Ann Oper Res
DOI 10.1007/s10479-015-2076-1

# f-Flip strategies for unconstrained binary quadratic programming

**Fred Glover**[1] · **Jin-Kao Hao**[2,3]



**Abstract** Unconstrained binary quadratic programming (UBQP) provides a unifying modeling and solution framework for solving a remarkable range of binary optimization problems, including many accompanied by constraints. Current methods for solving UBQP problems customarily rely on neighborhoods consisting of *flip moves* that select one or more binary variables and "flip" their values to the complementary value (from 1 to 0 or from 0 to 1). We introduce a class of approaches called *f-flip strategies* that include a fractional value f as one of those available to the binary variables during intermediate stages of solution. A variety of different f-flip strategies, particularly within the context of multi-start algorithms, are proposed for pursuing intensification and diversification goals in metaheuristic algorithms, accompanied by special rules for evaluating and executing f-flips efficiently.

**Keywords** 0–1 Optimization · Binary quadratic programming · Metaheuristics · Multi-start algorithms · Computational efficiency

## 1 Introduction

The unconstrained binary quadratic programming (UBQP) problem may be written as:

$$\text{Minimize } x_o = xQx, \text{ x binary}$$

where Q is an n by n matrix of constants and x is an n-vector of binary (0–1) variables (Hammer and Rudeanu 1968).

✉ Fred Glover
fredwglover@yahoo.com

Jin-Kao Hao
Jin-kao.Hao@univ-angers.fr

1 University of Colorado Leeds School of Business, Boulder, CO, USA

2 LERIA, Université d'Angers, 49045 Angers, France

3 Institut Universitaire de France, Paris, France

Published online: 11 December 2015

The UBQP formulation is notable for its ability to represent a wide range of important applications. Some applications appear naturally in the form of UBQP while others can be "re-cast" into the UBQP form by employing various transformations. The former case includes financial analysis (Laughunn 1970), computer aided design (Krarup and Pruzan 1978), traffic management (Gallo et al. 1980), machine scheduling (Alidaee et al. 1994), cellular radio channel allocation (Chardaire and Sutter 1994), and statistical physics and circuit layout design (Barahona et al. (1988)). Moreover, the UBQP model can conveniently formulate a number of well-known combinatorial optimization problems pertaining to graphs such as determining maximum cliques (Bomze et al. 1999), maximum-cuts (Boros and Hammer 1991; Kochenberger et al. 2013; Wang et al. 2013), set packing (Alidaee et al. 2008), set-partitioning (Lewis et al. 2008), maximum independent sets (Kochenberger et al. (2004)). A review of additional UBQP applications and formulations can be found in (Pardalos and Rodgers 1990; Pardalos and Xue 1994) and a recent survey (Kochenberger et al. 2014).

In this note we propose a class of approaches called *f-flip strategies* for transitioning from one solution to another in metaheuristic algorithms for the UBQP problem. In contrast to the customary 1-flip and 2-flip moves (Glover and Hao 2010a, b) that provide transitions by complementing one or two binary variables (from 1 or 0 to 0 or 1), an f-flip move includes the option of assigning a binary variable a fractional value f which may depend on the variable. Hence an f-flip move for a single variable can consist of changing from 0 or 1 to f, or from f to 0 or 1, as well as the complementation operation. We primarily focus here such flips for single variables (just as 1-flips provide the main moves for most UBQP methods) and give particular attention to multi-start strategies employing f-flips, where a selected subset of variables is assigned fractional values, and the transitions are from f to 0 or 1.

## 2 Efficient evaluations for f-flips

Before describing alternative strategies for exploiting f-flip moves, we examine the issue of how to evaluate such moves efficiently. As in the case of the customary single variable 1-flip move, a fast evaluation method can have a critical effect on algorithmic speed. The key to an efficient evaluation scheme, which identifies the change in the objective value $x_o$ caused by an f-flip move, lies in determining an effective updating process. We show how to do this by a rule that generalizes the rule of Glover and Hao (2010a) to the present context.

