BEST NETWORK FLOW BOUNDS FOR THE QUADRATIC KNAPSACK PROBLEM[1]

PAUL CHAILLOU
Renault, Paris, France

PIERRE HANSEN
RUTCOR, Rutgers University, USA

YVON MAHIEU
Solvay, Brussels, Belgium.

A Lagrangean relaxation of the quadratic knapsack problem is studied. It is shown, among other properties, that the best value of the Lagrangean multiplier, and hence the best bound for the original problem, can be determined in at most n−1 applications of a maximum flow algorithm to a network with n+2 vertices and n+m arcs, where n and m denote the numbers of variables and of quadratic terms. A branch-and-bound algorithm using this result is presented and computational experience is reported on.

1. INTRODUCTION.

The *quadratic knapsack problem*, introduced by Gallo, Hammer and Simeone ([3]), can be expressed as follows:

$$\text{Maximize } f(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_ix_j \tag{1}$$

subject to

$$\sum_{j=1}^{n} a_jx_j \leq b \tag{2}$$

$$x_j \in \{0,1\} \quad \text{for } j = 1, 2, \ldots, n \tag{3}$$

and $b > 0$, $a_j \geq 0$, $c_{ij} \geq 0$ for $j = 1, 2, \ldots, n$. Note that (1) contains linear terms as $c_{ii}x_ix_i = c_{ii}x_i$. This problem has several applications cited in [3] e.g. location of airports to maximize freight, location of weather measurement stations with least correlation between measures,

[1]Paper presented at the NETFLOW 83 International Workshop, Pisa, Italy, 28-31 March 1983.

detection of a k-clique in a graph.

Problem (1)-(3) is NP-complete, as the case $c_{ij} = 0$ for $i \neq j$ is the usual knapsack problem. Gallo, Hammer and Simeone ([3]) use *upper planes* of (1), i.e. linear majorizing functions to obtain knapsack problems as relaxations. They explore several variants and compare the performances of the corresponding branch-and-bound algorithms.

The present paper is devoted to a Lagrangean relaxation approach to problem (1)-(3), and is organized as follows: properties of the Lagrangean function are studied in the next section; a polynomial algorithm to obtain the best value of the Lagrangean multiplier, and hence the best bound for the original problem is presented in section 3, together with heuristic and exact algorithms for the quadratic knapsack problem itself; computational experience is reported on in the last section.

2. PROPERTIES OF THE LAGRANGEAN FUNCTION.

Using a non-negative multiplier λ, the introduction of the constraint (2) into the objective function (1) yields the Lagrangean function

$$h(x_1, x_2, \ldots, x_n, \lambda) = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_i x_j + \lambda (b - \sum_{j=1}^{n} a_j x_j). \tag{4}$$

We are interested here in the properties of the function

$$\overline{f}(\lambda) = \max_{x_1, \ldots, x_n \in \{0,1\}} h(x_1, x_2, \ldots, x_n, \lambda). \tag{5}$$

Let $f^* = f(x_1^*, \ldots, x_n^*)$ denote the value of the optimal solution of (1)-(3). The first three following properties are particular cases of well-known results on Lagrangean relaxation (see Minoux [8, Chap. 6]). The proofs are therefore omitted.

Property 1. *$\overline{f}(\lambda)$ is piecewise linear in λ.* ∎

Property 2. $\min_{\lambda \geq 0} \overline{f}(\lambda) \geq f^*$. ∎

Property 3. *$\overline{f}(\lambda)$ is convex.* ∎

The *inclusion property* which is next proved allows to find the value of λ which minimizes $\overline{f}(\lambda)$ in polynomial time. A similar property has been shown to be true for hyperbolic 0-1 programs by Picard and Queyranne ([9]), with a similar proof.

Property 4. *Let* $0 \leq \lambda^1 < \lambda^2$, $\max\{h(x_1, \ldots, x_n, \lambda^1)\} = h(x_1^1, \ldots, x_n^1, \lambda^1)$ *and* $\max\{h(x_1, \ldots, x_n, \lambda^2)\} = h(x_1^2, \ldots, x_n^2, \lambda^2)$. *Then* $x_j^1 \geq x_j^2$ *for* $j = 1, 2, \ldots, n$.

