algorithms

MDPI

Article

An Efficient Closed-Form Formula for Evaluating *r*-Flip Moves in Quadratic Unconstrained Binary Optimization

Bahram Alidaee [1], Haibo Wang [2] and Lutfu S. Sua [3,*]

[1] School of Business Administration, The University of Mississippi, University, Oxford, MS 38677, USA; balidaee@bus.olemiss.edu
[2] A.R. Sánchez Jr. School of Business, Texas A&M International University, Laredo, TX 78041, USA; hwang@tamiu.edu
[3] Department of Management and Marketing, Southern University and A&M College, Baton Rouge, LA 70807, USA
* Correspondence: lutfu.sagbansua@subr.edu

Abstract: Quadratic unconstrained binary optimization (QUBO) is a classic NP-hard problem with an enormous number of applications. Local search strategy (LSS) is one of the most fundamental algorithmic concepts and has been successfully applied to a wide range of hard combinatorial optimization problems. One LSS that has gained the 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 obtain a new solution if the changes improve the objective function. The main purpose of this paper is to develop several results for the implementation of *r*-flip moves in QUBO, including a necessary and sufficient condition that when a 1-flip search reaches local optimality, the number of candidates for implementation of the *r*-flip moves can be reduced significantly. The results of the substantial computational experiments are reported to compare an *r*-flip strategy-embedded algorithm and a multiple start tabu search algorithm on a set of benchmark instances and three very-large-scale QUBO instances. The *r*-flip strategy implemented within the algorithm makes the algorithm very efficient, leading to very high-quality solutions within a short CPU time.

Keywords: combinatorial optimization; quadratic unconstrained binary optimization; local optimality; *r*-flip local optimality

Citation: Alidaee, B.; Wang, H.; Sua, L.S. An Efficient Closed-Form Formula for Evaluating *r*-Flip Moves in Quadratic Unconstrained Binary Optimization. *Algorithms* **2023**, *16*, 557. https://doi.org/10.3390/a16120557

Academic Editor: Francisco Cuevas de la Rosa

Received: 11 November 2023
Revised: 28 November 2023
Accepted: 2 December 2023
Published: 5 December 2023

1. Introduction

Quadratic unconstrained binary optimization is a classic NP-hard problem that has been used as a unifying approach to many combinatorial optimization problems [1,2]. 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 [3–10]. However, due to the complexity and practicality of QUBO, it is still necessary to provide results suitable for solving large-scale problems. In recent years, researchers have developed theoretical results to reduce the algorithmic implementation difficulty of QUBO, [11–16]. Our results in this paper also help to reduce the size and difficulty of the algorithmic implementation of these problems.

Quadratic unconstrained binary optimization (QUBO) can be formulated as:

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

In Equation (1), $\frac{1}{2}q_{i,j}$ is the *i,j*-th entry of a given *n* by *n* symmetric matrix *Q*. QUBO is often referred to as the $x^T Qx$ model [17]. Since ${x_i}^2 = x_i \ \ for \ \ x_i = 0, 1$, and *Q* may be

written as an upper triangular matrix by doubling each entry of the upper triangle part of the matrix and letting $q_{i,i} = q_i$, then we can write Equation (1) as Equation (2).

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

Local search strategy (LSS) is one of the most fundamental algorithmic concepts and has been successfully applied to a wide range of hard combinatorial optimization problems. The basic ingredient of almost all sophisticated heuristics is some variation 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 [11,18–22]. In Table A4, we present a comprehensive review of *r*-flip strategies applied to QUBO. Let $N = \{1, 2, \ldots, n\}$. Given a binary solution, $x = (x_1, \cdots, x_n)$ of $x^T Q x$, 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) [23,24], variable neighborhood search (VNS) [25,26], and multi-exchange neighborhood search (MENS) [18–22] 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 execution of an implementation of an *r*-flip local search, for a larger value of r, can be computationally expensive to execute. This is because the size of the search space is of order n chosen r, and for fixed values of n, it grows quickly in r for the value of $r \leq \lfloor n/2 \rfloor$. Hence, 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 a small value of r (e.g., $r = 1$) can quickly reach local optimality. Thus, as a way to escape 1-flip local optimality, researchers have tried to dynamically change the value of r as the search progressed. This offers 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 but also possibly reach better solutions because the larger values of r provide an opportunity to search a more diverse part of the solution space.

Previous Works

The development of closed-form formulas for *r*-flip moves is desirable for developing heuristics for solving very-large-scale problem instances because it can reduce computational time consumed by the implementation of an algorithm. Alidaee and Kochenberger [11] introduced several theorems showing closed-form *r*-flip formulas for general pseudo-Boolean optimization. The authors in [12,13] recently provided closed-form formulas for evaluating *r*-flip rules in QUBO. In particular, Theorem 6 in [11] is specific to the $f(x) = x^T Q x$ problem. To explain the closed-form formula for the *r*-flip rule in $x^T Q x$, we first introduce a few definitions. Refer to Table A4 for an exhaustive list of literature on *r*-flip rules applied to QUBO.

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

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

Fact 1. Given a solution vector $x = (x_1, \cdots, x_i, \cdots, x_n)$, and a solution $x' = (x_1, \cdots, 1 - x_i, \cdots, x_n)$ obtained by flipping the i-th element of x, we have:

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

It is well known that any locally optimal solution to an instance of the QUBO problem with respect to a 1-flip search satisfies

$$Either\ ((x_i = 0)\ iff\ E(x_i) < 0)\ or\ (x_i = 1\ iff\ \mathrm{E}(x_i) \geq 0),\ for\ i = 1, \cdots, n \tag{5}$$

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

$$\begin{aligned} &\forall j < i, E(x_j) \leftarrow E(x_j) + q_{j,i}(x'_i - x_i) \\ &\forall j > i, E(x_j) \leftarrow E(x_j) + q_{i,j}(x'_i - x_i) \\ &=,\ E(x_j) \leftarrow E(x_j) \quad j = 1, \ldots, n \end{aligned} \tag{6}$$

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 Algorithm 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 [27].

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, $k \leq r$;
$\Pi(i)$: The i-th element of x in the order $\pi(1) \cdots \pi(n)$;
S: = {i:x_i is tentatively chosen to receive a new value to produce a new solution x_i'} restricting consideration to $|S| = r$;
D: The set of candidates for an improving move;
Tabu_ten: The maximum number of iterations for which a variable can remain tabu;
Tabu(i): A vector representing tabu status of x;
$E(x_i)$: Derivative of $f(x)$ with respect to x_i;
$E(x) = (E(x_1), \ldots, E(x_n))$ The vector of derivatives;
X(.): A vector representing the solution of x;
E(.): A vector representing the value of derivative $E(x_i)$.

Algorithm 1: 1-flip local search.

Initialize: n, x, evaluate the 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 the vector E(x) using Equation (6), Flag = 1*
6 *End do*
7 *End while*

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

(Theorem 6, Alidaee and Kochenberger [11]) Let x be a given solution of QUBO and x' obtained from x by the r-flip move (for a chosen set S), where $S \subseteq N$, $|S| = r$. The change in the value of the objective function is:

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

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

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

As explained in [11], the evaluation of change in the objective function of Equation (7) can be done in $O(r^2)$, i.e., evaluating $f(x')$ from $f(x)$. The update in Equation (8) requires r calculations for each j in $N\backslash S$, and $r-1$ calculations for each j in S. Thus, overall, the update for all n variables can be performed in $O(nr)$.

Note that for any two elements $i,j = 1, \ldots, n$, and $i < j$, we can define:

$$\begin{aligned} E'(x_i) &= E(x_i) - q_i - q_{i,j}x_j, \\ E'(x_j) &= E(x_j) - q_j - q_{i,j}x_i. \end{aligned} \tag{9}$$

Using Equation (9), a useful way to express Equation (7) is Equation (10).

