Journal of the Operational Research Society

ISSN: 0160-5682 (Print) 1476-9360 (Online) Journal homepage: https://orsociety.tandfonline.com/loi/tjor20

A note on heuristic approach based on UBQP formulation of the maximum diversity problem

Bahram Alidaee & Haibo Wang

To cite this article: Bahram Alidaee & Haibo Wang (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, DOI: 10.1057/s41274-016-0031-4

To link to this article: https://doi.org/10.1057/s41274-016-0031-4

Published online: 21 Dec 2017.

Journal of the Operational Research Society (2017) 68, 102–110

www.palgrave.com/journals

A note on heuristic approach based on UBQP formulation of the maximum diversity problem

Bahram Alidaee[1]* and Haibo Wang[2]

[1]*Marketing Department, School of Business Administration, The University of Mississippi, P.O. Box 1848, University, MS 38677-1848, USA; and* [2]*College of Business Administration, Texas A&M International University, Laredo, TX 78041, USA*

The maximum diversity problem (MDP) is a challenging NP-hard problem with a wide range of real applications. Several researchers have pointed out close relationship between the MDP and unconstrained binary quadratic program (UBQP). In this paper, we provide procedures to solve MDP ideas from the UBQP formulation of the problem. We first give some local optimality results for *r-flip* improvement procedures on MDP. Then, a set of highly effective diversification approaches based on sequential improvement steps for MDP are presented. Four versions of the approaches are used within a simple tabu search and applied to 140 benchmark MDP problems available on the Internet. The procedures solve all 80 small- to medium-sized problems instantly to the best known solutions. For 22 of the 60 large problems, the procedures improved by significant amounts the best known solutions in reasonably short CPU time.

Journal of the Operational Research Society (2017) **68(1),** 102–110. doi:10.1057/s41274-016-0031-4;
published online 27 September 2016

Keywords: combinatorial optimization problem; unconstrained binary quadratic programming; maximum diversity problem; *r-flip* local search; diversification strategy

1. Introduction

The maximum diversity problem (MDP) consists of selecting a subset of m elements from a set of n elements ($m < n$) in such a way that the sum of the distances between the chosen elements is maximized. The MDP can be stated as follows:

$$\text{Max } f(x) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_i x_j \qquad (1)$$

$$s.t. \sum_{i=1}^{n} x_i = m. \qquad (2)$$

Here d_{ij} ($i,j = 1,\ldots,n$) is the distance between elements i and j. The binary variable x_i ($i = 1,\ldots,n$) is 1 if element i is chosen and 0 otherwise.

The MDP was first introduced by Kuo *et al* (1993). It appears in the literature under many different names with a variety of applications (refer to Marti *et al*, 2013, for a survey). Due to strong NP-hardness, the MDP presents a challenge to heuristic methods, (refer to Marti *et al*, 2013, who computationally compared 30 different heuristics available in the literature). Several recent heuristics have also been proposed by Aringhieri and Cordone (2011), Wang *et al* (2012, 2014), Marti *et al* (2013), and Wu and Hao (2013).

*Correspondence: Bahram Alidaee, Marketing Department, School of Business Administration, The University of Mississippi, P.O. Box 1848, University, MS 38677-1848, USA.
E-mail: balidaee@bus.olemiss.edu

Many researchers have pointed out that the MDP is closely related to the unconstrained binary quadratic program (UBQP) (see, for example, Kochenberger and Hao, 2014; Wang *et al*, 2009, 2012, 2014). In fact, the MDP can be formulated as UBQP. Although the researchers have pointed to this relationship, they all avoid directly using heuristics based on UBQP formulation of MDP. Heuristics based on UBQP has attracted many researchers in recent years, (see Kochenberger and Hao, 2014, for a survey, and Kochenberger and Glover, 2013, for special issue on xQx models). It is well known that discrete Hopfield neural network (DHNN), which also have many applications, can be formulated as UBQP. In a recent paper, Wang *et al* (2009) used DHNN and provided an effective heuristic for the MDP. Although Wang *et al* (2009) did not use directly (3) to solve the problem, their heuristic approach is directly using (1) and (2). They work on the set of feasible solutions (2) to improve upon objective function (1). Alidaee *et al* (2007) provided an effective tabu search based on UBQP for the maximum edge weight clique problem where instead of single equality in (2) there is inequality ($\leq$). From the heuristic approach there are significant differences between the two approaches because with inequality the space of feasible solutions is a lot bigger than when we have equality. Other recent effective heuristics for MDP includes GRASP, Aringhieri, *et al* (2007) and Silva *et al* (2007), Iterated Tabu Search, Palubeckis (2007), and Variable Search Neighborhood, Brimberg *et al* (2009). Refer to Marti *et al*

(2013), for comparison of 30 different heuristics applied to MDP, and recent survey paper by Kochenberger and Hao (2014) and special issue on xQx models by Kochenberger and Glover (2013) that provide comprehensive overviews of available algorithms.

The main contribution of this paper is to provide a set of highly effective diversification approaches for MDP that can be implemented within meta-heuristics. Building upon UBQP formulation of the problem, we provide local optimality for *r-flip* improvement procedures. A novel diversification approach based on the sequence to implement these improvement rules is presented. Four implementations combined with multi-start strategy embedded in a version of 'adaptive memory tabu search with strategic oscillation,' which was applied to UBQP model by Glover *et al* (1998), is applied to MDP. Our extensive computational experiment on 140 benchmark problems, available via http://www.optsicom.es/mdp/, that have been used by many researchers shows the methods are highly effective. In fact, our methods solve all 80 small- and medium-sized problems, ($n \leq 500$ and $m \leq 200$), instantly. For large-sized problems with $n = 2000, 3000$, and $m = 200, 300, 400, 500, 600$, our methods improved by significant amounts over the best known solutions for 22 of the rest of 60 problems in reasonably short CPU time. Note that data for 20 of the large-sized problems with $n = 2000$, and $m = 200$ are real numbers, not necessarily integers, (i.e., problems MDG-b.21-40). Best known solutions for these problems have not been improved since the original generation by Duarte and Marti (2007). Our methods improved by significant amounts the best known solutions for 14 of these problems.