Proof. Let $I = \{j;\ x_j^1 = 1,\ x_j^2 = 0\}$, $J = \{j;\ x_j^1 = 1,\ x_j^2 = 1\}$, and $K = \{j;\ x_j^1 = 0,\ x_j^2 = 1\}$. We must show that $K = \emptyset$. Let us consider two more Boolean n-vectors defined by

$$x_j^3 = \begin{cases} 1 & \text{if } j \in I \cup J \cup K \\ 0 & \text{if } j \notin I \cup J \cup K \end{cases} \quad \text{and} \quad x_j^4 = \begin{cases} 1 & \text{if } j \in J \\ 0 & \text{if } j \notin J. \end{cases}$$

From the definitions above and the optimality of $(x_1^1, \ . \ . \ . \ , x_n^1)$, we have

$$h(x_1^1, \ . \ . \ . \ , x_n^1, \lambda^1) = \sum_{i \in I \cup J} \sum_{j \in I \cup J} c_{ij} + \lambda^1 (b - \sum_{j \in I \cup J} a_j)$$

$$\geq \ h(x_1^3, \ . \ . \ . \ , x_n^3, \lambda^1) = \sum_{i \in I \cup J} \sum_{j \in I \cup J} c_{ij} + \sum_{i \in K} \sum_{j \in K} c_{ij} +$$

$$\sum_{i \in I \cup J} \sum_{j \in K} c_{ij} + \sum_{i \in K} \sum_{j \in I \cup J} c_{ij} + \lambda^1 (b - \sum_{j \in I \cup J} a_j - \sum_{j \in K} a_j)$$

which implies

$$\lambda^1 \sum_{j \in K} a_j \ \geq \ \sum_{i \in I} \sum_{j \in K} c_{ij} + \sum_{i \in J} \sum_{j \in K} c_{ij} +$$

$$\sum_{i \in K} \sum_{j \in I} c_{ij} + \sum_{i \in K} \sum_{j \in J} c_{ij} + \sum_{i \in K} \sum_{j \in K} c_{ij}. \tag{6}$$

From the optimality of $(x_1^2, \ . \ . \ . \ , x_n^2)$ it follows that

$$h(x_1^2, \ . \ . \ . \ , x_n^2, \lambda^2) = \sum_{i \in J} \sum_{j \in J} c_{ij} + \sum_{i \in J} \sum_{j \in K} c_{ij} +$$

$$\sum_{i \in K} \sum_{j \in J} c_{ij} + \sum_{i \in K} \sum_{j \in K} c_{ij} + \lambda^2 (b - \sum_{j \in J} a_j - \sum_{j \in K} a_j)$$

$$\geq \ h(x_1^4, \ . \ . \ . \ , x_n^4, \lambda^2) = \sum_{i \in J} \sum_{j \in J} c_{ij} + \lambda^2 (b - \sum_{j \in J} a_j),$$

hence

$$\sum_{i\in J}\sum_{j\in K} c_{ij} + \sum_{i\in K}\sum_{j\in J} c_{ij} + \sum_{i\in K}\sum_{j\in K} c_{ij} \geq \lambda^2 \sum_{j\in K} a_j \tag{7}$$

and, as $c_{ij} \geq 0$, from (6) and (7)

$$\lambda^1 \sum_{j\in K} a_j \geq \lambda^2 \sum_{j\in K} a_j$$

which, as $\lambda^1 < \lambda^2$, is possible only if $K = \emptyset$. ∎

Corollary 4.1. *The number of linear segments of $\overline{f}(\lambda)$ is at most n.* ∎

Property 5. *Let $\overline{f}(\lambda^*) = \min_{\lambda\geq 0} \overline{f}(\lambda)$ (taking the smallest value of λ in case of multiple optima). Then the optimal solution of (5) is infeasible for all $\lambda < \lambda^*$ and feasible for all $\lambda \geq \lambda^*$.*

Proof. The slope of $\overline{f}(\lambda)$ is negative to the left of λ^* hence, for each linear segment, the coefficient of λ (i.e. $b - \sum_{j=1}^{n} a_j x_j$) is negative, and the corresponding solution is infeasible. The case $\lambda \geq \lambda^*$ is treated similarly. ∎