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

A simple exhaustive r-flip search is provided in Algorithm 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 for the efficient implementation of an r-flip search within an algorithm.

Algorithm 2: Exhaustive r-flip local search.

Initialize: n, x, evaluate the vector E(x), value of r
Flag = 1
1 *Do while (Flag = 1)*
2 *Flag = 0*
3 *For each combination* $S \subset N$ *and* $|S| \leq r$, *evaluate* Δf, *Equation (7):*
If $\Delta f > 0$:
$x_i = 1 - x_1$, *for* $i \in S$, *update E(x) using Equation (8), Flag = 1*
4 *End while*

2. New Results on Closed-Form Formulas

We first introduce some notations. For $m < n$, define $\binom{n}{m}$ to be the number of combinations of m elements out of n, and let $\varphi = \underset{i,j\in N}{\text{Max}}\{|q_{i,j}|\}$ and $M = \varphi * (2, r)$. Furthermore, Lemmas 1 and 2, presented below, help to prove the results. Note that Lemma 1 is a direct deduction from previous results [11].

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

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

Proof. The condition of local optimality in Equation (5) indicates that:

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

Using this condition, we thus have:

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

□

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

$$\sum_{i,j \in S} (x'_i - x_i)\left(x'_j - x_j\right) q_{i,j} \leq M \tag{12}$$

Proof. For each pair of elements, $i, j \in S$, the left-hand side can be $q_{i,j}$ or $-q_{i,j}$. 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 $x^T Qx$ 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:*

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

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

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

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 Equation (15); that completes the proof.

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

□

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 $x^T Qx$ 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, based on Theorem 1, we have:

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

Using Lemma 2, we also have Equation (17), which completes the proof.

$$\sum_{i \in S} |E(x_i)| < \sum_{i,j \in S} (x'_i - x_i)(x'_j - x_j) q_{i,j} \leq M \tag{17}$$

□

The 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 $|S| = r$ that satisfies Equation (13), then x is also locally optimal solution with respect to an r-flip search. Furthermore, if there is no subset S of any size that satisfies Equation (13), then x is also locally optimal solution with respect to an r-flip search for all $r \leq n$. Similar statements are also true regarding Proposition 1.

The result of Proposition 1 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 locally optimal solution, only those elements with the sum of the absolute value of derivatives less than M are eligible for consideration. Furthermore, when deciding on the elements of an r-flip search, we can easily check to see whether 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 x^TQx 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 Equation (18), must be satisfied. Of course, if the last inequality in Equation (18) is satisfied, all other inequalities are also satisfied. This means that each subset of the S is also an improving move. This is important in any dynamic neighborhood search strategy k-flip move for $k \leq r$ in consideration.*

$$\begin{array}{l} \text{Here we have } \varphi = \underset{i,j \in N}{Max}\{|q_{ij}|\} \text{ and } M = 6 * \varphi : \\ |E(x_a)| < M \text{ for } a = i, j, k, l, \\ |E(x_a)| + |E(x_b)| < M \text{ for } (a \neq b), a, b = i, j, k, l, \\ |E(x_a)| + |E(x_b)| + |E(x_c)| < M \text{ for } (a \neq b \neq c), a, b, c = i, j, k, l, \\ \qquad |E(x_a)| + |E(x_b)| + |E(x_c)| + |E(x_d)| < M \text{ for } a = i, b = j, c = k, d = l \end{array} \tag{18}$$

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. In addition, a numerical example is given in Appendix A.

2.1. Strategy 1

We first define a set, $D(n)$, of candidates for improving moves. 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 Equation (19).

$$\left|E\left(x_{\pi(1)}\right)\right| \leq \cdots \leq \left|E\left(x_{\pi(n)}\right)\right| \tag{19}$$

Here, $\pi(i)$ means the i-th element of x in the order $(\pi(1), \cdots, \pi(n))$. Let K be the largest value of k = 1, 2,. . ., n, where Inequality (20) is satisfied. The set $D(n)$ is now defined by Equation (21).

$$\sum_{i=1}^{k} \left|E\left(x_{\pi(i)}\right)\right| < M \text{ for } k = 1, 2, 3, \cdots, n \tag{20}$$

$$D(n) = \left\{x_{\pi(1)}, \cdots, x_{\pi(K)}\right\} \tag{21}$$

Lemma 3. *Any subset $S \subseteq D(n)$ satisfies 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., Inequality (19):

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 $k \leq r$ (this might be due to the fact that the right-hand-side value M in Equation (20) for a given r is constant. Thus, smaller values of $|E(x_i)|$ on the left-hand side might help to satisfy the inequality more easily.

II. Because the elements of $D(n)$ are in 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 Equation (22). Note that there are many more subsets of $D(n)$ compared to the sets in Equation (21) that are candidates for consideration in possible k-flip moves. Here, we only provide one possible efficient implementable strategy.

$$S \in \{\{\pi(1), \pi(2)\}, \{\pi(1), \pi(2), \pi(3)\}, \cdots, \{\pi(1), \cdots, \pi(K)\}\} \tag{22}$$

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 $r > 1$, the locally optimal solution 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 could 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 sparse. Proposition 2 is a weaker condition of Proposition 1 that can help be overcome to a certain 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:

$$Either \ \ (x_i = 0 \ iff \ E(i) < 0) \ \ or \ \ (x_i = 1 \ \ iff \ E(i) \geq 0) \text{ for } i \in S \tag{23}$$

□

Of course, if we have N in Equation (23) 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, $\varphi_S = \max_{i,j \in S} \{|q_{i,j}|\}$, and $M_S = \varphi_S * (2, r)$. With these definitions, we now state Proposition 2.

Proposition 2. *(Weak necessary condition): Let $S \subseteq N$, $|S| = r$, and φ_S and M_S are as defined above. Given any solution $x = (x_1, \cdots, x_n)$ of $x^T Qx$, 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 the 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 sparse, for many variables, the values of derivatives are "unaffected" by the change in values of the elements in S. This means that 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 using Equation (20), 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 for such a strategy.

$$D(1) = \{x_i : |E(x_i)| < M\} \tag{24}$$

Corollary 1 . *Let φ and M be as defined before. Given a solution $x = (x_1, \cdots, x_n)$ of $x^T Qx$, if the 1-flip local search cannot further improve the value of f(x), and $i \in S$ with $S \subseteq N$, 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 carried out some experimentation to find the size of set $D(1)$ for different sizes of instances. The steps of the experiment to find the size of $D(1)$ are provided below. The problems considered were taken from Ref. [26] and have been used by many researchers. We only used the larger-scale problems with 2500 to 6000 variables for a total of 38 instances.

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

- Step 1. Randomly initialize a solution to the problem. For each value of r, calculate M. Apply the algorithm in Algorithm 1 and generate a locally optimal solution x with respect to a 1-flip search. However, in Step 5 of Algorithm 1, only consider the derivatives with $|E(x_i)| < M$.
- Step 2. Find the number of elements in set $D(1)$ for x.
- Step 3. Repeat steps 1 and 2 200 times for each problem and find the average number of elements in 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, in general we can say that, as the density of matrix Q increased, the size of $D(1)$ decreased for all problem sizes and values of r. This is, of course, due to the fact that the larger density of Q made the derivative of each element in x more related to other elements. As the size of a problem increases, the size of $D(1)$ also increases.

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

	***r* = 2**				***r* = 3**				***r* = 4**			
	Density											
Prob. 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

An interesting observation in our experiment is that, in most cases, the size of $D(1)$ for better locally optimal solutions was smaller than those with the worse locally optimal solutions. This indicates that, as the search reaches closer to the globally optimal solutions, the time for an r-flip search decreases when we take advantage of Corollary 1.

