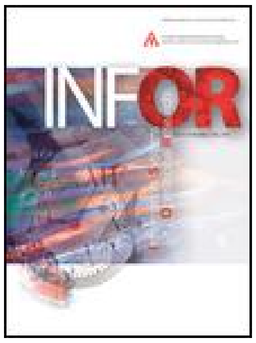


INFOR: Information Systems and Operational Research

ISSN: 0315-5986 (Print) 1916-0615 (Online) Journal homepage: http://www.tandfonline.com/loi/tinf20

Efficient Methods For Solving Quadratic 0–1 Knapsack Problems

Peter Hammaer & David J. Rader Jr.

To cite this article: Peter Hammaer & David J. Rader Jr. (1997) Efficient Methods For Solving Quadratic 0–1 Knapsack Problems, INFOR: Information Systems and Operational Research, 35:3, 170-182, DOI: 10.1080/03155986.1997.11732327

To link to this article: http://dx.doi.org/10.1080/03155986.1997.11732327

Published online: 25 May 2016.

Submit your article to this journal

View related articles

Full Terms & Conditions of access and use can be found at
http://www.tandfonline.com/action/journalInformation?journalCode=tinf20

Download by: [McMaster University] **Date:** 09 June 2016, At: 05:07

EFFICIENT METHODS FOR SOLVING QUADRATIC 0–1 KNAPSACK PROBLEMS[1]

PETER L. HAMMER, DAVID J. RADER, JR.

RUTCOR, Rutgers University, P.O. Box 5062, New Brunswick, NJ 08903.
hammer@rutcor.rutgers.edu *rader@rutcor.rutgers.edu*

ABSTRACT

We propose a heuristic algorithm and an exact branch and bound algorithm for the problem of maximizing a quadratic function in $0-1$ variables which are subject to a linear inequality. The proposed algorithms aim at determining, at every step of the procedure, variables which can be fixed to values that are both feasible and optimal for certain subproblems. The algorithms make repeated use of the L_2-best linear approximation of the objective function and use for variable fixation conclusions derived from Lagrangean relaxation, order relations and constraint pairing. We investigate the role of each of the three variable fixation techniques using computational tests. Extensive computational results are presented comparing the proposed heuristic and exact algorithm to previously known ones. The conclusions show that the solution produced by the proposed heuristic algorithm is, on the average, within 1% of the optimum and that the number of nodes examined in the proposed exact algorithm is dramatically smaller than the ones examined in published methods.

RÉSUMÉ

On propose un algorithme heuristique ainsi qu'un algorithme exact de "branch and bound" pour le problème de maximisation d'une fonction quadratique à variables bivalentes sujettes à une inégalité linéaire. Les algorithmes proposés ont pour but de déterminer, à chaque étape de la procédure, des variables dont les valeurs peuvent être fixés d'une façon qui est à la fois admissible et optimale pour certains sous-problèmes. Les algorithmes utilisent à plusieurs reprises des approximations linéaires L_2-optimales de la fonction objective, et emploient afin de fixer les valeurs des variables, des conclusions obtenues par relaxation Lagrangéenne, par relations d'ordre, et par couplages des contraintes. Nous étudions le rôle de chaqune des trois techniques de fixation de variables par tests sur l'ordinateur. On présente des nombreaux résultats de calcul afin de comparer l'heuristique proposée, ainsi que l'algorithme exact, avec les méthodes connues. Nos conclusions montrent que la solution obtenue par l'heuristique proposée différe de l'optimum en moyenne par moins de 1%, et que le nombre de noeuds examinés dans l'algorithme exact proposé est considérablement réduit en comparaison avec les méthodes connues.

1. INTRODUCTION

The *quadratic knapsack problem* (QKP), as defined in Gallo, Hammer and Simeone [5], consists of the following:

$$\text{Maximize} \quad Q(\mathbf{x}) = \sum_{k=1}^{n} c_k x_k + \sum_{1\le i<j\le n} c_{ij} x_i x_j \tag{1}$$

subject to

$$\sum_{k=1}^{n} a_k x_k \le b \tag{2}$$

$$\mathbf{x} \in \{0,1\}^n$$

where $c_k, a_k \geq 0$ for $k = 1, \ldots, n$, $c_{ij} \geq 0$ for $1 \leq i,j \leq n$ and $b > 0$. We assume, for notational expedience, that $c_{ij} = c_{ji}$. In addition, we assume for simplicity throughout that $a_1 \geq a_2 \geq \ldots \geq a_n$.