The following property uses the Boolean expression of the minimum-cut problem, due to Hammer ([4]), in a similar way as Picard and Ratliff ([10]). Let us associate with (5) a network $N = (V, U, \lambda)$ with a vertex set $V = \{s, 1, \ldots, n, p\}$, where s denotes the source and p the sink, and with an edge set $U = \{(s,i);\ i = 1, \ldots, n\} \cup \{(i,j);\ c_{ij} \neq 0\} \cup \{(i,p);\ i = 1, \ldots, n\}$. Moreover, let us give to the arcs the capacities

$$c_{sj}(\lambda) = \max(0, \sum_{i=1}^{n} c_{ji} - \lambda a_j), \quad c_{ij}(\lambda) = c_{ij} \text{ and } c_{jp}(\lambda) = \max(0, \lambda a_j - \sum_{i=1}^{n} c_{ji}).$$

Property 6. *We have $\overline{f}(\lambda) = \lambda b + \sum_{j=1}^{n} c_{sj}(\lambda) + \Psi(\lambda)$ where $\Psi(\lambda)$ denotes the maximum flow in N.*

Proof. From Ford and Fulkerson's theorem ([1]), we have

$$\Psi(\lambda) = \min\{\sum_{j=1}^{n} c_{sj}(\lambda)(1 - x_j) + \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} x_i (1 - x_j) + \sum_{j=1}^{n} c_{jp}(\lambda) x_j\}$$

$$= \sum_{j=1}^{n} c_{sj}(\lambda) + \min\{\sum_{j=1}^{n} (c_{jp}(\lambda) - c_{sj}(\lambda) + \sum_{i=1}^{n} c_{ji}) x_j - \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} x_i x_j\}$$

$$= \sum_{j=1}^{n} c_{sj}(\lambda) + \min\{\sum_{j=1}^{n} (\max(\lambda a_j - \sum_{i=1}^{n} c_{ji}, 0) + \min(\lambda a_j - \sum_{i=1}^{n} c_{ji}, 0) + \sum_{i=1}^{n} c_{ji})x_j - \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_i x_j\}$$

$$= \sum_{j=1}^{n} c_{sj}(\lambda) + \min(\sum_{j=1}^{n} \lambda a_j x_j - \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_i x_j)$$

$$= \sum_{j=1}^{n} c_{sj}(\lambda) - \overline{f}(\lambda) + \lambda b,$$ from which the result follows. ■

Let us now consider some particular values of λ. Let

$$\lambda_j = \sum_{i=1}^{n} c_{ij}/a_j \text{ for } j = 1, \ldots, n \text{ and } \lambda_0 = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}/\sum_{j=1}^{n} a_j = \sum_{j=1}^{n} \lambda_j a_j/\sum_{j=1}^{n} a_j.$$

Property 7. $\overline{f}(\lambda) = \lambda b$ *for all* $\lambda > \max\{\lambda_j\}$.

Indeed when $\lambda > \max\{\lambda_j\}$, $c_{sj} = 0$ for all j and hence $\Psi(\lambda^*) = 0$, from which the result follows. ■

Note also that $\overline{f}(0) = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}$.

Property 8. $\overline{f}(\lambda) \geq \lambda_0 b$ *for all* $\lambda \geq 0$.

Proof. $\overline{f}(\lambda)$ is greater than or equal to (5) for all Boolean n-vectors, in particular for the vectors $(0, 0, \ldots, 0)$ and $(1, 1, \ldots, 1)$. Hence

$$\overline{f}(\lambda) \geq \max(\lambda b, \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} + \lambda(b - \sum_{j=1}^{n} a_j))$$

and the minimum of this expression is attained for $\lambda = \lambda_0$. ■

The results of this section are summarized in Figure 1.

3. ALGORITHMS.

The following algorithm uses the results of the previous section to determine λ^* such that $\overline{f}(\lambda^*) = \min_{\lambda>0} \overline{f}(\lambda)$.