2.3. Implementation Details

We first implement two strategies in Sections 2.1 and 2.2 via Algorithm 3 for Strategy 1 and Algorithm 4 for Strategy 2, and then proposed Algorithm 5 for Strategy 2 embedded with a simple tabu search algorithm for the improvement in Algorithm 5. A summary of the methodology used in this research is provided as follows:

We started by choosing the appropriate subset S to implement a move. There were two strategies employed at this step that required the utilization of two different algorithms.

Algorithm 3 was applied for Strategy 1, which involved defining a set of candidates for improving moves, given a locally optimal solution with respect to a 1-flip move.

Algorithm 4 was applied for Strategy 2, which involved using an individual element to create a set of candidates for applying an r-flip search.

Algorithm 5 was applied for improvement, which involved a tabu search algorithm.

The detailed pseudo-code of Algorithms 3–5 is provided below:

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 $\left|E\left(x_{\pi(i)}\right)\right| \leq \left|E\left(x_{\pi(i+1)}\right)\right|$, *using Inequality (20) 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 $\sum_{i=1}^{j}\left|E\left(x_{\pi(i)}\right)\right| < M_{S_j}$, *evaluate* Δf *using Equation (7).*
7 If $\Delta f > 0$:
$x_i = 1 - x_i$, for $i \in S_j$, *update E(x) using Equation (8), Flag = 1, go to Step 1*
8 End for
9 End while

Algorithm 4: *r*-flip local search: strategy 2

Initialize: n, x, evaluate E(x), value of r, M
Flag = 1
1 Do while (Flag = 1)
2 Flag $= 0$, *and* $S = \varnothing$
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 $\left|E\left(x_j\right)\right| < M$, *and* $|S \cup \{j\}| \leq r$ *evaluate* Δf *for* $S \cup \{j\}$ *using Equation (7)*
7 If $\Delta f > 0$:
$x_i = 1 - x_i$, *for* $i \in S \cup \{j\}$, *update E(x) using Equation (8),* $S = S \cup \{j\}$,
Flag =1, go to Step 1
8 End for
9 End while

Algorithm 5: Hybrid *r*-flip/1-flip local search embedded with a simple tabu search algorithm

Initialize: n, x, tabu list, evaluate E(x), value of r, M, tabu tenure
Call local search: Algorithm 4
Do while (until some stopping criteria, e.g., CPU time limit, is reached)
Call Destruction()
Call Construction()
Call randChange()
End while

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

- Step 3a. Find the variable that is not on the tabu list and lead to the small change to the solution when the variable is flipped.
- Step 3b. Change its value, place it on the tabu list to update the tabu list, and update $E(x)$.
- Step 3c. Test whether there is any variable that is not on the tabu list and that can improve the solution. If not, go to Step 3a.

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

- Step 4a. Test all of the variables that are not on the tabu list. If a solution better than the current best solution is found, change its value, place it on the tabu list, update $E(x)$, update the tabu list, and go to Step 1.
- Step 4b. Find the index i corresponding to the greatest value of $E(x_i)$, change its value of x_i, place it on the tabu list to update the tabu list, and update $E(x)$.
- Step 4c. If this is the 15th iteration in the Construction() procedure, go to Step 1.

- Step 4d. Test whether there is any variable that is not on the tabu list and that can improve the solution. If not, 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. There is a probability of less than 2% of invoking after the Construction() procedure. To get the 2% probability, a random number generator is used to create an integer between 1 and 1000. If the value of the integer is smaller than 20, the randChange() is invoked. The variable chosen in the randChange() will lead to the change of $E(x)$ for Destruction().

Any local search algorithm, e.g., Algorithms 3 or 4, can be used in Step 1 of this simple tabu search heuristic. However, a limited preliminary implementation of Algorithms 3 and 4 within Algorithm 5 suggests that, due to its simplicity of implementation and computational saving time, Algorithm 4 with slight modification was quite effective; thus, we used it in Step 1 of Algorithm 5. The slight modification was as follows. If the solution found by a 1-flip is worse than the current best-found solution, quit the local search and go to Step 2.

In order to determine whether the hybrid r-flip/1-flip local search algorithms with two strategies (Algorithms 3 and 4) perform better than the hybrid r-flip/1-flip local search embedded with a simple tabu search implementation, we compared Algorithms 3 and 4 to 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 the three algorithms of a 2-flip on very-large-scale QUBO instances in the next section.

3. Computational Results

In this study, we performed substantial computational experiments to evaluate the proposed strategies for problem size, density, and r-value. We compared the performance of Algorithms 3–5 for $r = 2$ on very-large-scale QUBO instances. We also compared the best algorithm among Algorithms 3–5 to one of the best algorithms for $x^T Qx$, i.e., Palubeckis's multiple start tabu search. We coded the algorithms in C++ programming language.

In [28], there are five multiple start tabu search algorithms, and the MST2 algorithm had the best results reported by the author. We chose the MST2 algorithm with the default values for the parameters recommended by the author [28]. In the MST2 algorithm, the number of iterations as the stopping criterion for the first tabu search start subroutine is 25,000× the size of the problem, and then the MST2 algorithm reduces the number of iterations to 10,000× the size of problem as the stopping criterion 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 CPU time limit in the MST2 algorithm is checked at the end of the tabu search start subroutine. Thus, the computing time might exceed the CPU time limit for large instances when choosing short CPU time limits.

All algorithms in this study were 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 an 8 GB memory. All computing jobs were submitted through the Open PBS Job Management System (Version 3) to ensure that both methods used the same CPU for memory usage and CPU time limits in the same instance.

Preliminary results indicate that Algorithms 3–5 performed well in instances with a size of less than 3000 and low density. All algorithms found the best-known solution with a CPU time limit of 10 s. Thus, we only compared the results of large instances with high density and a size of 3000 to 8000 with the MST2 algorithm and the best algorithm among Algorithms 3–5. These benchmark instances with sizes of 3000 to 8000 have been reported by other researchers [5,12]. In addition, we generated some very-large-scale QUBO instances with a high density and size of 30,000 using the same parameters from the benchmark instances. We used a CPU time limit of 600 s and $r = 2$ for Algorithms 3–5 in the very-large-scale instances in Table 2. We adopted the following notation for the computational results:

Table 2. Results of Algorithms 3–5 on p30000 instances with a CPU time limit of 600 s and $r = 2$.

Instance ID	Size	Density	Algorithm 3		Algorithm 4		Algorithm 5	
			OFV	TB (s)	OFV	TB (s)	OFV	TB (s)
p30000_1	30,000	0.5	127,239,168	591	127,292,467	591	127,336,719	592
p30000_2	30,000	0.8	158,439,036	572	158,472,098	555	158,526,518	571
p30000_3	30,000	1	179,192,241	584	179,219,781	587	179,261,723	590

OFV: The value of the objective function for the best solution found by each algorithm;
BFS: The best solution found among algorithms within the CPU time limit;
TB [s]: Time for each algorithm to reach the best solution in seconds;
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 the comparison for Algorithms 3–5 on very-large-scale instances out of 10 runs. Algorithm 5 produced a better solution than Algorithms 3 and 4 with $r = 2$; thus, we used Algorithm 5 with $r = 1$ and $r = 2$ to compare to the MST2 algorithm. We imposed a CPU time limit of 60 s and 600 s per run, with 10 runs per instance on Algorithm 5 and the MST2 algorithm. We choose a tabu tenure value of 100 for 1-flip and 2-flip.

In our implementation, we chose the CPU time limit as the stopping criterion and checked the CPU 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 CPU time limit as the stopping criterion seemed to be a fair performance comparison method between the algorithms.

