Working Paper Series
WP 2021-002
April 2021

New Results on Closed-Form Formulas for Evaluating r-flip Moves in Quadratic Unconstrained Binary Optimization

Bahram Alidaee
The University of Mississippi

Haibo Wang
Texas A&M International University

Wade W. Liu
The University of Mississippi

NEW RESULTS ON CLOSED-FORM FORMULAS FOR EVALUATING *R*-FLIP MOVES IN QUADRATIC UNCONSTRAINED BINARY OPTIMIZATION

BAHRAM ALIDAEE
School of Business Administration
The University of Mississippi
University, Mississippi, USA

HAIBO WANG*
A.R. Sanchez, Jr. School of Business
Texas A&M International University
Laredo, Texas, USA

WADE W. LIU
Department of Computer Science
School of Engineering
The University of Mississippi
University, Mississippi, USA

The quadratic unconstrained binary optimization (QUBO) is a classic NP-hard problem with enormous applications. Local search strategy (LSS) is one of the most fundamental algorithmic concepts that has been successfully applied to a wide range of hard combinatorial optimization problems. A basic element of almost all sophisticated heuristics is some variations of LSS. One LSS that has gained attention of researchers is the r-flip (also known as r-Opt) strategy. Given a binary solution with n variables, the r-flip strategy 'flips' r binary variables to get a new solution if the changes improve the objective function. The implementation of an r-flip local search can be time consuming. This is due to the fact that the size of the search space is of order n^r *and grows quickly in r; hence, researchers adopt a 1-flip move in their algorithms. The r-flip strategy is fundamentally important in a local search where some variable neighborhood strategies, such as fan-and-filter (F&F), variable neighborhood search (VNS), and multi-exchange neighborhood search (MENS), are the basis of the search process. The main purpose of this paper is to prove several theoretical results for QUBO, including a necessary and sufficient condition that when a 1-flip search reaches a local optimality, the number of candidates for implementation of the r-flip moves can be reduced significantly. The results of the substantial computer exercise are reported to compare the r-flip strategy embedded algorithm and multiple start tabu search algorithm on a set of benchmark instances and three very-large-scale QUBO instances. The r-flip strategy embedded algorithm provides a very high-quality solution within the 60- and 600-seconds time limits.*

KEYWORDS *Combinatorial optimization, Quadratic unconstrained binary optimization, Local optimality, r-flip local optimality*

* Address correspondence to Haibo Wang, Division of International Business and Technology Studies, A.R. Sanchez, Jr. School of Business, Texas A&M International University, 5201 University Boulevard, Laredo, Texas 78041. Email: hwang@tamiu.edu.

1. Introduction

The quadratic unconstrained binary optimization (QUBO) is formulated as:

$$Max\ \ f(x)=\sum_{i=1}^{n}q_i x_i+\frac{1}{2}\sum_{i=1}^{n}\sum_{j\neq i}^{n}q_{ij}x_i x_j,\ \ \text{s.t. } x_i\in\{0,1\},\ \ i=1,\cdots,n. \tag{1}$$

In (1), $\frac{1}{2}q_{ij}$ is the *ij*-th element of a given *n* by *n* symmetric matrix *Q*. The QUBO is often referred to as the *xQx* model (Kochenberger and Glover 2013; Lewis et al. 2009; Lewis et al. 2005). Since $x_i^2=x_i$, and *Q* may be written as an upper triangular matrix by doubling each element of the upper triangle part of the matrix, and letting $q_{ii}=q_i$, then we can write (1) as (2).

$$Max\ \ f(x)=\sum_{i=1}^{n}\sum_{j\geq i}^{n}q_{ij}x_i x_j=xQx,\ \ \text{s.t. } x_i\in\{0,1\},\ \ i=1,\cdots,n \tag{2}$$

The problem has enormous applications and has been used as a unifying approach to many combinatorial optimization problems (Alidaee et al. 1994; Glover et al. 2019; Kochenberger et al. 2014; Kochenberger et al. 2004). The QUBO is a classic NP-hard problem. Due to its practicality, as well as theoretical interest, over the years researchers have proposed many theoretical results as well as simple and sophisticated approaches as solution procedures (Alidaee et al. 2010; Alidaee et al. 2017; Alidaee and Wang 2017; Anacleto et al. 2020; Beasley 1998; Boros et al. 1999; Boros et al. 2007; Chen 2015; Glover et al. 1998; Glover et al. 2018; Lewis et al. 2009; Lewis et al. 2005; Merz and Freisleben 2002; Palubeckis 2004). Local search strategy (LSS) is one of the most fundamental algorithmic concepts that has been successfully applied to a wide range of hard combinatorial optimization problems. The basic ingredient of almost all sophisticated heuristics is some variations of LSS. One LSS that has been used by many researchers as a stand-alone or as a basic component of more sophisticated algorithms is the *r-flip* (also known as *r-Opt*) strategy (Ahuja et al. 2002; Ahuja et al. 2004; Alidaee et al. 2010; Alidaee et al. 2017; Alidaee and Wang 2017; Anacleto et al. 2020; Li and Alidaee 2002; Lin and Kernighan 1973; Mladenović and Hansen 1997; Resende 2008; Szeider 2011; Yagiura and Ibaraki 1999, 2001; Yagiura et al. 2006). Let *N={1,2,...,n}*. Given a binary solution, $x=(x_1,\cdots,x_n)$ of *xQx*, the *r*-flip search chooses a subset, $S\subseteq N$, with $|S|\leq r$, and builds a new solution, *x'*, where $x'_i=1-x_i$ for all $i\in S$. If *x'* improves the objective function, it is called an *improving move (*or improving subset *S)*. The *r*-flip search starts with a solution *x*, chooses an improving subset *S*, and flips all elements in *S*. The process continues until there is no subset *S* with $|S|\leq r$ that improves the objective function. The result is called *a locally optimal solution with respect to the r*-flip *move* (or *r*-Opt). Often in strategies where variable neighborhood searches, such as *fan-and-filter (F&F)* (Alidaee 2004; Glover 1998), *variable neighborhood search (VNS)*(Mladenović and Hansen 1997), and *multi-exchange neighborhood search (MENS)*(Ahuja et al. 2002; Ahuja et al. 2004; Mladenović and Hansen 1997) are used, the value of *r* dynamically changes as the search progresses. Generally, there are two reasons for a dynamically changing search space strategy.

a) The implementation of an *r*-flip local search, for the larger value of *r*, can be computationally expensive. This is because the size of the search space is of the order n^r and grows quickly in *r*. Hence, the smaller values of *r*, especially, *r* equal to 1 and 2, have shown considerable success.
b) In practice, an *r*-flip local search process with the small value of *r* (e.g., *r=1*) can quickly reach local optimality. Thus, as one way to escape local optimality, researchers have tried to dynamically change the value of *r* as the search progressed. This gives an opportunity to expand the search to a more diverse solution space.

A clever implementation of (a) and (b) in an algorithm can not only save computational time, since the smaller value of *r* is less computationally expensive, but it can also possibly reach better solutions because the larger values of *r* provide an opportunity to search more a diverse part of the solution space.

The development of closed form formulas for *r*-flip moves is desirable for developing heuristics for the very-large-scale problems because it can reduce computational time in an algorithm. Alidaee et al. (2010) introduced several theorems showing closed form *r*-flip formulas for general pseudo-Boolean optimization. In particular, **Theorem 6** in (Alidaee et al. 2010) is specific to the *xQx* problem. To explain the closed form formula for the *r*-flip rule in *xQx*, we first introduce a few definitions.