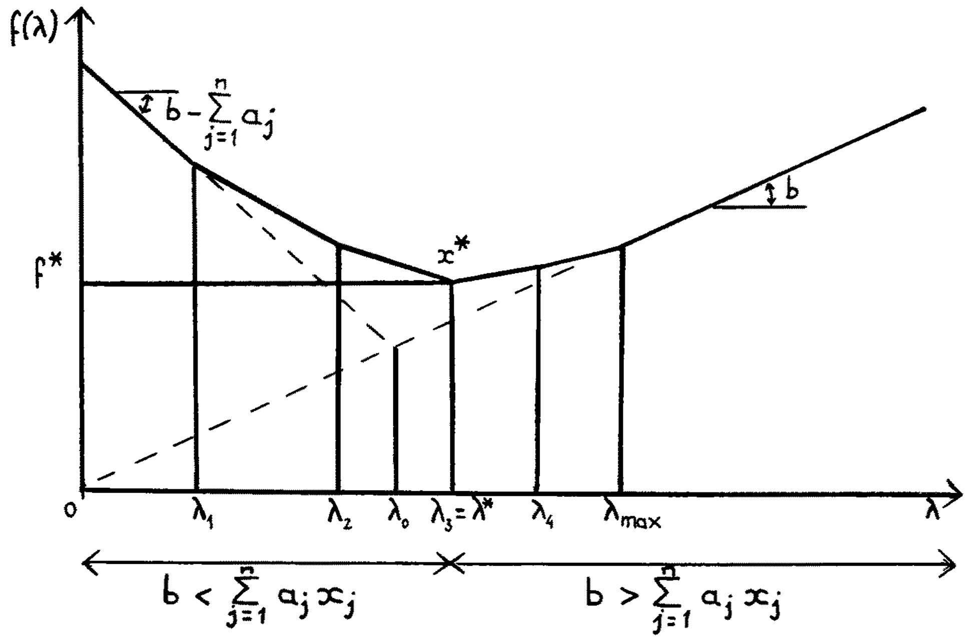

Figure 1. Properties of the Lagrangean function.

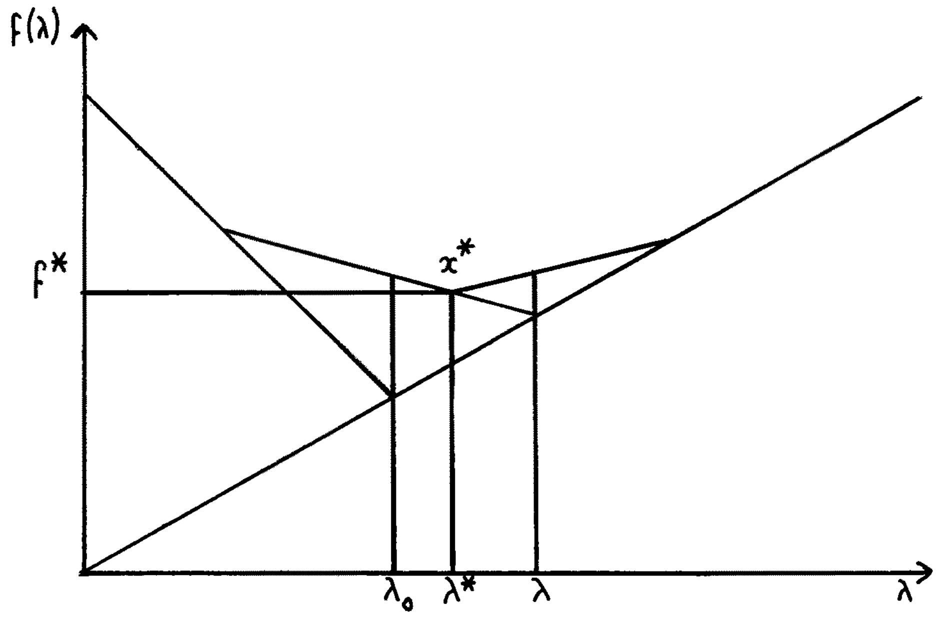

Figure 2. Obtention of λ^*.

