INFORMATICA, 1992, Vol.3, No.2, 225-240

HEURISTICS WITH A WORST-CASE BOUND FOR UNCONSTRAINED QUADRATIC 0-1 PROGRAMMING

Gintaras PALUBECKIS

Department of Practical Informatics
Kaunas University of Technology
3028 Kaunas, Studentų St. 50, Lithuania

Abstract. In this paper, we present two heuristics for solving the unconstrained quadratic 0–1 programming problem. First heuristic realizes the steepest ascent from the centre of the hypercube, while the second constructs a series of solutions and chooses the best of them. In order to evaluate their worst-case behaviour we define the performance ratio K which uses the objective function value at the reference point $x=1/2$. We show for both heuristics that K is bounded by 1 from above and this bound is sharp. Finally, we report on the results of a computational study with proposed and local improvement heuristics.

Key words: quadratic 0-1 programming, heuristics, performance ratio, local improvement algorithms.

1. Introduction. The unconstrained quadratic 0-1 programming problem is formulated as

$$\max f(x) = x^T Q x + C^T x, \tag{1}$$

$$\text{s.t. } x \in \{0,1\}^n, \tag{2}$$

where $Q = (q_{ij})$ is an $n \times n$ upper triangular matrix having zero diagonal and $C = (c_i)$ is a column n-vector. This model has a variety of applications in economics, operation research, physics, design automation and other areas (some examples are pointed out by Hansen (1979), Gallo, Hammer and Simeone (1980), Kőrner and Richter (1982), Gulati, Gupta and Mittal (1984)). A number of exact algorithms for solving (1), (2) have been developed so far, e.g., McBride and Yormark (1980), Carter (1984), Gulati, Gupta and Mittal (1984), Barahona, Jünger and Reinelt (1989). However, little is known about heuristic solution methods despite of the fact that the exact algorithms may consistently succeed in getting a solution and proving its optimality in a reasonable amount of time for the problems with at most 50 – 60 variables only (Barahona, Jünger and Reinelt, 1989). The most natural heuristic is the local improvement algorithm given in Gulati, Gupta and Mittal (1984). The main operation of it consists in the evaluation of the change in the value of some variable from 0 to 1 or vice versa. Another example is a greedy heuristic which starts at $x = 0$ and forces to 1 at each step one or two variables maximizing the increase in the value of f. Those heuristics and a more powerful local improvement algorithm as well were investigated by Palubeckis (1989, 1990). Hennet (1982) suggests a heuristic algorithm extracted from the dynamic programming scheme for the problem (1), (2) with one additional requirement that the sum of variables cannot exceed a prescribed limit.

It should be mentioned also that to date no polynomial-time algorithm is known for computing a local optimum for (1), (2). In fact, this problem is shown to be PLS-complete (Pardalos and Jha, 1992).

In this paper, we propose two alternative heuristics for solving the problem (1), (2). We adopt as a performance ratio a new solution quality measure (yet mentioned in Palubeckis, 1990) which uses the value of f at the reference point $x = 1/2$. We show for both heuristics that the ratio is bounded by 1 from above and this bound is sharp. In Section 4, computational results are reported comparing both heuristics against the local improvement algorithms on two sets of randomly generated problems with up to 300 variables.

2. Measuring the quality of a solution. A commonly used criterion for the evaluation of the solutions whenever applied to our problem is $K'(f,x) = (f(x^0) - f(x))/f(x^0) = 1 - f(x)/f(x^0)$, where x and x^0 are any and, respectively, optimal 0-1 n-vectors. However, following the definition from Zemel (1981) we may conclude that this measure is not proper for (1), (2), since it cannot guarantee that $K'(f,x) = K'(f',x')$ for any f,x and for any f',x' obtained from f,x by complementing some variables, i.e., replacing some x_i with $1-x_i$ and releasing a constant appearing in the expression for f' (in fact, we have $f(x) = f'(x') + e$, where e is obtained by summing some of q_{ij}, c_i). Yet the above condition is, obviously, satisfied by an alternative measure $K(f,x) = (f(x^0) - f(x))/(f(x^0) - f(1/2))$. Moreover, K also meets all other conditions from Zemel (1981) for a measure to be proper. On the other hand, K is not free from drawbacks too. The main criticism levelled at K is the lack of sensitivity for some types of problems. Consider, for instance, the problem of finding a maximal clique in a graph. It, clearly, may be put in the form (1), (2). Suppose that an n-vertex graph has an edge set E of cardinality d_1n^2, a maximal clique consisting of bn^2 edges, and d_2n^2 nonadjacent pairs of vertices. Here $d_1, d_2, b, b \leqslant d_1, d_1 + d_2 = (n-1)/(2n)$, are some positive constants. By choosing $c_i = 0$ for all i, $q_{ij} = 1$, if $(i,j) \in E$, and $q_{ij} = -cn$ with an appropriate constant $c < 1$, if $(i,j) \notin E$, (observe that $q_{ij} = -c$ does not work) we reduce a problem to (1), (2) with $f(1/2) = -n^3(cd_2 - d_1/n)/4 = -dn^3$, where d is a positive constant for sufficiently large n. For any edge (which gets a very bad solution to (1), (2)) of a graph, we have