The quadratic knapsack problem has been studied by various authors. Branch and bound methods were developed in Chaillou, Hansen and Mahieu [1] and [5] for the exact solution of the problem. Upper planes (linear majorization functions of (1)) were used in [5] to obtain upper

[1]Recd. Feb. 1995; Revd. Oct. 1996

Downloaded by [McMaster University] at 05:07 09 June 2016

Downloaded by [McMaster University] at 05:07 09 June 2016

bounds of the maximum, while Lagrangean relaxation techniques were used in [1]. The upper bounds found in [5] were improved upon in Körner [13], while the complexity of finding the Lagrangean dual of (QKP) has been improved upon in Gallo, Grigoriadis and Tarjan [4]. Some valid inequalities and facets for a linear programming formulation of (QKP) are described in Johnson, Mehrotra and Nemhauser [12].

Several generalizations of (QKP) have also been studied by various authors. Results on the Lagrangean dual of the *nonlinear knapsack problem*, and in particular the *supermodular knapsack problem*, were presented in Gallo and Simeone [6]. These results generalize and extend results found in [1]. Note that, since $c_{ij} \geq 0$ in (1), (QKP) is a particular case of the supermodular knapsack problem. Also, valid inequalities and facets for the nonlinear knapsack problem were described in Crama and Mazzola [2].

(QKP) and the nonlinear knapsack problem have been used to model a variety of problems, including flexible manufacturing systems, graph and hypergraph partitioning and compiler construction. For references and other details, see [2, 12, 14].

As noted in [5], (QKP) is NP-hard. However, since (1) is supermodular, in the absence of the knapsack constraint (2), it is polynomially-solvable. Thus, the Lagrangean Dual of (QKP) can be solved in polynomial time [1, 6]. In addition, this bound has the same optimal solution as the LP-relaxation of (QKP) [7].

Besides Lagrangean relaxation, other methods have been used to help solve both the unconstraind quadratic $0-1$ program as well as (QKP). The use of order relations of the form $x_i \leq x_j$ to help solve unconstrained quadratic $0-1$ programs was introduced in Hammer and Hansen [8]. Additional work on order relations was given in Hammer and Simeone [11]. The pairing of two linear constraints in order to fix variables to their optimal values was introduced in Hammer, Padberg and Peled [10]. Constraint pairing was used in [5] for the solution of (QKP).

In this paper, we shall first present a heuristic procedure for finding a "good" solution to (QKP). We shall then develop a branch and bound procedure, which incorporates Lagrangian duality, order relations and constraint pairing, for the exact solution to the problem. Finally, we shall present the results of extensive computational experimentation.

2. LINEARIZATION AND EXCHANGE (LEX) HEURISTIC

In this section we develop a heuristic algorithm called Linearization and Exchange (LEX) for the quadratic knapsack problem. For this, we use the best linear L_2-approximation of (1) in order to build a linear knapsack problem associated to (1), (2) and to make use of it for determining $0-1$ values to the variables in a (cautiosly) greedy way in order to form a good solution to (QKP). Once we have set all variables, an exchange algorithm between certain components of the initial solution vector is used to further improve the solution.

In order to initiate LEX, we first need a "good" linear approximation of the objective function (1). The best linear L_2-approximation of a pseudo-Boolean function f, as defined in Hammer and Holzman [9], is the function l_f for which

$$\sum_{\mathbf{x}\in B^n} [f(\mathbf{x}) - l_f]^2 = \min_{l \text{ linear}} \sum_{\mathbf{x}\in B^n} [f(\mathbf{x}) - l(\mathbf{x})]^2.$$

The linear function l_f was characterized as the unique linear pseudo-Boolean function that agrees with f in average value and in average first derivatives. To find l_f, we use the following closed formulas given by

Lemma 2.1 [9]

The best linear L_2-approximation of Q is

$$l_Q = c_0^* + \sum_{i=1}^{n} c_i^* x_i$$

where

$$c_0^* = -\frac{1}{4}\sum_{i<j} c_{ij}$$
$$c_i^* = c_i + \frac{1}{2}\sum_{j\neq i} c_{ij},\ i = 1,\ldots,n$$

Using this lemma, we may easily calculate l_Q, as well as update l_Q when variables have been assigned values of 0 or 1. Note that the linear knapsack problem (LKP)

$$\begin{aligned}&\text{Maximize} \quad l_Q(\mathbf{x}) = \sum_{k=1}^{n} c_k^* x_k\\ &\text{subject to}\\ &\qquad \sum_{k=1}^{n} a_k x_k \leq b\\ &\qquad \mathbf{x} \in \{0,1\}^n\end{aligned}$$

yields an upper bound to our original problem.

To form our heuristic solution, we use (LKP) in a greedy fashion, assigning the value 1 to the variable with the highest benefit-cost ratio. After setting to 0 all variables that can no longer fit in our knapsack, we update l_Q and repeat.

Example.