ALGORITHM 1.

a) *Initialization.* Set $\alpha_1 := b - \sum_{j=1}^{n} a_j$ and $\beta_1 := \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}$. Set $\alpha_2 := b$ and $\beta_2 = 0$.
b) *Maximum flow problem.* Compute $\lambda = (\beta_1 - \beta_2)/(\alpha_2 - \alpha_1)$. Construct the network $N = (V, U, \lambda)$ and determine a maximum flow between s and p. Let $\alpha_3\lambda+\beta_3$ denote the line in the $(\lambda, \overline{f}(\lambda))$-space corresponding to the optimal solution of (5), defined by Property 6.

c) *Optimality test.* If $\alpha_3\lambda+\beta_3 = \alpha_1\lambda+\beta_1$, then $\lambda^* = \lambda$, and so end. Otherwise if $\alpha_3 < 0$, set $\alpha_1 := \alpha_3$, $\beta_1 := \beta_3$ and go to b); if $\alpha_3 \geq 0$ set $\alpha_2 := \alpha_3$, $\beta_2 := \beta_3$ and go to b).

The application of Algorithm 1 to the function $\overline{f}(\lambda)$ of Figure 1 is illustrated in Figure 2; it is seen that λ^* is found in three iterations.

As two linear segments out of a maximum of n are considered at the first iteration and one more at each subsequent one, at most $n-1$ iterations are needed. As N has $n+2$ vertices and $n+m$ arcs, it follows that Algorithm 1 takes $O(n^4)$ operations in the worst case, if Karzanov's algorithm ([5]) or the "three Indians" one ([6]) is used in step b).

We now turn to the resolution of the quadratic knapsack problem itself, and first consider a heuristic algorithm of the "Attila" type.

ALGORITHM 2.

a) *Initialization.* Set $K_1 := \{1, 2, \ldots, n\}$, $K_0 := \emptyset$. Compute $q_j = \sum_{i=1}^{n} (c_{ij} + c_{ji})/a_j$ for $j = 1, 2, \ldots, n$ and $\sum_{j=1}^{n} a_j$.
b) If $\sum_{j\in K_1} a_j \leq b$, end with a heuristic solution X_h given by $x_j = 1$ for all j in K_1, $x_j = 0$ for all j in K_0.

c) *Setting a variable to 0.* Select $q_k = \min\{q_j;\ j \in K_1\}$. Set $K_1 := K_1 - \{k\}$, $K_0 := K_0 \cup \{k\}$. Update q_j for j in K_1 by setting $q_j := q_j - (c_{kj} + c_{jk})/a_j$. Return to step b).

If the quadratic terms with $c_{ij} \neq 0$ are given in linked lists, step a) requires $O(m)$ operations. The sum $\sum_{j\in K_1} a_j$ being updated, step b) is $O(1)$ and done n times at most, hence is $O(n)$ in all. If the q_j's are kept in a heap, updating the heap and selecting the smallest q_j can be done in $O(m\,\mathrm{Log}n)$ operations, which is also the worst-case complexity of Algorithm 2.

We now describe the exact branch-and-bound algorithm.

ALGORITHM 3.

a) *Heuristic solution.* Obtain a heuristic solution X_h by Algorithm 2. Set $X_{opt} := X_h$ and $f_{opt} := f(X_h)$.

b) *Lexicographic fixation of variables.*

b.1) Determine λ^* by Algorithm 1. Let X^* denote the corresponding optimal solution of (5). If

$\sum_{j=1}^{n} a_j x_j^* = b$, end, X^* being the optimal solution of the quadratic knapsack problem.

b.2) Set $K_1 := \emptyset$, $K_0 := \emptyset$, $K_2 := \{1, 2, \ldots, n\}$ i.e. consider all variables as free. Then, for all j in K_2 in turn:

b.3) Set $x_j := \bar{x}_j^*$ i.e. at the complement of its value in X^*. Then, if $x_j = 1$, set $x_k := 0$ for all k in K_2 such that $a_k > b - \sum_{l \in K_1} a_l - a_j$; if $x_j = 0$ set $x_k := 0$ for all k in K_2 such that $a_k > b - \sum_{l \in K_1} a_l$. Compute by Algorithm 1 the best Lagrangean bound $\bar{f}_j(\lambda_j^*)$ of the so-defined subproblem. If $\bar{f}_j(\lambda_j^*) \leq f_{opt}$ set $K_1 := K_1 \cup \{j\}$, $K_2 := K_2 - \{j\}$ if $x_j = 0$ and $K_0 := K_0 \cup \{j\}$, $K_2 := K_2 - \{j\}$ if $x_j = 1$.