Let $N = \{1, \ldots, n\}$ denote the index set for components of the x vector and the rows and columns of Q. [The matrix $Q = (q_{ij} : i, j \in N)$ is often preprocessed to put it in lower triangular form by setting $q_{ij} := q_{ij} + q_{ji}$ for $i < j$, followed by $q_{ji} = 0$, which leaves the value $x_o = xQx$ unchanged and allows memory to be saved by storing only the non-zero lower diagonal elements. Our development additionally makes it possible to reduce computational effort using Q in this triangular form.]

Let $x'$ and $x''$ represent two binary solutions where $x''$ is obtained from $x'$ by an f-flip move applied to a single variable $x_k$. Define $x'_o = x'Qx'$ and $x''_o = x''Qx''$.

Then, the value $\Delta_k(v) = x''_o - x'_o$, which depends on the choice of the variable $x_k$ and the new value $x''_k = v$ that produces $x''$ (where $x''_i = x'_i$ for all $i \neq k$) discloses whether the move that replaces $x'$ by $x''$ will cause $x_o$ to improve or deteriorate (respectively, decrease or increase) relative to the minimization objective. The goal of making such an evaluation rapidly in search methods that incorporate f-flip moves is achieved by the following result.

**Proposition 1** *Let* $N(k) = N - \{k\}$ *and define*

$$Q'(k) = \sum \left( \left( q_{ik} + q_{ki} \right) x'_i : i \in N(k) \right), \quad k \in N \tag{1}$$

*and assume* $Q'(k)$ *is pre-computed by this definition for each* $k \in N$ *relative to an initial solution* $x'$. *Then the associated value* $\Delta'_k(v)$ *for* $v \in \{0, f, 1\}$ *is given by*

$$\Delta'_k(v) = \left( v - x'_k \right) \left( Q'(k) + q_{kk} \left( v + x'_k \right) \right) \tag{2}$$

*Proof* First, we identify a computation for determining the value $\Delta'_k(v)$ relative to an arbitrary initial solution $x'$ where each $x'_k \in \{0, f, 1\}$. Let $x_{ok}(v)$ denote the new value of $x_o$ derived from $x_o = x'_o$ as a result of changing the value of $x'_k$ to v, while $x_i = x'_i$ for $i \in N - k$. Consequently, $\Delta'_k(v) = x_{ok}(v) - x'_o$. Then, we decompose the representation of $x_o$ to write:

$$\begin{aligned} x'_o = &\sum \left( q_{ij} x'_i x'_j : i \in N(k), \quad j \in N(k) \right) + \sum \left( q_{ik} x'_i x'_k : i \in N(k) \right) \\ &+ \sum \left( q_{kj} x'_k x'_j : j \in N(k) \right) + q_{kk} x'_k x'_k \end{aligned}$$

or, reorganizing

$$\begin{aligned} x'_o = &\sum \left( q_{ij} x'_i x'_j : i \in N(k), \quad j \in N(k) \right) + x'_k \left( \sum \left( q_{ik} x'_i : i \in N(k) \right) \right. \\ &\left. + \sum \left( q_{kj} x'_j : j \in N(k) \right) \right) + q_{kk} x'_k x'_k \end{aligned}$$

From $Q'(k) = \sum((q_{ik} + q_{ki})x'_i : i \in N(k)), k \in N$ in (1) we obtain

$$x'_o = \sum \left( q_{ij} x'_i x'_j : i \in N(k), \quad j \in N(k) \right) + x'_k Q'(k) + q_{kk} x'_k x'_k$$

By a corresponding reorganization, we also obtain

$$x_{ok}(v) = \sum \left( q_{ij} x'_i x'_j : i \in N(k), \quad j \in N(k) \right) + v Q'(k) + q_{kk} v v$$

Hence, from $\Delta'_k(v) = x_{ok}(v) - x'_o$:

$$\Delta'_k(v) = \left( v - x'_k \right) Q'(k) + q_{kk} \left( v - x'_k \right) \left( v + x'_k \right)$$

or

$$\Delta'_k(v) = \left( v - x'_k \right) \left( Q'(k) + q_{kk} \left( v + x'_k \right) \right)$$

which corresponds to (2), completing the proof.