Consider the following problem:

$$\begin{aligned}\text{Maximize} \quad Q(\mathbf{x}) = {} & 2x_1 + 5x_2 + 2x_3 + 4x_4 + 8x_1x_2\\ & +6x_1x_3 + 10x_1x_4 + 2x_2x_3 + 6x_2x_4 + 4x_3x_4\\ \text{subject to} \quad & 8x_1 + 6x_2 + 5x_3 + 3x_4 \leq 16\\ & x_k \in \{0,1\} \quad k = 1,\ldots,4\end{aligned} \tag{3}$$

Using Lemma 2.1, we obtain the best linear L_2 upper bound

$$l_Q = 14x_1 + 13x_2 + 8x_3 + 14x_4$$

of (3). Since

$$\frac{c_4^*}{a_4} = \frac{14}{3} = \max\left\{\frac{c_k^*}{a_k}\right\},$$

we set $x_4 = 1$. Updating our c^* and b, we obtain $\hat{b} = 13$ and

$$l_Q = 19x_1 + 15x_2 + 10x_3.$$

We now set $x_2 = 1$ because $c_2^*/a_2 = \max c_k^*/a_k$. This yields an updated $\hat{b} = 13 - 6 = 7$, which implies that $x_1 = 0$. This leaves x_3 as our only remaining free variable. Since $a_3 = 5 < 7 = \hat{b}$, we set $x_3 = 1$, yielding the feasible point $\mathbf{x} = (0,1,1,1)$ with value $= 23$.

In order to continue with a description of the LEX algorithm, we need a few further definitions. Given a quadratic $0-1$ program, the "derivatives" Δ_i were defined in [8] as

$$\Delta_i(\mathbf{x}) = c_i + \sum_{j\neq i} c_{ij}x_j$$

Downloaded by [McMaster University] at 05:07 09 June 2016

and the "second derivatives" Δ_{ij} as

$$\begin{aligned}\Delta_{ij}(\mathbf{x}) &= \Delta_i(\mathbf{x}) - \Delta_j(\mathbf{x}) + c_{ij}(x_i - x_j) \\ &= c_i - c_j + \sum_{k\neq i,j}(c_{ik} - c_{jk})x_k. \\ &= f(\mathbf{x}|x_i = 1, x_j = 0) - f(\mathbf{x}|x_i = 0, x_j = 1)\end{aligned} \tag{4}$$

These definitions give us conditions for "exchanging" the values of two coordinates as well as increasing the value of some coordinates.

Once we have found a feasible solution to (QKP), we employ as many times as possible a local improvement algorithm based on the method of Peterson [15]. This algorithm uses the following elementary operations:

Exchange: Suppose $x_i = 1$ and $x_j = 0$. If $\sum_{k\in P} a_k - a_i + a_j \leq b$ and $\Delta_{ij}(\mathbf{x}) = \min \Delta_{rs}(\mathbf{x}) < 0$, then transform $\mathbf{x}$ to $\mathbf{x}' = \mathbf{x} - \mathbf{e_i} + \mathbf{e_j}$, where $\mathbf{e_k}$ is the kth unit vector.

Fill-up: Suppose $x_i = 0$. If $\sum_{k\in P} a_k + a_i \leq b$ and $\Delta_i(\mathbf{x}) = \max_{k\in N} \Delta_k(\mathbf{x})$, then transform $\mathbf{x}$ to $\mathbf{x}' = \mathbf{x} + \mathbf{e_i}$.

where $P = \{i : x_i = 1\}$ and $N = \{i : x_i = 0\}$. These operations find a feasible point that is at least as good as our original point.

Example (continued).

At our first exchange operation, we see that the only feasible exchange is the replacement of $x_2 = 1, x_1 = 0$ by $x_2 = 0, x_1 = 1$. Now,

$$\begin{aligned}\Delta_{21}(\mathbf{x}) &= c_2 - c_1 + (c_{23} - c_{13})x_3 + (c_{24} - c_{14})x_4 \\ &= 5 - 3 + (2 - 6) + (6 - 10) \\ &= -6 < 0\end{aligned}$$

Thus we obtain a new point $\mathbf{x} = (1, 0, 1, 1)$ with value 29. Since we can no longer do exchanges nor fill-ups, this is our heuristic solution.

Our entire heuristic algorithm is summarized in Figure 1.

3. VARIABLE FIXATION BY LAGRANGIAN, ORDER AND PAIRING (FLOP)

When developing a branch and bound algorithm to solve a problem, one must decide upon a basic approach to undertake. There are, in general, two such approaches from which to choose: either search deep into the tree, moving between nodes very quickly, or do as much analysis at each node as possible. The first approach results in searching through many nodes, while spending as little time in each node as possible. The second approach searches through fewer nodes, but spends much more time at each node. The ideal approach to use, if any, depends on the type of problem one is trying to solve. Our approach to the quadratic knapsack problem adopts the second approach.

In this section we shall describe a three-step procedure which will be applied in every node of the branch and bound tree. The aim of this analysis is to fix the values of some of the variables by the use of order relations, Lagrangean techniques and constraint pairing. The various ingredients of the variable Fixation by Lagrangean, Order and Pairing(FLOP) method will be described below.