In the rest of the paper, we first provide several theoretical results for local optimality of UBQP formulation of the MDP. Then adapted from our earlier research, we provide an *r-flip* local optimality rule for the problem. A simple and effective implementation of a *2-flip* rule for improvement of local search is provided. A simple *1-flip* rule and a *2-flip* rule that generate local optimal solutions are provided. Our diversification strategy which is based on sequence of implementation of *2-flip* improvement rule is introduced next. We then explain combined implementation of the diversification with a multi-start strategy within a simple tabu search for the problem. Then, computational experiment is presented, and finally concluding remarks are given.

2. Local optimality rules

With an appropriate large enough choice of constant penalty P an equivalent UBQP formulation of MDP can be given as follows:

$$MDP(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_i x_j - P\left(\sum_{i=1}^{n} x_i - m\right)^2. \quad (3)$$

Of course Eq. (3) can be simply stated as Max xQx, with binary variables, where Q is an n by n symmetric matrix created by carrying out the power and the summation and taking second derivatives with respect to each variable. Let q_{ij} $(i,j = 1,\ldots,n)$ be the ij-th element of matrix Q, and $\Delta_i(x)$ the first derivative of *MDP(x)* with respect to x_i. Now we have

$$\Delta_i(x) = \sum_{j \neq i} 2 q_{ij} x_j, \quad \text{for } i = 1, \ldots, n. \quad (4)$$

Given a solution x, define,

$$x_i = 1, \text{ for all } \Delta_i(x) > 0, \text{ and } x_i = 0, \text{for all } \Delta_i(x) \leq 0. \quad (5)$$

H1 is a *1-flip* heuristic rule for any Max xQx problem. It is well known that any solution x is locally optimal with respect to *1-flip neighborhood structure* if and only if (5) is satisfied.

H1 *1-flip* Improvement Rule for Max xQx

Start: A binary vector x, an order $\pi = (\pi_1,\ldots,\pi_n)$ of integers $1,\ldots,n$
Do while (condition (5) is not satisfied for all x_i)
  For $k = 1, n$
    (if condition (5) is not satisfied for x_j where $j = \pi_k$)
      $x_j = 1 - x_j$
      Update $\Delta_i(x)$ for $i = 1,\ldots,n$
    End if
  End for
  Choose a new sequence $\pi = (\pi_1,\ldots,\pi_n)$
End while

The sequences π used in the algorithm H1 and the starting point x make a significant difference in the final solution. Most algorithms in the literature use the *1-flip improvement rule* as part of their more sophisticated procedures. Those procedures use the 'natural order' (or sometimes the next most improvement rule) for improvement process. Borrowing from sequencing problems (eg, traveling salesman problems) in the literature, recently Alidaee *et al* (2015) used limited versions of *2_Opt, 3_Opt, 4_Opt,* and a combination of the three as a diversification approach adapted for UBQP. The authors showed the procedures were highly effective for general UBQP. Note that at each iteration, a new order is created and an improvement process is implemented according to the new order. The process creates an effective diversification approach to search a larger solution space. Such diversification procedure is fundamentally different from other diversification procedures in the literature. All diversification approaches in the literature manipulate the solution x; however, in the procedure H1 it is based on the steps (sequence) to implement the local search improvement. Below we present a result regarding local optimality of a feasible solution of the MDP problem. The proposition shows as why *1-flip* rule may not be attractive to be applied to UBQP formulation of MDP, i.e., formulation (3).

Proposition 1 *Considering 1-flip improvement rule for MDP(x). Any solution x is locally optimal if and only if m variables are 1 and the rest are zero.*

Proof Given a binary vector x with k variables equal to 1 where $k > m$, since P is a large enough constant then we can improve the objective function MDP by flipping any variable from 1 to zero one at a time until we get m variables equal to 1. Similarly if $k < m$ we can improve the objective function by flipping any variable from 0 to 1 until we get m variables equal to 1. Now, if m variables are 1 and the rest 0, by changing any variable we will clearly reduce the objective function. This completes the proof. □

The significance of Proposition 1 is that each member of the set of feasible solutions is locally optimal with respect to *1-flip* rule. This makes the implementation of *1-flip improvement rule* very difficult when applied to (3). Given a feasible solution a change on any x_i reduces the value of the objective function. However, an appropriate use of the *r-flip* rule is very promising as a solution procedure for the problem when applied to (3). Proposition 2 provides such local optimality condition for *r-flip* rule for MDP. The proposition is an adaptation of Theorem 6 from Alidaee *et al* (2010) which was given for general UBQP. First, consider a solution x for UBQP and a set of variables S where $|S| = r \leq n$. Define an *r-flip* 'move' by changing all elements $x_i \in S$ to $1 - x_i$. Of course, *1-flip* rule is a special case of *r-flip* rule. Alidaee *et al* (2010) gave the following result that can be used to efficiently apply *r-flip improvement rule* to MDP. Given a solution x, flip all elements x_i of a set of S variables to $1 - x_i$ to create a new solution x'. Now, define a set of equalities as follows:

$$f(x') - f(x) = \sum_{i\in S}(x'_i - x_i)\Delta_i(x) + \sum_{i,j\in S}(x'_i - x_i)(x'_j - x_j)(2q_{ij}) \quad (16)$$

$$\Delta_j(x) = \Delta_j(x) + \sum_{i\in S}(x'_i - x_i)(2q_{ij}),\ \forall j \in \{1,\ldots,n\}\backslash S$$
$$\Delta_j(x) = \Delta_j(x) + \sum_{i\in S\backslash\{j\}}(x'_i - x_i)(2q_{ij}),\ \forall j \in S \quad (17)$$

Theorem 1 derived from the quadratic formulation of the problem assures conditions of the local optimality when flipping a set of variables in the search process.