Given a solution $x=(x_1,\cdots,x_n)$, *derivative* of *f(x)* with respect to x_i is:

$$E(x_i)=q_i+\sum_{j<i}q_{ji}x_j+\sum_{j>i}q_{ij}x_j,\ \ i=1,\cdots,n. \tag{3}$$

Fact 1. Given a solution vector $x=(x_1,\cdots,x_i,\cdots,x_n)$, obtain a new solution $x'=(x_1,\cdots,1-x_i,\cdots,x_n)$ from *x* by flipping an element *i*. Now we have:

$$E(f)=f(x')-f(x)=(x'_i-x_i)E(x_i)\ . \tag{4}$$

It is well known that any local optimal solution of QUBO with respect to a 1-flip search satisfy,

$$\textit{Either}\ \ (x_i=0\ \ \textit{iff}\ \ E(x_i)\le 0)\ \ \textit{or}\ \ (x_i=1\ \ \textit{iff}\ \ E(x_i)\ge 0),\ \ \text{for } i=1,\cdots,n. \tag{5}$$

Furthermore, after changing *x* to *x'*, the update for $E(x_j)$ *, j=1,...,n*, can be calculated as follows:

$$\begin{aligned}\forall j<i,&\ \ E(x_j)=E(x_j)+q_{ji}(x'_i-x_i)\\ \forall j>i,&\ \ E(x_j)=E(x_j)+q_{ij}(x'_i-x_i)\\ j=i,&\ \ E(x_j)=E(x_j)\end{aligned} \tag{6}$$

Note that x'_i-x_i may be written as $1-2x_i$, which can simplify the implementation process. A simple 1-flip search is provided in Figure 1. Note that in line 3 we chose a sequence to implement Fact 1. Using such a strategy has experimentally proven to be very effective in several recent studies (Alidaee et al. 2017; Alidaee and Wang 2017; Wang and Alidaee 2018, 2019a, 2019b; Wang et al. 2020).

Before we present the algorithms in this study for the r-flip strategy, the notations used are given as follows:

n	The number of variables
x	A starting feasible solution
x*	The best solution found so far by the algorithm
K	The largest value of k for r-flip, k≤r
π(i)	The i-th element of x in the order π(1)⋯π(n)
S	The index of improving subset of a binary solution
D	The set of candidates for an improving move
Tabu_ten	The maximum values a variable can remain Tabu
Tabu(i)	A vector representing Tabu status of x

Figure 1. Pseudo-code for a 1-flip local search subroutine for maximization problems.

```
Algorithm 1. 1-flip Local Search
Initialize: n, x, evaluate vector E(x)
Flag=1
1 Do while (Flag=1)
2    Flag=0
3    Randomly choose a sequence $\pi(1),\ldots,\pi(n)$ of integers 1,…,n.
4    Do i= $\pi(1),\ldots,\pi(n)$
5       If ( $E(x_i) < 0$ and $x_i = 1$ ) or ( $E(x_i) > 0$ and $x_i = 0$ ):
            $x_i = 1 - x_i$, update vector E(x) using Equation (6), Flag=1
6    End do
7 End while
```

The result of Fact 1 has been extended to the r-flip search, given below.

(**Theorem 6, Alidaee et al. (2010)**) Let *x* be a given solution of QUBO and *x'* obtained from *x* by *r*-flip move where $S \subseteq N$, and |*S*|=*r*. The change in the value of the objective function is:

$$E(f) = f(x') - f(x) = \sum_{i \in S}(x'_i - x_i)E(x_i) + \sum_{i,j \in S}(x'_i - x_i)(x'_j - x_j)q_{ij} \qquad (7)$$

Furthermore, after changing *x* to *x'*, the update for $E(x_j)$, *j=1,…,n*, can be calculated as follows:

$$\begin{aligned} &\forall j \in N \setminus S, \quad E(x_j) = E(x_j) + \sum_{i \in S} (x'_i - x_i) q_{ij} \\ &\forall j \in S, \quad E(x_j) = E(x_j) + \sum_{i \in S \setminus \{j\}} (x'_i - x_i) q_{ij} \end{aligned} \tag{8}$$

As explained in Alidaee et al. (2010), the evaluation of change in the objective function (7) can be done in $O(r^2)$, i.e., evaluating *f(x')* from *f(x)*, and update in (8) can be performed in *O(n)*.

Note that for any two elements *i,j=1,…,n*, we can define:

$$\begin{aligned} E'(x_i) &= E(x_i) - q_i - q_{ij} x_j \\ E'(x_j) &= E(x_j) - q_j - q_{ji} x_i \end{aligned} \tag{9}$$

Using (9), a useful way to express Equation (7) is Equation (10), as used in a recent paper (Anacleto et al. 2020).

$$E(f) = \sum_{i \in S} [(1 - 2x_i) E'(x_i) + \sum_{j \in S, j \le i} (1 - x_i - x_j) q_{ij}] \tag{10}$$

A simple exhaustive *r*-flip search is provided in Figure 2. The complexity of the problem indicates that the use of a larger value of *r* in the *r*-flip local search can make the implementation of the search process more time consuming. Meanwhile, the larger value of *r* can provide an opportunity to search a more diverse area of search space and, thus, possibly reach better solutions. To overcome such conflicts, researchers often use *r*=1 (and occasionally *r*=2) as the basic components of their more complex algorithms, such as F&F, VNS, and MENS. Below, in **Theorem 1** and **Proposition 1**, we prove that after reaching the locally optimal solution with respect to a 1-flip search, the implementation of an *r*-flip search can significantly be reduced. Further, related results are also provided to allow the efficient implementation of an *r*-flip search within an algorithm.

Figure 2. Pseudo-code for an exhaustive *r*-flip local search subroutine for maximization problems.

```
Algorithm 2. Exhaustive r-flip Local Search
Initialize: n, x, evaluate vector E(x), value of r
Flag=1
1  Do while (Flag=1)
2  Flag=0
3  For each combination $S \subset N$ and $|S| \le r$, evaluate E(f), Equation (7):
       If E(f) > 0:
           $x_i = 1 - x_1$, for $i \in S$, update vector E(x) using Equation (8), Flag=1
4  End While
```

2. New results

We first introduce some notations. For $m<n$, define *(m, n)* to be the number of combinations of m elements out of n, and let $\varphi = \underset{i,j\in N}{Max}\{|q_{ij}|\}$, and $M = \varphi * (2,r)$. Furthermore, **Lemma 1** and **Lemma 2**, given below, help to prove the results.

Lemma 1: Given a locally optimal solution $x=(x_1,\cdots,x_n)$ with respect to a 1-flip search, we have:

$$(x'_i - x_i)E(x_i) \le 0 \text{, for } i=1,\dots,n. \tag{11}$$

Proof: Condition of local optimality in (5) indicates that:

$$(E(x_i) \ge 0 \text{ iff } x_i = 1), \text{ and } (E(x_i) \le 0, \text{ iff } x_i = 0).$$

Using this condition, we thus have:

$$(x'_i - x_i)E(x_i) \le 0, \quad \text{for } i = 1,\cdots,n.$$

Lemma 2: Let $x=(x_1,\cdots,x_n)$ be any solution of the problem; then, we have:

$$\sum_{i,j\in S}(x'_i - x_i)(x'_j - x_j)q_{ij} \le M\,. \tag{12}$$

Proof: For each pair of elements, $i, j \in S$, the left-hand-side can be q_{ij} or $-q_{ij}$. Since $|S|=r$, the summation on the left-hand-side is at most equal to M.