Throughout this section we shall assume that, at each node, the set of variables are partitioned into three subsets: the subset P of variables fixed to 1, the subset N fixed to 0, and a subset F of remaining, or "free", variables. We shall denote the subproblem defined at the

Downloaded by [McMaster University] at 05:07 09 June 2016

```
$F = \{1, \dots, n\}$
$N = P = \emptyset$
$\hat{b} = b$
for $i = 1, \dots, n$
    $c_i^* = c_i + \frac{1}{2}\sum_{j \neq i} c_{ij}$
While $F \neq \emptyset$ do
    Find $c_p^*/a_p = \max\{c_i^*/a_i : i \in F\}$
    set $P = P \bigcup \{p\}$,  $F = F - \{p\}$,  $\hat{b} = \hat{b} - a_p$
    set $N0 = \{k \in F : a_k > \hat{b}\}$
    set $N = N \bigcup N0$
    for $i \in F - N0$
        $c_i^* \mathrel{+}= \frac{1}{2}[c_{ip} - \sum_{k \in N0} c_{ik}]$
    $F = F - N0$
Wend

Repeat
    Do Exchange.
    Do Fill-up.
Until no more exchanges nor fill-ups possible.
```

Figure 1: LEX algorithm

current node by the triplet (P, N, F). Thus, at each node, we can consider the partial 0-1 vector $\mathbf{x}$ as

$$\mathbf{x}_i = \begin{cases} 1 & \text{if } i \in P \\ 0 & \text{if } i \in N \\ \mathbf{x}_i & \text{if } i \in F \end{cases}$$

Also, we shall let LB denote the value of the best known integer solution $\mathbf{x}^* = (x_1^*, \dots, x_n^*)$.

3.1 Extracting Order Relations Among Variables

In this section we develop sufficient conditions for an order relation of the form $x_i \leq x_j$ to hold in some optimal solution to (QKP). These relations will then be used both to fix variables at the given node, and to develop a branching strategy.

Suppose we are given the linear knapsack problem

$$\begin{aligned} &\text{Maximize} \quad \sum_{k=1}^{n} \beta_k x_k \\ &\text{subject to} \\ &\qquad \sum_{k=1}^{n} \alpha_k x_k \leq \gamma \\ &\qquad \mathbf{x} \in \{0,1\}^n \end{aligned}$$

where all $\beta_k, \alpha_k \geq 0$. If $\beta_i < \beta_j$ and $\alpha_i \geq \alpha_j$, then it is possible to improve upon a feasible solution $\mathbf{y}$ having $y_i = 1$ and $y_j = 0$ by the exchange $y_i = 0$ and $y_j = 1$. This remark simply shows that $x_i \leq x_j$ in *all* optimal solutions of the problem. Similarly, if $\beta_i \leq \beta_j$ and $\alpha_i > \alpha_j$, then $x_i \leq x_j$ in *some* optimal solution.

Downloaded by [McMaster University] at 05:07 09 June 2016

It was noted in [8] that, given the unconstrained quadratic $0-1$ problem

$$\max_{\mathbf{x}\in\{0,1\}^n} \sum c_k x_k + \sum c_{ij} x_i x_j, \tag{5}$$

if the second order derivative Δ_{ij} (as defined in Section 2) is negative in every $0-1$ point $\mathbf{x}$, then $x_i \le x_j$ in every optimal solution of (5). For the case of the quadratic knapsack problems, that result can be extended to the following:

Lemma 3.1

Let $\Delta_{ij}(\mathbf{x})$ be as defined in (5) for $i < j$, and $a_i \ge a_j$ in (QKP). Then:

(a) if $\Delta_{ij}(\mathbf{x}^*) < 0$ at some optimal solution $\mathbf{x}^*$, then $x_i^{**} \le x_j^{**}$ in every optimal solution $\mathbf{x}^{**}$ of (QKP).

(b) if $\Delta_{ij}(\mathbf{x}^*) = 0$ at some optimal solution $\mathbf{x}^*$, then there exists an optimal solution $\mathbf{x}^{**}$ such that $x_i^{**} \le x_j^{**}$.

Proof

(a) Let $\mathbf{x}^*$ be some optimal solution such that $\Delta_{ij}(\mathbf{x}^*) < 0$. and having $x_i^* = 1$ and $x_j^* = 0$ Define a vector $\mathbf{x}^{**}$ as follows:

$$x_k^{**} = \begin{cases} x_k^* & \text{if } k \ne i, j \\ 1 & \text{if } k = j \\ 0 & \text{if } k = i \end{cases}$$

Since $a_i \ge a_j$, $\mathbf{x}^{**}$ is feasible. Also, since $\Delta_{ij}(\mathbf{x}^*) = f(\mathbf{x}^*) - f(\mathbf{x}^{**}) < 0$ it follows that $f(\mathbf{x}^*) < f(\mathbf{x}^{**})$.