Theorem 1 (Alidaee *et al*, 2010): *Consider a problem f(x) =* Max *xQx with binary variables. Given a solution x, if all elements x_i in a set of S variables are flipped to $1 - x_i$, the amount of change in the objective function is equal (6) where x' is the resulting solution after the changes. Furthermore, when x is changed to x', the derivatives can be updated by (7).*

Let x be a feasible solution for MDP(x). Let S_1 and S_2 be two subsets of variables where if $x_i \in S_1$ then $x_i = 0$ and if $x_i \in S_2$ then $x_i = 1$ such that $|S_1| = |S_2| = l$, and let $S = S_1 \cup S_2$. If each $x_i \in S$ is flipped to $1 - x_i$, then we keep feasibility in the problem. Now, for the feasible solution x, define

$$\bar{\Delta}_i(x) = \sum_{j:x_j\notin S_2} 2q_{ij}x_j \quad \text{for } i = 1,\ldots,n. \quad (8)$$

Proposition 2 *Given a feasible solution x, and two sets of variables S_1 and S_2 as defined above. If each $x_i \in S$ is flipped to $1 - x_i$, the amount of change in the objective function is equal to (9) where x' is the resulting feasible solution after the changes.*

$$MDP(x') - MDP(x) = \left(\sum_{i\in S_1}\bar{\Delta}_i(x) + \sum_{i,j\in S_1}2q_{ij}(x)\right) - \left(\sum_{i\in S_2}\bar{\Delta}_i(x) + \sum_{i,j\in S_2}2q_{ij}\right). \quad (9)$$

Proof Note that S includes $r = 2l$ elements where l elements have values 1 and l elements have values 0. Now since feasibility is kept, using Equations (6–8) the desired result follows. □

Note 1 If the quantity (9) is larger than 0, flipping all elements $x_i \in S$ to $1 - x_i$ will improve the objective function. This can be used to give an *r-flip improvement rule*, H2 for MDP. Furthermore, assume we have an x with $m - l$ variables equal to 1 and the rest equal to 0. Now, Proposition 2 can be interpreted as the two sets S_1 and S_2 competing to be included in the rest of $m - l$ variables to create a feasible solution. We use this fact and give two rules, H3 and H4, respectively, *1-flip* and *2-flip* greedy rules. Feasible solutions generated by H3 and H4 will be used as starting solutions in our final implementation. In H3, we start with $x = 1$ and one-at-a-time change one variable to 0 until we reach a feasible solution. In H4, however, we start with $x = 0$ and at each iteration $k = 2,\ldots,m$, we use two-variable exchange to create a local optimal solution with only k components equal to 1. The process continues until we reach a local optimal feasible solution with $k = m$. Also note that in implementing rules, formula (10) and update of values $\bar{\Delta}_i$ for $i = 1,\ldots,n$, can significantly be simplified as follows. When $x_i = 1$ and $x_j = 0$ are flipped to $x_i = 0$ and $x_j = 1$, then we have

$$MDP(x') - MDP(x) = \bar{\Delta}_j(x) - \bar{\Delta}_i(x). \quad (9')$$

The update for $\bar{\Delta}_i$ are simplified by as follows:

$$\bar{\Delta}_k(x) = \bar{\Delta}_k(x) - 2d_{ki} + 2d_{kj},\ \forall k \neq i,j$$
$$\bar{\Delta}_i(x) = \bar{\Delta}_i(x) + 2d_{ij} \quad . \quad (10)$$
$$\bar{\Delta}_j(x) = \bar{\Delta}_j(x) - 2d_{ij}$$

H2 *r-flip* Improvement Rule for *MDP(x)*

Start with a feasible solution x. In some sequence order π, choose $r = 2l$ variables (l variables with values 1 and l variables with values 0). Flip all r variables from x_i to $1 - x_i$ if $MDP(x') - MDP(x) > 0$. Continue the process until there is no more set of r variables that can be flipped.

Note that in H2, similar to H1 after each complete iteration we change the sequence π in some fashion to start with a new sequence.

H3 *1-flip* Greedy Improvement for *MDP(x)*

Start: with $x = 1$
Do $k = 1, n - m$
$j - \arg\min_{i:x_i=1}\{\bar{\Delta}_i(x)\}$
Set $x_j = 1 - x_j$
Update $\bar{\Delta}_i(x)$ (for $i = 1,\ldots,n$)
End do

H4 *2-flip* Local Optimal Rule for *MDP(x)*

Start with $x = 0$, let i, j be $\arg\max_{k,l}\{2d_{kl}\}$, and set $x_i = x_j = 1$.
Do $k = 1, m - 2$
$j - \arg\max_{i:x_i=0}\{\bar{\Delta}_i(x)\}$
Set $x_i = 1 - x_i$
Update $\bar{\Delta}_i(x)$ (for $i = 1,\ldots,n$)
Do while (there is $x_i = 1$ and $x_j = 0$ with $\bar{\Delta}_i(x) < \bar{\Delta}_j(x)$)
Set $x_i = 1 - x_i$ & $x_j = 1 - x_j$
Update $\bar{\Delta}_i(x)$ (for $i = 1,\ldots,n$)
End while
End do

Note that H3 has been used in iterative tabu search by Palubeckis (2007), and GRASP by Aringhieri *et al* (2008) for MDP. Also, H4 in a different context, traveling salesman problems, has been used as initial solution, Gendreau *et al* (1992). In the next section, we first present diversification approach based on sequencing steps to improve the solution then we provide a simple tabu search for the MDP.