Table 3 describes the size and density of each instance and the number of times out of 10 runs the OFV was reached as well as the solution deviation within the CPU time limit for the MST2 algorithm and Algorithm 5 with $r = 1$ and $r = 2$. The MST2 algorithm produced a stable performance and reached the same OFV frequently out of 10 runs. Algorithm 5 started from a random initial solution and could search for a more diverse solution space within a short CPU time limit. When the CPU time limit was changed to 600 s, the MST2 algorithm and Algorithm 5 produced a better-quality solution in terms of the relative standard deviation [29]. The relative standard deviation (RSD) in Table 3 in parenthesis was measured by $RSD = 100\frac{\sigma}{\mu}$, $\sigma = \sqrt{\frac{\sum(f(x)-\overline{f(x)})^2}{n}}$, and $\mu = \overline{f(x)}$, where $f(x)$ is the OFV of each run and $\overline{f(x)}$ is the mean value of the OFV out of $n = 10$ runs. For some instances, the relative standard deviation (RSD) was less than 5.0×10^{-4} even though not all runs found the same OFV. We used 0.000 as the value of RSD when the value was rounded up to three decimal points.

Table 3. The solution quality of the MST2 algorithm and Algorithm 5 with CPU time limits of 60 and 600 s out of 10 runs.

Instance ID	Size	Density	MST2 with 60 s	*r*-Flip with 60 s		MST2 with 600 s	*r*-Flip with 600 s	
				$r = 1$	$r = 2$		$r = 1$	$r = 2$
p3000_1	3000	0.5	10 (0)	10 (0)	10 (0)	10 (0)	10 (0)	10 (0)
p3000_2	3000	0.8	10 (0)	10 (0)	10 (0)	10 (0)	10 (0)	10 (0)
p3000_3	3000	0.8	4 (0.01)	7 (0.007)	10 (0)	10 (0)	10 (0)	10 (0)
p3000_4	3000	1	10 (0)	10 (0)	10 (0)	10 (0)	10 (0)	10 (0)
p3000_5	3000	1	10 (0)	9 (0.003)	7 (0.002)	9 (0.001)	10 (0)	10 (0)

Table 3. *Cont.*

Instance ID	Size	Density	MST2 with 60 s	*r*-Flip with 60 s		MST2 with 600 s	*r*-Flip with 600 s	
				r = 1	*r* = 2		*r* = 1	*r* = 2
p4000_1	4000	0.5	10 (0)	10 (0)	10 (0)	10 (0)	10 (0)	10 (0)
p4000_2	4000	0.8	10 (0)	10 (0)	10 (0)	9 (0.004)	10 (0)	10 (0)
p4000_3	4000	0.8	10 (0)	10 (0)	10 (0)	10 (0)	10 (0)	10 (0)
p4000_4	4000	1	1 (0.033)	10 (0)	10 (0)	10 (0)	10 (0)	10 (0)
p4000_5	4000	1	10 (0)	10 (0)	10 (0)	10 (0)	10 (0)	10 (0)
p5000_1	5000	0.5	6 (0.000)	1 (0.002)	2 (0.002)	10 (0)	3 (0.002)	2 (0.002)
p5000_2	5000	0.8	10 (0)	4 (0.003)	1 (0.002)	6 (0.012)	10 (0)	10 (0)
p5000_3	5000	0.8	10 (0)	7 (0.001)	3 (0.002)	10 (0)	10 (0)	10 (0)
p5000_4	5000	1	10 (0)	1 (0.002)	1 (0.001)	10 (0)	3 (0.002)	1 (0.001)
p5000_5	5000	1	6 (0.021)	9 (0.003)	4 (0.004)	10 (0)	10 (0)	10 (0)
p6000_1	6000	0.5	10 (0)	10 (0)	4 (0.001)	10 (0)	10 (0)	10 (0)
p6000_2	6000	0.8	10 (0)	4 (0.001)	4 (0.001)	1 (0.006)	10 (0)	9 (0)
p6000_3	6000	1	9 (0.002)	3 (0.002)	1 (0.007)	10 (0)	10 (0)	10 (0)
p7000_1	7000	0.5	1 (0.002)	1 (0.006)	1 (0.007)	10 (0)	2 (0.002)	4 (0.002)
p7000_2	7000	0.8	7 (0.000)	1 (0.008)	1 (0.008)	10 (0)	1 (0.004)	2 (0.004)
p7000_3	7000	1	8 (0.011)	3 (0.021)	5 (0.023)	10 (0)	10 (0)	10 (0)
p8000_1	8000	0.5	10 (0)	1 (0.004)	1 (0.005)	9 (0.001)	10 (0)	1 (0.002)
p8000_2	8000	0.8	10 (0)	1 (0.009)	1 (0.008)	10 (0)	7 (0.003)	10 (0)
p8000_3	8000	1	10 (0)	1 (0.013)	1 (0.01)	10 (0)	4 (0.001)	3 (0.002)
p30000_1	30,000	0.5	1 (0.002)	1 (0.023)	1 (0.019)	7 (0.018)	1 (0.017)	1 (0.011)
p30000_2	30,000	0.8	10 (0)	1 (0.017)	1 (0.016)	6 (0.01)	1 (0.019)	1 (0.013)
p30000_3	30,000	1	10 (0)	1 (0.015)	1 (0.019)	2 (0.037)	1 (0.025)	1 (0.019)

Table 4 reports the computational results of the CPU time limit of 60 s, and Table 5 reports the computational results of the CPU time limit of 600 s. In Table 4, it can be seen that the MST2 algorithm matched 5 out of 27 of the best solutions within the CPU time limit. The 1-flip strategy in Algorithm 5 matched 26 out of 27 of the best solutions, whereas the 2-flip strategy in Algorithm 5 matched 18 out of 27 of the best solutions. For the MST2 algorithm, the computing time to find the initial solution exceeded the CPU time limit of 60 s for two large instances.

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

Instance ID	BFS (60 s)	MST2 (60 s)		*r*-Flip (60 s)			
		OFV	TB (s)	OFV (*r* = 1)	TB (s)	OFV (*r* = 2)	TB (s)
p3000_1	3,931,583	3,931,583	10	3,931,583	3	3,931,583	8
p3000_2	5,193,073	5,193,073	25	5,193,073	2	5,193,073	2
p3000_3	5,111,533	5,111,533	52	5,111,533	8	5,111,533	4
p3000_4	5,761,822	5,761,437	10	5,761,822	2	5,761,822	2
p3000_5	5,675,625	5,675,430	24	5,675,625	7	5,675,625	17
p4000_1	6,181,830	6,181,830	40	6,181,830	3	6,181,830	4
p4000_2	7,801,355	7,797,821	12	7,801,355	13	7,801,355	4
p4000_3	7,741,685	7,741,685	31	7,741,685	5	7,741,685	8
p4000_4	8,711,822	8,709,956	58	8,711,822	5	8,711,822	11
p4000_5	8,908,979	8,905,340	27	8,908,979	4	8,908,979	13
p5000_1	8,559,680	8,556,675	56	8,559,680	21	8,559,680	7
p5000_2	10,836,019	10,829,848	34	10,836,019	59	10,836,019	11
p5000_3	10,489,137	10,477,129	28	10,489,137	20	10,489,137	16
p5000_4	12,251,710	12,245,282	52	12,251,710	54	12,251,520	42
p5000_5	12,731,803	12,725,779	56	12,731,803	17	12,731,803	16