(b) Suppose $\Delta_{ij}(\mathbf{x}^*) = 0$. Then, if $\mathbf{x}^{**}$ is defined as above, $f(\mathbf{x}^*) = f(\mathbf{x}^{**})$, implying the result.

●

In our branch-and-bound algorithm, we use the results of Lemma 3.1 in two ways: fixing some of the variables at our current node and branching on a variable.

To demonstrate how Lemma 3.1 is used to fix variables, suppose that, at some node, $\Delta_{ij}(P, N, F) \le 0$ for some $i < j$; here

$$\Delta_{ij}(P, N, F) = \sum_{k\in P}(c_{ik} - c_{jk}) + c_i - c_j + \sum_{k\in F}(c_{ik} - c_{jk})x_k.$$

This conclusion can be drawn by noting that, if

$$\sum_{k\in P}(c_{ik} - c_{jk}) + c_i - c_j + \sum_{k\in F}\max(0, c_{ik} - c_{jk}) \le 0$$

then $\Delta_{ij}(P, N, F) \le 0$ for any $0-1$ assignment to x_k, $k \in F$. If follows that, if $i \in P$, then $x_j = 1$ in some optimal solution of the subproblem defined by (P, N, F). Similarly, if $j \in N$, then $x_i = 0$ in some optimal solution of the subproblem defined by (P, N, F). Stating this result precicely, we obtain the following:

Rule 1. If $\Delta_{ij}(P, N, F) \le 0$ and $x_i = 1 (x_j = 0)$, then $x_j = 1 (x_i = 0)$ in some optimal solution to the subproblem defined by (P, N, F).

To show how to use Lemma 3.1 to branch on a variable, suppose we have the following situation

$$\Delta_{15}(P, N, F) \le 0$$
$$\Delta_{16}(P, N, F) \le 0$$
$$\Delta_{23}(P, N, F) \le 0$$

where $x_1, x_2, x_3, x_5, x_6 \in F$. Obviously, if we branch on x_1 and consider first the node where $x_1 = 1$, we will be able to fix both x_5 and x_6 to 1. This is the best we can do, since no other branching gives us more variable fixations. Generalizing this idea, we have the following:

Downloaded by [McMaster University] at 05:07 09 June 2016

Rule 2. Assume that at least one implication is present. Let $A_i = \{j > i : j \in F,\ \Delta_{ij} \leq 0\}$ and $B_j = \{i < j : i \in F,\ \Delta_{ij} \leq 0\}$. Branch on the variable x_k which maximizes $\{\max\{|A_i|, |B_i|\}\}$; examine first the node $x_k = 1$ if $|A_k| \geq |B_k|$, otherwise examine the node $x_k = 0$ first.

3.2 Variable Fixation via Lagrangean Relaxation

In this section we describe how Lagrangean relaxation is used to fix variables at the current node

Let us define the Lagrangean relaxation of (QKP) as

$$\min_{\lambda \geq 0} \max_{\mathbf{x} \in B^n} \left[\sum c_k x_k + \sum c_{ij} x_i x_j - \lambda\left(\sum a_k x_k - b\right)\right]. \tag{6}$$

Since $c_{ij} \geq 0$ for $i < j$, (6) can be solved by a sequence of network flow problems. For details, see [1, 4, 6]. The method of variable fixation via Lagrangean relaxation was introduced for the linear case in [3] and for the quadratic case in [1].

Consider the current node (P, N, F). For each $k \in F$, suppose we set $x_k = \overline{x}_k^*$, where $\overline{y} = 1 - y$ for $y \in \{0, 1\}$, and consider the subproblem $(P, N, x_k = \overline{x}_k^*, F - \{k\})$. Let V_k denote the Lagrangean upper bound for this subproblem. Obviously, if V_k is strictly less than the value of the best known integer solution (LB), then the only solutions to the subproblem (P, N, F) that can possibly improve upon $\mathbf{x}^*$ has $x_k = x_k^*$. Thus, we may fix $x_k = x_k^*$ and delete k from F. Repeating this procedure for each $k \in F$ until no more variable fixations occurs, we obtain a reduction in the number of nodes our branch and bound algorithm has to examine.

# variables	Heuristic Time (in seconds)		Heuristic/Optimal (%)	
	average	worst case	average	worst case
30	0.02	0.03	99.98	99.83
40	0.03	0.07	99.95	99.53
50	0.06	0.17	99.71	98.15
75	0.12	0.40	99.81	97.96
100	0.20	0.62	99.86	98.72

Table 1: Performance of LEX

3.3 Constraint Pairing

At each node in our branch-and-bound tree, we are interested in determining whether it is possible, given the partial assignment (P, N, F), to improve upon LB. In order to assist in this, we employ the constraint pairing techniques of [10]. To use these techniques, we need a linear inequality to pair with (2). Recall problem (LKP) from Section 2. The linear function $\sum c_k^* x_k$ derived from the best linear L_2-approximation l_Q is an upper bound to (1), since $c_{ij} \geq 0$ for all $i < j$. Knowing this, we use the techniques of [10] on the following system:

$$\sum_{x \in F} c_j^* x_j \geq LB - \sum_{i,j \in P} c_{ij} \tag{7}$$

$$\sum_{x \in F} a_j x_j \leq b - \sum_{j \in P} a_j \tag{8}$$

$$x_j \in \{0, 1\}, \quad j = 1, \ldots, n$$

This system will give us conditions on when we may prune our tree, if there are no feasible $0-1$ solutions to (7) - (8). Also, the system (7) - (8) may indicate the fact that the values of certain variables may be fixed in every optimal solution of our current subproblem.