3. Diversification strategy and a simple tabu search for MDP

3.1. Diversification processes

The diversification procedures in this paper depend on what improvement step is taken next. All diversification approaches in the literature use some manipulation of the solution x. However, steps taken to implement improvement were first shown to be an effective diversification approach for general UBQP by Alidaee *et al* (2015). Appropriate sequences to implement improvement process can create a set of a huge number of possible diversification approaches. Borrowed from sequencing problems (eg, traveling salesman problem), the authors adapted limited versions of *2_Opt, 3_Opt, 4_Opt*, and a combination of the three (called in this paper *ALL*) for UBQP to implement improvement processes. We simply use several ways to create diverse sets of orders to implement improvement procedures. As we mentioned, *n_Opt* neighborhood procedures are well studied in the context of sequencing problems; however, here we are using them as steps to implement the 2_Opt procedure in a 0–1 optimization problem, MDP.

To illustrate the diversification procedures used in this paper, consider the following processes. At any stage of a process let $x = (x_1, \ldots, x_n)$ be a feasible solution and $\pi = (\pi_1, \ldots, \pi_n)$ a sequence of the *1,…,n* numbers where π_i is the position of the *i*-th variable. Assume that we are in the process of implementing a *2_Opt* strategy, and we have just applied the *2-flip* rule one at a time according to the sequence π. Next, we want again to apply the *2-flip* rule. We first randomly choose two numbers $1 \leq j < k < n$. In sequence π we now create 3 blocks of variables $B_1 = (\pi_1, \ldots, \pi_j)$,$B_2 = (\pi_{j+1}, \ldots, \pi_k)$, and $B_3 = (\pi_{k+1}, \ldots, \pi_n)$, see Figure 1. We randomly implement one of the 5 choices of sequences as shown in Figure 1. For example, if choice 1 is picked, we start to implement *2-flip* rule clockwise (*right arrow*) on variables in block 1, then counterclockwise (*left arrow*) on variables in block 2, and finally clockwise (*right arrow*) on variables in block 3. Similarly, if choice 2 is to be implemented, we start to implement *2-flip* rule clockwise (*right arrow*) on variables in block 2, then counterclockwise (*left arrow*) on variables in block 1, and finally counterclockwise (*left arrow*) on variables in block 3. These five *2_Opt* choices create many possible sequences as the search continues. Thus as the search continues, implementing *2-flip* rule according to each sequence creates many possible diverse solutions x. In the similar fashion, we can apply *3_Opt* or *4_Opt* strategies. In *3_Opt*, we are limiting the process to 24 cases, Figure 2, and in *4_Opt* to only 8 possible cases, Figure 3, which is a limited version of the so-called double-bridge *4_Opt* moves. We also applied the process by randomly choosing one of the 37 possible moves, which we called *ALL*.

3.2. Simple tabu search with sequencing rule and multi-start strategy

We use our diversification approach with a multi-start strategy within a simplified version of 'adaptive memory tabu search with strategic oscillation'. To simplify presentation of the pseudocode for our implementation of the procedures, we use the following definitions.

x, a binary starting feasible solution of n variables,

x^*, best solution found so far by an algorithm,

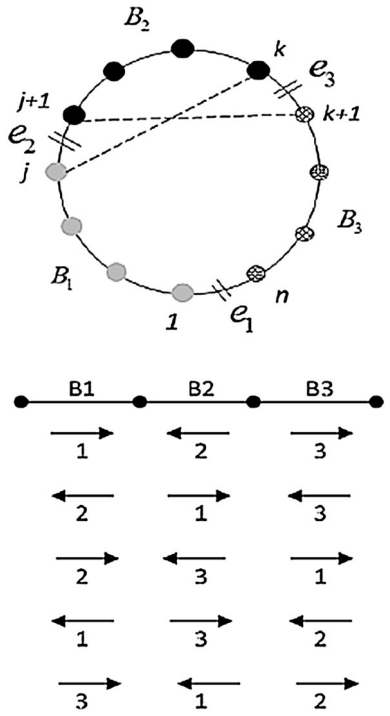

Figure 1 Implementation of a limited *2_Opt* strategy, 5 possible cases.

$Z = xQx$, value of the objective function for the variable x,

$Z^* = x^*Qx^*$, value of the objective function for the best solution found,

$J_{next}(x)$, set of variables where conditions of local optimality for *2-flip* rule are not satisfied, $J_{next}(x) = \{i \neq j : \bar{\Delta}_i(x) > \Delta_j(x)$ with $x_i = 0, x_j = 1\}$,

p1, p2, two integer constants where $p1 < p2 \leq n$, their values depend on the problem,

Tabu_ten, a constant integer number as tabu tenure (maximum value a variable can remain tabu),

Tabu(i), for $i = 1,\ldots,n$, a vector representing tabu status of variables,

π, a sequence of $1,\ldots,n$.