Table 4. *Cont.*

Instance ID	BFS (60 s)	MST2 (60 s)		*r*-Flip (60 s)			
		OFV	TB (s)	OFV (*r* = 1)	TB (s)	OFV (*r* = 2)	TB (s)
p6000_1	11,384,976	11,377,315	42	11,384,976	12	11,384,976	5
p6000_2	14,333,855	14,330,032	39	14,333,855	27	14,333,767	14
p6000_3	16,132,915	16,122,333	51	16,130,731	24	16,132,915	48
p7000_1	14,477,949	14,467,157	56	14,477,949	41	14,476,263	21
p7000_2	18,249,948	18,238,729	55	18,249,948	47	18,246,895	47
p7000_3	20,446,407	20,431,354	59	20,446,407	15	20,446,407	12
p8000_1	17,340,538	17,326,259	47	17,340,538	26	17,340,538	35
p8000_2	22,208,986	22,180,465	55	22,208,986	54	22,208,683	53
p8000_3	24,670,258	24,647,248	56	24,670,258	43	24,669,351	50
p30000_1	127,252,438	126,732,48,3	60	127,252,438	58	127,219,336	60
p30000_2	158,384,175	157,481,366	69	158,384,175	59	158,339,497	60
p30000_3	179,103,085	178,093,109	89	179,103,085	58	179,029,747	54

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

Instance ID	BFS (600 s)	MST2 (600 s)		*r*-Flip (600 s)			
		OFV	TB (s)	OFV (*r* = 1)	TB (s)	OFV (*r* = 2)	TB (s)
p3000_1	3,931,583	3,931,583	11	3,931,583	5	3,931,583	5
p3000_2	5,193,073	5,193,073	25	5,193,073	1	5,193,073	3
p3000_3	5,111,533	5,111,533	52	5,111,533	30	5,111,533	8
p3000_4	5,761,822	5,761,822	269	5,761,822	1	5,761,822	2
p3000_5	5,675,625	5,675,625	505	5,675,625	43	5,675,625	29
p4000_1	6,181,830	6,181,830	40	6,181,830	4	6,181,830	2
p4000_2	7,801,355	7,800,851	530	7,801,355	8	7,801,355	8
p4000_3	7,741,685	7,741,685	30	7,741,685	5	7,741,685	2
p4000_4	8,711,822	8,711,822	67	8,711,822	2	8,711,822	7
p4000_5	8,908,979	8,906,525	65	8,908,979	4	8,908,979	13
p5000_1	8,559,680	8,559,075	324	8,559,680	9	8,559,680	27
p5000_2	10,836,019	10,835,437	541	10,836,019	17	10,836,019	21
p5000_3	10,489,137	10,488,735	400	10,489,137	29	10,489,137	38
p5000_4	12,252,318	12,249,290	265	12,252,318	127	12,251,848	143
p5000_5	12,731,803	12,731,803	265	12,731,803	19	12,731,803	32
p6000_1	11,384,976	11,384,976	406	11,384,976	8	11,384,976	39
p6000_2	14,333,855	14,333,767	498	14,333,855	62	14,333,855	17
p6000_3	16,132,915	16,128,609	239	16,132,915	60	16,132,915	71
p7000_1	14,478,676	14,477,039	344	14,478,676	92	14,478,676	397
p7000_2	18,249,948	18,242,205	587	18,249,948	115	18,249,844	43
p7000_3	20,446,407	20,431,833	109	20,446,407	47	20,446,407	21
p8000_1	17,341,350	17,337,154	546	17,340,538	45	17,341,350	141
p8000_2	22,208,986	22,207,866	122	22,208,986	49	22,208,986	89
p8000_3	24,670,924	24,669,797	402	24,670,924	185	24,670,924	386
p30000_1	127,336,719	127,323,304	568	127,332,912	598	127,336,719	592
p30000_2	158,561,564	158,438,942	573	158,561,564	580	158,526,518	571
p30000_3	179,329,754	179,113,916	575	179,329,754	599	179,261,723	590

When the CPU time limit was increased to 600 s, the MST2 algorithm matched 10 out of 27 of the best solutions. The 1-flip strategy matched 25 out of 27 of the best solutions, and the 2-flip strategy matched 23 out of 27 of the best solutions. The 1-flip and 2-flip strategies in Algorithm 5 performed well on the large, high-density instances. There was no clear pattern of the 2-flip strategy being used more often than the 1-flip strategy to find the same OFV. The 1-flip and 2-flip strategies in Algorithm 5 chose the initial solution randomly and independently. The 1-flip strategy had better performance when the CPU time limits were 60 and 600 s.

Tables 6 and 7 present the time deviation of each algorithm upon reaching the OFV for each instance. The MST2 algorithm had less variation in computing time when it finding the OFV, whereas the *r*-flip strategy in Algorithm 5 had a wider range of computing time. If the algorithm only found the OFV once out of 10 runs, the time deviation was zero.

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

Instance ID	MST2		1-Flip		2-Flip	
	AT (s)	DT	AT (s)	DT	AT (s)	DT
p3000_1	12.5	15.663	15.7	84.075	24.1	56.875
p3000_2	32.5	20.059	4.9	45.583	4.8	68.606
p3000_3	54.3	5.901	29.3	46.180	12.0	60.477
p3000_4	13.3	18.434	12.0	107.798	10.0	51.640
p3000_5	30.4	15.829	33.2	45.470	32.3	48.240
p4000_1	49.0	10.227	7.7	50.131	9.3	36.570
p4000_2	14.2	9.848	22.1	62.351	21.1	70.476
p4000_3	35.4	11.236	15.1	69.699	16.5	75.542
p4000_4	58.0	0	22.2	65.408	35.4	45.780
p4000_5	31.1	11.082	18.1	73.038	29.6	40.109
p5000_1	56.8	2.339	4.0	0	10.0	42.426
p5000_2	35.0	3.563	28.3	74.329	11.0	0
p5000_3	30.0	8.607	38.9	33.038	33.0	50.069
p5000_4	54.0	5.238	54.0	0	42.0	0
p5000_5	57.2	1.317	38.8	40.504	30.5	63.379
p6000_1	43.3	5.112	32.9	47.380	21.8	67.507
p6000_2	39.8	4.238	31.8	51.521	42.3	46.315
p6000_3	52.1	5.027	32.3	44.640	48.0	0
p7000_1	56.0	0	41.0	0	21.0	0
p7000_2	55.3	1.367	47.0	0	47.0	0
p7000_3	59.0	0	42.0	56.293	35.2	45.958
p8000_1	48.1	7.232	26.0	0	35.0	0
p8000_2	55.9	0.566	54.0	0	53.0	0
p8000_3	57.2	1.806	43.0	0	50.0	0
p30000_1	60.0	0	58.0	0	60.0	0
p30000_2	70.0	1.166	59.0	0	60.0	0
p30000_3	90.3	0.912	58.0	0	54.0	0

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

Instance ID	MST2		1-Flip		2-Flip	
	AT (s)	DT	AT (s)	DT	AT (s)	DT
p3000_1	11.9	16.067	21.3	62.678	37.8	130.045
p3000_2	29.1	18.144	5.5	69.234	7.7	81.231
p3000_3	58.5	16.889	126.4	99.610	143.6	116.812
p3000_4	292.1	11.285	10.8	58.365	13.7	49.512
p3000_5	543.9	6.365	109.1	88.328	164.3	72.144
p4000_1	42.8	7.773	13.5	63.553	15.1	63.861
p4000_2	552.2	2.597	30.7	49.923	37.0	110.712
p4000_3	31.7	7.293	18.8	57.442	22.9	47.545
p4000_4	70.8	4.094	42.2	102.331	48.7	115.376
p4000_5	70.5	8.596	25.3	81.984	43.0	69.595
p5000_1	337.2	4.265	70.7	126.287	48.0	61.872
p5000_2	557.8	4.565	135.4	96.462	210.6	67.670
p5000_3	428.5	7.257	115.8	96.524	115.5	104.238