Theorem 1: Let φ and M be as defined above, and let $x=(x_1,\cdots,x_n)$ be a locally optimal solution of xQx with respect to a 1-flip search. A subset $S \subseteq N$, with $|S|=r$, is an improving r-flip move *if and only if* we have:

$$\sum_{i\in S}|E(x_i)| < \sum_{i,j\in S}(x'_i - x_i)(x'_j - x_j)q_{ij} \tag{13}$$

Proof: Using (7), a subset $S \subseteq N$ of r elements is an improving r-flip move if and only if we have:

$$f(x') - f(x) = \sum_{i\in S}(x'_i - x_i)E(x_i) + \sum_{i,j\in S}(x'_i - x_i)(x'_j - x_j)q_{ij} > 0 \tag{14}$$

Since x is a locally optimal solution with respect to a 1-flip search, it follows from **Lemma 1** that inequality (14) is equivalent to (15); that completes the proof.

$$\sum_{i,j \in S} (x'_i - x_i)(x'_j - x_j) q_{ij} > -\sum_{i \in S} (x'_i - x_i) E(x_i) = \sum_{i \in S} | E(x_i) | \qquad (15)$$

Proposition 1: Let φ and M be as defined above and let $x = (x_1, \cdots, x_n)$ be any locally optimal solution of the *xQx* problem with respect to a 1-flip search. If a subset $S \subseteq N$, with $|S|=r$, is an improving *r*-flip move, then we must have: $\sum_{i \in S} | E(x_i) | < M$.

Proof: Since *x* is a locally optimal solution with respect to a 1-flip search and *S* is an improving *r*-flip move, by **Theorem 1,** we have:

$$\sum_{i \in S} | E(x_i) | < \sum_{i,j \in S} (x'_i - x_i)(x'_j - x_j) q_{ij} \qquad (15)$$

Using **Lemma 2,** we also have (16), which completes the proof.

$$\sum_{i \in S} | E(x_i) | < \sum_{i,j \in S} (x'_i - x_i)(x'_j - x_j) q_{ij} \le M \qquad (16)$$

A consequence of ***Theorem 1*** is as follows. Given a locally optimal solution *x* with respect to a 1-flip search, if there is no subset of *S* with $|S|=r$ which satisfies (13), then *x* is also locally optimal with respect to an r-flip search. Furthermore, if there is no subset *S* of any size that (13) is satisfied, then *x* is also locally optimal with respect to an *r*-flip search for all $r \le n$. Similar statements are also true regarding **Proposition 1**.

The result of **Proposition1** is significant in the implementation of an *r*-flip search. It illustrates that, after having a 1-flip search implemented, if an *r*-flip search is next served as a local optimal solution, only those elements with the sum of absolute value of derivatives less than *M* are eligible for consideration. Furthermore, when deciding about the elements of an *r*-flip search, we can easily check to see if any element x_i by itself or with a combination of other elements is eligible to be a member of an improving *r*-flip move *S*. Example 1 below illustrates this situation.

Example 1: Consider an *xQx* problem with *n* variables. Let $x = (x_1, \cdots, x_n)$ be a given locally optimal solution with respect to a 1-flip search. Consider *S=(i,j,k,l)* for a possible 4-flip move. In order to have *S* for an improving move, all 15 inequalities, given below in (17), must be satisfied. Of course, if the last inequality in (17) is satisfied, all other inequalities are also satisfied. This means each subset of the *S* is also an improving move. This is important in any dynamic neighborhood search strategies *k*-flip moves for $k \le r$ in consideration.

Here we have $\varphi = \underset{i,j \in N}{Max}\{| q_{ij} |\}$, and $M = 6 * \varphi$:

$$\begin{aligned}
&| E(x_a) | < M, \text{ for } a = i, j, k, l, \\
&| E(x_a) | + | E(x_b) | < M, \text{ for } (a \ne b), a, b = i, j, k, l, \qquad (17)\\
&| E(x_a) | + | E(x_b) | + | E(x_c) | < M, \text{ for } (a \ne b \ne c), a, b, c = i, j, k, l, \\
&| E(x_a) | + | E(x_b) | + | E(x_c) | + | E(x_d) | < M, \text{ for } a = i, b = j, c = k, d = l
\end{aligned}$$

Obviously, choosing the appropriate subset *S* to implement a move is critical. There are many ways to check for an improving subset *S*. Below, we explain two such strategies.

2.1. Strategy 1

We first define a set *D(n)* where all its subsets are the candidates for an improving move. Given a locally optimal solution *x* with respect to a 1-flip move, let the elements of *x* be ordered in ascending absolute value of derivatives, as given in (18).

$$|E(x_{\pi(1)})| \leq \cdots \leq |E(x_{\pi(n)})| \tag{18}$$

Here, $\pi(i)$ means the *i*-th element of x in the order $(\pi(1),\cdots,\pi(n))$. Now, one at a time in the given order, check the summation in (19) for *k=1,2,...,n.* Let *K* be the largest value of *k* where the inequality is satisfied. The set *D(n)* is now defined by (20).

$$\sum_{i=1}^{k} |E(x_{\pi(i)})| < M, \quad \text{for } k = 1,2,3,\cdots,n \tag{19}$$

$$D(n) = \{x_{\pi(1)},\cdots,x_{\pi(K)}\} \tag{20}$$

Lemma 3: Any subset $S \subseteq D(n)$ satisfy the *necessary condition* for an improving move.
Proof. It follows from **Proposition 1**.

There are some advantages to having elements of *x* in an ascending order, i.e., inequalities (18): (i) the smaller the value of $|E(x_i)|$ is, the more likely that x_i is involved in an improving *k*-flip move for $k \leq r$; (ii) because the elements of *D(n)* are in an ascending order of absolute values of derivatives, a straightforward implementable series of alternatives to be considered for improving subsets, *S*, may be the elements of the set given in (21). Note that, there are a lot more subsets of *D(n)* compared to the sets in (21) that are the candidates for consideration in possible *k*-flip moves. Here we only gave one possible efficient implementable strategy.

$$S = \{\{\pi(1),\pi(2)\},\{\pi(1),\pi(2),\pi(3)\},\cdots,\{\pi(1),\cdots,\pi(K)\}\} \tag{21}$$

It is important to note that, if **Proposition 1** is used in the process of implementing an algorithm, given a locally optimal solution *x* with respect to a 1-flip search, after an *r*-flip implementation for a subset |*S*|=*r* with *r*>1, local optimality with respect to a 1-flip search for the new solution, *x'*, can be destroyed. Thus, if an *r*-flip search needed to be continued, a 1-flip search might be necessary on solution *x'* before a new *r*-flip move can continue. However, there are many practical situations where this problem may be avoided for many subsets, especially when the problem is very-large-scale, i.e., the value of *n* is large, and/or *Q* is sparce. **Proposition 2** is a weaker condition of **Proposition 1** that can help to overcome up to some point in the aforementioned problem.

In the proof of **Theorem 1** and **Proposition 1**, we only used a condition of optimality for a 1-flip search satisfied for the members of the subset *S*. We now define a condition as follows and call it '*condition of optimality with respect to a* 1-flip *search for a set S'*, or simply *'condition of optimality for S'*.

Given a solution *x*, the *condition of optimality* for any subset $S \subseteq N$ is satisfied if and only if we have:

$$Either \ \ (x_i = 0 \ \ iff \ \ E(i) \le 0) \ \ or \ \ (x_i = 1 \ \ iff \ \ E(i) \ge 0), \ \ \text{for } i \in S \qquad (22)$$

Of course, if we have *N* in (22) instead of *S*, *x* is a locally optimal solution as was defined in **Fact 1**. For *m*<*n*, let *(m, n)* be the number of combinations of *m* elements out of *n* elements, and $\varphi_S = \underset{i,j \in S}{Max}\{|\, q_{ij} \,|\}$, and $M_S = \varphi_S * (2, r)$. With these definitions, now we state **Proposition 2**.

Proposition 2 (weak necessary condition): Let $S \subseteq N$, |*S*|=*r*, and φ_S, and M_S as defined above. Given any solution $x = (x_1, \cdots, x_n)$ of *xQx*, and assume the condition of optimality is satisfied for a subset *S*. If *S* is an *r*-flip improving move, we must have $\sum_{i \in S} |\, E(x_i) \,| < M_S$.

Proof: Similar to proof of **Proposition 1**.