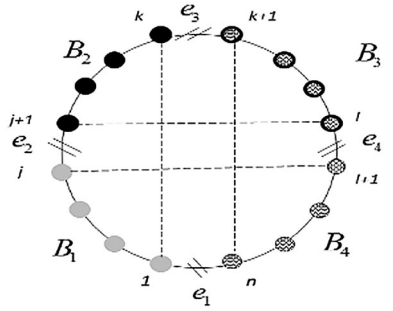

Figure 3 Implementation of a limited double-bridge *4_Opt* strategy, 8 possible cases.

We present a pseudocode of the simple tabu search, H5, that we applied using the 4 procedures. Note that in the pseudocode we can start with any feasible solution including results generated by H3 and H4. The "*N_Opt*" in the code calls for a change on sequence π. This is done after each complete *2-flip* iteration for $1,\ldots,n$. If the algorithm is in the process of implementing *2_Opt, 3_Opt*, or *4_Opt* strategies, we randomly choose one of the 5, 24, and 8 choices, respectively, to change the sequence (Figures 1, 2, and 3). However, if the algorithm is in the process of implementing *ALL*, we randomly choose one of the 37 moves to change the sequence. In each case, the output is a new sequence π that provides an order to implement the *2-flip* improvement process.

The procedure also randomly chooses a set of variables, x_i, and flips them $1 - x_i$. This is done after the local optimality condition is reached by calling "*rand_var_change(.).*" This is done according to the following rule. When local optimality is reached a random number, *p,* is generated in the interval $p \in [1, K]$ where K is changing in the interval *[p1, p2]*. Then we randomly choose two sets of variables S_1 and S_2 where $|S_1| = |S_2| = p$, $x_i = 0$ if $x_i \in S_1$, and $x_i = 0$ if $x_i \in S_2$. Finally we flip all $x_i \in S_1 \cup S_2$ to $1 - x_i$. The idea is related to the strategic oscillation used in many tabu search settings, (see, for example, Glover *et al* (1998), for implementation on UBQP). Also note that in our algorithm we use short-term so-called, 'recency,' tabu tenure structure, but we do not use long-term, 'frequency' structure, as Glover *et al* (1998) used in their procedure. We also use a simple *aspiration* criterion in our

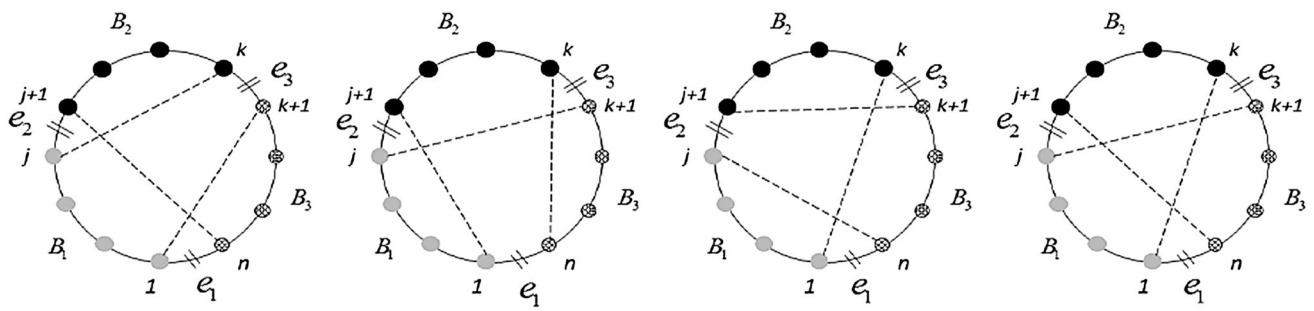

Figure 2 Implementation of a limited *3_Opt* strategy, 24 possible cases.

tabu search. If the objective value of a new solution is strictly better than the best known solution found so far, we override the tabu status for the variable to be changed. Our procedure expands over 'adaptive memory tabu search with strategic oscillation,' Glover *et al* (1998), however similar to all meta-heuristics it has elements of many different algorithms and thus is hybrid in nature. We use multi-start strategy which has been used by many authors, eg, Palubeckis (2007), and random change of variables, used for example by Brimberg *et al* (2009), called 'shaking.'

H5 Simple Tabu Search for MDP

Initialization:
Set: x= a starting feasible solution, $x^*=x$, $Z=Z^*=xQx$, *p1, p2, Tabu_ten,*
Tabu(i)=0, for *i=1,...,n,* and $\pi=(\pi_1,\ldots,\pi_n)$, calculate J_{next} .
Do while (until some stopping criteria, e.g., time limit, is reached)
Do *K=p1, p2*
Do while ($J_{next}\neq\varnothing$)
Do *i=1, n-1*
$L1=\pi_i$
Do *j=i+1, n*
$L2=\pi_j$
If ($L1, L2\in J_{next}$) **Then**
$\bar{Z}=\bar{x}Q\bar{x}$, for $\bar{x}_j=x_j$, $j\neq L1, L2$, and $\bar{x}_{L1}=1-x_{L1}, \bar{x}_{L2}=1-x_{L2}$,
If *(((Tabu(L1)=0) .and. (Tabu(L2)=0)) .or.* ($\bar{Z}>Z^*$)) **Then**
$x_{L1}=1-x_{L1}, x_{L2}=1-x_{L2}$,
Update: $J_{next}(x)$, and *Tabu(j), for j=1,...,n,*
$Z=\bar{Z}$,
If ($\bar{Z}>Z^*$) $Z^*=\bar{Z}$,
End If
End If
End do
End do
Call *N_Opt(.)*
End while
Call *rand_var_change(.)*
End do
End while

