

Working Paper Series

**HEARIN CENTER
FOR
ENTERPRISE SCIENCE**

HCES-09-05

A Hybrid Multi-Exchange Local Search for Unconstrained Binary Quadratic Program

By

Wei Liu^a

Dawn Wilkins^a

Bahram Alidaee



The University of Mississippi

Interim Director, Bahram Alidaee
School of Business Administration
The University of Mississippi
Post Office Box 1848
University, MS 38677-1848
(662) 915-5820
<http://hces.bus.olemiss.edu>

A Hybrid Multi-Exchange Local Search for Unconstrained Binary Quadratic Program

Wei Liu^a
Dawn Wilkins^a

^aComputer Science Department
Engineering School
The University of Mississippi

Bahram Alidaee^b
^bHearin Center for Enterprise Science
The School of Business Administration
The University of Mississippi

^bCorrespondent author

A Hybrid Multi-Exchange Local Search for Unconstrained Binary Quadratic Program

Abstract

This paper proposes a generic hybrid r -opt/1-opt local search algorithm for the unconstrained binary quadratic programming problem. A new algorithm which combines a simple tabu search and the proposed hybrid local search is described and experimentally compared with an algorithm that uses the 1-opt local search with the same simple tabu search and a multiple start tabu search algorithm proposed by Palubeckis (2004). Computational results are provided for the 38 largest problem instances used by Palubeckis (2004), and in addition, for 50 randomly generated problems with size from 2500 to 6000. The new tabu search algorithm using the proposed hybrid r -opt/1-opt local search performs well, as confirmed by the experimental results.

1. Introduction

The unconstrained binary quadratic programming (UBQP) problem can be formulated as

$$\begin{aligned} \text{MAX } f(x) &= \sum_{i=1}^n \sum_{j=1}^n q_{ij} x_i x_j \\ \text{s.t. } & x_i \in \{0, 1\}, \quad i = 1, \dots, n. \end{aligned}$$

where q_{ij} , (for $i, j \in N = \{1, 2, \dots, n\}$), are entries of a given $n \times n$ symmetric rational matrix Q and x_i , $i = 1, \dots, n$, are elements of an n -vector x . Since $x_i^2 = x_i$ we can take the diagonal element of Q out and represent it as linearly as $q_{ii} = q_i$, $i = 1, \dots, n$. Furthermore, since Q is symmetric, the UBQP can be presented as follows which has the advantage of faster implementation and reduced memory. Thus, we only consider Q to be an upper triangular matrix.

$$\begin{aligned} \text{MAX } f(x) &= \sum_{i=1}^n q_i x_i + \sum_{i=1}^n \sum_{j>i}^n q_{ij} x_i x_j \\ \text{s.t. } & x_i \in \{0, 1\}, \quad i = 1, \dots, n. \end{aligned}$$

UBQP is a unified model for a variety of combinatorial optimization problems (Kocjenberger *et al* 2004). Numerous hard problems have been formulated as UBQP problems, including VLSI design (Barahona *et al* 1988, Boros *et al* 1999), statistical mechanics (De Simone *et al* 1995), reliability theory and statistics (Papaioannou 1977, Rao 1977), economics and finance (Hammer and Shliffer 1971, Hillier 1969), manufacturing (Alidaee *et al.* 1994, Badics and Boros 1998, Lewis *et al.* 2005, Alidaee *et al.* 2005), discrete mathematics (Fraenkel and Hammer 1984, Hammer 1977), computer aided design (Krarup and Pruzan 1978), traffic management (Gallo *et al* 1980), and molecular conformation (Phillips and Rosen 1994, Kochenberger *et al.* 2005), to name a few. Furthermore, many graph problems can be converted to UBQP including the maximum clique problem, maximum cut problem, maximum vertex packing problem, minimum covering problem, maximum independent set problem, and graph coloring problem (Pardalos and Rodgers 1992, Pardalos and Xue 1994, Kochenberger *et al.* 2005).

Hammer and Rudeanu (1968) and Hansen (1979) showed that any quadratic or linear functions with bounded integer variables and linear equation constraints can be reformulated into UBQP form. Kochenberger *et al* (2004) argued that “such reformulation into the UBQP format is not merely a representation novelty, but a unified framework of practical consequences” by casting a set of well-known problems in a variety of domains into the UBQP framework and solving them in reasonable time.

Solution approaches for UBQP have been an active research area for many years due to the computational challenge and the vast potential applications. Several exact methods such as branch-and-cut and branch-and-bound have been developed for UBQP problems

(Barahona *et al* 1989, Billionnet and Sutter 1994, Carter 1984, Helmberg and Rendl 1998, Kalantari and Bagchi 1990, Palubeckis 1995). These exact methods work well on small UBQP problems with less than 100 variables, but they quickly become excessively slow when the number of variables increases. Several meta-heuristics based on different ideas were proposed to solve large UBQP problems. These heuristics include tabu-search (Glover and Laguna 1997, Glover *et al* 1998, Glover *et al* 1999, Beasley 1998, Palubeckis 2004), scatter-search (Amini *et al* 1999), GRASP (Palubeckis and Tomkevicius 2002), simulated annealing (Alkhamis *et al* 1998, Katayama and Narihisa 2001) and genetic algorithms (Katayama *et al* 2000, Lodi *et al* 1999, Merz and Freisleben 1999, Merz and Katayama, 2004). These heuristics are often used with a local search algorithm embedded in to explore the solution space. A simple 1-opt method is the most common local search algorithm used within meta-heuristics. A 1-opt (also used in the literature as 1-flip, and 1-move) local search in the context of UBQP starts with a solution x and changes value of an element x_i to $1 - x_i$ (for an $i \in N$) until no more change (or move) is possible. Boros *et al* (2005) proposed several heuristics to improve 1-opt for UBQP problems.

In solving difficult combinatorial optimization problems, some researchers have noticed that r -opt (also known as r -move, r -flip, r -exchange, multi-exchange strategies) is able to search a larger solution space, and thus a better solution might be found using such strategy. However, most such strategies in the literature are highly problem specific (Ahuja *et al* 2002, Ahuja *et al* 2004, Deineko and Woeginger 2000, Frangioni *et al* 2004, Glover 1996, Li and Alidaee 2002, Magazine *et al* 2002, Vredeveld and Lenstra 2003, Yaguiura and Ibaraki 1999, Yaguiura and Ibaraki 2001). Merz and Freisleben (2002) and Merz and Katayama (2004) applied a special case of r -opt local search algorithm to UBQP problems. In the context of UBQP, an r -Opt strategy starts with a solution x and changes values of all elements x_i to $1 - x_i$, for $i \in S \subseteq N$, and $|S| = r$, until no more changes (or moves) are possible. However, in application such strategy can be very time consuming which depends on size of exchange (i.e., value of r). Thus, in implementation depending on size of r some promising subset of all moves is should be considered. For example see (Merz and Freisleben, 2002, and Merz and Katayama, 2004) who applied a special case of r -opt local search algorithm to UBQP problems.

