A tabu search based memetic algorithm for the maximum diversity problem

Yang Wang [a,b], Jin-Kao Hao [b,*], Fred Glover [c], Zhipeng Lü [d]

[a] *School of Management, Northwestern Polytechnical University, 127 Youyi West Road, 710072 Xi'an, China*

[b] *LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers, France*

[c] *OptTek Systems, Inc., 2241 17th Street Boulder, CO 80302, USA*

[d] *School of Computer Science and Technology, Huazhong University of Science and Technology, 430074 Wuhan, China*

Accepted to *Engineering Applications of Artificial Intelligence*, 2 Sept. 2013

Abstract

This paper presents a highly effective memetic algorithm for the maximum diversity problem based on tabu search. The tabu search component uses a successive filter candidate list strategy and the solution combination component employs a combination operator based on identifying strongly determined and consistent variables. Computational experiments on three sets of 40 popular benchmark instances indicate that our tabu search/memetic algorithm (TS/MA) can easily obtain the best known results for all the tested instances (which no previous algorithm has achieved) as well as improved results for 6 instances. Analysis of comparisons with state-of-the-art algorithms demonstrate statistically that our TS/MA algorithm competes very favorably with the best performing algorithms. Key elements and properties of TS/MA are also analyzed to disclose the benefits of integrating tabu search (using a successive filter candidate list strategy) and solution combination (based on critical variables).

keywords: combinatorial optimization; maximum diversity problem; metaheuristics; tabu search; memetic algorithm

* Corresponding author.

Email addresses: `sparkle.wy@gmail.com` (Yang Wang), `hao@info.univ-angers.fr` (Jin-Kao Hao), `glover@opttek.com` (Fred Glover), `zhipeng.lv@hust.edu.cn` (Zhipeng Lü).

1 Introduction

The maximum diversity problem (MDP) is to identify a subset M of a given cardinality m from a set of elements N, such that the sum of the pairwise distance between the elements in M is maximized. More precisely, let $N = \{e_1, \ldots, e_n\}$ be a set of elements and d_{ij} be the distance between elements e_i and e_j. The objective of the MDP can be formulated as follows [18]:

$$\begin{aligned} &Maximize \quad f(x) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij} \cdot x_i \cdot x_j \\ &\text{subject to} \quad \sum_{i=1}^{n} x_i = m, \quad x_i \in \{0,1\}, \quad i = 1, \ldots, n \end{aligned} \tag{1}$$

where each x_i is a binary (zero-one) variable indicating whether an element $e_i \in N$ is selected to be a member of the subset M.

The MDP is closely related to the unconstrained binary quadratic programming (UBQP) problem [17,20,30]. Given a symmetric $n \times n$ matrix $Q = (q_{ij})$, the UBQP problem is to identify a binary vector x of length n for the following function:

$$Maximize \quad g(x) = x'Qx = \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}x_i x_j \tag{2}$$

Contrasting the objective functions of the MDP and the UBQP, we observe that the MDP is a special UBQP with a cardinality constraint.

The MDP is an NP-hard problem and provides practical applications mainly including location, ecological systems, medical treatment, genetics, ethnicity, product design, immigration and admissions policies, committee formation, curriculum design, and so on [16,23].

Due to its theoretical significance and many potential applications, various solution procedures have been devised for the MDP problem. Exact algorithms are able to solve instances with less than 150 variables in reasonable computing time [3,22]. However, because of the high computational complexity, heuristic and metaheuristic algorithms are commonly used to produce approximate solutions to larger problem instances. Examples of these methods include various GRASP variants [1,2,7,26,27], tabu search based algorithms [4,5,25,29], variable neighborhood search [5,6], scatter search [5], iterated greedy algorithm [19] and hybrid evolutionary algorithm [8,16]. A comprehensive review concerning the MDP and an interesting comparison among the best MDP algorithms can be found in [23].

Our proposed TS/MA falls within the memetic algorithm classification as laid out in [24] (and in particular adopts the scatter search template described in [11]). First, we use tabu search to improve each solution generated initially or created by combining members of a current population. The TS moves are simple swaps that flip (or add and drop) solution elements, drawing on the successive filter candidate list strategy to accelerate the move evaluations. Second, we design a solution combination operator to take advantage of solution properties by reference to the analysis of strongly determined and consistent variables. Finally, we introduce a population rebuilding strategy that effectively maintains population diversity.