$$K = (bn^2 - 1)/(bn^2 + dn^3) \to +0,$$

as $n \to \infty$. Hence, whenever estimated by K this bad solution is almost equally good as an optimal one. The lack of sensitivity of K was also observed for the second set of test problems used in our computational experiments (see Section 4). For those problems, $-f(1/2)/f(x^0)$ is about 9. In both examples, the measure K' seems to be more appropriate.

One of the tasks associated with K is to look for the heuristics that for any f would produce x with

$$K(f,x) < 1. \tag{3}$$

Two heuristics of such a type are described in the following Section. In order to evaluate the performance of some heuristic A for (1), (2) we use the worst-case performance ratio of A, denoted $K_n(A)$, which is defined for each integer n to be the maximum of $K(f,x)$ over all n variable functions f and corresponding solutions x delivered by A for (1), (2) with this f.

3. Heuristics. A natural way to find a 0-1 vector satisfying (3) is to start at the centre of the n-dimensional hypercube and compel the variables to equal 0 or 1 one at a time, selecting at each step that variable which maximizes the increase in the value of f. This steepest ascent concept is realized in the following algorithm (referred to as $A1$ later on).

Step 1. Let $I = \{1,2,\ldots,n\}$, $a_i = c_i$ and $b_i = \sum_{j=1}^{i-1} q_{ji} + \sum_{j=i+1}^{n} q_{ij}$, $i \in I$.

Step 2. Let $k_i = 2a_i + b_i$ and $k'_i = a_i$, $i \in I$. For each $i \in I$, if $k_i > 0$, or $k_i = 0$ and $k'_i \geqslant 0$, set $v_i = 1$; otherwise set $k_i = -k_i$, $k'_i = -k'_i$, $v_i = 0$. Find $j \in I$ such that $k_j \geqslant k_i$ for all $i \in I$ and $k'_j \geqslant k'_i$ for all $i \in \{l \mid l \in I,\ k_l = k_j\}$.

Step 3. Fix $x_j = v_j$. Delete j from I. If $I = \varnothing$, go to Step 4. Otherwise for each $i \in I$, subtract q_{ij} (or q_{ji}, if $j < i$) from b_i and, if $x_j = 1$, add q_{ij} (or q_{ji}) to a_i. Return to Step 2.

Step 4. Stop; the solution to (1), (2) is $x_{A1} = (x_i)$.

This algorithm, obviously, runs in time $O(n^2)$. Its worst-case behaviour is characterized by the following statement.

PROPOSITION 1. For $n > 2$ and any fixed positive number ε,

$$1 - 1/(2n-3) - \varepsilon < K_n(A1) < 1.$$

Proof. (i) Denote by u_i the value of k_j during the i-th arrival at Step 2 of $A1$. Changing the value of x_j from $1/2$ to v_j in Step 3 increases f by $k_j/4$. Therefore

$$f(x_{A1}) = f(1/2) + \sum_{i=1}^{n} u_i/4.$$

Since $u_i \geqslant 0$ for all i, and $u_1 > 0$ or $u_2 > 0$ (if $k_i = 0$, $i = 1,2,\ldots,n$, initially), it follows that $f(x_{A1}) > f(1/2)$. This establishes the upper bound.