The paper is organized as follows. In section 2, we propose a hybrid r -opt/1-opt local search algorithm for UBQP problems. We introduce and describe a new algorithm which combines a simple tabu search heuristic and the proposed hybrid r -opt/1-opt local search algorithm in section 3. We tested the new algorithm introduced in section 3 against an algorithm using 1-opt local search with the same simple tabu search and the best multi-start tabu search algorithm proposed by Palubeckis (2004) on the 38 largest problem instances used in (Palubeckis 2004) and 50 randomly generated problems with size from 2500 to 6000. The results are reported in section 4. We chose Palubeckis' multi-start tabu search algorithm to compare against, because it is recent, available on Internet (<http://www.soften.ktu.lt/~gintaras/binqopt.html>), and reportedly produced very good results (Palubeckis 2004). Section 5 concludes this paper and outlines our future research.

2. Local Search Algorithms for UBQP

In this section, we first briefly review the 1-opt and r -opt local search algorithms, then propose a hybrid r -opt/1-opt local search algorithm for UBQP problems.

2.1 1-opt and r -opt

Among the local search algorithms for UBQP, 1-opt is the simplest and most widely used. It can be implemented efficiently by using Theorem 2-1 proposed by Glover *et al* (1999).

Definition 2-1: Let Q be a upper triangular matrix, let $\Delta_i(x)$ be *the first derivative* of $f(x)$

with respect to x_i , then $\Delta_i(x) = q_i + \sum_{j=1}^{i-1} q_{ji}x_j + \sum_{j=i+1}^n q_{ij}x_j$.

Theorem 2-1: For $x = (x_1, \dots, x_i, \dots, x_n)$ and $x' = (x_1, \dots, \bar{x}_i, \dots, x_n)$, where $\bar{x}_i = 1 - x_i$, the change in the value of the objective function is

$$f(x') - f(x) = (\bar{x}_i - x_i)\Delta_i(x).$$

Thus, in any local optimal solution of the UQP with respect to 1-opt search we have

$$x_i = 0 \text{ iff } \Delta_i(x) < 0; \text{ and } x_i = 1 \text{ iff } \Delta_i(x) > 0, \forall i = 1, \dots, n.$$

Furthermore, after changing x_i to \bar{x}_i the update for all $\Delta_j(x)$ can be calculated as follows

$$\begin{aligned} \forall j < i, \Delta_j(x) &= \Delta_j(x) + q_{ji}(\bar{x}_i - x_i), \\ \forall j > i, \Delta_j(x) &= \Delta_j(x) + q_{ij}(\bar{x}_i - x_i), \\ j = i, \Delta_j(x) &= \Delta_j(x). \end{aligned}$$

Let $\Delta(x)$ denote the n vector with i^{th} element equal to $\Delta_i(x)$ for $i = 1, \dots, n$. $\Delta(x)$ can be initialized using formula $\Delta_i(x) = q_i + \sum_{j=1}^{i-1} q_{ji}x_j + \sum_{j=i+1}^n q_{ij}x_j$. Figure 2-1 is the pseudo-code for 1-opt local search for maximization problems.

```

Void 1-opt_local_search ( $Q, x, \Delta(x), n$ )
Begin
  Set changed = true
  Do while changed == true
    Set changed = false
    For  $i = 1, \dots, n$ 
      If ( $\Delta_i(x) > 0$  and  $x_i == 0$ ) or ( $\Delta_i(x) < 0$  and  $x_i == 1$ )
        Set changed = true
        Set  $x_i = 1 - x_i$ 
        Update  $\Delta(x)$ 
           $\forall j < i, \Delta_j(x) = \Delta_j(x) + q_{ji}(\bar{x}_i - x_i),$ 
           $\forall j > i, \Delta_j(x) = \Delta_j(x) + q_{ij}(\bar{x}_i - x_i),$ 
           $j = i, \Delta_j(x) = \Delta_j(x).$ 
        End If
      End For
    End Do
  End

```

Figure 2-1 pseudo-code for 1-opt local search for maximization problems

Definition 2-2: Let Q be an upper triangular matrix, then $(ij)^{th}$ element, q_{ij} , is equal to the second derivative of $f(x)$, (i.e., $\Delta_{ij}(x) = q_{ij}$) with respect to x_i and x_j .

If we change r elements instead of one in the vector x at each iteration, the algorithm is then called r -opt. Alidaee (2004) introduced Theorem 2-2 which can serve as the basis for any r -opt local search algorithm.

Theorem 2-2: Let $x = (x_1, \dots, x_n)$ be a given vector and let x' be a vector obtained from x by changing x_i to $\bar{x}_i = 1 - x_i$ for all $i \in S \subseteq N$, and $|S| = r$. Now, the change to the value of the objective function is

$$f(x') - f(x) = \sum_{i \in S} (\bar{x}_i - x_i) \Delta_i(x) + \sum_{\forall i, j \in S} (\bar{x}_i - x_i)(\bar{x}_j - x_j) \Delta_{ij}(x).$$

Furthermore, after changing x to x' the update for all $\Delta_i(x)$ (i.e., calculating $\Delta_i(x')$ for $i = 1, \dots, n$) can be calculated as follows,

$$\begin{aligned} \forall j \in N \setminus S \quad \Delta_j(x) &= \Delta_j(x) + \sum_{i \in S} (\bar{x}_i - x_i) \Delta_{ij}(x), \text{ and} \\ \forall j \in S \quad \Delta_j(x) &= \Delta_j(x) + \sum_{i \in S \setminus \{j\}} (\bar{x}_i - x_i) \Delta_{ij}(x). \end{aligned}$$