In order to evaluate the performance of TS/MA, we conduct experimental tests on 3 sets of challenging benchmarks with a total of 40 instances. The test results indicate that TS/MA yields highly competitive outcomes on these instances by finding improved best known solutions for 6 instances and matching the best known results for the other instances. Furthermore, we analyze the influence of some critical components and demonstrate their key roles to the performance of the proposed TS/MA algorithm.

The rest of the paper is organized as follows. Section 2 describes the proposed TS/MA algorithm. Section 3 presents experimental results and comparisons with state-of-the-art algorithms in the literature. Section 3.3 analyzes several essential components of TS/MA. Concluding remarks are given in Section 4.

2 Tabu Search/Memetic Algorithm

Algorithms that combine improvement methods with population-based solution combination algorithms, and hence that can be classified as memetic algorithms [24], often prove effective for discrete optimization [13]. By linking the global character of recombinant search with the more intensive focus typically provided by local search, the memetic framework offers interesting possibilities to create a balance between intensification and diversification within a search procedure. Our TS/MA algorithm follows the general memetic framework and is mainly composed of four components: a population initialization and rebuilding procedure, a tabu search procedure, a specific solution combination operator and a population updating rule. As previously noted, our procedure more specifically adopts the form of a scatter search procedure, and utilizes combinations from the structured class proposed for scatter search in [10].

2.1 Main scheme

The general architecture of our TS/MA algorithm is described in Algorithm 1. It starts with the creation of an initial population P (line 3, see Sect. 2.3). Then, the solution combination is employed to generate new offspring solution (line 8, see Sect. 2.5), whereupon a TS procedure (line 9, see Sect. 2.4) is launched to optimize each newly generated solution. Subsequently, the population updating rule decides whether such an improved solution should be inserted into the population and which existing individual should be replaced (line 14, see Sect. 2.6). Finally, if the population is not updated for a certain number of generations, the population rebuilding procedure is triggered to build a new population (line 21, see Sect. 2.3). In the following subsections, the main components of our TS/MA algorithm are described in detail.

Algorithm 1 Pseudo-code of TS/MA for the MDP

1: **Input**: an $n \times n$ matrix (d_{ij}), a given cardinality $m \leq n$
2: **Output**: the best solution x^* found
3: $P = \{x^1, \ldots, x^q\} \leftarrow$ Pop_Init() /* Section 2.3 */
4: $x^* = arg\ max\{f(x^i)|i = 1, \ldots, q\}$
5: **while** a stop criterion is not satisfied **do**
6:    $UpdateNonSucc = 0$
7:    **repeat**
8:       randomly choose two solutions x^i and x^j from P
9:       $x^0 \leftarrow$ Combination_Operator(x^i, x^j) /* Section 2.5 */
10:       $x^0 \leftarrow$ Tabu_Search(x^0) /* Section 2.4 */
11:       **if** $f(x^0) > f(x^*)$ **then**
12:          $x^* = x^0$
13:       **end if**
14:       $\{x^1, \ldots, x^q\} \leftarrow$ Pop_Update($x^0, x^1, \ldots, x^q$) /* Section 2.6 */
15:       **if** P does not change **then**
16:          $UpdateNonSucc = UpdateNonSucc + 1$
17:       **else**
18:          $UpdateNonSucc = 0$
19:       **end if**
20:    **until** $UpdateNonSucc > \theta$
21:    $P = \{x^1, \ldots, x^q\} \leftarrow$ Pop_Rebuild() /* Section 2.3 */
22: **end while**

2.2 Search space and evaluation function

Given an n element set $N = \{e_1, \ldots, e_n\}$, the search space Ψ of the MDP consists of all the m-element subsets of N; i.e., $\Psi = \{S|S \subset N, |S| = m\}$. Thus the search space size equals $\binom{n}{m}$. A feasible solution of the MDP can be conveniently represented as an n-vector of binary variables x such that exactly

m variables receive the value of 1 and the other $n - m$ variables receive the value of 0. Given a solution $x \in \Psi$, its quality or fitness is directly measured by the objective function $f(x)$ of Eq. (1).

2.3 Population initialization and rebuilding

The initial population contains q different local optimal solutions (q is a parameter and called the population size) and is constructed as follows. First, we randomly generate an initial feasible solution, i.e., any binary n-vector with exactly m elements assigned the value of 1. Then this solution is submitted to the tabu search procedure to obtain an improved solution which is also a local optimum but not necessarily a first local optimum encountered (see Sect. 2.4). Then, the solution improved by tabu search is added in the population if it does not duplicate any solution in the population. This procedure is repeated until the population size reaches the specified value q.