c) *Branch-and-bound routine.*

c.1) Set $M = \emptyset$.

c.2) Compute $\underline{b} = b - \sum_{l \in K_1} a_l$. If $\underline{b} < 0$ go to c.7).

c.3) For all j in K_2, if $a_j > \underline{b}$ set $K_0 := K_0 \cup \{j\}$, $K_2 := K_2 - \{j\}$, add j to M by the right and underline it.

c.4) Compute by Algorithm 1 the best Lagrangean bound $\bar{f}(\lambda^*)$ for the current subproblem; let X^* denote the corresponding solution. If $\bar{f}(\lambda^*) \leq f_{opt}$ go to c.7).

c.5) If $\sum_{j=1}^{n} a_j x_j^* \leq b$, set $X_{opt} = X^*$, $f_{opt} = f(X^*)$; moreover if $\sum_{j=1}^{n} a_j x_j^* = b$ go to c.7).

c.6) Choose j in K_2 according to rule d) hereunder and fix x_j at the specified value. Add j to M by the right. Go to c.2).

c.7) Seek from right to left an index j in M that is not underlined. If no such index exists, end. Otherwise set free the variables with an index k to the right of j in M and erase those indices, set $x_j = \bar{x}_j$, underline j in M, update K_0, K_1 and K_2 accordingly and go to c.2).

d) *Branching.*

d.1) Determine for each j in K_2 the decrease in value of $\bar{f}(\lambda)$ when x_j^* is replaced by $\bar{x}_j^*$, the value of λ being that one corresponding to the last bound computed.

d.2) Choose j in K_2 such that the decrease computed in the previous step is the smallest possible. Give to x_j the value $\bar{x}_j^*$.

COMPUTATIONAL EXPERIENCE.

The algorithms of the previous section have been extensively tested on a fairly small IBM 360/65 computer ([7]). Four hundred problems, with 10 to 50 variables and between 25 and 100 % of the largest possible number of quadratic terms have been randomly generated. All coefficients come from uniform distributions on [1, 100] for the c_{ij}, on [1, 50] for the a_{ij} and on $[50, \sum_{j=1}^{n} a_j]$ for b. Each problem was allocated a maximum CPU time of 2700 seconds, which was not enough to complete resolution for some of the largest and densest problems. The results are summarized in Table 1.

It appears clearly that a large proportion (about two thirds) of the variables are fixed in the preliminary procedure (step b) of Algorithm 3. Also, the number of subproblems solved and the number of backtracking remain fairly moderate.

The values of the upper and lower bounds given by Algorithms 1 and 2 were compared to the optimal values of two further series of 30 problems with 50 and 40 variables and a maximum number of quadratic terms. The results are summarized in Table 2. It is seen that the heuristic often yields an optimal solution or near-optimal solution, while the uper bound is quite precise. Comparing these computational results with those of Gallo, Hammer and Simeone ([3]) we find that computer times are larger but, on a smaller computer, bounds are more precise and subproblems less numerous. Finally, we note that a few larger problems, with up to 150 variables were solved in several hours of computer time.

Key for Table 1: D = density of the c_{ij} matrix, n = number of variables, m = number of quadratic terms, N_{sp} = number of subproblems solved, t = computational time in CPU seconds, N_p = number of variables fixed in the preliminary procedure, N_b = number of backtrackings in the branch-and-bound procedure, N_{bb} = number of variables fixed in the branch-and-bound procedure, N_{spbb} = number of subproblems solved in the branch-and-bound procedure, S = percentage of problems solved in 2700 seconds.

Figures on odd and even lines are mean values and standard deviations respectively for 20 problems in each case.