In Figure 2.2 we illustrate a pseudo-code of an r -opt strategy.

```

Void  $r$ -opt_local_search ( $Q, x, \Delta(x), n$ )
Begin
  Set changed = true
  Do while changed == true
    Set changed = false
    For all combinations of  $S$ 
      
$$\Delta f(x) = f(x') - f(x) = \sum_{i \in S} (\bar{x}_i - x_i) \Delta_i(x) + \sum_{\forall i, j \in S} (\bar{x}_i - x_i)(\bar{x}_j - x_j) \Delta_{ij}(x).$$

      If  $\Delta f(x) > 0$ 
        Set changed = true
        Set  $x_i = 1 - x_i$  for  $x_i \in S$ 
        Update  $\Delta(x)$ 
          
$$\forall j \in N \setminus S \quad \Delta_j(x) = \Delta_j(x) + \sum_{i \in S} (\bar{x}_i - x_i) \Delta_{ij}(x), \text{ and}$$

          
$$\forall j \in S \quad \Delta_j(x) = \Delta_j(x) + \sum_{i \in S \setminus \{j\}} (\bar{x}_i - x_i) \Delta_{ij}(x).$$

        End If
      End For
    End Do
  End

```

Figure 2-2 pseudo-code for exhaustive r -opt local search for maximization problems

Note: Theorem 2-2 is significant from an implementation point of view, because by flipping more variables, i.e., a larger value for r , we possibly explore a larger area of the solution space. Given a solution x , iteratively we make an r -opt move until no such move results in a better solution. In that case, we have reached a local optimal solution with respect to r -opt moves. Further, from a computational point of view, calculation of such an r -opt move only depends on the 1 -flip move, because the value of $\Delta_i(x)$ for all $i \in N$ is the *first derivative* of f with respect to x_i evaluated at x and $\Delta_{ij}(x) = q_{ij}$ is a constant. Note that in Theorem 2-2, updating the vector $\Delta(x) = (\Delta_1(x), \dots, \Delta_n(x))$ after one r -opt move can be done in $O(n)$ time. Also, evaluation of change in the value of the objective function can be done in $O(r^2)$ time.

Though r -opt can explore a larger area of the solution space and thus find better solutions than 1 -opt, however it is computationally much more expensive. This limits the direct use of r -opt in UBQP problems. One solution proposed by Merz and Freisleben (2002) and Merz and Katayama (2004) introduced a randomized r -opt local search algorithm which searches only a small portion of the r -opt search space. Their r -opt algorithm was based on the ideas of Lin and Kernighan (1973). In the following we present a result, Theorem 2.3, which allows limiting moves in r -opt to only a subset of promising moves.

2.2 A Hybrid r -opt/1-opt Local Search Algorithm

Before we describe the new hybrid r -opt/1-opt local search algorithm for UBQP problems, we first introduce the following theorem on which the hybrid r -opt/1-opt algorithm is based.

Theorem 2-3: Let M be the maximum absolute value of all $\Delta_{ij}(x) = q_{ij}$, for $i, j \in N$, and let $\binom{m}{n}$ be the number of combinations of m out of n elements. If 1-opt local search cannot further improve the value of $f(x)$ and x_i is an element of a subset $S \subseteq N$ where an r -flip move of elements of S improves $f(x)$, then $|\Delta_i(x)|$ must be less than or equal to $M \times \binom{2}{r}$.

Proof: From Theorem 2-2, we get for r -flip move

$$f(x') - f(x) = \sum_{i \in S} (\bar{x}_i - x_i) \Delta_i(x) + \sum_{\forall i, j \in S} (\bar{x}_i - x_i)(\bar{x}_j - x_j) \Delta_{ij}(x). \quad (1)$$

$$\text{Since this } r\text{-flip move improves the value of } f(x), f(x') - f(x) > 0 \quad (2)$$

Because 1-opt cannot further improve the value of $f(x)$, from Theorem 2-1, we have

$$\text{if } \Delta_i(x) > 0, \text{ then } (\bar{x}_i - x_i) = -1 \text{ and if } \Delta_i(x) < 0, \text{ then } (\bar{x}_i - x_i) = 1 \quad (3)$$

From (1-3), we get

$$\sum_{\forall i, j \in S} (\bar{x}_i - x_i)(\bar{x}_j - x_j) \Delta_{ij}(x) > \sum_{i \in S} |\Delta_i(x)| \quad (4)$$

If $|\Delta_i(x)| > M \times \binom{2}{r}$, then inequality (4) can never be true. Therefore, $|\Delta_i(x)| \leq M \times \binom{2}{r}$.

Theorem 2-3 illustrates that if r -opt search is served as a local optimal solution, only those elements x_i where $|\Delta_i(x)| \leq M \times \binom{2}{r}$ need to be considered by the r -opt strategy. Thus, the

set $D = \{x_i \mid |\Delta_i(x)| \leq M \times \binom{2}{r}\}$ is the elements of x needs to be considered in the r -opt

exchange strategy. In order to gain some insight into the size of D , we did the following experiment: First randomly initialize a problem and apply 1-opt search to it, then count the number of x_i in D . We repeated the above experiment 200 times for each of the UBQP problems used in this paper. Since the sizes of D for problems of the same size and density are similar, Table 2-1 lists the experimental result about the size of D as a variable only on

problem size and density. The first column is the size of the UBQP problems. The second row is the density of the UBQP problems. The values of D 's size reported in Table 2-1 are true for most cases. That means, in very rare cases, the size of D can be larger than the value reported in Table 2-1.

	$r = 2$				$r = 3$				$r = 4$			
	0.1	0.3	0.5	0.8	0.1	0.3	0.5	0.8	0.1	0.3	0.5	0.8
2500	<100	<40	<30	<20	<400	<200	<100	<100	<1000	<500	<300	<200
3000	<100	<40	<30	<20	<400	<200	<100	<100	<1100	<500	<400	<250
4000	<100	<30	<30	<20	<500	<200	<100	<100	<1200	<600	<400	<250
5000	<100	<30	<30	<20	<500	<200	<100	<100	<1300	<600	<400	<250
6000	<100	<30	<30	<20	<500	<200	<100	<100	<1400	<600	<400	<250

Table 2-1 Size of Set $D = \{x_i \mid |\Delta_i(x)| \leq M \times \binom{2}{r}\}$

From table 2-1, we can see that the size of D is usually very small compared to the size of x when r is small. The size of D increases as r increases. The size of D also depends on the size and density of the UBQP problems. When density increases, the size of D decreases.

Our experimental result also shows that in most cases the sizes of D at better local optimums are smaller than those at worse local optimums. This observation is very important because it means that as the global search improves solutions, the time for r -opt search decreases if we take advantage of Theorem 2-3.