The rationale of this simple strategy is based on the following experimental observation. Given two local optimal solutions x^1 and x^2 which are achieved by the TS procedure, the average distance between x^1 and x^2 is generally no less than 10%, the distance being defined to be equal to $1 - c/m$ where c is the number of common elements of x^1 and x^2 and m the given cardinality.

This procedure is also used by the TS/MA algorithm when the population is not updated for θ consecutive generations (θ is a parameter and called the *population rebuilding threshold*, see Algorithm 1, line 20). In this case, the population is recreated as follows. First, the best solution x^* from the old population becomes the first member of the new population. Second, for each of the remaining solutions in the old population, we carry out the following steps: (1) randomly interchange $\rho \cdot m$ variables with the value of 1 and $\rho \cdot m$ variables with the value of 0 where $0 < \rho < 1$ (ρ is a parameter and called the *perturbation fraction*); (2) this perturbed solution is submitted to tabu search to obtain an improved solution; (3) if this refined solution is not a duplication of any solution in the new population, it is added in the new population; otherwise, the method returns to step (1).

2.4 Tabu search procedure

To improve the quality of a solution, we use a tabu search procedure which applies a constrained *swap* operator to exchange a variable having the value of 1 with a variable having the value of 0. More formally, given a feasible solution $x = \{x_1, \ldots, x_n\}$, let U and Z respectively denote the set of variables with the value of 1 and 0 in x. Then, the neighborhood $N(x)$ of x consists of all the solutions obtained by swapping two variables $x_i \in U$ and $x_j \in Z$. Since this swap operator keeps the m cardinality constraint satisfied, the neighborhood contains only feasible solutions. Clearly, for a given solution x, its neighborhood $N(x)$ has a size of $m \cdot (n - m)$.

To rapidly determine the move gain (the objective change on passing from the current solution to its neighboring solution), we apply the following technique which is similar to the technique used in [4].

First, we employ a vector Δ to record the objective variation of moving a variable x_i from its current subset U/Z into the other subset Z/U. This vector can be initialized as follows:

$$\Delta_i = \begin{cases} \sum_{j \in U} \ -d_{ij} \ \ (x_i \in U) \\ \sum_{j \in U} \ \ d_{ij} \ \ (x_i \in Z) \end{cases} \tag{3}$$

Then, the move gain of interchanging two variables $x_i \in U$ and $x_j \in Z$ can be calculated using the following formula:

$$\delta_{ij} = \Delta_i + \Delta_j - d_{ij} \tag{4}$$

Finally, once a move is performed, we just need to update a subset of move gains affected by the move. Specifically, the following abbreviated calculation can be performed to update Δ upon swapping variables x_i and x_j [21]:

$$\Delta_k = \begin{cases} -\Delta_i + d_{ij} \ \ (k = i) \\ -\Delta_j + d_{ij} \ \ (k = j) \\ \Delta_k + d_{ik} - d_{jk} \ \ (k \neq \{i, j\}, x_k \in U) \\ \Delta_k - d_{ik} + d_{jk} \ \ (k \neq \{i, j\}, x_k \in Z) \end{cases} \tag{5}$$

Given the size of the swap neighborhood which is equal to $m \cdot (n - m)$, it could be computationally costly to identify the best move at each iteration of tabu search. To overcome this obstacle, we employ the successive filter candidate list strategy of [12] that breaks a compound move (like a swap) into component operations and reduces the set of moves examined by restricting consideration to those that produce high quality outcomes for each separate operation.

For the swap move, we first subdivide it into two successive component operations: (1) move the variable x_i from U to Z; (2) move the variable x_j from Z to U. Since the resulting objective difference of each foregoing operation can be easily obtained from the vector Δ, we then pick for each component operation the top cls variables (cls is a parameter and called the candidate list size) in terms of their Δ values recorded in a non-increasing order to construct the candidate lists UCL and ZCL. Finally, we restrict consideration to swap moves involving variables from UCL and ZCL. The benefits of this strategy will be verified in Section 3.3.

It should be clear that cls impacts the performance of the TS procedure. A too large cls value may include some non-improving (unattractive) moves in the neighborhood exploration while a too small value could exclude some attractive moves. A parameter sensitivity analysis is provided in Section 3.3.1 where this parameter is studied in detail.