Notice that, the values of φ_S and M_S in **Proposition 2** depend on *S*; however, these values can be updated efficiently as the search progresses. As explained above, in situations where the problem is very-large-scale and/or *Q* is sparce, for many variable, the values of derivatives are 'unaffected' by the change of values of elements in *S*. This means a large set of variables still satisfies the condition of optimality and, thus, the search can continue without applying a 1-flip search each time before finding a new set *S* for *r*-flip implementation.

2.2. Strategy 2

Another efficient and easily implementable strategy is when instead of (19), we only use an individual element to create a set of candidates for applying an *r*-flip search, set *D(1)* as defined below. **Corollary 1** is a special case of **Proposition 1** that suffices such a strategy.

$$D(1) = \{x_i : |\, E(x_i) \,| < M\} \qquad (23)$$

Corollary 1: Let φ and *M* be as defined before, given a solution $x = (x_1, \cdots, x_n)$ of *xQx*, if the 1-flip 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 we must have $|\, E(x_i) \,| < M$.

To gain insight into the use of **Corollary 1,** we did some experimentation to find the size of the set *D(1)* for different sizes of instances. The steps of the experiment to find the size of *D(1)* are given below. Problems considered are taken from the OR-Library. These are all maximization problems created by Beasley (1998) and Palubeckis (2004), and used by many researchers. We only used the larger-scale problems with 2500 to 6000 variables, a total of 38 instances.

Find_D(): Procedure for finding the size of the set *D(1)*:

Step 1. Randomly initialize a solution of the problem. Apply the algorithm in **Figure 1** and generate a locally optimal solution *x* with respect to a 1-flip search.

Step 2. Find the number of elements in the set *D(1)* for *x*.

Step 3. Repeat Step 1 and 2, 200 times for each problem, and find the average number of elements in the set *D(1)* for the same size problem, density, and *r* value.

The results of the experiment are shown in **Table 1**. From **Table 1**, we can see that for the larger value of *r*, the size of *D(1)* increases for the same problem size and density. As the density of matrix *Q* increases, the size of *D(1)* decreases for all problem sizes and values of *r*. This is, of course, due to the fact that the larger density of *Q* makes the derivative of each element in an *x* more related to other elements. As the size of a problem increases, the size of *D(1)* also increases. An interesting observation in our experiment was that, in most cases, the size of *D(1)* for better local optimum solutions were smaller than those with the worse local optimal solutions. This indicates that as the search reaches closer to the lobal optimal solutions, the time for an *r*-flip search decreases when we take advantage of **Corollary 1**.

Table 1. Size of the set D(1).

	r=2				*r=3*				*r=4*			
	Density				*Density*				*Density*			
Problem size	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

2.3. Implementation details

We first implement two strategies in Sections 2.1 and 2.2 via **Algorithm 3** for **Strategy 1 (**see **Figure 3),** and **Algorithm 4** for **Strategy 2 (**see **Figure 4)**, then propose **Algorithm 5** for **Strategy 2** embedded with a simple tabu search algorithm for the improvement in **Figure 5.**

Figure 3. Pseudo-code for hybrid *r*-flip/1-flip local search, **Strategy 1**, for maximization problems.

```
Algorithm 3. r-flip Local Search: Strategy 1
Initialize: n, x, evaluate vector E(x), value of r, M
Flag=1
1  Do while (Flag=1)
2      Flag=0
3      Call 1-flip local search: Algorithm 1
4      Sort variables according to $|E(x_{\pi(i)})| \leq |x_{\pi(i+1)}|$, using Inequality (19) evaluate value of K
5      For $j = \pi(1), \cdots, \pi(K)$:
6          For $S_j = \{\pi(1), \cdots, \pi(j)\}$, evaluate $M_{S_j}$
              If $\sum_{i=1}^{j} |E(x_{\pi(i)})| < M_{S_j}$, evaluate E(f) using Equation (7).
7          If E(f)>0:
              $x_i = 1 - x_i$, for $i \in S_j$, update vector E(x) using Equation (8), Flag=1, go to Step 1
8      End for
9  End while
```

Figure 4. Pseudo-code for a hybrid *r*-flip/1-flip local search, **Strategy 2**, for maximization problems.

```
Algorithm 4. r-flip Local Search: Strategy 2
Initialize: n, x, evaluate vector E(x), value of r, M
Flag=1
1  Do while (Flag=1)
2      Flag=0, and $S = \emptyset$
3      Call 1-flip local search: Algorithm 1
4      Randomly choose a sequence $\pi(1), \cdots, \pi(n)$ of integers $1, \cdots, n$
5      For $j = \pi(1), \cdots, \pi(n)$:
6          If $|E(x_j)| < M$, and $|S \cup \{j\}| \leq r$ evaluate E(f) for $S \cup \{j\}$ using Equation (7)
7          If E(f) >0:
              $x_i = 1 - x_i$, for $i \in S \cup \{j\}$, update vector E(x) using Equation (8), $S = S \cup \{j\}$,
              Flag =1, go to Step 1
8      End for
9  End while
```

Figure 5. Pseudo-code for hybrid r-flip/1-flip local search embedded with a simple tabu search algorithm.

Algorithm 5. Hybrid *r*-flip/1-flip Local Search embedded with a simple tabu search algorithm

Initialize: n, x, tabu list, evaluate vector *E(x),* value of *r, M, tabu tenure*

```
1  Call local search: Algorithm 4
2  Do While (until some stopping criteria, e.g., CPU time limit, is reached)
3    Call Destruction()
4    Call Construction()
5    Call randChange()
6  End While
```

In the Destruction() procedure, there are three steps:

Step 3a:	Find the variable that is not on the tabu list and does the least damage to the solution.
Step 3b:	Change its value; place it on the tabu list; update the E(x) vector; update the tabu list.
Step 3c:	Test if there is any variable that is not on the tabu list and can improve the solution. If no, go to **Step 3a.**

In the Construction() procedure, there are four steps:

Step 4a:	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 E(x) vector, update the tabu list, and go to **Step 1**.
Step 4b:	Pick the variable that improves the solution most, change its value, place it on the tabu list, update the E(x) vector, and update the tabu list.
Step 4c:	If this is the fifteenth iteration in the Construction procedure, go to **Step 1**.
Step 4d:	Test if there is any variable that is not on the tabu list and can improve the solution. If no, go to **Step 3a**. If yes, go to **Step 4a**.

The randChange() procedure is invoked occasionally and randomly to select an *x* for the Destruction() using a random number generator and there is less than a 2% probability to invoke after the Construction() procedure.

Any local search algorithm can be used in **Step 1** of this simple tabu search heuristic.

The hybrid *r*-flip/1-flip local search, **Algorithm 4**, was slightly modified and used with this simple tabu search heuristic (Step 1): If the solution found by a 1-flip is worse than the current best-found solution, quit the local search and go to Step 3. Thus, the hybrid *r*-flip/1-flip local search degrades into a 1-flip search when a local optimal solution is worse than the current best solution found. This slight modification ensures that not too much computational time is spent on a local search. This hybrid *r*-flip/1-flip local search embedded with a simple tabu search algorithm is referred to as **Algorithm 5**.

In order to determine whether the hybrid *r*-flip/1-flip local search algorithms with two strategies (**Algorithms 3** and **4**) do better than the hybrid r-flip/1-flip local search embedded with a simple tabu search implementation, we compared **Algorithms 3** and **4** with **Algorithm 5**.

The goal of the new strategies is to reach local optimality on large scale instances with less computing time. We report the comparison of three algorithms of a 2-flip on very-large-scale QUBO instances in the next section.