Table 7. *Cont.*

Instance ID	MST2		1-Flip		2-Flip	
	AT (s)	DT	AT (s)	DT	AT (s)	DT
p5000_4	279.4	5.987	270.3	48.842	143.0	0
p5000_5	287.1	9.234	194.5	94.641	172.7	92.268
p6000_1	424.8	4.555	152.9	95.641	145.9	46.657
p6000_2	498.0	0	142.5	97.375	73.8	122.070
p6000_3	252.3	5.571	248.0	51.129	318.1	61.672
p7000_1	344.5	0.153	272.5	93.675	441.5	12.974
p7000_2	587.0	0	115.0	0	265.1	76.694
p7000_3	109.0	0	84.5	39.472	131.1	55.401
p8000_1	548.6	1.398	251.3	63.346	141.0	0
p8000_2	145.4	11.829	258.3	56.352	300.7	56.001
p8000_3	514.3	11.365	368.8	43.667	417.3	12.797
p30000_1	572.0	1.365	598.0	0	592.0	0
p30000_2	581.5	1.526	580.0	0	571.0	0
p30000_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. Clever implementation of the *r*-flip strategy can reduce the computing time and improve the solution quality. We reported the time and solutions out of 10 runs for each instance. The time deviation and solution deviation of 10 runs were computed with the short CPU time limits due to the computing resources available to this study.

The observed results show the effectiveness of the employed methodology in finding the best solutions within very competitive CPU time limits. This suggests the possibility of using alternative stopping criteria as a potential improvement in the future. Also, the application of an efficient strategy in evaluating flip moves has the potential to further improve the algorithm performance, as suggested by Anacleto et al. [12].

The *r*-flip method we developed presents itself as a compelling alternative for quantum computing solvers, which are capable of efficiently handling QUBO problems as large as 5000 by 5000 variables due to the design of quantum processors. The advantages of the *r*-flip method become particularly pronounced when dealing with substantial, complex problem instances, showcasing superior computing performance. In essence, our method emerges as a favorable choice for addressing large-scale, real-world challenges with efficacy and computational prowess.

4. Conclusions

In this study, we explored the quadratic unconstrained binary optimization (QUBO) problem, which produced significant findings. We established a necessary and sufficient condition for the local optimality of an *r*-flip search after a 1-flip search had already achieved local optimality. Our computational experiments demonstrated a substantial reduction in candidate options for *r*-flip implementation. The new *r*-flip strategy efficiently solved extremely large QUBO instances within 600 s, outperforming MST2 in terms of the time taken to reach the best-known solutions in benchmark instances. These results are particularly promising for implementing variable neighborhood strategies on extensive problems or sparse matrices.

Similar to other quadratic unconstrained binary optimization (QUBO) methods, our approach is subject to scalability constraints. The size of the Q matrix, representing the number of variables in the problem, is inherently restricted by the memory limitations of the computer system. In practical terms, our methodology exhibits robust performance, successfully tackling problem instances of up to 200,000 by 200,000 in dense matrices, with the potential to extend its capabilities to 500,000 by 500,000 in sparse matrices. This underscores the versatility and efficiency of our approach within the specified computational constraints, demonstrating its applicability to a wide range of problem sizes and structures.

Author Contributions: Conceptualization, B.A. and H.W.; methodology, B.A.; validation, B.A., H.W. and L.S.S.; formal analysis, H.W.; investigation, H.W.; data curation, H.W.; writing—original draft preparation, B.A.; writing—review and editing, H.W. and L.S.S.; visualization, B.A.; All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The instance data and solutions will be made available upon request.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Numerical Example for Theorem 1 and Proposition 1

Consider the problem with the Q matrix provided below in Figure A1. A locally optimal solution x with $f(x) = 1528$ and the vector of derivatives $E(x)$ is shown in Table A1. For $r = 2$, $M = 100$, every two elements that satisfy Proposition 1 are shown in Table A1. However, the only sets of two-element combinations that were improved are those shown with the red font in Figure A2. These are the two elements that satisfied Theorem 1. The new results of $f(x)$ and $E(x)$ for the two elements are shown in Table A2. In this simple problem, instead of comparing 380 two-element combinations, we only needed to compare 21 combinations. Out of these 21 combinations, the 6 possible improved combinations are shown in Table A3.

-30	50	-68	-66	94	-100	-20	-4	80	94	0	86	76	98	-36	76	52	-8	48	42
	-7	50	28	60	56	-54	78	4	30	-42	-28	-50	90	-42	90	-26	6	22	0
		-26	-46	26	82	-50	-90	44	-94	54	52	-50	54	-68	-52	20	96	64	-22
			-35	8	-82	-84	-46	-34	-22	-62	-34	-56	-86	18	44	-84	2	-72	-60
				31	-86	48	90	-50	-70	-50	8	-42	-80	10	92	64	-36	4	14
					19	-66	-98	50	60	-94	88	76	100	-46	-72	-36	-66	-46	2
						-4	-60	-20	86	36	94	-94	-52	-34	-36	12	-28	64	-70
							-17	-74	-2	-32	50	-10	46	52	-68	-64	84	-46	94
								-30	-14	54	6	-24	26	-80	-56	22	82	-64	-16
									25	26	70	2	34	-68	100	70	-48	82	0
										32	-22	-58	-56	62	86	-8	-90	-68	52
											-47	72	-42	52	-40	2	10	58	28
												40	-38	48	-16	76	92	-4	-96
													-16	72	76	66	-70	-22	64
														-31	-62	88	-66	0	-34
															37	64	-72	-12	10
																13	-6	-18	-26
																	-3	74	-80
																		20	-54
																			22

Figure A1. Matrix Q for maximizing $x^T Qx$.

Table A1. A locally optimal solution, $x(.)$, and values of $E(x_i)$, $i = 1,\ldots, n$.

	$x(.)$	1	1	0	0	1	0	1	0	0	1	0	1	1	1	0	1	1	0	1	0
	$E(.)$	624	177	−74	−459	209	−7	44	−7	−120	523	−124	233	22	114	−3	431	375	−89	242	−66
$f(x)$	1528	624	127	0	0	55	0	70	0	0	383	0	3	58	104	0	89	−5	0	20	0

In Figure A1 and Table A1, we have:

$E(x_1) = -30 + (50 * 1 - 68 * 0 - 66 * 0 + 94 * 1 - 100 * 0 - 20 * 1 - 4 * 0 + 80 * 0 + 94 * 1 + 0 * 0 + 86 * 1 + 76 * 1 + 98 * 1 - 36 * 0 + 76 * 1 + 52 * 1 - 8 * 0 + 48 * 1 + 42 * 0) = 624$

$E(x_2) = -7 + 50 * 1 + (50 * 0 + 28 * 0 + 60 * 1 + 56 * 0 - 54 * 1 + 78 * 0 + 4 * 0 + 30 * 1 - 42 * 0 - 28 * 1 - 50 * 1 + 90 * 1 - 42 * 0 + 90 * 1 - 26 * 1 + 6 * 0 + 22 * 1 + 0 * 0) = 177$

$E(x_3) = -26 + (-68 * 1 + 50 * 1) + (-46 * 0 + 26 * 1 + 82 * 0 - 50 * 1 - 90 * 0 + 44 * 0 - 94 * 1 + 54 * 0 + 52 * 1 - 50 * 1 + 54 * 1 - 68 * 0 - 52 * 1 + 20 * 1 + 96 * 0 + 64 * 1 - 22 * 0) = -74$

$E(x_4) = -35 + (-66 * 1 + 28 * 1 - 46 * 0) + (8 * 1 - 82 * 0 - 84 * 1 - 46 * 0 - 34 * 0 - 22 * 1 - 62 * 0 - 34 * 1 - 56 * 1 - 86 * 1 + 18 * 0 + 44 * 1 - 84 * 1 + 2 * 0 - 72 * 1 - 60 * 0) = -459$