We give below a small example to illustrate how this method works. Suppose we have a matrix D and a solution $U = \{x_1, x_3, x_5, x_8\}$, $Z = \{x_2, x_4, x_6, x_7\}$, then the best swap move is obtained with the following steps:

$$D = \begin{pmatrix} 0 & 5 & 6 & 2 & 4 & 2 & 1 & 6 \\ 5 & 0 & 4 & 2 & 6 & 8 & 9 & 2 \\ 6 & 4 & 0 & 5 & 3 & 6 & 2 & 5 \\ 2 & 2 & 5 & 0 & 7 & 4 & 8 & 4 \\ 4 & 6 & 3 & 7 & 0 & 3 & 2 & 2 \\ 2 & 8 & 6 & 4 & 3 & 0 & 6 & 7 \\ 1 & 9 & 2 & 8 & 2 & 6 & 0 & 1 \\ 6 & 2 & 5 & 4 & 2 & 7 & 1 & 0 \end{pmatrix} \tag{6}$$

Step1: compute the objective variation of moving a variable from U to Z according to the Eq. (3) and obtain:

$\Delta_1 = -16$, $\Delta_3 = -14$, $\Delta_5 = -9$, $\Delta_8 = -13$

Likewise, compute the objective variation of moving a variable from Z to U and obtain:

$\Delta_2 = 17$, $\Delta_4 = 18$, $\Delta_6 = 18$, $\Delta_7 = 6$

Step2: pick cls variables, say $cls = 2$ with the best objective variation from U and Z respectively, then the selected candidate subsets will be $UCL = \{x_5, x_8\}$, $ZCL = \{x_4, x_6\}$.

Step3: compute move gains according to Eq. (4), where a move consists in swapping two variables from UCL and ZCL.

$\delta_{54} = -9 + 18 - 7 = 2$, $\delta_{84} = -13 + 18 - 4 = 1$, $\delta_{56} = -9 + 18 - 3 = 6$,

$\delta_{86} = -13 + 18 - 7 = -2$

Step4: determine the best move to interchange variables x_5 from UCL and x_6 from ZCL with the largest move gain of 6.

To ensure solutions visited within a certain span of iterations will not be revisited, tabu search typically incorporates a short-term memory, known as *tabu list* [12]. In our implementation, each time two variables x_i and x_j are swapped, two random integers are taken from an interval $tt = [a, b]$ (where a and b are chosen integers) as the tabu tenure of variables x_i and x_j to prevent any move involving either x_i or x_j from being selected for a specified number of iterations. (The integers defining the range of tt are parameters of our procedure, identified later.) Specifically, our tabu list is defined by a n-element vector T. When x_i and x_j are swapped, we assign the sum of a random integer from tt and the current iteration count $Iter$ to the i_{th} element $T[i]$ of T and the sum of another random integer from tt and $Iter$ to $T[j]$. Subsequently, for any iteration $Iter$, a variable x_k is forbidden to take part in a swap move if $T[k] > Iter$.

Tabu search then restricts consideration to variables not currently tabu, and at each iteration performs a swap move that produces the best (largest) move gain according to Eq. (4). In the case that two or more swap moves have the same best move gain, one of them is chosen at random.

To accompany this rule, a simple aspiration criterion is applied that permits a move to be selected in spite of being tabu if it leads to a solution better than the best solution found so far. The tabu search procedure terminates when the best solution cannot be improved within a given number α of iterations (α is a parameter and called the *improvement cutoff*).

The pseudo-code of the tabu search procedure is shown in Algorithm 2.

2.5 Solution combination by reference to critical variables

Our memetic algorithm uses a dedicated solution combination operator to generate promising offspring solutions. The combination operator is based on the idea of critical variables which are given the name *strongly determined and consistent variables* in [9]. In the context of the MDP, the notion of strongly determined and consistent variables can be defined as follows.

Definition 1 (Strongly determined variables). Relative to a given solution $x = \{x_1, x_2, ..., x_n\}$, let U denote the set of variables with the value of 1 in x. Then, for a specific variable $x_i \in U$, the (objective function) *contribution* of x_i in relation to x is defined as:

Algorithm 2 Pseudo-code of TS for the MDP

Input: a given solution x and its objective function value $f(x)$
Output: an improved solution x^* and its objective function value $f(x^*)$
Initialize vector Δ according to Eq. (3), initialize tabu list vector T by assigning each element with value 0, initialize U and Z composed of variables with value of 1 and 0 in x, respectively, $Iter = 0$, $NonImpIter = 0$, $x^* = x$, $f(x^*) = f(x)$
while $NonImpIter < \alpha$ **do**
  Identify top cls variables from U and top cls variables from Z in terms of the Δ value to construct UCL and ZCL
  Identify the index i^*_{nt} and j^*_{nt} of non-tabu variables from UCL and ZCL that leads to the maximum δ value (computed according to Eq. (4)) by swapping $x_{i^*_{nt}}$ and $x_{j^*_{nt}}$ (break ties randomly); Similarly identify i^*_t and j^*_t for tabu variables
  if $\delta_{i^*_t j^*_t} > \delta_{i^*_{nt} j^*_{nt}}$ and $f(x^*) + \delta_{i^*_t j^*_t} > f(x^*)$ **then**
    $i^* = i^*_t, j^* = j^*_t$
  else
    $i^* = i^*_{nt}, j^* = j^*_{nt}$
  end if
  $x_{i^*} = 0, x_{j^*} = 1$, $f(x) = f(x) + \delta_{i^* j^*}$, $U = U \backslash \{x_{i^*}\} \cup \{x_{j^*}\}$, $Z = Z \cup \{x_{i^*}\} \backslash \{x_{j^*}\}$
  Update Δ according to Eq. (5)
  Update T by assigning $T[i] = Iter + rand(tt), T[j] = Iter + rand(tt)$
  if $f(x) > f(x^*)$ **then**
    $x^* = x, f(x^*) = f(x)$
    $NonImpIter = 0$
  else
    $NonImpIter = NonImpIter + 1$
  end if
  $Iter = Iter + 1$
end while

$$VC_i(x) = \sum_{x_j \in U} d_{ij} \tag{7}$$

Obviously, the objective function of the MDP can be computed with regard to VC as follows:

$$f(x) = \frac{1}{2} \cdot \sum_{x_i \in U} VC_i(x) \tag{8}$$

We sort all the variables in a non-increasing order according to their objective function contribution and select the top β variables (β is a parameter) as strongly determined variables SD.

Definition 2 (Consistent variables). Relative to two local optimal (high quality) solutions x^i and x^j, let U_i and U_j respectively denote the set of variables with the value of 1 in x^i and x^j. Then, the consistent variables are defined as:

$$C = \{x_k | x_k \in U_i \cap U_j\} \quad (9)$$

Given two local optimal solutions x^i and x^j and a set of variables N, our critical variable combination operator constructs one offspring solution according to the following steps:

(1) Identify strongly determined variables SD_i and SD_j with regard to x^i and x^j, respectively;
(2) Select consistent variables that simultaneously emerges in SD_i and SD_j; i.e., $CS = SD_i \cap SD_j$;
(3) Randomly pick m-$|CS|$ variables from the set N-CS to satisfy the cardinality constraint (maintaining the number of variables with the value of 1 equal to m);
(4) Construct a feasible offspring solution by assigning the value 1 to the variables selected in steps (2) and(3) and assigning the value 0 to the remaining variables.

2.6 Population updating

The population updating procedure is invoked each time a new offspring solution is generated by the combination operator and then improved by tabu search. As in a simple version of the scatter search template of [11], the improved offspring solution is added into the population if it is distinct from any solution in the population and better than the worst solution, while the worst solution is removed from the population.

3 Experimental results and analysis

3.1 Benchmark instances

Three sets of benchmarks with a total of 40 large instances (with at least 2000 variables) are utilized to evaluate the performance of the proposed approach. Small and medium scale benchmarks are excluded in our experimentation because these problem instances can be easily solved by many heuristics in a very short time and present no challenge for our TS/MA algorithm.

(1) Random Type 1 instances (Type1_22): 20 instances with $n = 2000, m = 200$, where d_{ij} are integers generated from a [0,10] uniform distribution.

These instances are first introduced in [7] and can be downloaded from: http://www.uv.es/∼rmarti/paper/md p.html.

(2) ORLIB instances (b2500): 10 instances with $n = 2500, m = 1000$, where d_{ij} are integers randomly generated from [-100,100]. They all have a density of 0.1. These instances are derived from the UBQP problem by ignoring the diagonal elements and are available from ORLIB.

(3) Palubeckis instances (p3000 and p5000): 5 instances with $n = 3000, m = 0.5n$ and 5 instances with $n = 5000, m = 0.5n$, where d_{ij} are integers generated from a [0,100] uniform distribution. The density of the distance matrix is 10%, 30%, 50%, 80% and 100%. The sources of the generator and input files to replicate these problem instances can be found at: http://www.soften.ktu.lt/∼gintaras/max_div.html.