D	n	m	N_{sp}	t	N_p	N_b	N_{bb}	N_{spbb}	S
100	50	1275	279	655.25	36	120	14	243	60
		0	265	397.33	8	136	8	271	
	40	820	352	607.83	24	165	16	328	78.26
		0	445	672.46	11	238	11	452	
	30	465	240	281.35	18	117	12	222	100
		0	309	409.82	9	173	9	317	
	20	210	98	55.56	11	42	9	87	100
		0	103	60.40	6	54	6	109	
	10	55	17	3.85	5	5	5	12	100
		0	12	3.06	3	7	3	4	
75	50	951	336	650.89	32	148	18	304	75
		13	300	519.85	14	151	14	305	
	40	619	256	443.11	24	111	16	232	90
		8	228	473.92	14	115	14	241	
	30	348	144	121.20	22	60	8	122	100
		7	230	147.02	7	115	7	235	
	20	157	123	63.50	9	59	11	114	100
		6	185	92.90	7	105	7	189	
	10	40	16	3.68	6	4	4	10	100
		3	9	2.04	2	5	2	10	
50	50	639	255	428.84	38	106	12	217	90.48
		20	309	277.10	8	157	8	315	
	40	412	171	277.47	28	68	12	143	100
		15	181	356.55	10	91	10	190	
	30	233	158	104.75	20	67	10	138	100
		11	250	118.05	9	125	9	255	
	20	105	64	29.46	13	24	7	51	100
		7	68	31.63	5	35	5	73	
	10	27	18	3.56	6	5	4	11	100
		4	14	3.13	3	8	3	17	
25	50	317	194	313.00	36	74	14	157	100
		15	209	347.38	11	104	11	219	
	40	204	158	145.48	28	61	12	130	100
		12	141	123.59	8	71	8	148	
	30	117	86	51.45	22	30	8	64	100
		8	105	47.80	8	52	8	109	
	20	51	31	11.29	16	7	4	14	100
		7	33	13.79	4	17	4	36	
	10	15	20	3.63	6	6	4	14	100
		4	15	3.44	3	9	3	19	

Table 1. Computational Results.

n	$UB - opt$	$opt - LB$	$UB - LB$
50	4.230	0.191	4.421
	7.701	0.332	7.773
40	3.985	0.051	4.037
	5.152	0.177	5.131

Table 2. Comparison of upper bounds, lower bounds and optimal values. Odd lines give mean values and even lines standard deviations for 30 problems (in %).

Note added on proofs: Gallo, Grigoriadis and Tarjan ([2]) have recently obtained an $O(n^3)$ algorithm for the parametric network flow problem. it can be used to find the best multiplier with a worst-case complexity an order of magnitude lower than that of Algorithm 1.

REFERENCES

[1] FORD L.R. and D.R. FULKERSON, *Flows in networks*, Princeton University Press, 1962.

[2] GALLO G., M. GRIGORIADIS and R.E. TARJAN, "A Fast Parametric Network Flow Algorithm", Research Report, Department of Computer Science, Rutgers University, N.J. (forthcoming).

[3] GALLO G., P.L. HAMMER and B. SIMEONE, "Quadratic Knapsack Problem", *Math. Prog. Study* ***12*** (1980), 132-149.

[4] HAMMER P.L., "Some Network Flow Problem Solved with Pseudo-Boolean Programming", *Operations Research* ***13*** (1965), 388-399.

[5] KARZANOV A.V., "Determining the Maximum Flow in a Network by the Method of Preflows", *Soviet Math. Dokl.* ***15*** (1974), 434-437.

[6] MALHOTRA V.M., M.P. KUMAR and S.N. MAHESHWARI, "An $O(V^3)$ Algorithm for Finding Maximum Flows in Networks", *Information Processing Letters* ***7*** (1978), 277-278.

[7] MAHIEU Y., Un Algorithme pour le problème du sac de campeur quadratique, Mémoire de fin d'études, Faculté Universitaire Catholique de Mons, Belgium, 1981.

[8] MINOUX M., *Programmation Mathématique*, 2 Volumes, Dunod, Paris, 1983.

[9] PICARD J.C. and M. QUEYRANNE, "Networks, Graphs and Some Non-Linear 0-1 Programming Problems", Techn. Rep. EP77-R-32, Ecole Polytechnique de Montréal, Canada, 1977.

[10] PICARD J.C. and H.D. RATLIFF, "Minimum Cuts and Related Problems", *Networks* ***5*** (1975), 357-370.