. . . for $E(x_5)$ to $E(x_{19})$

$$E(x_{20}) = 22 + (42 * 1 + 0 * 1 - 22 * 0 - 60 * 0 + 14 * 1 + 2 * 0 - 70 * 1 + 94 * 0 - 16 * 0 + 0 * 1 + 52 * 0 + 28 * 1 - 96 * 1 + 64 * 1 - 34 * 0 + 10 * 1 - 26 * 1 + 80 * 0 - 54 * 1) = -66$$

$$f(x_1) = 1 * (-30 + 50 + 94 - 20 + 94 + 86 + 76 + 98 + 76 + 52 + 48) = 624$$

$$f(x_2) = 1 * (-7 + 60 - 54 + 30 - 28 - 50 + 90 + 90 - 26 + 22) = 127$$

$$f(x_3) = 0 * (26 - 50 - 94 + 52 - 50 + 54 - 52 + 20 + 64) = 0$$

$$\ldots for\ f(x_4)\ to\ f(x_{19})$$

$$f(x_{20}) = 0 * (22) = 0$$

$$f(x) = 624 + 127 + 55 + 70 + 383 + 3 + 58 + 104 + 89 - 5 + 20 = 1528$$

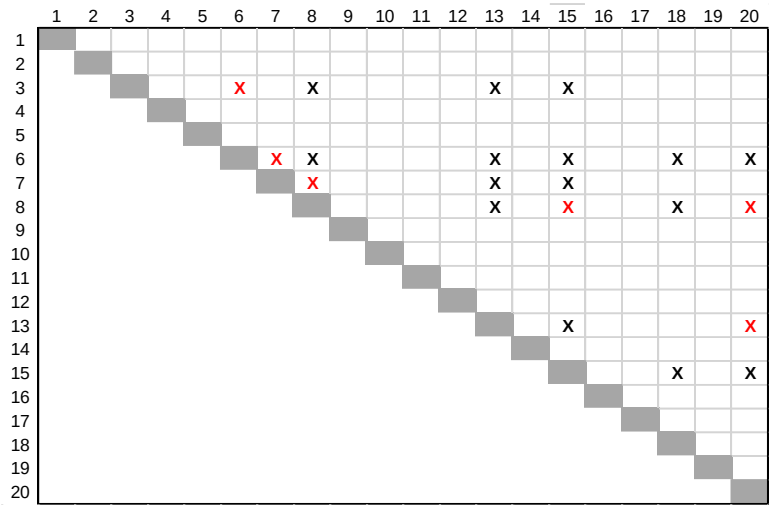

Figure A2. The indicated elements satisfy Proposition 1. Only the two-element combinations indicated by the red font can improve $f(x)$, which satisfies Theorem 1, whereas the ones indicated by the black font cannot.

Table A2. An improvement using 2-flip for the locally optimal solution, x, and values of $E(x_i)$, $i = 1, \ldots, n$.

	$x(.)$	1	1	0	0	1	0	0	1	0	1	0	1	1	1	1	1	1	0	1	1
	$E(.)$	646	267	−204	−463	275	−83	−120	199	−270	367	−78	269	58	348	49	347	361	−123	78	64
$f(x)$	1684	646	217	0	0	121	0	0	35	0	315	0	83	10	240	−39	99	−31	0	−34	22

In Figure A2 and Table A2, we have:

$$f(x_1) = 1 * (-30 + 50 + 94 - 4 + 94 + 86 + 76 + 98 - 36 + 76 + 52 + 48 + 42) = 646$$

$$f(x_2) = 1 * (-7 + 60 + 78 + 30 - 28 - 50 + 90 - 42 + 90 - 26 + 22) = 217$$

$$\ldots for\ f(x_3)\ to\ f(x_{18})$$

$$f(x_{19}) = 1 * (20 - 54) = -34$$

$$f(x_{20}) = 1 * (22) = 22$$

$$f(x) = 646 + 217 + 121 + 35 + 315 + 83 + 10 + 240 - 39 + 99 - 31 - 34 + 22 = 1684$$

Table A3. All 6 possible improved combinations for the locally optimal solution ($f(x) = 1528$).

1	1	0	0	1	0	0	1	0	1	0	1	1	1	1	1	1	0	1	1	1684
1	1	1	0	1	1	0	0	1	1	0	1	1	1	0	1	1	1	1	0	1646
1	1	0	0	1	0	1	1	0	1	0	1	0	1	0	1	1	0	1	1	1633
1	1	0	0	1	0	1	1	0	1	1	1	0	1	1	1	1	0	1	1	1616
1	1	1	0	1	1	0	0	0	1	0	1	1	1	0	1	1	0	1	0	1601
1	1	1	0	1	0	1	0	0	1	1	1	0	1	0	1	1	0	1	1	1530
1	1	0	0	1	0	1	0	0	1	0	1	1	1	0	1	1	0	1	0	**1528**

Table A4. An exhaustive search of *r*-flip rules for QUBO.

Study	***r*-Flip Rules**
Debevre, P, Sugimura, M, and Parizy, M, Ieee Access, 2023 [2].	The automotive paint shop problem was formulated as QUBO, then 1-flip and multiple flip moves provided for the solution.
Glover, F, and Hao, J-K, Int. J. Metaheuristics, 2010 [6].	An efficient evaluation of 2-flip moves for QUBO was presented.
Glover, F, and Hao, J-K, Annals of Operations Research, 2016 [7].	A class of approaches called f-flip strategies that include a fractional value for f is provided for QUBO.
Lozano, M, Molina, D, and Garcia-Martinez, C, European Journal of Operational Research, 2011 [10].	A maximum diversity QUBO problem was considered with an added number of variables that must be equal to an integer m. A 2-flip strategy with computational experiment was presented.
Alidaee, B., G. Kochenberger, and H. Wang, Int. J. Appl. Metaheuristic Comput., 2010 [11].	Several theoretical results for *r*-flip moves in a general pseudo-Boolean optimization including QUBO were demonstrated.
Anacleto, E, Meneses, C, Ravelo, S, Computers & Operations Research, 2020 [12].	Two closed-form formulas for evaluating *r*-flip moves were presented. The effectiveness of the moves was evaluated experimentally.
Anacleto, E, Meneses, C, and Liang, T, Computers & Operations Research, 2021 [13].	An *r*-flip move strategy for a quadratic assignment problem that can be transferred to QUBO was considered. A closed-form formula and experimental evaluations were considered.
Katayama, K, and Naihisa, H, in: W. Hart, N. Krasnogor, J.E. Smith (Eds.), Recent Advances in Memetic Algorithms, Springer, Berlin, 2004 [30].	A *k*-flip move in the context of a memetic algorithm was presented.
Liang, R.N, Anacleto, E.A.J, and Meneses, C.N. Computers & Operations Research, 2023 [31].	A closed-form formula for pseudo-Boolean optimization and data structure for efficient implementation of the 1-flip rule was presented.
Merz, P, and Freisleben, B, Journal of Heuristics, 2002 [32].	Provided 1-flip and multiple-flip strategies for QUBO and provided a computational experiment.
Rosenberg, G., Vazifeh, M, Woods, B, and Haber, E, Computational Optimization and Applications, 2016 [33].	A *k*-flip strategy in the context of a quantum annealer was presented.
Alidaee, B., Wang, H., and Liu, W. Working Paper, WP 2021-002. Texas A&M International University, 2022 [34].	New Results on Closed-Form Formulas for Evaluating r-flip Moves in Quadratic Unconstrained Binary Optimization.