4. Computational experiment

To test effectiveness of the procedures using multi-start and *n_Opt* (*2, 3, 4_Opt,* and a combination of these 3 sequential moves, *ALL*) diversification strategies, we applied the 4 procedures on the sets of 140 benchmark problems available on the Internet that have been used by many researchers. The problems are available via http://www.optsicom.es/mdp/. Best known solutions were taken from this website and the articles (Aringhieri and Cordone, 2011; Kochenberger and Glover, 2013; Marti *et al*, 2013; Wang *et al*, 2009, 2012, 2014; Wu and Hao, 2013). We wrote codes using FORTRAN and ran them on a super computer where there is the capability of parallelism. Although we used fairly strong data structures to cut much of the unnecessary computations, we did not use the capability of parallelism available on the Internet. The FORTRAN program was running sequentially on a single core of a SGI Altix 8XE cluster with Intel Xeon Quad-core E5420 Harpertown processors, which are 2.5 GHz CPU with 8 GB memory. Table 1 categorizes data for these sets of problems. Below we present the result of our experiment for each set of problems.

4.1. Initial parameter settings

To solve each problem using one of the procedures, *2_Opt, 3_Opt, 4_Opt,* and *ALL*, we used three starting feasible solutions: generated feasible solution by H3, H4, and a random feasible solution. Furthermore, for each problem, each procedure, and each starting solution, we solved the problem with two different starting sequences: one starting with a sequence π as the natural numbers and the other in the reverse order of natural numbers. Thus, each problem was solved by each procedure six times and best results are reported. Initial

Table 1 Set of 140 problems used in the computational experiment

ID	*n*	*m*	*dij*
SOM-b. 1-20	100–500	10–200	Integer
GKD-c. 1-20	500	50	Real
MDG-a. 1-20	500	50	Real
MDG-a. 21-40	2000	200	Integer
MDG-b. 1-20	500	50	Real
MDG-b. 21-40	2000	200	Real
MDG-c. 1-20	3000	300, 400, 500, 600	Integer

computations for solving some of the problems showed the values of *Tabu_ten, p1, p2*, and the total time *'Total_time'* were the most important consideration in reaching quality solutions. The value of *Total_time* given to a problem was divided equally to each of the six ways of solving the problem by a procedure. For example, if a *2_Opt* strategy was applied to a problem six times, assuming $n = 3000$, $m = 600$, and $Total\_time = 0.5(n + m) = 1800$ s, then each of the six ways of solving the problem was given exactly $0.083(n + m) =$ 300 s of CPU time.

From each set of problems in Table 1, except those with $m < 50$, we randomly chose one problem and solved it using combinations of *Tabu_ten*, *p1* and *p2*, with a $Total\_Time = 0.5(n + m)$ *seconds*, given as follows:

Tabu_ten: 5, 10, 15, 20, 25 (total of 5 cases)

p1: 2, 5, 10, 15, 0.01(n + m) (total of 5 cases)

p2: 10, 30, 50, 75, 100, for problems of (total of 5 cases)

Each of the 7 randomly chosen problems was solved by each of the 4 procedures using all combinations of the above parameters. This is 5^3 possible cases; however, since we always used $p1 < p2$, the number of combinations used to solve each problem by each procedure was a few less than 5^3. For problems of size $m < 50$, we used: *Tabu_ten: 5, 10; p1: 2, 5; p2: 5, 10*. Using the result of these tests, we used $Tabu\_ten = 5$, $p1 = 2$, $p2 = 0.015(n + m)$, and $Total\_Time = (n + m)$ in the final experimentation for problems of $m \geq 50$. For problems of $m < 50$, we used $p1 = 2$, $p2 = 5$, $Tabu\_ten = 5$, and $Total\_Time = 10$ *s* in the final experimentation.

4.2. Analysis of final experimentation

We discuss the analysis for a set of problems based on size of the problems as follows. To save space, we will be prudent in presenting details of the results.

4.2.1. (n = 100–500, m = 10–200) SOM-b.1-20, GKD-c.1-20, MDG-a.1-20, MDG-b.1-20. Data in some of these sets some are integers, and some are real as shown in Table 1. All 4 procedures reached best known solutions instantly.

4.2.2. (n = 2000, m = 200) MDG-a.21-40, MDG-b.21-40. Data in the set MDG-a are integers. All 4 procedures reached the best known solutions. However, none of the procedures improved upon the best known solutions. Minimum, average, and maximum time to reach best known solutions for *2_Opt, 3_Opt, 4_Opt,* and *ALL* for all 20 problems, respectively, were (4.3, 10.9, 72.7), (9.9, 20.1, 46.6), (3.9, 13.4, 59.4), and (7.2, 13.5, 49.1) CPU seconds.

Data in the set MDG-b are real numbers. All 4 procedures reached the best known solutions. Minimum, average, and maximum time to reach best known solutions for *2_Opt, 3_Opt, 4_Opt,* and *ALL* for all 20 problems, respectively, were (15.4, 25.0, 112.2), (11.1, 20.4, 120.8), (10.5, 15.8, 60.1), and (14.2, 21.1, 93.9) CPU seconds. Since the introduction of these problems by Duarte and Marti (2007), no research has improved upon the best known solutions. As we ran the algorithms until the time limit, all procedures improved by a significant amount upon the best known solutions for 14 of the problems. Table 2 shows details of the results. As shown in the table, on average, performances of the algorithms were in the order of *2_Opt, 3_Opt, 4_Opt*, and *ALL*. This means for these problems a simpler sequence was more effective. The bold numbers in the tables indicate improvement by the algorithm over best known solutions.