Downloaded by [McMaster University] at 05:07 09 June 2016

Downloaded by [McMaster University] at 05:07 09 June 2016

		GHS	CHM		LEX	
n	density	ratio	ratio	fix	ratio	fix
30	25	100.0	100.0	18	100.0	27
	50	100.0	97.94	12	99.98	21
	75	99.90	97.99	13	99.90	22
	100	100.0	98.55	9	100.0	26
40	25	99.42	98.70	22	99.76	31
	50	99.77	98.48	21	99.84	32
	75	99.94	97.98	9	99.96	21
	100	99.98	98.41	12	99.98	23
50	25	99.91	98.98	28	99.86	41
	50	99.74	97.33	6	99.85	31
	75	99.81	98.61	14	99.77	28
	100	99.86	98.52	10	99.84	21
75	25	99.85	99.11	33	99.93	60
	50	99.80	99.07	35	99.81	63
	75	99.85	98.84	14	99.88	48
	100	99.91	99.39	23	99.92	31
100	25	—	99.14	38	99.84	79
	50	—	99.30	32	99.96	71
	75	—	99.09	30	99.85	61
	100	—	99.16	24	99.98	63

Table 2: Comparison of Heuristics

4. COMPUTATIONAL RESULTS

In this section we present computational results obtained using the FLOP algorithm described in Sections 2 and 3. We shall present results showing the efficiency of the heuristic algorithm LEX, as well as results comparing it to other known heuristics. Also, we shall give results that demonstrate the importance of each of the three steps of FLOP described in Section 3, in terms of the increase in the number of nodes to be searched when one of the three steps is left out. We shall also give results that compare the FLOP algorithm to two known methods, both in the number of nodes examined and in CPU times. These results will be given on two sets of data.

We note that all examples were run on a SPARCstations 1+ and a SPARCserver 1000 with two processors. All algorithms were written in C.

Unless otherwise stated all problems generated in this section have the following characteristics: all coefficients c_k, c_{ij} and a_k were integers uniformly distributed between 1 and 100. b was an integer uniformly distributed between $0.5\sum a_k$ and $0.9\sum a_k$.

4.1 Heuristic Results

In this section we shall give average and worst case running times and "closeness" to the optimal solution of a set of problems. We shall then compare LEX to other known heuristics previously given in the literature.

Table 1 summarizes the results of 100 test problems grouped in 5 classes of 20 problems each. All problems had 100% dense quadratic functions, meaning that all terms were present in our objective function.

The second and third columns give results on the CPU times for our algorithm, while the fourth and fifth columns show how close the heuristic solution is to the optimal solution.

Table 2 compares the solution obtained from LEX to those given by heuristics found in Gallo, Hammer, Simeone (GHS) [5] and Chaillou, Hansen, Mahieu (CHM) [1]. Each row in the table gives the average results of 10 test problems. The "fix" column gives the number of variables fixed by Lagrangean relaxation before any branch-and-bound algorithm begins. The algorithm described in [5] does not include this step, therefore there is no "fix" column for GHS. Finally, the GHS algorithm was not run on problems of size $n = 100$, because of time limitations.

Note that LEX initially fixes more variables than the CHM method. This will be important in comparing branch-and-bound methods, since we will be starting with fewer variables than the others.

The most important conclusion from the computational results given in Table 2 is that the solution provided by LEX does as well, and almost uniformly better, than the other heuristic solutions.

# variables	30	40	50	75	100
Reduction in # of branch nodes	54%	66%	66%	69%	88%

Table 3: Effect of Using Order Relations in FLOP

# variables	30	40	50	75	100
Reduction in # of branch nodes	74%	76%	78%	74%	78%

Table 4: Effect of Using Lagrangian Relaxation in FLOP

# variables	30	40	50	75	100
Reduction in # of branch nodes	0.5%	1.7%	1.8%	4.1%	2.3%

Table 5: Effect of Using Constraint Pairing in FLOP

4.2 Importance of FLOP's Three Components

It is natural to ask whether the inclusion of each of the three components of the FLOP method is well justified. In order to answer this question the FLOP method, as described in Section 3, is compared with three similar procedures, each of which differs from FLOP in having exactly one of the three components of FLOP removed.