(ii) Let $m = n - 1$. Define $\alpha = \varepsilon/3$, $\beta = (\varepsilon + 2)/(m + 1)$. Consider the problem (1), (2) with f having $q_{1i} = (2 - \beta)/m$, $i = 2, \ldots, n$, $q_{ij} = 2\alpha/[m(m-1)]$, $i = 2, \ldots, m$, $j = i + 1, \ldots, n$, $c_1 = -1$, and all other c_i equal to 0. On the first iteration of $A1$, $k_1 = \beta > k_i = 2\alpha/m + (2 - \beta)/m$, $i \in \{2, \ldots, n\}$, and thus the variable x_1 is chosen and forced to 0. All the remaining variables are fixed at 1. Therefore $f(x_{A1}) = \alpha$, while $f(x^0) = 1 + \alpha - \beta$. Furthermore $f(1/2) = (\alpha - \beta)/4$. Hence

$$K(f, x_{A1}) = (4 - 4\beta)/(4 + \varepsilon - 3\beta) = 1 - (2 + \varepsilon m + 2\varepsilon)/(4m - 2 + \varepsilon m - 2\varepsilon)$$

from what the lower bound follows.

Remark. Note that the use of the additional criterion k_i' in $A1$ has, in fact, no impact on the worst-case behaviour of $A1$ (with respect to K). This rule of breaking the ties enhances the efficiency of $A1$ on random and, hopefully, real problems.

Now we approach the problem (1), (2) in a different way. Define the r-th shell of the hypercube to be the set of vertices x with exactly r coordinates equal to 1. Let F_r, $r \in \{0, 1, \ldots, n\}$, be an average value of f on the r-th shell. The underlying idea of the second algorithm is to find for each r a solution $x(r)$ that belongs to the r-th shell and satisfies $f(x(r)) \geqslant F_r$, and pick out the best of them as an output.

For some $r > 0$ assume that some $s - 1 < r$ variables are forced to 1 while the other being free. In order to reveal a proper criterion for the choice of the s-th variable we have to explore the average $F_{r,s-1}(l)$ of f over the set of extensions of this partial assignment to those vertices of the r-th shell that have $x_l = 1$ for a free variable x_l. Let I_1 (respectively I_2) stand for the set of indices of the fixed (free) variables. Let $u_{s-1}, v_{s-1}, z_{s-1}$ be the averages over

$$\{q_{ij} | i < j,\ i \in I_1,\ j \in I_2 \text{ or } i \in I_2,\ j \in I_1\}, \quad \{q_{ij} | i < j \in I_2\}$$

and $\{c_i | i \in I_2\}$ respectively. Given any free x_l with the sums a_l', b_l of q_{il} or q_{li} with i in I_1 and I_2 respectively, we may write

$$F_{r,s-1}(l) = c + a_l' + c_l + u_s s(r - s) + v_s \binom{r-s}{2} + z_s(r - s), \qquad (4)$$

where $u_s = [u_{s-1}(s-1)(n-s+1)-a_l'+b_l]/[s(n-s)]$, $v_s = \left[v_{s-1}\binom{n-s+1}{2} - b_l\right] / \binom{n-s}{2}$, $z_s = [z_{s-1}(n-s+1)-c_l]/(n-s)$ and c is a term not depending on l. It is readily checked that the right hand side of (4) is maximized by x_l having largest $a_l' + c_l + b_l(r-s)/(n-s-1)$. This fact is exploited in the following algorithm (on occasion, referred to as $A2$).

Step 1. Choose x_{A2} to be that from $(0,\dots,0)$ and $(1,\dots,1)$, which has a larger value of f. Set $r = 1$.

Step 2. Let $a_i = c_i$ and $b_i = \sum_{j=1}^{i-1} q_{ji} + \sum_{j=i+1}^{n} q_{ij}$, $i \in I = \{1,\dots,n\}$, and set $s = 1$.

Step 3. Let $k_i = a_i + b_i(r-s)/(n-s-1)$ or $k_i = a_i$, if $s = n-1$, $i \in I$. Find $j \in I$ such that $k_j \geqslant k_i$ for all $i \in I$ and $a_j \geqslant a_i$ for all $i \in \{l | l \in I, k_l = k_j\}$.

Step 4. Set $x_j = 1$. Delete j from I. If $|I| = n - r$, force all x_i, $i \in I$, to 0 and go to Step 5. Otherwise for each $i \in I$, subtract q_{ij} (or q_{ji}, if $j < i$) from b_i and add to a_i. Set $s = s + 1$ and return to Step 3.