3. Computational results

In this study, we perform substantial computer exercises to evaluate the proposed strategies for problem size, density, and *r* value. We compare the performance of **Algorithms 3**, **4,** and **5** for *r*=2 on very-large-scale QUBO instances. We also compare the best algorithm among **Algorithms 3, 4,** and **5** with one of the best algorithms for xQx, i.e., Palubeckis's multiple start tabu search. We code the algorithms in C++ programming language. The source code of Palubeckis' multiple start tabu search algorithms coded in C++ can be downloaded from: https://www.personalas.ktu.lt/~ginpalu/

In (Palubeckis, 2004), there are five multiple start tabu search algorithms reported, and the **MST2** algorithm had the best results reported by the author. We choose the **MST2** algorithm with the default values for the parameters recommended by the author (Palubeckis, 2004). In the **MST2** algorithm, the number of iterations as the stopping criteria for the first tabu search start subroutine is 25000 * size of problem, then the **MST2** algorithm reduces the number of iterations to 10000*size of problem as the stopping criteria for the subsequent tabu search starts. Within the tabu search subroutine, if an improved solution is found, then the **MST2** algorithm invokes a local search immediately. The time-limit is checked at the end of the tabu search start subroutine. Thus, the computing time might exceed the time limit for large instances when we choose short time limits.

All algorithms in this study are compiled by GNU C++ compiler v4.8.5 and run on a single core of Intel Xeon Quad-core E5420 Harpertown processors, which have a 2.5 GHz CPU with 8 GB memory. All computing jobs were submitted through the Open PBS Job Management System to ensure both methods using the same CPU for memory usage and computing time limits on the same instance.

Preliminary results indicated that **Algorithms 3, 4,** and **5** perform well on instances with the size less than 3,000 and low density. All algorithms found the best-known solution with the time limit of 10 seconds. Thus, we only compare the results of large instances with high density and size from 3,000 to 8,000 by the **MST2** algorithm and the best algorithms among **Algorithms 3, 4,** and **5**. These benchmark instances with the size from 3,000 to 8,000 have been reported by other researchers (Alidaee et al. 2017; Glover et al. 2010; Hanafi et al. 2013; Katayama and Narihisa 2001; Lü et al. 2010; Rosenberg et al. 2016; Samorani et al. 2019; Shi et al. 2017; Wang et al. 2012). In addition, we generate some very-large-scale QUBO instances with high density and size of 30,000 using the same parameters from the benchmark instances. The largest instances in the literature, reported by Samorani et al. (2019), have the size of 15,000. We use a time limit of 600 seconds and r=2 for **Algorithms 3, 4,** and **5** on the very-large-scale instances in Table 2. We adopted the following notation for computational results:

OFV	The value of the objective function for the best solution found by each algorithm.
BFS	Best found solution among algorithms within the time limits.
TB[s]	Time to reach the best solution in seconds of each algorithm.
AT[s]	Average computing time out of 10 runs to reach OFV.
DT	%Deviation of computing time out of 10 runs to reach OFV.

Table 2 shows the results of comparison for **Algorithms 3, 4,** and **5** on very-large-scale instances out of 10 runs. **Algorithm 5** produces a better solution than **Algorithms 3** and **4** with r=2; thus, we use **Algorithm 5** with r=1 and r=2 to compare with the **MST2** algorithm. We impose a time limit of 60 seconds and 600 seconds per run with 10 runs per instance on **Algorithm 5** and the **MST2** algorithm. We choose tabu tenure value of 100 for 1-flip and 2-flip. The instance data and solutions files are available at: https://doi.org/10.18738/T8/WDFBR5.

Table 2. Results of Algorithms 3, 4, and 5 on bqp30000 instances with the time limit of 600 seconds and r=2.

Instance ID	Size	Density	Algorithm 3		Algorithm 4		Algorithm 5	
			OFV	TB[s]	OFV	TB[s]	OFV	TB[s]
bqp30000_1	30000	0.5	127239168	591	127292467	591	127336719	592
bqp30000_2	30000	0.8	158439036	572	158472098	555	158526518	571
bqp30000_3	30000	1	179192241	584	179219781	587	179261723	590

In our implementation, we choose the time limit as the stopping criteria and check the time limit before invoking the tabu search in **Algorithm 5**. Because the **MST2** algorithm and **Algorithm 5** are not single point-based search methods, the choice of the time limit as the stopping criteria seems to be a fair performance comparison method between algorithms.

Table 3 describes the size and density of each instance and the number of times out of 10 runs to reach the OFV as well as solution deviation within the time limit for the **MST2 algorithm** and **Algorithm 5** with r=1 and r=2. The **MST2** algorithm produces a stable performance and reaches the same OFV frequently out of 10 runs. **Algorithm 5** starts from a random initial solution and can search a more diverse solution space in a short time limit. When the time limit is changed to 600 seconds, the **MST2** algorithm and **Algorithm 5** produce a better solution quality in terms of solution deviation. The %Deviation in Table 3 inside the parenthesis is measured by: 100* $(\sqrt{(x-\bar{x})^2/n}/\bar{x})$, where x is the OFV of each run and $\bar{x}$ is the mean value of OFV out of 10 runs. For some instances, the %Deviation is less than 5.0E-4 even though not all runs found the same OFV. We use 0.000 as the value %Deviation when the value is rounded up to three decimal points.

Table 4 reports the computational results of a 60-second time limit, and Table 5 reports the computational results of a 600-second time limit. In Table 4, the **MST2** algorithm matches 5 out of 27 best solutions within the time limit. The 1-flip strategy in **Algorithm 5** matches 26 out of 27 best solutions while the 2-flip strategy in **Algorithm 5** matches 18 out of 27 best solutions. For the **MST2** algorithm, the computing time to find the initial solution exceeded the 60-second time limit for two large instances.

Table 3. The solution quality of MST2 and Algorithm 5 with 60- and 600-seconds time limits out of 10 runs.

Instance ID	**Size**	**Density**	**MST2 with 60s**	**r-flip with 60s**		**MST2 with 600s**	**r-flip with 600s**	
				r=1	**r=2**		**r=1**	**r=2**
bqp3000_1	3000	0.5	10(0)	10(0)	10(0)	10(0)	10(0)	10(0)
bqp3000_2	3000	0.8	10(0)	10(0)	10(0)	10(0)	10(0)	10(0)
bqp3000_3	3000	0.8	4(0.01)	7(0.007)	10(0)	10(0)	10(0)	10(0)
bqp3000_4	3000	1	10(0)	10(0)	10(0)	10(0)	10(0)	10(0)
bqp3000_5	3000	1	10(0)	9(0.003)	7(0.002)	9(0.001)	10(0)	10(0)
bqp4000_1	4000	0.5	10(0)	10(0)	10(0)	10(0)	10(0)	10(0)
bqp4000_2	4000	0.8	10(0)	10(0)	10(0)	9(0.004)	10(0)	10(0)
bqp4000_3	4000	0.8	10(0)	10(0)	10(0)	10(0)	10(0)	10(0)
bqp4000_4	4000	1	1(0.033)	10(0)	10(0)	10(0)	10(0)	10(0)
bqp4000_5	4000	1	10(0)	10(0)	10(0)	10(0)	10(0)	10(0)
bqp5000_1	5000	0.5	6(0)	1(0.002)	2(0.002)	10(0)	3(0.002)	2(0.002)
bqp5000_2	5000	0.8	10(0)	4(0.003)	1(0.002)	6(0.012)	10(0)	10(0)
bqp5000_3	5000	0.8	10(0)	7(0.001)	3(0.002)	10(0)	10(0)	10(0)
bqp5000_4	5000	1	10(0)	1(0.002)	1(0.001)	10(0)	3(0.002)	1(0.001)
bqp5000_5	5000	1	6(0.021)	9(0.003)	4(0.004)	10(0)	10(0)	10(0)
bqp6000_1	6000	0.5	10(0)	10(0)	4(0.001)	10(0)	10(0)	10(0)
bqp6000_2	6000	0.8	10(0)	4(0.001)	4(0.001)	1(0.006)	10(0)	9(0)
bqp6000_3	6000	1	9(0.002)	3(0.002)	1(0.007)	10(0)	10(0)	10(0)
bqp7000_1	7000	0.5	1(0.002)	1(0.006)	1(0.007)	10(0)	2(0.002)	4(0.002)
bqp7000_2	7000	0.8	7(0)	1(0.008)	1(0.008)	10(0)	1(0.004)	2(0.004)
bqp7000_3	7000	1	8(0.011)	3(0.021)	5(0.023)	10(0)	10(0)	10(0)
bqp8000_1	8000	0.5	10(0)	1(0.004)	1(0.005)	9(0.001)	10(0)	1(0.002)
bqp8000_2	8000	0.8	10(0)	1(0.009)	1(0.008)	10(0)	7(0.003)	10(0)
bqp8000_3	8000	1	10(0)	1(0.013)	1(0.01)	10(0)	4(0.001)	3(0.002)
bqp30000_1	30000	0.5	1(0.002)	1(0.023)	1(0.019)	7(0.018)	1(0.017)	1(0.011)
bqp30000_2	30000	0.8	10(0)	1(0.017)	1(0.016)	6(0.01)	1(0.019)	1(0.013)
bqp30000_3	30000	1	10(0)	1(0.015)	1(0.019)	2(0.037)	1(0.025)	1(0.019)