Tables 3 – 5 show the average percentage of reduction found when, after test problems were first solved using two of the three steps, we resolve the problems using all three steps. For instance, Table 3 says that for test problems on 30 variables, if we first solve the problems using only constraint pairing and variable fixation via Lagrangean relaxation, we can get a 54% reduction in the number of nodes searched by simply adding in our order relation techniques.

Downloaded by [McMaster University] at 05:07 09 June 2016

We note that all problems solved were of 100% density for the quadratic objective function.

Note that each step makes a contribution to the reduction of nodes in our search tree, which can be very significant (as in the case with Lagrangean relaxation and order relations) or more modest (as in the case of constraint pairing).

4.3 Comparison of Branch and Bound Algorithms

In this section we present results summarizing some comparison of the FLOP algorithm to two previously known methods, GHS [5] and CHM [1]. Those methods include specific techniques for building initial solutions. The FLOP method starts with a solution obtained using the LEX algorithm.

The average results of ten test runs for various numbers of variables and various densities are presented in Table 6. Table 7 gives the average running times of each method in CPU seconds. Times are given as hour:minute:second. A node limit of 20000 was set, as well as a time limit of 2 hours, 30 minutes. Thus, some of the numbers in the GHS and CHM columns may have been higher had we increased our limits to other plateaus.

Parameters			Nodes Examined		
n	m	density	GHS	CHM	FLOP
30	112	25	114	132	4
	232	50	636	531	14
	349	75	398	574	12
	465	100	388	1349	20
40	205	25	1074	285	17
	412	50	1589	509	14
	614	75	2264	2030	37
	820	100	752	1749	26
50	324	25	6197	442	13
	635	50	10792	1975	52
	953	75	8024	3138	39
	1275	100	5903	14882	182
75	709	25	11599	1707	33
	1427	50	14569	7251	63
	2139	75	16638	5670	70
	2850	100	15916	7090	118
100	1243	25	-	3783	26
	2547	50	-	5478	60
	3793	75	-	16308	204
	5050	100	-	12508	159

Table 6: Node comparison of branch-and-bound methods

The most significant conclusion of these computational experiments is that, in every case, the number of nodes examined by FLOP represents a very small fraction of those examined by GHS and CHM. Moreover, we also note that FLOP's running times are also significantly better than those of GHS and CHM. This is a side benefit, since we were aiming at the reduction in the number of nodes in the search tree.

Let us note that, on the average, FLOP uses only 2.26% of the nodes and 35.2% of the time used by the best of the two methods it was compared. Even in the worst case, in which it uses 5.96% of the nodes and needs 92.77% of the time, FLOP's advantages are evident.

Downloaded by [McMaster University] at 05:07 09 June 2016

Parameters		Time (Seconds)		
n	density	GHS	CHM	FLOP
30	25	3	3	1
	50	14	14	3
	75	9	13	4
	100	9	31	8
40	25	40	14	6
	50	1:03	25	6
	75	1:35	1:31	22
	100	30	1:14	21
50	25	6:44	30	11
	50	12:08	3:48	1:00
	75	8:59	3:57	47
	100	6:23	16:44	5:49
75	25	40:40	10:06	1:13
	50	40:08	34:04	1:41
	75	48:54	26:00	6:29
	100	49:25	24:44	17:15
100	25	-	53:38	2:13
	50	-	28:47	2:49
	75	-	58:06	7:49
	100	-	1:32:48	23:48

Table 7: CPU time comparison of branch-and-bound methods

Downloaded by [McMaster University] at 05:07 09 June 2016

Up to this point, we used different starting points for the branch and bound methods; to be more specific, for each algorithm we used the intialization method proposed by its authors. In order to avoid any possible advantage of the FLOP method resulting from the use of a highly efficient initialization procedure (LEX), we ran a second set of experiments in which each of the three exact methods used the same initial solution provided by the LEX procedure. 125 problems were run, grouped in 6 classes of 25 problems each. Each class is of varying size, and all problems in the class are of full (100%) density. We note that, unlike the previous experiments, we set a time limit of around 2 hours and 30 minutes, but we did not set a node limit. This was done in order to get a better measure of the performance of the three algorithms. For these experiments, the average node count and average CPU time for each class is given in Tables 8 and 9.

As before, we note that the number of nodes examined by the FLOP method represents an extremely small fraction of those examined by GHS and CHM. Moreover, the difference becomes even more pronounced for larger problems, in which FLOP examines less that 3% of the nodes examined by the other methods. In addition, FLOP's average CPU time is generally smaller in these experiments than the other methods. Again, the difference in CPU time is even more obvious for larger problems.