*Comments 1* For a matrix Q in lower triangular form, we can re-write (1) to give the simpler expression $Q'(k) = \sum(q_{ik} x'_i : i \in I(k)), k \in N$, where $I(k) = \{i \in N, i < k\}$, and by convention $Q'(1) = 0$ (since $I(1) = \emptyset$).

*Comments 2* $Q'(k)$ can be computed in $O(n)$ time by (1) and hence in $O(n^2)$ time for all $k \in N$.

Given $Q'(k)$, $\Delta'_k(v)$ can be computed in $O(1)$ time by (2), and hence in $O(n)$ time for all $k \in N$.

As a basis for our next result which refers to a solution $x''$ obtained from $x'$, we assume the values $Q'(k)$ and $\Delta'_k(v)$ are initialized as indicated in Proposition 1 and the result is applied recursively (redefining $x' = x''$) to create evaluations for a current $x'$.

**Proposition 2** *Let $x'$ denote a current solution and consider a new solution $x''$ produced from $x'$ by an f-flip that changes the value of a selected variable $x_h = x'_h$ to $x_h = v''$. Then the associated new values $Q''(k)$ and $\Delta''_k(v)$ for $x''$ can be obtained as follows:*

$$\text{For } k \in N(h)\text{: } Q''(k) = Q'(k) + (q_{hk} + q_{kh})\left(v'' - x'_h\right) \tag{3}$$

$$\text{For } k = h\text{: } Q''(h) = Q'(h) \tag{4}$$

*and*

$$\text{For } k \in N(h)\text{: } \Delta''_k(v) = \left(v - x'_k\right)\left(Q''(k) + q_{kk}\left(x'_k + v\right)\right) \tag{5}$$

$$\text{For } k = h\text{: } \Delta''_h(v) = (v - v'')(Q''(h) + q_{hh}(v'' + v) \tag{6}$$

*Proof* To update $Q'(k)$ to $Q''(k)$ for $k \in N(h)$, we rewrite $Q'(k)$ in the form

$$Q'(k) = \left(\sum (q_{ik} + q_{ki})x'_i : i \in N(h) - \{k\}\right) + \left(q_{hk} + q_{kh}\right) x'_h$$

and, from $x''_i = x'_i$ for $i \neq h$, correspondingly write

$$Q''(k) = \left(\sum (q_{ik} + q_{ki})x'_i : i \in N(h) - \{k\}\right) + \left(q_{hk} + q_{kh}\right) v''.$$

This gives

$$Q''(k) = Q'(k) + \left(q_{hk} + q_{kh}\right)\left(v'' - x'_h\right)$$

as stipulated in (3). Applying the Definition (1) to $x''$, again with $x''_i = x'_i$ for $i \neq h$, we obtain

$$Q''(h) = \left(\sum (q_{ih} + q_{hi})x'_i : i \in N(h)\right) = Q'(h)$$

thereby establishing (4). Next, applying (2) of Proposition 1, we have:

$$\Delta''_k(v) = \left(v - x''_k\right)\left(Q''(k) + q_{kk}\left(x''_k + v\right)\right) \tag{7}$$

For $k \in N(h)$, (7) is the same as

$$\Delta''_k(v) = \left(v - x'_k\right)\left(Q''(k) + q_{kk}\left(x'_k + v\right)\right)$$

which establishes (5), and for $k = h$, using $x''_h = v''$, (7) is the same as

$$\Delta''_h(v) = (v - v'')\left(Q''(h) + q_{hh}(v'' + v)\right)$$

which establishes (6). This completes the proof.

*Comments 3* Each of the operations (3), (4), (5) and (6) has $O(1)$ computational complexity, hence $O(n)$ complexity over all $k \in N$ (for a matrix Q in lower triangular form, (3) can be slightly simplified in the same manner as noted in Comment 1).

*Comments 4* Propositions 1 and 2 can also be applied to the more general case that includes transitions to multiple fractional values f between 0 and 1.