When the time limit is increased to 600 seconds, the **MST2** algorithm matches 10 out of 27 best solutions. The 1-flip strategy matches 25 out of 27 best solutions, and the 2-flip strategy matches 23 out of 27 best solutions. The 1-flip and 2-flip strategies in **Algorithm 5** perform well on the high-density large instances. There is no clear pattern that the 2-flip strategy uses more time than a 1-flip strategy on finding the same OFV. The 1-flip and 2-flip strategies in **Algorithm 5** choose the initial solution randomly and independently. The 1-flip strategy has a better performance when the time limits are 60 seconds and 600 seconds.

Tables 6 and 7 present the time deviation of each algorithm on reaching the OFV for each instance. The **MST2** algorithm has less variation in computing time when it finds the same OFV while the *r*-flip strategy in **Algorithm 5** has a wider range of computing time. If the algorithm only finds the OFV once out of 10 runs, the time deviation will be zero.

Table 4. Results of MST2 and *r*-flip strategy in **Algorithm 5** within the 60 seconds time limit.

Instance ID	BFS (60s)	MST2 (60s)		r-flip (60s)			
		OFV	TB[s]	OFV(r=1)	TB[s]	OFV(r=2)	TB[s]
bqp3000_1	3931583	3931583	10	3931583	3	3931583	8
bqp3000_2	5193073	5193073	25	5193073	2	5193073	2
bqp3000_3	5111533	5111533	52	5111533	8	5111533	4
bqp3000_4	5761822	5761437	10	5761822	2	5761822	2
bqp3000_5	5675625	5675430	24	5675625	7	5675625	17
bqp4000_1	6181830	6181830	40	6181830	3	6181830	4
bqp4000_2	7801355	7797821	12	7801355	13	7801355	4
bqp4000_3	7741685	7741685	31	7741685	5	7741685	8
bqp4000_4	8711822	8709956	58	8711822	5	8711822	11
bqp4000_5	8908979	8905340	27	8908979	4	8908979	13
bqp5000_1	8559680	8556675	56	8559680	21	8559680	7
bqp5000_2	10836019	10829848	34	10836019	59	10836019	11
bqp5000_3	10489137	10477129	28	10489137	20	10489137	16
bqp5000_4	12251710	12245282	52	12251710	54	12251520	42
bqp5000_5	12731803	12725779	56	12731803	17	12731803	16
bqp6000_1	11384976	11377315	42	11384976	12	11384976	5
bqp6000_2	14333855	14330032	39	14333855	27	14333767	14
bqp6000_3	16132915	16122333	51	16130731	24	16132915	48
bqp7000_1	14477949	14467157	56	14477949	41	14476263	21
bqp7000_2	18249948	18238729	55	18249948	47	18246895	47
bqp7000_3	20446407	20431354	59	20446407	15	20446407	12
bqp8000_1	17340538	17326259	47	17340538	26	17340538	35
bqp8000_2	22208986	22180465	55	22208986	54	22208683	53
bqp8000_3	24670258	24647248	56	24670258	43	24669351	50
bqp30000_1	127252438	126732483	60	127252438	58	127219336	60
bqp30000_2	158384175	157481366	69	158384175	59	158339497	60
bqp30000_3	179103085	178093109	89	179103085	58	179029747	54

Table 5. Results of MST2 and *r*-flip strategy in **Algorithm 5** within the 600 seconds time limit.

Instance ID	**BFS (600s)**	**MST2 (600s)**		**r-flip (600s)**			
		OFV	**TB[s]**	**OFV(r=1)**	**TB[s]**	**OFV(r=2)**	**TB[s]**
bqp3000_1	3931583	3931583	11	3931583	5	3931583	5
bqp3000_2	5193073	5193073	25	5193073	1	5193073	3
bqp3000_3	5111533	5111533	52	5111533	30	5111533	8
bqp3000_4	5761822	5761822	269	5761822	1	5761822	2
bqp3000_5	5675625	5675625	505	5675625	43	5675625	29
bqp4000_1	6181830	6181830	40	6181830	4	6181830	2
bqp4000_2	7801355	7800851	530	7801355	8	7801355	8
bqp4000_3	7741685	7741685	30	7741685	5	7741685	2
bqp4000_4	8711822	8711822	67	8711822	2	8711822	7
bqp4000_5	8908979	8906525	65	8908979	4	8908979	13
bqp5000_1	8559680	8559075	324	8559680	9	8559680	27
bqp5000_2	10836019	10835437	541	10836019	17	10836019	21
bqp5000_3	10489137	10488735	400	10489137	29	10489137	38
bqp5000_4	12252318	12249290	265	12252318	127	12251848	143
bqp5000_5	12731803	12731803	265	12731803	19	12731803	32
bqp6000_1	11384976	11384976	406	11384976	8	11384976	39
bqp6000_2	14333855	14333767	498	14333855	62	14333855	17
bqp6000_3	16132915	16128609	239	16132915	60	16132915	71
bqp7000_1	14478676	14477039	344	14478676	92	14478676	397
bqp7000_2	18249948	18242205	587	18249948	115	18249844	43
bqp7000_3	20446407	20431833	109	20446407	47	20446407	21
bqp8000_1	17341350	17337154	546	17340538	45	17341350	141
bqp8000_2	22208986	22207866	122	22208986	49	22208986	89
bqp8000_3	24670924	24669797	402	24670924	185	24670924	386
bqp30000_1	127336719	127323304	568	127332912	598	127336719	592
bqp30000_2	158561564	158438942	573	158561564	580	158526518	571
bqp30000_3	179329754	179113916	575	179329754	599	179261723	590

Table 6. Computing the time deviation of MST2 and r-flip strategy in Algorithm 5 within the 60 seconds time limit.