Step 5. If $f(x) > f(x_{A2})$, set $x_{A2} = x$. Set $r = r+1$, $I = \{1,\dots,n\}$, and, if $r < n$, return to Step 2.

Step 6. Stop with the solution x_{A2}.

The complexity of this algorithm is $O(n^3)$. Yet the upper bound on K_n remains the same as for $A1$.

PROPOSITION 2. For $n > 3$ and any fixed positive number ε,

$$1 - 1/(2n-5) - \varepsilon < K_n(A2) < 1. \tag{5}$$

Proof. (i) To settle (3) we have to show that

$$f(x_{A2}) > f(1/2). \tag{6}$$

Letting $B = \sum_{i<j} q_{ij}$, $D = \sum_i c_i$, $F = \max\{F_r | 0 \leqslant r \leqslant n\}$, we may write the following chain of inequalities

$$f(x_{A2}) \geqslant F \geqslant f(1/2) + |B|/[4(n-\gamma)] \geqslant f(1/2) = B/4 + D/2\,, \tag{7}$$

where $\gamma = 0$ for n odd and $\gamma = 1$ for n even. The first inequality follows directly from the derivation of the criterion k_i and the definition of $A2$. To prove the second we distinguish between three cases according to the signs of B and D. Denote by H the third member in (7). If $B \geqslant 0$ and $B \geqslant -D$, then $F \geqslant F_n \geqslant H$. Moreover, the last inequality is strict provided $B \neq 0$ or $D \neq 0$. If $D \leqslant 0$ and $B < -D$, then $F \geqslant F_0 > H$. Finally, let $B < 0$, $D > 0$. Then for n being even, $F \geqslant F_{n/2} = H$. For n odd, $F_{(n-1)/2} = H$, if $-B \geqslant D$, and $F_{(n+1)/2} = H$, if $-B \leqslant D$.

Now, (6) follows from the last inequality in (7) when $B \neq 0$, from the second when $B = 0$, $D \neq 0$, and from the first when $B = 0$, $D = 0$ (in this case, $f(x(1)) > 0$ or $f(x(2)) > 0$ depending on the existence of a linear part in (1)).

(ii) Let $m = n - 2$. For some positive $\alpha < \varepsilon$ consider the problem (1), (2) with f having the following nonzero coefficients: $c_1 = 1 + \alpha$, $c_2 = -m$, $q_{12} = m$, $c_i = 1$, $q_{1i} = -1$, $i = 3, \dots, n$. For any $r \in \{1, \dots, n-1\}$, the first variable selected in Step 3 is x_1. Consequently, $f(x_{A2}) = 1 + \alpha$ and

$$K = [m - (1+\alpha)]/[m - (1+\alpha)/2] = 1 - (1+\alpha)/(2m - 1 - \alpha)$$

is greater than the leftmost term in (5).

Corollary. For any instance of (1), (2), heuristic $A2$ yields a solution x_{A2} satisfying

$$f(x_{A2}) \geqslant F \quad \text{and} \quad f(x_{A2}) > \frac{1}{2^n} \sum_{x \in \{0,1\}^n} f(x).$$

Proof. From (7) and definition of F we have

$$f(x_{A2}) \geqslant F \geqslant \frac{1}{2^n} \sum_{x \in \{0,1\}^n} f(x).$$

The second relation becomes an equality only when $B = D = 0$. But in this case, $F = 0$ while $f(x_{A2}) > 0$ since $f(x(i)) > 0$ for $i = 1$ or 2 (as already established in the proof of Proposition 2).

The first estimate in Corollary is best possible for $A2$ as the following fact implies.

PROPOSITION 3. (Palubeckis, 1990, p. 98–99). If $P \neq NP$, then no polynomial-time heuristic for (1), (2) can guarantee a solution x with $f(x) > F$ (though such a solution exists).

The assertion below shows that $A1$ cannot be estimated similarly.

PROPOSITION 4. Let ε be any fixed positive number. For each $n > 3$, there exists an instance of (1), (2) such that for x_{A1} computed by $A1$

$$f(x_{A1})/F < \varepsilon.$$

Proof. The same instance as in the proof of Proposition 1 works well. Indeed, $F = F_n$ and $f(x_{A1})/F = \alpha/(1+\alpha-\beta) = \varepsilon/(3+\varepsilon-3\beta) < \varepsilon$ since $(3+\varepsilon-3\beta) > 1$ for $n \geqslant 4$.