*Comments 5* To reduce round-off error when updating f-flip evaluations (and to eliminate round-off error for a matrix Q with integer data), in the situation where each $x_k$ has the same set of rational-valued options, these options can be scaled to integer values to allow all operations on Q to be integer (for example, the options {0, 1/2, 1} and {0, 1/3, 2/3, 1} can be scaled to {0, 1, 2} and {0, 1, 2, 3}).

*Comments 6* There is a scale-factor distortion in choosing f-values between 0 and 1, which is retained when the f-values are scaled to integers as in Comment 5. For example, the effect of setting f = 1/2, gives each cross product term $x_i x_j$ only 1/4 the impact when both $x_i$ and $x_j = 1/2$. That is, $x_i x_j = 1/4$ in this case compared to $x_i x_j = 1$ when $x_i = x_j = 1$ (on the other hand, the $x_i x_j = 1/2$ case arises when $x_i = 1/2$ and $x_j = 1$). Thus, to more closely mirror the "1/2 impact option", a value such as f = 2/3 may be chosen, so that the values of $x_i x_j$ range over 0, 4/9, 6/9, 1 (or 0, 4, 6, 9 when {0, 2/3, 1} is scaled to {0, 2, 3}). Note this implies that the true $x_o$ value for a binary solution must be divided by the square of the largest $x_i$ value produced by integer scaling.

## 3 UBQP strategies using f-flips

There are several ways to use the foregoing results to produce strategies for UBQP exhibiting varying degrees of tradeoffs between intensification and diversification. We comment first on simple strategies that use f-flips in a multi-start setting, where each new start is launched from an $x'$ solution containing a specified subset $N^o$ of variables at fractional values (i.e., more precisely, $x'_k = f$ for $k \in N^o$). In these approaches each iteration selects a fractional variable $x_h$, $h \in N^o$ and executes an f-flip to produce a solution $x''$ in which $x''_h = 0$ or 1, followed by redefining $x' = x''$ and removing h from $N^o$. Once $N^o$ becomes empty, the method proceeds by employing a UBQP algorithm that retains binary values for all variables. After a chosen cutoff point is reached, the method launches a new re-start from a new selected $x'$.

To initiate a simple strategy of this type, the fractional $x'$ solution itself may come from a precursor binary $x'$ solution, which may be generated randomly or produced by prior solution effort (e.g., selecting one of the best solutions found to date). Then $N^o$ can be populated by choosing a set of components $x'_k$ of $x'$ to receive fractional values, as by making use of the $\Delta'_k(v)$ evaluations. For example, $N^o$ may then be generated by selecting

(i) a subset of variables whose $\Delta'_k(v)$ evaluations are worst.
(ii) a subset of variables whose $\Delta'_k(v)$ evaluations are best.
(iii) a subset composing a mix of (i) and (ii).
(iv) a subset picked at random.

The evaluation $\Delta'_k(v)$ for a binary valued $x'_k$ can be based on either $v = f$ or $v = 1 - x'_k$ (although $v = f$ might seem more natural since this is the value potentially to be assigned to $x'_k$). The process can select all elements of $N^o$ at once, or can select elements one at a time, updating $x'$ after each choice. Once $N^o$ is thus populated with a target number of variables for which $x'_k = f$, the method proceeds as indicated above by progressively driving these variables to integer values.

In contrast to these simple strategies, more intricate procedures can be employed that modify a customary neighborhood search process (without necessarily re-starting), by allowing variables to become fractional and then driven back to integer values. To prevent fractional-valued variables from recovering integer values too soon after becoming fractional, and to prevent integer-valued variables from being re-assigned a fractional value too soon after becoming integer, tabu lists can be employed in natural ways. Additional control can be exercised by periodically penalizing fractional assignments to drive all variables to receive integer values. These illustrative options represent only a portion of the ways that f-flips can be used to create UBQP strategies.