Instance ID	MST2		1-flip		2-flip	
	AT[s]	DT	AT[s]	DT	AT[s]	DT
bqp3000_1	12.5	15.663	15.7	84.075	24.1	56.875
bqp3000_2	32.5	20.059	4.9	45.583	4.8	68.606
bqp3000_3	54.3	5.901	29.3	46.180	12.0	60.477
bqp3000_4	13.3	18.434	12.0	107.798	10.0	51.640
bqp3000_5	30.4	15.829	33.2	45.470	32.3	48.240
bqp4000_1	49.0	10.227	7.7	50.131	9.3	36.570
bqp4000_2	14.2	9.848	22.1	62.351	21.1	70.476
bqp4000_3	35.4	11.236	15.1	69.699	16.5	75.542
bqp4000_4	58.0	0	22.2	65.408	35.4	45.780
bqp4000_5	31.1	11.082	18.1	73.038	29.6	40.109
bqp5000_1	56.8	2.339	4.0	0	10.0	42.426
bqp5000_2	35.0	3.563	28.3	74.329	11.0	0
bqp5000_3	30.0	8.607	38.9	33.038	33.0	50.069
bqp5000_4	54.0	5.238	54.0	0	42.0	0
bqp5000_5	57.2	1.317	38.8	40.504	30.5	63.379
bqp6000_1	43.3	5.112	32.9	47.380	21.8	67.507
bqp6000_2	39.8	4.238	31.8	51.521	42.3	46.315
bqp6000_3	52.1	5.027	32.3	44.640	48.0	0
bqp7000_1	56.0	0	41.0	0	21.0	0
bqp7000_2	55.3	1.367	47.0	0	47.0	0
bqp7000_3	59.0	0	42.0	56.293	35.2	45.958
bqp8000_1	48.1	7.232	26.0	0	35.0	0
bqp8000_2	55.9	0.566	54.0	0	53.0	0
bqp8000_3	57.2	1.806	43.0	0	50.0	0
bqp30000_1	60.0	0	58.0	0	60.0	0
bqp30000_2	70.0	1.166	59.0	0	60.0	0
bqp30000_3	90.3	0.912	58.0	0	54.0	0

Table 7. Computing the time deviation of MST2 and *r*-flip strategy in **Algorithm 5** within the 600 seconds time limit.

Instance ID	MST2		1-flip		2-flip	
	AT[s]	DT	AT[s]	DT	AT[s]	DT
bqp3000_1	11.9	16.067	21.3	62.678	37.8	130.045
bqp3000_2	29.1	18.144	5.5	69.234	7.7	81.231
bqp3000_3	58.5	16.889	126.4	99.610	143.6	116.812
bqp3000_4	292.1	11.285	10.8	58.365	13.7	49.512
bqp3000_5	543.9	6.365	109.1	88.328	164.3	72.144
bqp4000_1	42.8	7.773	13.5	63.553	15.1	63.861
bqp4000_2	552.2	2.597	30.7	49.923	37.0	110.712
bqp4000_3	31.7	7.293	18.8	57.442	22.9	47.545
bqp4000_4	70.8	4.094	42.2	102.331	48.7	115.376
bqp4000_5	70.5	8.596	25.3	81.984	43.0	69.595
bqp5000_1	337.2	4.265	70.7	126.287	48.0	61.872
bqp5000_2	557.8	4.565	135.4	96.462	210.6	67.670
bqp5000_3	428.5	7.257	115.8	96.524	115.5	104.238
bqp5000_4	279.4	5.987	270.3	48.842	143.0	0
bqp5000_5	287.1	9.234	194.5	94.641	172.7	92.268
bqp6000_1	424.8	4.555	152.9	95.641	145.9	46.657
bqp6000_2	498.0	0	142.5	97.375	73.8	122.070
bqp6000_3	252.3	5.571	248.0	51.129	318.1	61.672
bqp7000_1	344.5	0.153	272.5	93.675	441.5	12.974
bqp7000_2	587.0	0	115.0	0	265.1	76.694
bqp7000_3	109.0	0	84.5	39.472	131.1	55.401
bqp8000_1	548.6	1.398	251.3	63.346	141.0	0
bqp8000_2	145.4	11.829	258.3	56.352	300.7	56.001
bqp8000_3	514.3	11.365	368.8	43.667	417.3	12.797
bqp30000_1	572.0	1.365	598.0	0	592.0	0
bqp30000_2	581.5	1.526	580.0	0	571.0	0
bqp30000_3	586.5	2.773	599.0	0	590.0	0

The *r*-flip strategy can be embedded in other local search heuristics as an improvement procedure. The clever implementation of the *r*-flip strategy can reduce the computing time as well as improve the solution quality. We reported the time and solutions out of 10 runs for each instance. If all 10 runs report the same solution by the **MST2** algorithm and the *r*-flip strategy in **Algorithm 5**, the computational time of the *r*-flip strategy in **Algorithm 5** is much smaller regardless of the size of the instances. The time deviation and solution deviation of 10 runs with the short time limits are computed due to the computing resources available to this study; the samples from 10 runs might not follow a normal distribution. Thus, there is no statistical evidence to distinguish the computing time and solution quality between the 1-flip and 2-flip strategies.

4. Conclusion

In this paper, we considered the QUBO problem and provided several results including a necessary and sufficient condition for local optimality of an *r*-flip search when a 1-flip search has already reached local optimality. Our computational results show candidates to be considered for an *r*-flip implementation can be reduced significantly, and the new *r*-flip strategy is able to solve very large scale QUBO instances within 600 seconds while it uses a fractional of time to reach the best-known solutions on the benchmark instances. The results are attractive in the situations where variable neighborhood strategies are being implemented on the very-large-scale problems or sparce matrices.

References

Ahuja, R. K., Ergun, Ö., Orlin, J. B., & Punnen, A. P. (2002). A survey of very large-scale neighborhood search techniques. *Discrete Applied Mathematics*, *123*(1), 75-102. https://doi.org/https://doi.org/10.1016/S0166-218X(01)00338-9

Ahuja, R. K., Orlin, J. B., Pallottino, S., Scaparra, M. P., & Scutellà, M. G. (2004). A Multi-Exchange Heuristic for the Single-Source Capacitated Facility Location Problem. *Management Science*, *50*(6), 749-760. https://doi.org/10.1287/mnsc.1030.0193

Alidaee, B. (2004). *Fan-and-Filter Neighborhood Strategy for 3-SAT Optimization, Hearin Center for Enterprise Science*. The University of Mississippi.

Alidaee, B., Kochenberger, G., & Wang, H. (2010). Theorems Supporting r-flip Search for Pseudo-Boolean Optimization. *Int. J. Appl. Metaheuristic Comput.*, *1*(1), 93-109. https://doi.org/10.4018/jamc.2010102605

Alidaee, B., Kochenberger, G. A., & Ahmadian, A. (1994). 0-1 Quadratic programming approach for optimum solutions of two scheduling problems. *International Journal of Systems Science*, *25*(2), 401-408. https://doi.org/10.1080/00207729408928968

Alidaee, B., Sloan, H., & Wang, H. (2017). Simple and fast novel diversification approach for the UBQP based on sequential improvement local search. *Computers & Industrial Engineering*, *111*, 164-175. https://doi.org/http://dx.doi.org/10.1016/j.cie.2017.07.012

Alidaee, B., & Wang, H. (2017). A note on heuristic approach based on UBQP formulation of the maximum diversity problem. *Journal of the Operational Research Society*, *68*(1), 102-110. https://doi.org/10.1057/s41274-016-0031-4

Anacleto, E. A. J., Meneses, C. N., & Ravelo, S. V. (2020). Closed-form formulas for evaluating r-flip moves to the unconstrained binary quadratic programming problem. *Computers & Operations Research*, *113*, 104774. https://doi.org/https://doi.org/10.1016/j.cor.2019.104774

Beasley, J. (1998). Heuristic algorithms for the unconstrained binary quadratic programming problem.

Boros, E., Hammer, P. L., Minoux, M., & Rader, D. J. (1999). Optimal cell flipping to minimize channel density in VLSI design and pseudo-Boolean optimization. *Discrete Applied Mathematics*, *90*(1), 69-88. https://doi.org/https://doi.org/10.1016/S0166-218X(98)00114-0

Boros, E., Hammer, P. L., & Tavares, G. (2007). Local search heuristics for Quadratic Unconstrained Binary Optimization (QUBO). *Journal of Heuristics*, *13*(2), 99-132. https://doi.org/10.1007/s10732-007-9009-3

Chen, W. (2015). Optimality Conditions for the Minimization of Quadratic 0-1 Problems. *SIAM Journal on Optimization*, *25*(3), 1717-1731. https://doi.org/10.1137/140968409

Glover, F. (1998). A template for scatter search and path relinking. In J.-K. Hao, E. Lutton, E. Ronald, M. Schoenauer, & D. Snyers, *Artificial Evolution* Berlin, Heidelberg.