In closing this section, we will note that the local improvement algorithms, contrary to $A1, A2$, cannot guarantee (3) to be held for any f. Denote by $L(m)$ the local improvement algorithm that changes in each step the values of at most m variables (if this leads to a better solution, of course). The above operation with exactly $k \leqslant m$ variables will be called the k-change.

PROPOSITION 5. For $n > m$ and any fixed positive number ε,

$$K_n(L(m)) > \frac{4}{3}\Big(1 - (m-1)/(3n-2m-1) - \varepsilon\Big).$$

Proof. Define $\alpha = \varepsilon/n$ and consider the problem (1), (2) with $c_i = -(m-1)/2-\alpha$, $i = 1, \ldots, n$, $c_{ij} = 1$, $i = 1, \ldots, n-1$, $j = i+1, \ldots, n$. Suppose that $L(m)$ starts at $x = 0$. Then k-changes, $k \leqslant m$, cannot improve this solution and therefore $f(x) = 0$, $f(x^0) = n(n-m)/2 - n\alpha$. Consequently

$$K = 4(n-m-2\alpha)/(3n-2m-1-4\alpha) = \frac{4}{3}\Big[1-(m-1+2\alpha)/(3n-2m-1-4\alpha)\Big]$$

and the result follows.

4. Computational results.

The main purpose of the computational experiments was to compare heuristics $A1$, $A2$ described in Section 3 against the local improvement algorithms $L(1)$, $L(2)$

perceived as the most natural and at the same time rather good heuristics for (1), (2). All the algorithms were coded in the C language and tested on two sets of randomly generated problems, using an IBM PC/AT. Initially, two versions of $L(1)$ (and $L(2)$ also) differing in the 1-change (also 2-change for $L(2)$) selection strategy were compared on 60 smaller size problems from those used in the main experiments. The first of them seeks at each step for the 1-change that maximally increases the value of f, while the other one selects the first 1-change found that leads to an improvement. In the case of $L(2)$, both versions start at each step to examine 2-changes only when all 1-changes failed to improve a solution. When applied to $x = 0$, the second version of $L(1)$ was able to produce superior solutions than the first for 35 problems and equally good for 6. For $L(2)$, the corresponding numbers were 28 and 13. So, the second version was kept as a representative of $L(i)$, $i = 1, 2$.

The test problems forming the first set are constructed similarly as in Carter (1984), Barahona, Jünger and Reinelt (1989). All the coefficients q_{ij} and c_i are integral and chosen uniformly distributed in the interval $[-100, 100]$. Note that the problems with Q having full density are most difficult for exact solution methods (see Barahona, Jünger and Reinelt, 1984). Tables 1 and 2 present averaged results for this set of problems. In Table 1, for all the algorithms except $L(1)$ the average difference between the function value achieved by an algorithm and that for $L(1)$ is given. In Table 2, the same is done with respect to $L(2)$. The couple $A1 + L(1)$, for example, denotes that $L(1)$ is applied to a solution delivered by $A1$. Yet, either of the local improvement algorithms alone has $x = 0$ initially.

Table 1 shows that the heuristic $A1$ is better than $L(2)$ on both counts – the quality of the solutions and the computing time. $L(2)$ following $A1$ further improves a solution, and for larger n this couple is less expensive than $L(2)$ applied to $x = 0$. The heuristic $A2$ turns out to be by far more computer-time consuming (what agrees with the time bound given in Section 3). Nevertheless, for the first set of problems the performance of $A2$ is inferior to that of $A1$.

Table 1. Performance and CPU time comparison among $L(1)$, $L(2)$ and $A1$ for the first set of problems (Averages over 10 test problems for each n)

Heuristic	Dimension n							
	70	100	130	160	190	220	250	300
Function value								
$L(1)$	11271	18041	29028	36819	47604	61143	69578	94188
$L(2)$	+43	+9	+41	+150	+82	+210	+168	+244
$A1$	+226	+36	+173	+361	+309	+444	+609	+356
$A1 + L(1)$	+255	+150	+228	+483	+564	+576	+930	+782
$A1 + L(2)$	+270	+152	+257	+496	+606	+630	+985	+818
CPU time (sec.) under IBM PC/AT								
$L(1)$	2.6	5.9	10.9	17.1	25.6	38.0	49.8	75.0
$L(2)$	7.9	16.2	29.1	48.0	69.7	100.9	130.4	202.6
$A1$	4.8	9.8	16.5	24.9	35.1	47.0	60.5	86.8
$A1 + L(1)$	6.0	12.0	20.2	30.2	42.8	56.9	73.5	105.9
$A1 + L(2)$	9.4	18.7	31.1	47.3	65.9	90.5	115.2	163.8