Table 2-2 compares the number of possibilities that need to be considered for r -opt search when D is used against the number of possibilities that need to be considered for r -opt search when D is not used. Only possibilities for 2-opt and 3-opt are listed. The first column is the size of the UBQP problems. The second row is the density of the UBQP problems.

		$r = 2$				$r = 3$			
		0.1	0.3	0.5	0.8	0.1	0.3	0.5	0.8
2500	Use D	<4,950	<780	<435	<190	<10,586,800	<1,313,400	<161,700	<161,700
	Not use D	3,123,750				2,601,042,500			
3000	Use D	<4,950	<780	<435	<190	<10,586,800	<1,313,400	<161,700	<161,700
	Not use D	4,498,500				4,495,501,000			
4000	Use D	<4,950	<435	<435	<190	<20,708,500	<1,313,400	<161,700	<161,700
	Not use D	7,998,000				10,658,668,000			
5000	Use D	<4,950	<435	<435	<190	<20,708,500	<1,313,400	<161,700	<161,700
	Not use D	12,497,500				20,820,835,000			
6000	Use D	<4,950	<435	<435	<190	<20,708,500	<1,313,400	<161,700	<161,700
	Not use D	17,997,000				35,982,002,000			

Table 2-2 Number of possibilities that need to be considered for r -opt search

From Table 2-2, we can see that Theorem 2-3 reduces the number of possibilities that need to be considered by several orders, especially when density is large. In fact, most of the time, the number of possibilities that need to be considered for r -opt search if Theorem 2-3 is used is much smaller than the numbers listed in the table. Later in this section, we will show that we can further reduce the number of possibilities to be considered for r -opt search by using a simple heuristic.

In summary, Theorem 2-3 and Table 2-1 indicate that r -opt (especially when r is small) can be implemented quickly because only a small portion of variables in x need to be searched. The hybrid r -opt/1-opt local search algorithm we propose takes advantage of this useful observation to reduce the computation time.

Furthermore, in our hybrid r -opt/1-opt local search algorithm, we also use a simple heuristic: the smaller $|\Delta_i(x)|$ is, the more likely x_i is involved in a k -flip (for all $k \leq r$) move which improves the solution. To implement this simple heuristic, elements in set D need be sorted by the values of $|\Delta_i(x)|$ in ascending order. This gives us another opportunity to further reduce the number of possibilities that need to be considered for r -opt search. Based on inequation (4) in the proof of Theorem 2-3, we can finish the r -opt

search prematurely if we find S , a set of r x that satisfy inequation $\sum_{i \in S} |\Delta_i(x)| > M \times \binom{2}{r}$,

because any left combination of r x in D will violate inequation (4) due to the fact that elements in D are sorted by the values of $|\Delta_i(x)|$ in ascending order.

Denote D ($D \subseteq x$ and $|D| = m$) as the set to be considered by r -opt search. For $\forall x_i \in D$,

$|\Delta_i(x)| \leq c \leq M \times \binom{2}{r}$. If $c < M \times \binom{2}{r}$, the r -opt search is not complete, but it requires less

computational time. Figure 2-3 is the pseudo-code for the hybrid r -opt/1-opt local search algorithm for maximization problems.

```

Void r-opt/1-opt_local_search (r, Q, x,  $\Delta(x)$ , n, D, m, c)
Begin
  1-opt_local_search (Q, x,  $\Delta(x)$ , n)
  select_D (x,  $\Delta(x)$ , n, D, m, c)
  Do
    If improve (r, Q, x,  $\Delta(x)$ , n, D, m) == true
      1-opt_local_search (Q, x,  $\Delta(x)$ , n)
      select_D (x,  $\Delta(x)$ , n, D, m, c)
    Else
      Exit Do
    End if
  End do
End

```

Figure 2-3 pseudo-code for the hybrid *r*-opt/1-opt local search for maximization problems

```

Void select_D (x,  $\Delta(x)$ , n, D, m, c)
Begin
  Set m = 0
  For i = 1, ..., n
    If  $x_i = 1$ 
      Set  $D_m = i$  and  $m = m + 1$  if  $\Delta_i(x) \leq c$ 
    Else
      Set  $D_m = i$  and  $m = m + 1$  if  $\Delta_i(x) \geq -c$ 
    End if
  End for
  Sort D into ascending order
End

```

Figure 2-4 pseudo-code for the select_*D* subroutine

```

Boolean improve(r, Q, x,  $\Delta(x)$ , n, D, m)
Begin
  For i = 1, ..., r
    If r-opt_local_search_2 (i, Q, x,  $\Delta(x)$ , n, D, m) == true
      Return true
    End if
  End for
  Return false
End

```

Figure 2-5 pseudo-code for the improve subroutine

```

Boolean  $r$ -opt_local_search_2 ( $r, Q, x, \Delta(x), n, D, m$ )
Begin
  For all combinations of  $S$  where  $S \subseteq D$  and  $|S| = r$ 
     $\Delta f(x) = f(x') - f(x) = \sum_{i \in S} (\bar{x}_i - x_i) \Delta_i(x) + \sum_{\forall i, j \in S} (\bar{x}_i - x_i)(\bar{x}_j - x_j) \Delta_{ij}(x)$ .
    If  $\Delta f(x) > 0$ 
      Set  $x_i = 1 - x_i$  for  $x_i \in S$ 
      Update  $\Delta(x)$ 
       $\forall j \in N \setminus S \quad \Delta_j(x) = \Delta_j(x) + \sum_{i \in S} (\bar{x}_i - x_i) \Delta_{ij}(x)$ , and
       $\forall j \in S \quad \Delta_j(x) = \Delta_j(x) + \sum_{i \in S \setminus \{j\}} (\bar{x}_i - x_i) \Delta_{ij}(x)$ .
    Return true
  End If
End For
Return false
End

```

Figure 2-6 pseudo-code for the r -opt_local_search_2 subroutine

What is the best value for r depends on the density and size of the problems and the distribution of $\Delta_{ij}(x)$. Larger values of r lead to better solutions but more computational time. In our experiments, after some preliminary testing, we fixed r to be 3.

We have compared this hybrid r -opt/1-opt local search algorithm with 1-opt and other r -opt local search algorithms. Experiments show that it is an excellent local search algorithm which achieves a good balance between solution quality and computational time. The experimental results are reported and discussed in (Liu *et al* 2005). In this paper, we focus on the new algorithm which uses the hybrid r -opt/1-opt local search in conjunction with tabu search heuristic.

