$\text{s.t.} \quad \sum_j x_j - c = 0, \quad (\lambda), \tag{5}$$

$$x_j - b_j \leq 0, \quad (u_j), \tag{6}$$

$$-x_j \leq 0, \quad (v_j), \tag{7}$$

where λ, u_j, and v_j are the Kuhn–Tucker multipliers associated with the three types of constraints. The Kuhn–Tucker conditions for (4)–(7) may be stated as follows:

$$d_j x_j - a_j + u_j - v_j + \lambda = 0 \quad \text{for all } j, \tag{8}$$

$$u_j(x_j - b_j) = 0 \quad \text{for all } j, \tag{9}$$

$$v_j x_j = 0 \quad \text{for all } j, \tag{10}$$

$$u_j,\ v_j \geq 0 \quad \text{for all } j, \tag{11}$$

plus (5), (6) and (7)

where d_j is the jth diagonal element of D. Consider the following solution as a function of λ.

$$\left.\begin{aligned} x_j(\lambda) &= \max\left\{\min\left(\frac{a_j - \lambda}{d_j}, b_j\right), 0\right\} \\ u_j(\lambda) &= \max\{a_j - \lambda - d_j b_j, 0\} \\ v_j(\lambda) &= \max\{\lambda - a_j, 0\}. \end{aligned}\right\} \tag{12}$$

For any selection of λ, the above solution clearly satisfies (6), (7), and (11). We now show that this solution will also satisfy (8), (9), and (10).

Proposition 1. *The solution given by* (12) *satisfies* (9).

Proof. *Case* 1:

$$a_j - \lambda - d_jb_j < 0 \Rightarrow u_j = 0 \Rightarrow u_j(x_j - b_j) = 0.$$

Case 2:

$$a_j - \lambda - d_jb_j \geq 0 \Rightarrow \frac{a_j - \lambda}{d_j} \geq b_j \Rightarrow x_j = b_j \Rightarrow u_j(x_j - b_j) = 0.$$

This completes the proof of Propositon 1.

Proposition 2. *The solution given by* (12) *satisfies* (10).

Proof. *Case* 1:

$$a_j - \lambda < 0 \Rightarrow \frac{a_j - \lambda}{d_j} < 0 \Rightarrow x_j = 0 \Rightarrow v_jx_j = 0.$$

Case 2:

$$a_j - \lambda \geq 0 \Rightarrow v_j = 0 \Rightarrow v_jx_j = 0.$$

This completes the proof of Proposition 2.

Proposition 3. *The solution given by* (12) *satisfies* (8).

Proof. *Case* 1:

$$\frac{a_j - \lambda}{d_j} \geq b_j \Rightarrow x_j = b_j, \quad a_j - \lambda - d_jb_j \geq 0, \quad a_j - \lambda > 0.$$

$$a_j - \lambda - d_jb_j \geq 0 \Rightarrow u_j = a_j - \lambda - d_jb_j.$$

$$a_j - \lambda > 0 \Rightarrow v_j = 0.$$

Thus

$$d_jx_j - a_j + u_j - v_j + \lambda = d_jb_j - a_j + a_j - \lambda - d_jb_j - 0 + \lambda = 0.$$

Case 2:

$$0 < \frac{a_j - \lambda}{d_j} < b_j \Rightarrow x_j = \frac{a_j - \lambda}{d_j}, \quad a_j - \lambda - d_jb_j < 0, \quad a_j - \lambda > 0.$$

$$a_j - \lambda - d_jb_j < 0 \Rightarrow u_j = 0.$$

$$a_j - \lambda > 0 \Rightarrow v_j = 0.$$

Thus

$$d_jx_j - a_j + u_j - v_j + \lambda = d_j\left(\frac{a_j - \lambda}{d_j}\right) - a_j + 0 - 0 + \lambda = 0.$$

Case 3:

$$\frac{a_j-\lambda}{d_j}\leq 0 \Rightarrow x_j=0, \quad a_j-\lambda\leq 0.$$

$$a_j-\lambda\leq 0\Rightarrow a_j-\lambda-d_jb_j\leq 0, \quad v_j=\lambda-a_j.$$

$$a_j-\lambda-d_jb_j\leq 0\Rightarrow u_j=0.$$

Thus

$$d_jx_j-a_j+u_j-v_j+\lambda=0-a_j+0-(\lambda-a_j)+\lambda=0.$$

This completes the proof of Proposition 3.