## 4 Conclusions

As noted, the introduction of f-flips, together with rules for evaluating them efficiently, provide opportunities for a wide range of strategies for UBQP problems, ranging from multi-start approaches to modified neighborhood search approaches. An interesting possibility that lies outside these types of strategies concerns the use of f-flips in conjunction with path relinking methods (Glover et al. 2000, 2004), which have emerged as components of some of the currently most effective UBQP algorithms (Wang et al. 2012). These methods generate trajectories from selected initiating solutions by moves that follow directions determined by associated guiding solutions. By modifying these methods to incorporate f-flips, the resulting fractional-step trajectories can delay the execution of full integer steps until the evaluations identify strong winners, thus modifying the sequence in which integer steps are made in the path. The use of f-flips can also change the path destination ultimately selected when more than one guiding solution is used with a given initiating solution. Once again, the foundation provided by the results of Sect. 2 enables such approaches to be carried out efficiently.

**Acknowledgments** We are indebted to an insightful referee who discovered the omission of a variable in our formulation of Proposition 1.

## References

Alidaee, B., Kochenberger, G., & Ahmadian, A. (1994). 0–1 Quadratic programming approach for the optimal solution of two scheduling problems. *International Journal of Systems Science*, *25*, 401–408.

Barahona, F., Grotschel, M., Junger, M., & Reinelt, G. (1988). An application of combinatorial optimization to statistical physics and circuit layout design. *Operations Research*, *36*(3), 493–513.

Bomze, I.M., Budinich, M., Pardalos, P.M., & Pelillo M. (1999). The maximum clique problem. In *Handbook of Combinatorial Optimization* (pp. 1–74). Springer.

Boros, E., & Hammer, P. L. (1991). The max-cut problem and quadratic 0–1 optimization polyhedral aspects, relaxations and bounds. *Annals of Operations Research*, *33*(1–4), 151–180.

Chardaire, P., & Sutter, A. (1994). A decomposition method for quadratic zero-one programming. *Management Science*, *4*(1), 704–712.

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

Glover, F., & Hao, J. K. (2010a). Efficient evaluations for solving large 0–1 unconstrained quadratic optimization problems. *International Journal of Metaheuristics*, *1*(1), 1–10.

Glover, F., & Hao, J. K. (2010a). Fast 2-flip move evaluations for binary unconstrained quadratic optimization problems. *International Journal of Metaheuristics*, *1*(2), 100–107.

Glover, F., Laguna, M., & Marti, R. (2000). Fundamentals of scatter search and path relinking. *Control and Cybernetics*, *39*, 654–684.

Glover, F., Laguna, M., & Marti, R. (2004). Scatter search and path relinking: Foundations and advanced designs, new optimization technologies in engineering. In G. C. Onwubolu & B. V. Babu (Eds.), *Studies in Fuzziness and Soft Computing* (Vol. 141, pp. 87–100). Berlin: Springer.

Hammer, P. L., & Rudeanu, S. (1968). *Boolean methods in operations research and related areas. Econometrics and operations research* (Vol. 5). Berlin: Springer.

Kochenberger, G., Glover, F., Alidaee, B., & Rego, C. (2004). A unified modeling and solution framework for combinatorial optimization problems. *OR Spectrum*, *26*, 237–250.

Kochenberger, G. A., Hao, J.-K., Lü, Z., Wang, H., & Glover, F. (2013). Solving large scale max cut problems via tabu search. *Journal of Heuristics*, *19*(4), 565–571.

Kochenberger, G., Hao, J. K., Glover, F., Lewis, M., Lü, Z., Wang, H., et al. (2014). The unconstrained binary quadratic programming problem: A survey. *Journal of Combinatorial Optimization*, *28*, 58–81.

Krarup, J., & Pruzan, A. (1978). Computer aided layout design. *Mathematical Programming Study*, *9*, 75–94.

Laughunn, D. J. (1970). Quadratic binary programming. *Operations Research*, *14*, 454–461.

Lewis, M., Kochenberger, G., & Alidaee, B. (2008). A new modeling and solution approach for the set-partitioning problem. *Computers and Operations Research*, *35*(3), 807–813.

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.
Wang, Y., Lü, Z., Glover, F., & Hao, J. K. (2013). Probabilistic GRASP-tabu search algorithms for the UBQP problem. *Computers and Operations Research*, *40*(12), 3100–3107.