3. Tabu Search Algorithm using the Hybrid Local Search

Tabu search has many variants. See (Glover and Laguna, 1997) for a comprehensive discussion of tabu search. In our research, we used a simple tabu search implementation which contains only the main ingredient of the tabu search: a short-term memory tabu list. The following is the procedure for the simple tabu search.

- A. Initialization
 1. Initialize the x vector and the $\Delta_i(x)$ vector.
 2. Initialize the tabu list to zeros.
- B. Call the local search algorithm.

- C. If the run time exceeds the pre-set limit, stop.
- D. Destruction
 1. Find the variable that is not on the tabu list and does the least damage to the solution.
 2. Change its value; place it on the tabu list; update the $\Delta_i(x)$ vector; update the tabu list.
 3. Test if there is any variable that is not on the tabu list and can improve the solution. If no, go to D.1.
- E. Construction
 1. Test all the variables that are not on the tabu list. If a solution better than the best solution is found, change its value, place it on the tabu list, update the $\Delta_i(x)$ vector, update the tabu list and go to B.
 2. Pick the variable that improves the solution most, change its value, place it on the tabu list, update the $\Delta_i(x)$ vector and update the tabu list.
 3. If this is the fifteenth iteration in E, go to B.
 4. Test if there is any variable that is not on the tabu list and can improve the solution. If no, go to D.1. If yes, go to E.1.

Any local search algorithm can be used in step B of this simple tabu search heuristic.

The hybrid r -opt/1-opt local search introduced in section 2 was slightly modified to be used with this simple tabu search heuristic: If the solution found by 1-opt is worse than the current best solution found by tabu search, quit the local search and return to tabu search. Thus, the hybrid r -opt/1-opt local search degrades into 1-opt search when a local optimal solution worse than the current best solution is found. This slight modification ensures that not too much computational time is spent on local search. This simple tabu search algorithm with the hybrid r -opt/1-opt local search is referred to as NTS, New Tabu Search.

In order to determine whether the hybrid r -opt/1-opt local search does better than the 1-opt local search in the simple tabu search implementation, we compared NTS with an algorithm that uses only 1-opt local search in step B of the tabu search. The latter algorithm is referred to as STS, simple tabu search.

We also compared NTS and STS with the best Multiple Start Tabu search algorithm (MST) proposed by Palubeckis (2004). We chose Palubeckis' multiple start tabu search algorithm to compare against, because it is one of the latest heuristic algorithms for UBQP problems and it has excellent performance according to Palubeckis (2004). Another important reason is that the source code of Palubeckis' algorithm is available on the Internet, so we can test it against our algorithms on the same computer to ensure that the results are comparable.

4. Computational Results

The main purpose of experimentation was to find out whether the hybrid r -opt/1-opt local search does better than 1-opt local search when they are used as local search within tabu search heuristics. Furthermore, we wanted to know how the new tabu search algorithm performs compared to one of latest successful algorithms for UQBP, i.e., Palubeckis' multiple start tabu search.

The NTS and STS algorithms we have described were coded in the C++ programming language. The sources are publicly available at <http://www.cs.olemiss.edu/~wliu/ropt.htm>. MS Visual C++ 6.0 was used to compile all the programs. The source code of Palubeckis' multiple start tabu search algorithms were downloaded from <http://www.soften.ktu.lt/~gintaras/binqopt.html> and compiled with MS Visual C++ 6.0 also. Palubeckis proposed five multiple start tabu search algorithms. We only chose the best one, namely MST2 (Palubeckis, 2004, indicated MST2 is the best of their algorithms) to compare with our algorithms. Moreover, in our experiments, we used the author-recommended values for all the parameters in MST. All of the experimentation was done on a PC with Pentium 4 (2.66GHz) CPU and 512MB of RAM.

Preliminary experimental results showed that all the three algorithms did very well on problems with size less than 1000. Therefore, in this paper we only chose the 38 largest problem instances (sizes from 1000 to 6000) used by Palubeckis (2004). These problems include two series with 10 instances in each of size 1000 and 2500, respectively, introduced by Beasley (1998). All problems are maximization problems.

Each algorithm was run five times for each problem instance. We imposed a time limit of 40s for each run. Table 4-1 lists the computational results. The first three columns are the problem instance identifier, size of the instance and density of the instance, respectively. The fourth column shows the best solution found by all runs of the three algorithms. The rest of the table evaluates specific algorithms. The column under heading "NTS" displays the *difference* between the best value (column 4) and the value of the best solution found by NTS. The next columns represent the same characteristic for other algorithms.

problem	size	density	max	NTS	STS	MST
bpq1000_1	1000	0.1	371438	0	0	0
bpq1000_2	1000	0.1	354932	0	0	0
bpq1000_3	1000	0.1	371236	0	0	0
bpq1000_4	1000	0.1	370675	0	0	0
bpq1000_5	1000	0.1	352760	0	0	0
bpq1000_6	1000	0.1	359629	0	0	0
bpq1000_7	1000	0.1	371193	0	0	0
bpq1000_8	1000	0.1	351994	0	0	0
bpq1000_9	1000	0.1	349337	0	0	0
bpq1000_10	1000	0.1	351415	0	0	0
bpq2500_1	2500	0.1	1515944	0	0	0
bpq2500_2	2500	0.1	1471392	602	0	0
bpq2500_3	2500	0.1	1414192	0	926	0

bqp2500_4	2500	0.1	1507701	0	0	0
bqp2500_5	2500	0.1	1491816	0	0	0
bqp2500_6	2500	0.1	1469162	0	327	0
bqp2500_7	2500	0.1	1479040	280	728	0
bqp2500_8	2500	0.1	1484199	14	0	0
bqp2500_9	2500	0.1	1482413	0	0	0
bqp2500_10	2500	0.1	1483355	767	872	0
bqp3000_1	3000	0.5	3931583	3102	2571	0
bqp3000_2	3000	0.8	5193073	0	0	0
bqp3000_3	3000	0.8	5111533	717	717	0
bqp3000_4	3000	1	5761437	3277	5388	0
bqp3000_5	3000	1	5674778	0	0	788
bqp4000_1	4000	0.5	6181830	933	2552	0
bqp4000_2	4000	0.8	7799703	2589	2602	0
bqp4000_3	4000	0.8	7738563	393	3483	0
bqp4000_4	4000	1	8709343	194	4503	0
bqp4000_5	4000	1	8905075	0	1124	1475
bqp5000_1	5000	0.5	8557104	14128	6068	0
bqp5000_2	5000	0.8	10825765	619	0	3376
bqp5000_3	5000	0.8	10488783	0	0	16057
bqp5000_4	5000	1	12249534	0	8292	16674
bqp5000_5	5000	1	12729156	7624	0	14916
bqp6000_1	6000	0.5	11377874	0	307	11961
bqp6000_2	6000	0.8	14315679	0	112	112
bqp6000_3	6000	1	16117635	2110	8657	0