Glover, F., Kochenberger, G., & Du, Y. (2019). Quantum Bridge Analytics I: a tutorial on formulating and using QUBO models. *4OR*, *17*(4), 335-371. https://doi.org/10.1007/s10288-019-00424-y

Glover, F., Kochenberger, G. A., & Alidaee, B. (1998). Adaptive Memory Tabu Search for Binary Quadratic Programs. *Management Science*, *44*(3), 336-345. https://doi.org/10.1287/mnsc.44.3.336

Glover, F., Lewis, M., & Kochenberger, G. (2018). Logical and inequality implications for reducing the size and difficulty of quadratic unconstrained binary optimization problems. *European Journal of Operational Research*, *265*(3), 829-842. https://doi.org/https://doi.org/10.1016/j.ejor.2017.08.025

Glover, F., Lü, Z., & Hao, J.-K. (2010). Diversification-driven tabu search for unconstrained binary quadratic problems. *4OR*, *8*(3), 239-253. https://doi.org/10.1007/s10288-009-0115-y

Hanafi, S., Rebai, A.-R., & Vasquez, M. (2013). Several versions of the devour digest tidy-up heuristic for unconstrained binary quadratic problems. *Journal of Heuristics*, *19*(4), 645-677. https://doi.org/10.1007/s10732-011-9169-z

Katayama, K., & Narihisa, H. (2001). Performance of simulated annealing-based heuristic for the unconstrained binary quadratic programming problem. *European Journal of Operational Research*, *134*(1), 103-119. https://doi.org/https://doi.org/10.1016/S0377-2217(00)00242-3

Kochenberger, G., & Glover, F. (2013). Introduction to special xQx issue. *Journal of Heuristics*, *19*(4), 525-528. https://doi.org/10.1007/s10732-013-9227-9

Kochenberger, G., Hao, J.-K., Glover, F., Lewis, M., Lü, Z., Wang, H., & Wang, Y. (2014). The unconstrained binary quadratic programming problem: a survey. *Journal of Combinatorial Optimization*, *28*(1), 58-81. https://doi.org/10.1007/s10878-014-9734-0

Kochenberger, G. A., Glover, F., Alidaee, B., & Rego, C. (2004). A unified modeling and solution framework for combinatorial optimization problems. *OR Spectrum*, *26*(2), 237-250. https://doi.org/10.1007/s00291-003-0153-3

Lewis, M., Alidaee, B., Glover, F., & Kochenberger, G. (2009). A note on xQx as a modelling and solution framework for the Linear Ordering Problem. *International Journal of Operational Research*, *5*(2), 152-162. https://doi.org/10.1504/ijor.2009.025005

Lewis, M., Alidaee, B., & Kochenberger, G. (2005). Using xQx to model and solve the uncapacitated task allocation problem. *Operations Research Letters*, *33*(2), 176-182. https://doi.org/https://doi.org/10.1016/j.orl.2004.04.014

Li, W., & Alidaee, B. (2002). Dynamics of local search heuristics for the traveling salesman problem. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, *32*(2), 173-184. https://doi.org/10.1109/TSMCA.2002.1021106

Lin, S., & Kernighan, B. W. (1973). An Effective Heuristic Algorithm for the Traveling-Salesman Problem. *Operations Research*, *21*(2), 498-516. https://doi.org/10.1287/opre.21.2.498

Lü, Z., Glover, F., & Hao, J.-K. (2010). A hybrid metaheuristic approach to solving the UBQP problem. *European Journal of Operational Research*, *207*(3), 1254-1262. https://doi.org/https://doi.org/10.1016/j.ejor.2010.06.039

Merz, P., & Freisleben, B. (2002). Greedy and Local Search Heuristics for Unconstrained Binary Quadratic Programming. *Journal of Heuristics*, *8*(2), 197-213. https://doi.org/10.1023/A:1017912624016

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, *24*(11), 1097-1100. https://doi.org/https://doi.org/10.1016/S0305-0548(97)00031-2

Palubeckis, G. (2004). Multistart Tabu Search Strategies for the Unconstrained Binary Quadratic Optimization Problem. *Annals of Operations Research*, *131*(1), 259-282. https://doi.org/10.1023/B:ANOR.0000039522.58036.68

Resende, M. G. C. (2008). Metaheuristic Hybridization with Greedy Randomized Adaptive Search Procedures. In *State-of-the-Art Decision-Making Tools in the Information-Intensive Age* (pp. 295-319). https://doi.org/10.1287/educ.1080.0045

Rosenberg, G., Vazifeh, M., Woods, B., & Haber, E. (2016). Building an iterative heuristic solver for a quantum annealer. *Computational Optimization and Applications*, *65*(3), 845-869. https://doi.org/10.1007/s10589-016-9844-y

Samorani, M., Wang, Y., Lv, Z., & Glover, F. (2019). Clustering-driven evolutionary algorithms: an application of path relinking to the quadratic unconstrained binary optimization problem. *Journal of Heuristics*, *25*(4-5), 629-642. https://doi.org/http://dx.doi.org/10.1007/s10732-018-9403-z

Shi, J., Zhang, Q., Derbel, B., & Liefooghe, A. (2017, 5-8 June 2017). A Parallel Tabu Search for the Unconstrained Binary Quadratic Programming problem. 2017 IEEE Congress on Evolutionary Computation (CEC),

Szeider, S. (2011). The parameterized complexity of k-flip local search for SAT and MAX SAT. *Discrete Optimization*, *8*(1), 139-145. https://doi.org/https://doi.org/10.1016/j.disopt.2010.07.003

Wang, H., & Alidaee, B. (2018). Unrelated Parallel Machine Selection and Job Scheduling With the Objective of Minimizing Total Workload and Machine Fixed Costs. *IEEE Transactions on Automation Science and Engineering*, *15*(4), 1955 - 1963. https://doi.org/10.1109/TASE.2018.2832440

Wang, H., & Alidaee, B. (2019a). Effective heuristic for large-scale unrelated parallel machines scheduling problems. *Omega*, *83*, 261-274. https://doi.org/https://doi.org/10.1016/j.omega.2018.07.005

Wang, H., & Alidaee, B. (2019b). The multi-floor cross-dock door assignment problem: Rising challenges for the new trend in logistics industry. *Transportation Research Part E: Logistics and Transportation Review*, *132*, 30-47. https://doi.org/https://doi.org/10.1016/j.tre.2019.10.006

Wang, H., Alidaee, B., Ortiz, J., & Wang, W. (2020). The multi-skilled multi-period workforce assignment problem. *International Journal of Production Research*, 1-18. https://doi.org/10.1080/00207543.2020.1783009

Wang, Y., Lü, Z., Glover, F., & Hao, J.-K. (2012). Path relinking for unconstrained binary quadratic programming. *European Journal of Operational Research*, *223*(3), 595-604. https://doi.org/https://doi.org/10.1016/j.ejor.2012.07.012

Yagiura, M., & Ibaraki, T. (1999). Analyses on the 2 and 3-Flip Neighborhoods for the MAX SAT. *Journal of Combinatorial Optimization*, *3*(1), 95-114. https://doi.org/10.1023/A:1009873324187

Yagiura, M., & Ibaraki, T. (2001). Efficient 2 and 3-Flip Neighborhood Search Algorithms for the MAX SAT: Experimental Evaluation. *Journal of Heuristics*, *7*(5), 423-442. https://doi.org/10.1023/A:1011306011437

Yagiura, M., Kishida, M., & Ibaraki, T. (2006). A 3-flip neighborhood local search for the set covering problem. *European Journal of Operational Research*, *172*(2), 472-499. https://doi.org/https://doi.org/10.1016/j.ejor.2004.10.018