4.2.3. (n = 3000, m = 300, 400, 500, 600) MDG-c.1-20 Data in the set MDG-c are integers. Table 3 shows results for the algorithms. All 4 procedures reached best known solutions. Minimum, average, and maximum time to reach best known solutions for *2_Opt, 3_Opt, 4_Opt*, and *ALL* for all 20 problems, respectively, were (40.6, 202.0, 666.3), (56.1, 200.1, 664.5), (30.3, 131.4, 670.3), and (50.8, 179.5, 641.7) CPU seconds. As we ran the algorithms until the time limit was reached, all procedures improved by a significant amount upon the best known solutions for 8 of the problems. Table 3 shows details of the results. As shown in the table, on average, performances of the algorithms for these problems were in the order of 4*_Opt, ALL, 2_Opt*, and *3_Opt*. The bold numbers in the tables indicate improvement by the algorithm over best known solutions.

From the above analysis, we cannot make a clear conclusion whether simpler or more sophisticated sequences are more effective as diversification approaches. This also was confirmed by implementing the Friedman ranking test on the results of 4 options in each Tables 2 and 3, as the *p* value in each case was significantly larger than 0.05.

Table 2 Results for a set of MDG-b.21-40 problems

ID	*Best*	*Objective function value*			
	Known	*2_Opt*	*3_Opt*	*4_Opt*	*All*
MDG-b.21	11299894.86	11299894.83	11299894.83	11299894.83	11299894.83
MDG-b.22	11286775.59	**11292398.16**	**11292398.16**	**11292398.16**	**11292398.16**
MDG-b.23	11299940.87	11299940.87	11299940.87	11299940.87	11299940.87
MDG-b.24	11290874.16	**11290943.67**	**11291119.49**	**11291119.49**	**11290989.34**
MDG-b.25	11296066.66	**11298026.88**	**11298005.37**	**11298005.37**	**11297736.67**
MDG-b.26	11292295.89	**11298430.42**	**11298430.42**	**11298430.42**	**11298430.42**
MDG-b.27	11305676.85	11305676.85	11305676.85	11305676.85	11305676.85
MDG-b.28	11279916.06	**11282693.51**	**11282881.84**	**11282908.53**	**11282910.88**
MDG-b.29	11297188.33	**11297285.16**	**11297339.48**	**11297339.48**	**11297339.48**
MDG-b.30	11296414.93	**11298064.59**	**11298064.59**	**11297466.80**	**11297126.06**
MDG-b.31	11288901.49	11288901.49	11288901.49	11288901.49	11288901.49
MDG-b.32	11279820.13	**11283476.86**	**11283476.86**	**11283476.86**	**11283512.95**
MDG-b.33	11296297.93	**11298038.13**	**11298038.13**	**11297549.38**	**11297641.61**
MDG-b.34	11281245.5	**11290482.64**	**11290202.88**	**11290482.64**	**11290202.88**
MDG-b.35	11307424.26	11307424.26	11307424.26	11307424.26	11307424.26
MDG-b.36	11289469.27	**11302892.01**	**11302892.01**	**11302892.01**	**11302892.01**
MDG-b.37	11290544.79	**11295773.76**	**11295773.76**	**11295683.63**	**11295683.63**
MDG-b.38	11288570.85	**11296535.52**	**11296535.52**	**11296535.52**	**11296535.52**
MDG-b.39	11295054.2	11295054.20	11295006.25	11294853.69	11295006.25
MDG-b.40	11307104.8	**11309162.63**	**11308969.82**	**11309162.63**	**11309162.63**
Avg.	11293473.9	11296554.8	11296548.6	11296507.1	11296470.3

Table 3 Results for a set of MDG-c.1-20 problems

ID	*Best*	*Objective function value*			
	Known	*2_Opt*	*3_Opt*	*4_Opt*	*All*
MDG-c.1	24924685	**24926344**	**24926344**	**24925476**	**24926344**
MDG-c.2	24909199	**24910996**	24909199	**24912546**	**24912546**
MDG-c.3	24900820	**24905218**	**24905218**	**24905218**	**24903963**
MDG-c.4	24904964	**24909710**	**24909710**	**24909710**	**24909710**
MDG-c.5	24899703	24899703	24899703	24899703	24899703
MDG-c.6	43465087	43465087	43465087	43465087	43465087
MDG-c.7	43477267	43477267	43477267	43477267	43477267
MDG-c.8	43458007	**43465572**	**43465544**	**43465572**	**43465572**
MDG-c.9	43448137	43448137	43448137	43448137	43448137
MDG-c.10	43476251	43476251	43476251	43476251	43476251
MDG-c.11	67009114	**67021132**	**67021132**	**67021132**	**67021132**
MDG-c.12	67021888	67021888	67021888	67021888	67021888
MDG-c.13	67024373	67024373	67024373	67024373	67024373
MDG-c.14	67024804	**67030190**	**67030190**	**67030190**	**67030190**
MDG-c.15	67056334	67056334	67056334	67056334	67056334
MDG-c.16	95637733	**95638929**	**95638929**	**95638929**	**95638929**
MDG-c.17	95645826	95645826	95645826	95645826	95645826
MDG-c.18	95629207	95629207	95629207	95629207	95629207
MDG-c.19	95633549	95633549	95633549	95633549	95633549
MDG-c.20	95643586	95643586	95643586	95643586	95643586
Avg.		57761465.0	57761373.7	57761499.1	57761479.7