Table 4-1 Best solutions for the problems used in (Palubeckis 2004)

From Table 4-1, we can see that the three algorithms did equally well on the problems of size 1000. Besides this, Table 4-1 also indicates that the size and density of a problem have a significant impact on the performance of these three algorithms. To better understand how size and density affect the performance of these algorithms, we averaged the difference between the best solution and the best solution found by each algorithm over problems grouped by size and density. The problem instances whose size is 1000 were excluded from the calculation. The averages by size and density are listed in Table 4-2 and 4-3, respectively.

size	NTS	STS	MST
2500	166	285	0
3000	1419	1735	158
4000	822	2853	295
5000	4474	2872	10205
6000	703	3025	4024

Table 4-2 Average of difference between the best solution and the best solution found by each algorithm over problems grouped by size

From Table 4-2, we can see that MST outperformed NTS and STS on problems with size 4000 and less. However, for problems with size above 4000, MST did worse than NTS and STS. Comparing NTS to STS, NTS was better than STS in general except size 5000.

density	NTS	STS	MST
0.1	166	285	0
0.5	4541	2875	2990
0.8	617	988	2792
1	1886	3995	4836

Table 4-3 Average of difference between the best solution and the best solution found by each algorithm over problems grouped by density

Table 4-3 indicates that MST was better than NTS and STS when density is 0.1, and was worse when density is above 0.5. If we compare NTS to STS, we can see NTS did better than STS except when density is 0.5.

We may conclude from Tables 4-2 and 4-3 that MST works better on problems with smaller size and density while NTS and STS become more competitive as problem size and density increase. As for NTS and STS, NTS outperforms STS in general no matter size and density. The exceptions in Tables 4-2 and 4-3 were caused by just one problem, bqp5000_1, on which NTS happened to do poorly.

The problem instances from Palubeckis (2004) do not have high density instances when size is 2500 and do not have low density instances when size is above 2500. Therefore, in order to get more reliable results, we randomly generated 50 UBQP problems using the program provided by Palubeckis (2004). The problem instances we generated are five series in size of 2500, 3000, 4000, 5000 and 6000, respectively. Each series consists of 10 instances whose densities range from 0.1 to 1.

We repeated our experiments on these new problem sets and the results are listed in Table 4-4. The first three columns are the problem instance identifier, size of the instance and density of the instance, respectively. The fourth column shows the best solution found by all runs of the three algorithms. The rest of the table evaluates specific algorithms. The column under heading “NTS” displays the difference between the best value (column 4) and the value of the best solution found by NTS. The next columns represent the same characteristic for other algorithms.

problem	size	density	max	NTS	STS	MST
p2500_1	2500	0.1	1373321	0	0	0
p2500_2	2500	0.1	1361358	80	432	0
p2500_3	2500	0.3	2354297	0	0	0
p2500_4	2500	0.3	2350923	0	0	0
p2500_5	2500	0.5	3061193	0	1204	0
p2500_6	2500	0.5	2980254	1101	1101	0
p2500_7	2500	0.8	3732631	132	823	0
p2500_8	2500	0.8	3945371	0	0	0
p2500_9	2500	1	4367240	0	0	660
p2500_10	2500	1	4351642	0	0	0
p3000_1	3000	0.1	1752534	1395	1395	0
p3000_2	3000	0.1	1796489	763	1319	0
p3000_3	3000	0.3	3186179	0	0	0
p3000_4	3000	0.3	3117382	1455	2997	0
p3000_5	3000	0.5	3943235	2012	5291	0
p3000_6	3000	0.5	3937937	0	1233	0
p3000_7	3000	0.8	4869557	0	374	942
p3000_8	3000	0.8	5297065	0	0	0
p3000_9	3000	1	5725282	0	0	171
p3000_10	3000	1	5812655	0	4200	0
p4000_1	4000	0.1	2728401	242	0	0
p4000_2	4000	0.1	2799380	410	0	44
p4000_3	4000	0.3	4812903	1336	2729	0
p4000_4	4000	0.3	4751294	3434	837	0
p4000_5	4000	0.5	6155212	283	0	494
p4000_6	4000	0.5	6100166	5526	8664	0
p4000_7	4000	0.8	7780649	4070	28	0
p4000_8	4000	0.8	7921544	0	431	1913
p4000_9	4000	1	8776795	0	5482	1219
p4000_10	4000	1	8748287	0	5623	2677
p5000_1	5000	0.1	3821690	3085	6087	0
p5000_2	5000	0.1	3840041	2086	2435	0
p5000_3	5000	0.3	6780924	1524	5879	0
p5000_4	5000	0.3	6680005	2121	2949	0
p5000_5	5000	0.5	8433127	7948	6939	0
p5000_6	5000	0.5	8609364	0	1725	733
p5000_7	5000	0.8	10987316	0	1060	2631
p5000_8	5000	0.8	10884771	0	975	7084
p5000_9	5000	1	12546292	0	914	5277
p5000_10	5000	1	12256686	1422	0	8329
p6000_1	6000	0.1	5079900	5614	6560	0
p6000_2	6000	0.1	5050957	0	37	336
p6000_3	6000	0.3	8994975	1639	4727	0

p6000_4	6000	0.3	8746719	3982	4355	0
p6000_5	6000	0.5	11489782	0	399	4345
p6000_6	6000	0.5	11017766	2055	0	4108
p6000_7	6000	0.8	14271931	0	4316	10046
p6000_8	6000	0.8	14510254	4808	0	7170
p6000_9	6000	1	15687664	0	7431	17184
p6000_10	6000	1	16040695	76	0	2368

Table 4-4 Best solutions for the new randomly generated problems

As we did for Table 4-1, we also averaged the last three columns of Table 4-4 over problems grouped by size and density, respectively. The averages are listed in Tables 4-5 and 4-6.

size	NTS	STS	MST
2500	131.3	356	66
3000	562.5	1680.9	111.3
4000	1530.1	2379.4	634.7
5000	1818.6	2896.3	2405.4
6000	1817.4	2782.5	4555.7

Table 4-5 Average of the last 3 columns of table 4-4 over problems grouped by size

density	NTS	STS	MST
0.1	1367.5	1826.5	38
0.3	1549.1	2447.3	0
0.5	1892.5	2655.6	968
0.8	901	800.7	2978.6
1	149.8	2365	3788.5

Table 4-6 Average of the last 3 columns of table 4-4 over problems grouped by density

Tables 4-5 and 4-6 confirm the conclusions drawn from Tables 4-2 and 4-3. MST showed best performance on problems with smaller size and lower density, while NTS and STS, especially NTS, delivered strong performance on large and high density problems. In addition, NTS once again outperformed STS in general.