Hence to solve (1)–(3) one need only find the appropriate λ such that (5) is satisfied.

Let

$$g(\lambda)=\sum_j x_j(\lambda)=\sum_j \max\left\{\min\left(\frac{a_j-\lambda}{d_j},\ b_j\right),0\right\}.$$

Then we must find λ^* such that $g(\lambda^*)=c$. Note that since $d_j>0$ and $b_j\geq 0$, $x_j(\lambda)$ may be expressed as follows:

$$x_j(\lambda)=\begin{cases} b_j, & \lambda\leq a_j-d_jb_j,\\ \dfrac{a_j-\lambda}{d_j}, & a_j-d_jb_j<\lambda\leq a_j,\\ 0, & \lambda>a_j.\end{cases}$$

Clearly each $x_j(\lambda)$ is piece-wise linear and monotone nonincreasing. Since the sum of such functions preserves this property, $g(\lambda)$ is piece-wise linear and montone nonincreasing. A typical g is illustrated in Fig. 1.

3. Algorithm

Suppose there are n $x_j(\lambda)$, then the breakpoints for the piece-wise linear function $g(\lambda)$ occur at the $2n$ points $a_j-d_jb_j$ and a_j for $j=1,\dots,n$. Let $y_1,\dots,y_{2n}$ denote these breakpoints where $y_1\leq y_2\leq\dots\leq y_{2n}$. Then for $\lambda\leq y_1$, $g(\lambda)=\sum_j b_j$ and for $\lambda\geq y_{2n}$, $g(\lambda)=0$.

We now present an algorithm for obtaining the value λ^*, such that $g(\lambda^*)=c$. The procedure consists of a binary search to bracket λ^* between two breakpoints followed by a linear interpolation.

Algorithm for λ^*

Step 0. *Initialization.* If $c>\sum_j b_j$ or $c<0$, terminate with no feasible solution; otherwise, set $l\leftarrow 1$, $r\leftarrow 2n$,
$L\leftarrow\sum_j b_j$ and $R\leftarrow 0$.

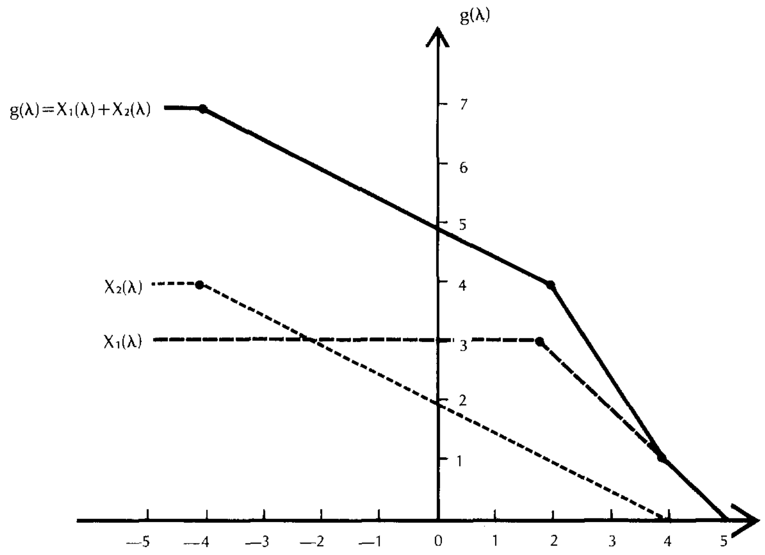

Fig. 1. Illustration of $g(\lambda)$ ($a_1 = 5$, $d_1 = 1$, $b_1 = 3$, $a_2 = 4$, $d_2 = 2$, $b_2 = 4$).