Table 2. Performance and CPU time comparison among $L(2)$, $A1$ and $A2$ for smaller size problems from the first set (Averages over 10 test problems for each n)

Heuristic	Dimension n					
	50	60	70	80	90	100
Function value						
$L(2)$	6275	8703	11314	12957	17032	18050
$A1$	+106	−4	+183	+169	+76	+27
$A1 + L(2)$	+118	+33	+227	+257	+109	+143
$A2$	+77	−6	+157	+213	−52	−32
$A2 + L(2)$	+121	+38	+205	+259	+77	+130
CPU time (sec.) under IBM PC/AT						
$L(2)$	3.5	5.8	7.9	10.5	13.0	16.2
$A1$	2.5	3.5	4.8	6.3	8.0	9.8
$A1 + L(2)$	4.8	6.5	9.4	11.6	15.8	18.7
$A2$	75.8	130.9	207.7	310.2	441.5	605.2
$A2 + L(2)$	78.0	134.0	212.1	315.4	450.2	614.3

The algorithms were also tested on the special problems produced by a generator described in Palubeckis (1990). Along with Q and C defining an instance of (1), (2) it delivers also a solution which is known to be provably optimal for this instance. The elements of randomness fitted in the generator allow to produce a sequence of different problems for fixed n. The second set of test problems is constructed using the generator with the following values of parameters (see Palubeckis, 1990): $h = 100$, $\lambda = 101$, $\delta = 1$, $t_0 = 0.1$, $m_x = 5$, if $n \leqslant 100$, $m_x = 6$, if $n = 132, 156$, $m_x = 7$, if $n = 182, 210$, $m_x = 8$, if $n \doteq 256$, $m_x = 9$, if $n = 306$. Tables 3 and 4 summarize the computational results on these problems. In each table, the row corresponding to $f(1/2)$ makes it possible to calculate the value of the performance criterion K. However, to avoid doubling of information those values for different algorithms are not shown. Clearly, in most cases K is very close to 0.

As can be seen from Table 3, the heuristic $A1$ consistently outperforms $L(2)$ on the second set of problems and coupled with $L(2)$ produces the solutions that are not far from the optimum. For smaller n (and we believe for larger as well), the heuristic $A2$ beats $A1$ in performance and when followed by $L(2)$ in about 93% of the cases leads to an optimal solution. It seems that the main reason of such an excellent behaviour of $A2$ lies in the nature of the test problem generator. The problems constructed by it have all near optimal (and optimal, of course) solutions in the $\lfloor n/2 \rfloor$-th and neighbouring shells (despite of the fact that $F = F_0 = F_n = 0$ for chosen h, λ).

5. Conclusions. We have presented and tested two heuristics for solving the unconstrained quadratic 0-1 programming problem (1), (2). Both of them for any instance of (1), (2) meet the performance bound given by (3). An open problem is to devise an efficient algorithm with a better worst-case bound, say $K_n < 0.5$, or to show that no polynomial-time algorithm for (1), (2) can have unless $P = NP$, a performance ratio $K_n < 1 - \varepsilon$ for small positive ε.

We conclude from the results of experimentation that the steepest ascent heuristic shows robust behaviour and jointly with the

Table 3. Performance and CPU time comparison among $L(1)$, $L(2)$ and $A1$ for the second set of problems (Averages over 10 test problems for each n)