In conclusion, we remark that it appears that the LEX algorithm compares favorably with previously known heuristics for solving (QKP), and that the FLOP algorithm represents a significant improvement over other branch and bound algorithms in terms of both the number of nodes examined and CPU running times.

n	Nodes Examined		
	GHS	CHM	FLOP
20	111	147	10
30	205	240	10
40	879	1244	23
50	3151	10133	52
75	38662	54920	290
100	62908	23739	517

Table 8: Node comparison of branch-and-bound methods using LEX heuristic

n	Time (seconds)		
	GHS	CHM	FLOP
20	1	1	1
30	2	2	2
40	16	19	11
50	1:16	4:34	50
75	30:03	44:49	11:41
100	2:06:16	1:01:05	32:47

Table 9: CPU time comparison of branch-and-bound methods using LEX heuristic

ACKNOWLEDGEMENTS

The authors acknowledge fruitful discussions with Professor Yves Crama and the partial support provided by the Office of Naval Research through the grants numbered N00014-92-J1375, N00014-92-J4083 and N00014-93-1-0891, by the Air Force Office of Scientific Research through grant number F49620-93-1-0041, and by the NSF through grant number NSF-INT-83-21811.

BIBLIOGRAPHY

[1] Chaillou, P., Hansen, P. and Mahieu, Y., "Best Network Flow Bounds for the Quadratic Knapsack Problem," *Lecture Notes in Mathematics* 1403(1986), 226-235.

[2] Crama, Y. and Mazzola, J.B., "Valid Inequalities and Facets for a Hypergraph Model of the Nonlinear Knapsack and the FMS Part Selection Problems," *Annals of Operations Research* 58(1995), 99-128.

[3] Dembo, R.S. and Hammer, P.L., "A Reduction Algorithm for Knapsack Problems," *Methods of Operations Research*, 36(1980), 49-60.

[4] Gallo, G., Grigoriadis, M. and Tarjan, R., "A Fast Parametric Maximum Flow Algorithm and Applications," *SIAM Journal on Computing*, 18(1989), 30-55.

[5] Gallo, G., Hammer, P.L., and Simeone, B., "Quadratic Knapsack Problems," *Mathematical Programming* 12(1980), 132-149.

[6] Gallo, G. and Simeone, S., "On the Supermodular Knapsack Problem," *Mathematical Programming*, 45(1988), 295-309.

[7] Geoffrion, A.M., "Lagrangean Relaxation for Integer Programming," *Mathematical Programming Study*, 2(1974), 82-114.

[8] Hammer, P.L. and Hansen, P., "Logical Relations in Quadratic 0-1 Programming," *Revue Roumaine de Math. Pures et Appl.* 26(1981), 421-429.

[9] Hammer, P.L. and Holzman, R., "Approximations of Pseudo-Boolean Functions; Applications to Game Theory," *ZOR - Methods and Models of Operations Research*, 36(1992), 3-21.

Downloaded by [McMaster University] at 05:07 09 June 2016

[10] Hammer, P.L, Padberg, M., Peled, U., "Constraint Pairing in Integer Programming," *INFOR* 13(1975), 68-81.

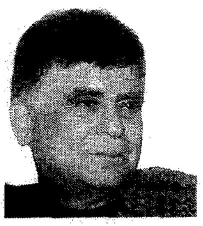
[11] Hammer, P.L. and Simeone, B., "Order Relations of Variables in $0-1$ Programming", *Annals of Discrete Mathematics* (North-Holland, 1987)

[12] Johnson, E.L., Mehrotra, A. and Nemhauser, G.L, "Min-Cut Clustering," *Mathematical Programming*, 62(1993), 133-151.

[13] Körner, F., "A New Bound for the Quadratic Knapsack Problem and its Use in a Branch and Bound Algorithm," *Optimization*, 17(1986), 643-648.

[14] Lengauer, T., *Combinatorial Algorithms for Integrated Circuit Layout* (Wiley, Chichester, 1990).

[15] Peterson. C.C., "A Capital Budgeting Heuristic Algorithm using Exchange Operations," *AIIE Transactions* 6(1974), 143-150.



Peter Hammer received his Doctorate in Mathematics at the University of Bucharest, Romania, and the Honorary Doctorate degree from the Swiss Federal Institute of Technology, in 1966 and 1986, respectively. Currently, he is the director of RUTCOR, Rutgers University's Center for Operations Research, New Brunswick, NJ, and is the Editor-in-Chief and Founder of *Discrete Mathematics*, *Discrete Applied Mathematics*, *Annals of Discrete Mathematics*, and *Annals of Operations Research*. He has done extensive research on the theory and applications of Boolean and pseudo-Boolean functions in operations research and other areas. Currently his main interest is in the theory and applications (in medicine, economics, technology, etc.) of the new methodology Logical Analysis of Data, which he and his collaboratos have introduced and developed.

David Rader, Jr. is currently working towards his Ph. D. in Operations Research at RUTCOR after receiving his undergraduate degree in Mathematics at the University of Richmond, VA. His current research interests include nonlinear $0-1$ functions, computational integer programming and applications of discrete mathematics in industry.

Downloaded by [McMaster University] at 05:07 09 June 2016