To further compare the hybrid r -opt/1-opt local search to the widely-used 1-opt local search when used in conjunction with the tabu search heuristic, we counted the number of times NTS found a better solution than STS, the number of times STS found a better solution than NTS, and how many times NTS and STS found the same solution. The results are listed in Table 4-7.

	NTS	STS	both
bpq series	14	6	8
p series	29	10	11
total	43	16	19

Table 4-7 Comparing NTS to STS

Table 4-7 shows that NTS did better than STS on 43 instances of the 78 test problems, excluding those of size 1000, while STS did better on only 16 of the 78 instances. They found the same solution on 19 instances all together. Table 4-7 confirms that NTS is a better algorithm than STS, which supports our belief that the hybrid r -opt/1-opt local search does better in tabu search than 1-opt.

5. Conclusions

We proposed a generic hybrid r -opt/1-opt local search algorithm for UBQP problems. A new algorithm combining a simple tabu search and the proposed hybrid local search is experimentally compared with an algorithm that uses the 1-opt local search with the same simple tabu search and with the best multiple start tabu search algorithm proposed by Palubeckis (2004). Computational results are provided for the 38 largest problem instances used by Palubeckis (2004), and in addition, for an additional 50 randomly generated problems with size from 2500 to 6000 variables.

The simple tabu search that uses the hybrid r -opt/1-opt local search in general outperformed the simple tabu search that uses the 1-opt local search. Besides this, we found that the multiple start tabu search algorithm proposed by Palubeckis (2004) is an excellent algorithm for problems of small to medium sizes and densities. However, the simple tabu search algorithms, especially the one using the hybrid r -opt/1-opt local search, did better on problems with very large size and density.

There are several issues for future work. It will be interesting to replace the 1-opt local search in the multiple start tabu search with the hybrid r -opt/1-opt local search to see if this improves the solution quality. Additional work is needed to find the best value for r . In our research, we arbitrarily chose three. Finally, the hybrid r -opt/1-opt local search should be tested with other global search heuristics such as genetic algorithm and simulated annealing.

References:

- Ahuja, R., O. Ergun, J. Orlin and A. Punnen, A Survey of Very Large-Scale Neighborhood Search techniques, *Discrete Applied Mathematics*, 2002, 123, 75-102.
- Ahuja, R., J. Orlin, S. Pallottino, M. Scaparra and M. Scutellà, A Multi-Exchange Heuristic for the Single-Source Capacitated Facility Location Problem, *Management Science*, 2004, 50(6), 749-760.

- Alidaee, B., G.A. Kochenberger and A. Ahmadian, 0-1 quadratic programming approach for the optimal solution of two scheduling problems, *International Journal of Systems Science*, 1994, 25, 401-408.
- Alidaee, B., Fan-and-Filter Neighborhood Strategy for 3-SAT Optimization, Working Paper, Hearin Center for Enterprise Science, School of Business Administration, The University of Mississippi, 2004.
- Alidaee, B, G. Kochenberger, F. Glover and C. Rego., A New Modeling and Solution Approach for the Number Partitioning Problem, *Journal of Applied Mathematics and Decision Sciences*, 2005, 9, 2, 113-121.
- Alkhamis, T.M., M. Hasab and M.A. Ahmed, Simulated annealing for the unconstrained quadratic pseudo-Boolean function, *European Journal of Operational Research*, 1998, 108, 641-652.
- Amini, M.M., B. Alidaee and G.A. Kochenberger, A scatter search approach to unconstrained quadratic binary programs, *New ideas in optimization*, 1999, McGraw-Hill, London, 317-329.
- Badics, T. and E. Boros, Minimization of half-products, *Mathematics of Operations Research*, 1998, 23, 649-660.
- Barahona, F., M. Grotscchel, M. Junger, and G. Reinelt, An application of combinatorial optimization to statistical physics and circuit layout design, *Operation Research*, 1988, 36, 493-513.
- Barahona, F., M. Junger and G. Reinelt, Experiments in quadratic 0-1 programming, *Mathematical Programming*, 1989, 44, 127-137.
- Beasley, E., Heuristic algorithms for the unconstrained binary quadratic programming problem, 1998, Working Paper, Imperial College.
- Billionnet, A. and A. Sutter, Minimization of a quadratic pseudo-Boolean function, *European Journal of Operational Research*, 1994, 78, 106-115.
- Boros, E., P.L. Hammer, M. Minoux and D. Rader, Optimal cell flipping to minimize channel density in VLSI design and pseudo-Boolean optimization, *Discrete Applied Mathematics*, 1999, 90, 69-88.
- Boros, E., P.L. Hammer and G. Tavares, Local search heuristics for unconstrained quadratic binary optimization, *Rutcor Research Report*, 2005.
- Carter, M.W., The indefinite zero-one quadratic problem, *Discrete Applied Mathematics*, 1984, 7, 23-44.