Heuristic	Dimension n					
	132	156	182	210	256	306
Function value						
Optimal	4356	6084	8281	11025	16384	23409
$L(1)$	1954	1933	2245	2891	6369	9014
$L(2)$	2634	2980	3357	6340	8316	11878
$A1$	3169	5319	6142	8695	13787	18606
$A1 + L(1)$	3508	5784	7153	9441	15047	20551
$A1 + L(2)$	3838	6050	7662	10074	15887	22288
$f(1/2)$	−38877	−54405	−74115	−98728	−146880	−209962
CPU time (sec.) under IBM PC/AT						
$L(1)$	1.0	1.0	1.1	1.7	4.1	5.8
$L(2)$	11.4	16.4	21.0	42.9	48.4	88.1
$A1$	16.3	22.7	30.8	41.2	61.1	87.3
$A1 + L(1)$	18.2	26.0	34.7	46.0	69.1	97.6
$A1 + L(2)$	30.5	46.2	60.0	82.9	124.7	196.1

Table 4. Performance and CPU time comparison among $L(2)$, $A1$ and $A2$ for smaller size problems from the second set (Averages over 10 test problems for each n)

Heuristic	Dimension n					
	50	60	70	80	90	100
Function value						
Optimal	625	900	1225	1600	2025	2500
$L(2)$	612	659	894	973	1032	1035
$A1$	535	762	917	1376	1576	2198
$A1 + L(2)$	605	876	1151	1514	1868	2475
$A2$	616	857	1151	1592	1929	2383
$A2 + L(2)$	625	883	1225	1600	2025	2480
$f(1/2)$	−5487	−7965	−10842	−14220	−17998	−22275
CPU time (sec.) under IBM PC/AT						
$L(2)$	2.0	2.2	3.0	3.9	4.2	5.1
$A1$	2.4	3.4	4.6	6.1	7.6	9.4
$A1 + L(2)$	4.2	5.9	8.5	11.1	13.9	18.0
$A2$	77.1	130.8	208.5	309.7	440.2	602.6
$A2 + L(2)$	78.9	133.4	212.0	313.8	446.0	610.1

simple local improvement algorithm produces good solutions with modest computational requirements. We note that the local improvement algorithms applied to a fairly good starting solution x achieve in most cases a better final one than that obtained when algorithms are applied to $x = 0$.

In future, the experiments may be continued in a way of testing some new, perhaps more sophisticated, heuristics and using additional sets of instances of (1), (2), e.g., those constructed by Pardalos (1991).

REFERENCES

Barahona, F., M.Jünger and G.Reinelt (1989). Experiments in quadratic 0-1 programming. *Mathematical Programming,* **44**(2), 127–137.

Carter, M.W. (1984). The indefinite zero-one quadratic problem. *Discrete Applied Mathematics,* **7**(1), 23–44.

Gallo, G., P.L.Hammer and B.Simeone (1980). Quadratic knapsack problems. *Mathematical Programming Study,* **12**, 132–149.

Gulati, V.P., S.K.Gupte and A.K.Mittal (1984). Unconstrained quadratic bivalent programming problem. *European J. of Operational Research,* **15**(1), 121–125.

Hansen, P. (1979). Methods of nonlinear 0-1 programming. *Annals of Discrete Mathematics,* **5**, 53–70.

Hennet, J.C. (1982). Resolution of a quadratic combinatorial problem by dynamic programming. *Lecture Notes in Control and Information Sciences,* **38**, 455–464.

Körner, F., and C.Richter (1982). Zur effektiven lösung von booleschen, quadratischen optimierungsproblemen. *Numerische Mathematik,* **40**(1), 99–109.

McBride, R.D., and J.S.Yormark (1980). An implicit enumeration algorithm for quadratic integer programming. *Management Science,* **26**(3), 282–296.

Palubeckis, G. (1989). Analysis of algorithms in quadratic unconstrained 0-1 optimization. *Litovskij Matematicheskij Sbornik,* **29**(2), 336–346 (in Russian).

alubeckis, G. (1990). Quadratic 0-1 optimization. *Informatica,* **1**(1), 89–106.

ardalos, P.M. (1991). Construction of test problems in quadratic bivalent programming. *ACM Transactions on Math. Software,* **17**(1), 74–87.

Pardalos, P.M., and S.Jha (1992). Complexity of uniqueness and local search in quadratic 0-1 programming. To appear in *Operations Research Letters*, 11(2).

Zemel, E. (1981). Measuring the quality of approximate solutions to zero-one programming problems. *Math. of Operations Research*, 6(3), 319–332.

Received March 1992

G. Palubeckis received the degree of Candidate of Technical Sciences from the Kaunas Polytechnic Institute, Kaunas, Lithuania, in 1987. His major research interests are in graph theory, combinatorial optimization and computer-aided design.