However, in the algorithms we tracked to see which of the starting solutions, i.e., results from H3, H4, and a random feasible solution, provided best known solutions more often. Starting solution generated from heuristic H4 at each iteration k ($k \leq m$) gives a locally optimal solution with k variables equal to 1 until we reach a feasible solution with m variables equal to 1. The solution is generated in instant CPU time and the result is fairly good. However, by itself it

did not provide the best known solution for any of the 140 problems. Starting solution generated by H3 is also done in instant CPU time; however, the result of course is not as good a solution as H4 provides since it does not go through the process of local optimality as rigorously as H4 does. However, using the two starting solutions within the simple tabu search in combination with the diversification sequences, H3 provided best known solutions more often. The reason was because the starting solution generated by H4 stuck 'strongly' in locally optimal solution from the beginning. Using the procedures to get out of local optimal solutions was more difficult and time consuming. In fact, similarly a random starting solution did a better job overall compared to starting solution by H4. Thus, from our experiment with these sets of 140 benchmark problems we conclude a starting solution which is not too good from the beginning and not too random from the beginning provided better final solutions.

We should mention that although our procedures improved upon 8 of the 20 largest problems, MDG-c set, since this set of problems is fairly new, introduced in 2013, it is not too surprising to see improvement upon them. However, we were very surprised that our procedures improved upon 14 of the 20 large MDG-b set problems since these problems were introduced in 2007, and no improvement by any researcher has been reported in published research. We think such improvements are due to the diversification nature of our procedures.

5. Conclusion

In this paper, the maximum diversity problem (MDP) was considered. Local optimality results for *r-flip improvement rule* based on UBQP formulation were presented. Appropriate simplification for implementation was presented. A highly effective set of diversification approaches based on sequencing procedure were presented. Four implementation versions of the approaches combined with three multi-start strategies within a simple tabu search were considered. An extensive computational experiment on 140 benchmark problems that are available on the Internet and used by many researchers was presented. Computational results showed the procedures were highly effective. Our methods produced new best solutions for 22 of the large problems. The next step is to implement a more clever implementation of the diversifications provided here to very-large-scale problems of MDP, as well as extension of MDP such as maximally diverse grouping problems.

Acknowledgments—We thank the two referees and the editors for providing valuable comments that greatly improved the presentation of the paper.

References

Alidaee B, Glover F, Kochenberger G and Wang H (2007). Solving the maximum edge weight clique problem via unconstrained quadratic programming. *European Journal of Operational Research* **181**(2):592–597.

Alidaee B, Kochenberger G and Wang H (2010). Theorems supporting r-flip search for pseudo-boolean optimization. *International Journal of Applied Metaheuristic Computing* **1**(1):93–109.

Alidaee B, Sloan H and Wang H (2015). Simple and fast novel diversification approach for the UBQP based on sequential improvement local search. submitted.

Aringhieri R, Cordone R and Melzani Y (2008). Tabu search versus GRASP for the maximum diversity problem. *4OR* **6**(1):45–60.

Aringhieri R and Cordone R (2011). Comparing local search metaheuristics for the maximum diversity problem. *Journal of the Operational research Society* **62**(2):266–280.

Brimberg J, Mladenovic N, Urosevic D and Ngai E (2009). Variable neighborhood search for the heaviest k-subgraph. *Computers & Operations Research* **36**(11):2885–2891.

Duarte A and Martí R (2007). Tabu search and grasp for the maximum diversity problem. *European Journal of Operational Research* **178**(1):71–84.

Gendrea M, Hertz A and Laporte G (1992). New insertion and postoptimization procedures for the traveling salesman problem. *Operations Research* **40**(6):1086–1094.

Glover F, Kochenberger G and Alidaee B (1998). Adaptive memory tabu search for binary quadratic programs. *Management Science* **44**(3):336–345.

Kochenberger G, Hao J-K, Glover F, Lewis M, Lu Z, Wang H and Wang Y (2014). The unconstrained binary quadratic programming problem: a survey, *Journal of Combinatorial Optimization* **28**(1):58–81.

Kochenberger G and Glover F (2013). Introduction to special xQx issue. *Journal of Heuristics* **19**(4):525–528.

Kuo CC, Glover F and Dhir KS (1993). Analyzing and modeling the maximum diversity problem by zero-one programming. *Decision Science* **24**(6):1171–1185.

Marti R, Gallego M, Duarte A and Pardo E (2013). Heuristics and metaheuristics for the maximum diversity problem. *Journal of Heuristics* **19**(4):591–615.

Palubeckis G (2007). Iterated tabu search for the maximum diversity problem. *Applied Mathematics and Computation* **189**(1):271–383.

Silva GC, de Andrade MRQ, Ochi LS, Martins SL and Plastino A (2007). New heuristics for the maximum diversity problem. *Journal of Heuristics* **13**(4):315–336.

Wang J, Zhou Y, Yin J, Zhang Y (2009). Competitive hopfield network combined with estimation of distribution for maximum diversity problems. *IEEE Transactions* on *Systems*, *Man*, and *Cybernetics*, *Part B*: *Cybernetics* **39**(4):1048–1066.

Wang JH, Zhou Y, Cai Y, Yin J (2012). Learnable tabu search guided by estimation of distribution for maximum diversity problems. *Soft Computing: A Fusion of Foundations Methodologies and Applications* **16**(4):711–728.

Wang Y, Hao J-K, Glover F and Lu Z (2014). A tabu search based memetic algorithm for the maximum diversity problem. *Engineering Applications of Artificial Intelligence* **27:**103–114.

Wu Q and Hao J-K (2013). A hybrid metaheuristic method for the Maximum Diversity Problem. *European Journal of Operational Research* **231**(2):452–464.

Received 12 October 2015;
accepted 6 July 2016