References

1. Boros, E.; Hammer, P.L.; Minoux, M.; Rader, D.J. Optimal cell flipping to minimize channel density in VLSI design and pseudo-Boolean optimization. *Discret. Appl. Math.* **1999**, *90*, 69–88. [CrossRef]
2. Debevre, P.; Sugimura, M.; Parizy, M. Quadratic Unconstrained Binary Optimization for the Automotive Paint Shop Problem. *IEEE Access* **2023**, *11*, 97769–97777. [CrossRef]
3. Glover, F.; Kochenberger, G.; Du, Y. Quantum Bridge Analytics I: A tutorial on formulating and using QUBO models. *Ann. Oper. Res.* **2019**, *17*, 335–371. [CrossRef]
4. Glover, F.; Kochenberger, G.A.; Alidaee, B. Adaptive Memory Tabu Search for Binary Quadratic Programs. *Manag. Sci.* **1998**, *44*, 336–345. [CrossRef]
5. Glover, F.; Lü, Z.; Hao, J.-K. Diversification-driven tabu search for unconstrained binary quadratic problems. *4OR* **2010**, *8*, 239–253. [CrossRef]
6. Glover, F.; Hao, J.-K. Fast two-flip move evaluations for binary unconstrained quadratic optimisation problems. *Int. J. Metaheuristics* **2010**, *1*, 100–108. [CrossRef]
7. Glover, F.; Hao, J.-K. f-Flip strategies for unconstrained binary quadratic programming. *Ann. Oper. Res.* **2016**, *238*, 651–657. [CrossRef]
8. Kochenberger, G.; Hao, J.-K.; Glover, F.; Lewis, M.; Lü, Z.; Wang, H.; Wang, Y. The unconstrained binary quadratic programming problem: A survey. *J. Comb. Optim.* **2014**, *28*, 58–81. [CrossRef]
9. Kochenberger, G.A.; Glover, F.; Wang, H. Binary Unconstrained Quadratic Optimization Problem. In *Handbook of Combinatorial Optimization*; Pardalos, P.M., Du, D.-Z., Graham, R.L., Eds.; Springer: New York, NY, USA, 2013; pp. 533–557.
10. Lozano, M.; Molina, D.; Garcia-Martinez, C. Iterated greedy for the maximum diversity problem. *Eur. J. Oper. Res.* **2011**, *214*, 31–38. [CrossRef]

11. Alidaee, B.; Kochenberger, G.; Wang, H. Theorems Supporting *r*-flip Search for Pseudo-Boolean Optimization. *Int. J. Appl. Metaheuristic Comput.* **2010**, *1*, 93–109. [CrossRef]
12. Anacleto, E.A.J.; Meneses, C.N.; Ravelo, S.V. Closed-form formulas for evaluating *r*-flip moves to the unconstrained binary quadratic programming problem. *Comput. Oper. Res.* **2020**, *113*, 104774. [CrossRef]
13. Anacleto, E.; Meneses, C.; Liang, T. Fast *r*-flip move evaluations via closed-form formulae for Boolean quadraticprogramming problems with generalized upper bound constraints. *Comput. Oper. Res.* **2021**, *132*, 105297. [CrossRef]
14. Glover, F.; Lewis, M.; Kochenberger, G. Logical and inequality implications for reducing the size and difficulty of quadratic unconstrained binary optimization problems. *Eur. J. Oper. Res.* **2018**, *265*, 829–842. [CrossRef]
15. Lewis, M.; Glover, F. Quadratic Unconstrained Binary Optimization Problem Preprocessing: Theory and Empirical Analysis. *Networks* **2017**, *70*, 79–97. [CrossRef]
16. Vredeveld, T.; Lenstra, J.K. On local search for the generalized graph coloring problem. *Oper. Res. Lett.* **2003**, *31*, 28–34. [CrossRef]
17. Kochenberger, G.; Glover, F. Introduction to special xQx issue. *J. Heuristics* **2013**, *19*, 525–528. [CrossRef]
18. Ahuja, R.K.; Ergun, Ö.; Orlin, J.B.; Punnen, A.P. A survey of very large-scale neighborhood search techniques. *Discret. Appl. Math.* **2002**, *123*, 75–102. [CrossRef]
19. Ahuja, R.K.; Orlin, J.B.; Pallottino, S.; Scaparra, M.P.; Scutellà, M.G. A Multi-Exchange Heuristic for the Single-Source Capacitated Facility Location Problem. *Manag. Sci.* **2004**, *50*, 749–760. [CrossRef]
20. Yagiura, M.; Ibaraki, T. Analyses on the 2 and 3-Flip Neighborhoods for the MAX SAT. *J. Comb. Optim.* **1999**, *3*, 95–114. [CrossRef]
21. Yagiura, M.; Ibaraki, T. Efficient 2 and 3-Flip Neighborhood Search Algorithms for the MAX SAT: Experimental Evaluation. *J. Heuristics* **2001**, *7*, 423–442. [CrossRef]
22. Yagiura, M.; Kishida, M.; Ibaraki, T. A 3-flip neighborhood local search for the set covering problem. *Eur. J. Oper. Res.* **2006**, *172*, 472–499. [CrossRef]
23. Alidaee, B. *Fan-and-Filter Neighborhood Strategy for 3-SAT Optimization, Hearin Center for Enterprise Science*; The University of Mississippi, Hearin Center for Enterprise Science: Oxford, MS, USA, 2004.
24. Glover, F. *A Template for Scatter Search and Path Relinking*; Springer: Berlin/Heidelberg, Germany, 1998.
25. Ivanov, S.V.; Kibzun, A.I.; Mladenović, N.; Urošević, D. Variable neighborhood search for stochastic linear programming problem with quantile criterion. *J. Glob. Optim.* **2019**, *74*, 549–564. [CrossRef]
26. Mladenović, N.; Hansen, P. Variable neighborhood search. *Comput. Oper. Res.* **1997**, *24*, 1097–1100. [CrossRef]
27. Alidaee, B.; Sloan, H.; Wang, H. Simple and fast novel diversification approach for the UBQP based on sequential improvement local search. *Comput. Ind. Eng.* **2017**, *111*, 164–175. [CrossRef]
28. Palubeckis, G. Multistart Tabu Search Strategies for the Unconstrained Binary Quadratic Optimization Problem. *Ann. Oper. Res.* **2004**, *131*, 259–282. [CrossRef]
29. Glen, S. Relative Standard Deviation: Definition & Formula. 2014. Available online: https://www.statisticshowto.com/relative-standard-deviation/ (accessed on 10 November 2023).
30. Katayama, K.; Naihisa, H. *An Evolutionary Approach for the Maximum Diversity Problem*; Hart, W., Krasnogor, N., Smith, J.E., Eds.; Recent Advances in Memetic Algorithms; Springer: Berlin/Heidelberg, Germany, 2004; pp. 31–47.
31. Liang, R.N.; Anacleto, E.A.J.; Meneses, C.N. Fast 1-flip neighborhood evaluations for large-scale pseudo-Boolean optimization using posiform representation. *Comput. Oper. Res.* **2023**, *159*, 106324. [CrossRef]
32. Merz, P.; Freisleben, B. Greedy and Local Search Heuristics for Unconstrained Binary Quadratic Programming. *J. Heuristics* **2002**, *8*, 197–213. [CrossRef]
33. Rosenberg, G.; Vazifeh, M.; Woods, B.; Haber, E. Building an iterative heuristic solver for a quantum annealer. *Comput. Optim. Appl.* **2016**, *65*, 845–869. [CrossRef]
34. Alidaee, B.; Wang, H.; Liu, W. *New Results on Closed-Form Formulas for Evaluating r-flip Moves in Quadratic Unconstrained Binary Optimization*; Working Paper Series, WP 2021-002; Texas A&M International University: Laredo, TX, USA, 2022.