Step 1. ***Test for bracketing.*** If $r - l = 1$, go to step 4; otherwise, set $m \leftarrow [\frac{1}{2}(l + r)]_I$, where $[k]_I$ is the greatest integer $\leq k$.

Step 2. *Compute new value.*
Set

$$C \leftarrow \sum_j \max\left\{\min\left(\frac{a_j - y_m}{d_j}, b_j\right), 0\right\}.$$

Step 3. *Update.* If $C = c$, terminate with $\lambda^* \leftarrow y_m$.
If $C > c$, set $l \leftarrow m$, $L \leftarrow C$, and go to Step 1.
If $C < c$, set $r \leftarrow m$, $R \leftarrow C$, and go to Step 1.

Step 4. *Interpolate.* Terminate with

$$\lambda^* \leftarrow y_l + \frac{(y_r - y_l) \cdot (c - L)}{(R - L)}.$$

The above algorithm for (1)–(3) is considered "good" (see Lawler [6]) because the number of elementary computational steps is bounded by a polynomial in the size of the problem, n. Let c, a, and d denote the times required to execute one comparison, one addition, and one division, respectively. Table 1 presents a worst case analysis for each step of the algorithm. From Table 1, we see that the computational time is bounded by $(2c + 2a + d)[n \log_2(2n - 2)] + (a)n + (2c + 2a + d)\log_2(2n - 2) + 3c + 6a + 3d$. Hence, the algorithm for the singly constrained quadratic program, presented in this exposition, is a member of the class known as "good" algorithms.

Table 1
Worst case analysis of algorithm operations

		Operations count		
Step number	Maximum executions of step	Comparisons (c)	Additions (a)	Divisions (d)
0	1	2	n	0
1	$\log_2(2n-2)+1$	1	2	1
2	$\log_2(2n-2)$	$2n$	$2n$	n
3	$\log_2(2n-2)$	1	0	0
4	1	0	4	2*
Totals	—	$(2n+2)\times\log_2(2n-2)+3$	$(2n+2)\times\log_2(2n-2)+n+6$	$(n+1)\times\log_2(2n-2)+3$

* Includes one multiplication.

Acknowledgment

This research was supported in part by the Air Force Office of Scientific Research under Grant Number AFOSR 77-3151.

References

[1] A. Ali, R. Helgason, J. Kennington and H. Lall, "Solving multicommodity network flow problems", *Operations Research*, to appear.

[2] G.R. Bitran and A.C. Hax, "On the solution of convex knapsack problems with bounded variables", in: A. Prékopa, ed., *Survey of mathematical programming*, Vol. 1 (North-Holland, Amsterdam, 1979) pp. 357–367.

[3] A. Charnes and W.W. Cooper, "The theory of search: optimum distribution of search effort", *Management Science* 5 (1958) 44–50.

[4] M. Held, P. Wolfe and H. Crowder, "Validation of subgradient optimization", *Mathematical Programming* 6 (1974) 62–88.

[5] J. Kennington and M. Shalaby, "An effective subgradient procedure for minimal cost multicommodity flow problems", *Management Science* 23 (1977) 994–1004.

[6] E.L. Lawler, *Combinatorial optimization: networks and matroids* (Holt, Rinehart, and Winston, New York, 1976).

[7] H. Luss and S.K. Gupta, "Allocation of effort resources among competing activities", *Operations Research* 23 (1975) 360–365.

[8] C.J. McCallum Jr., "An algorithm for certain quadratic integer programs", Bell Laboratories Technical Report, Holmdel, NJ (undated).

[9] L. Sanathanan, "On an allocation problem with multistage constraints", *Operations Research* 19 (1971) 1647–1663.

[10] K.S. Srikantan, "A problem in optimum allocation", *Operations Research* 11 (1963) 265–273.