- Deineko, V. and G. Woeginger, A Study of Exponential Neighborhoods for the Travelling Salesman Problem and for the Quadratic Assignment Problem, *Mathematical Programming*, Ser. A, 2000, 87, 519-542.
- De Simone, C., C.M. Diehl, M. Junger, P. Mutzel, G. Reinelt and G. Rinaldi, Exact ground states of Ising spin glasses: New experimental results with a branch and cut algorithm, *Journal of Statistical Physics*, 1995, 80, 487-496.
- Fraenkel, A.S. and P.L. Hammer, Pseudo-Boolean functions and their graphs, *Annals of Discrete Mathematics*, 1984, 20, 137-146.
- Frangioni, A., E. Necciari and M. Scutellà, A Multi-Exchange Neighborhood for Minimum Makespan Parallel Machine Scheduling Problems, *Journal of Combinatorial Optimization*, 2004, 8(2), 195-220.
- Gallo, G., P.L. Hammer and B. Simeone, Quadratic knapsack problems, *Mathematical programming*, 1980, 12, 132-149.
- Glover, F., Finding The Best Traveling salesman 4-opt Move in the Same Time as a Best 2-opt Move, *Journal of heuristics*, 1996, 2, 169-179.
- Glover, F. and M. Laguna, *Tabu Search*, 1997, Kluwer Academic Publishers.
- Glover, F., G. Kochenberger and B. Alidaee, Adaptive memory tabu search for binary quadratic programs, *Management Science*, 1998, 44(3), 336-345.
- Glover, F., G. Kochenberger, B. Alidaee and M. Amini, Tabu Search with critical event memory: An enhanced application for binary quadratic programs, *Meta-Heuristics, Advances and Trends in Local Search Paradigms for Optimization*, S. Voss, S. Martello, I. Osman, and C. Roucarinol (Eds.), Kluwer, 1999, 93-109.
- Hammer, P. and S. Rudeanu, *Boolean Methods in Operations Research*, Springer-Verlag, New York, 1968.
- Hammer, P.L. and E. Shliffer, Applications of pseudo-Boolean methods to economic problems, *Theory and Decision*, 1971, 1, 296-308.
- Hammer, P.L., Pseudo-Boolean remarks on balanced graphs, *International Series of Numerical Mathematics*, 1977, 36, 69-78.
- Hansen, P.B., Methods of nonlinear 0-1 programming, *Annals of Discrete Math*, 1979, 5, 53-70.
- Helmberg, C. and F. Rendl, Solving quadratic (0,1)-problems by semi-definite programs and cutting planes, *Mathematical Programming*, 1998, 82, 291-315.

- Hillier, F.S., *The evaluation of risky interrelated investments*, North-Holland, Amsterdam, 1969.
- Kalantari, B. and A. Bagchi, An algorithm for quadratic zero-one programs, *Naval Research Logistics*, 1990, 37, 527-538.
- Katayama, K. and H. Narihisa, Performance of simulated annealing-based heuristic for the unconstrained binary quadratic programming problem, *European Journal of Operational Research*, 2001, 134, 103-119.
- Katayama, K., M. Tani and H. Narihisa, Solving large binary quadratic programming problems by effective genetic local search algorithm, *In Proceedings of the genetic and evolutionary computation conference*, 2000, Morgan Kaufman, 643-650.
- Kochenberger, G., F. Glover, B. Alidaee and C. Rego, A unified modeling and solution framework for combinatorial optimization problems, *OR Spectrum*, 2004, 26, 1-14.
- Kochenberger, G, F. Glover, B. Alidaee and C. Rego., An Unconstrained Quadratic Binary Programming Approach to the Vertex Coloring Problem., *Annals of Operations Research*, 2005, to appear.
- Kochenberger, G, F. Glover, B. Alidaee, and K. Lewis., Using the Unconstrained Quadratic Program to Model and Solve Max 2-SAT Problems, *International Journal of Operations Research*, 2005, 1, 1/2, 89-100.
- Kochenberger, G, F. Glover, B. Alidaee, and H. Wang., Clustering of Microarray Data via Clique Partitioning, *Journal of Combinatorial Optimization*, 2005, to appear.
- Krarup, J. and A. Pruzan, Computer aided layout design, *Mathematical Programming Study*, 1978, 9, 75-94.
- Lewis, M, B. Alidaee, and G. Kochenberger., Using xQx to Model and Solve the Uncapacitated Task Allocation Problem, *Operations Research Letters*, 2005, 33, 176-182.
- Li, W. and B. Alidaee, Dynamics of Local Search Heuristics for the Traveling Salesman Problem, *IEEE Transactions on Systems, Man, and Cybernetics – Part A*, 2002, 32, 173-184.
- Lin, S. and B. Kernighan, An Effective Heuristic Algorithm for the Traveling Salesman Problem, *Operations Research*, 1973, 21, 498-516.
- Liu, W., B. Alidaee and D. Wilkins, A new r -opt search for UBQP problems, TR2005-14, Department of Computer and Information Science, University of Mississippi, 2005

- Lodi, A., K. Allemand and T.M. Liebling, An evolutionary heuristic for quadratic 0-1 programming, *European Journal of Operational Research*, 1999, 119, 662-670.
- Magazine, M., G. Polak and D. Sharma, A Multi-Exchange Neighborhood Search for Integrated Clustering and Machine Setup Model for PCB Manufacturing, *Journal of Electronic Manufacturing*, 2002, 11(2), 107-119.
- Merz, P. and B. Freisleben, Genetic algorithm for binary quadratic programming, *In Proceedings of the genetic and evolutionary computation conference*, 1999, Morgan Kaufman, 417-424.
- Merz, P. and K. Katayama, Memetic algorithms for the unconstrained binary quadratic programming problem, *BioSystems*, 2004, 78, 99-118.
- Palubeckis, G., A heuristic-based branch and bound algorithm for unconstrained quadratic zero-one programming, *Computing*, 1995, 54, 283-301.
- Palubeckis, G. and A. Tomkevicius, GRASP implementations for the unconstrained binary quadratic optimization problem, *Information Technology and Control*, 2002, 3, 14-20.
- Palubeckis, G., Multi-start tabu search strategies for the unconstrained binary quadratic optimization problem, *Annals of Operations Research*, 2004, 131, 259-282.
- Papaiouannou, S.G., Optimal test generation in combinatorial networks by pseudo-Boolean programming, *IEEE Transactions on Computers*, 1977, 26, 553-560.
- Pardalos, P. and G.P. Rodgers, A Branch and Bound algorithm for maximum clique problem, *Computer & OR*, 1992, 19, 363-375.
- Pardalos, P. and J. Xue, The maximum clique problem, *The Journal of Global Optimization*, 1994, 4, 301-328.
- Phillips, A.T. and J.B. Rosen, A quadratic assignment formulation for the molecular conformation problem, *Journal of Global Optimization*, 1994, 4, 229-241.
- Rao, M.R., Cluster analysis and mathematical programming, *Journal of the American Statistical Association*, 1977, 66, 622-626.
- Vredeveld, T. and J. Lenstra, On Local Search for the Generalized Graph Coloring Problem, *Operations Research Letters*, 2003, 31, 28-34.
- Yaguiura, M. and T. Ibaraki, Analyses on the 2 and 3-flip Neighborhoods for the MAX SAT, *Journal of Combinatorial Optimization*, 1999, 3, 95-114.
- Yaguiura, M. and T. Ibaraki, Efficient 2 and 3-flip Neighborhood Search Algorithms for the MAX SAT: Experimental Evaluation, *Journal of Heuristics*, 2001, 7, 423-442.