3.2 Experimental Protocol

Our TS/MA algorithm is programmed in C and compiled using GNU g++ on a Xeon E5440 with 2.83GHz CPU and 8GB RAM [1]. Following the DIMACS machine benchmark [2], our machine requires 0.43, 2.62 and 9.85 CPU seconds respectively for graphs r300.5, r400.5, and r500.5 compiled with gcc -O2. All computational results were obtained with the parameter values shown in Table 1 which are identified with the parameter sensitivity analysis provided in Section 3.3.1.

Given the stochastic nature of our algorithm, we solve each instance in the Type1_22 and ORLIB benchmarks 30 times, and solve each instance in the Palubeckis benchmark 15 times. For the comparative study reported in Section 3.5, TS/MA uses time limit as the stopping condition, as other reference algorithms did. For this purpose, we use SPEC - Standard Performance Evaluation Corporation (www.spec.org) to determine the performance difference between our computer and the reference machine of [29]. According to SPEC, our computer is slightly faster with a factor of 1.17. Hence, we set the time limit to 17, 256, 513 and 1538 seconds respectively for the instances of Type1_22, b2500, p3000 and p5000, which correspond to the stop condition used in [29].

[1] Upon the publication of the paper, the source code of the TS/MA algorithm will be made freely available to the public at: `http://www.info.univ-angers.fr/pub/hao/mdp.html`

[2] dfmax: ftp://dimacs.rutgers.edu/pub/dsj/clique/

Table 1
Settings of important parameters of the TS/MA algorithm

Parameters	Section	Description	Value
q	2.3	population size	10
θ	2.3	population rebuilding threshold	30
ρ	2.3	perturbation fraction	0.3
tt	2.4	tabu tenure interval	[15,25]
α	2.4	tabu search improvement cutoff	$6 \cdot m$
cls	2.4	candidate list size of each component operation	$\min(\sqrt{m}, \sqrt{n-m})$
β	2.5	number of strongly determined variables	$0.7 \cdot m$

Table 2
Post-hoc test for solution sets obtained by varying q

$p =$	5	10	20	30	40
10	0.00057				
20	0.00665	0.98856			
30	0.00954	0.97713	1.00000		
40	0.67881	0.08822	0.33919	0.40233	
50	0.99509	0.00447	0.03686	0.05000	0.93341

Table 3
Post-hoc test for solution sets obtained by varying θ

$\theta =$	10	20	30	40
20	0.94381			
30	0.97070	0.99994		
40	0.06421	0.32684	0.26205	
50	0.00410	0.04573	0.03171	0.90493

Table 4
Post-hoc test for solution sets obtained by varying tt

$tt =$	[1,15]	[15,25]	[15,50]	[25,50]	[25,100]
[15,25]	0.00039				
[15,50]	0.03906	0.80788			
[25,50]	0.03907	0.80786	1.00000		
[25,100]	0.98891	0.00492	0.19158	0.19122	
[50,100]	0.99959	0.00009	0.01470	0.01478	0.93517

Table 5
Post-hoc test for solution sets obtained by varying α

$\alpha =$	m	2 · m	3 · m	4· m	5· m	6· m	7· m	8· m	$9 \cdot m$
$2 \cdot m$	0.85330								
$3 \cdot m$	0.40765	0.99961							
$4 \cdot m$	0.18981	0.98742	0.99999						
$5 \cdot m$	0.07000	0.90758	0.99887	1.00000					
$6 \cdot m$	0.00000	0.00091	0.01459	0.05167	0.15025				
$7 \cdot m$	0.09187	0.93831	0.99961	1.00000	1.00000	0.11842			
$8 \cdot m$	0.56656	0.99999	1.00000	0.99983	0.99200	0.00640	0.99624		
$9 \cdot m$	0.00238	0.30550	0.76509	0.93839	0.99371	0.74514	0.98743	0.61304	
$10 \cdot m$	0.00033	0.10856	0.45123	0.72441	0.91867	0.94677	0.88215	0.30514	0.99999

3.3 Analysis of TS/MA parameters and key components

In this section, we conduct an parameter sensitivity analysis of the proposed TS/MA algorithm and study some of its key components.

3.3.1 Parameter sensitivity analysis

We first show a parameter sensitivity analysis based on a subset of 11 diverse instances. For each TS/MA parameter, we test a number of possible values

Table 6
Post-hoc test for solution sets obtained by varying cls

$cls =$	$m^{0.1}$	$m^{0.2}$	$m^{0.3}$	$m^{0.4}$	$m^{0.5}$	$m^{0.6}$	$m^{0.7}$	$m^{0.8}$	$m^{0.9}$
$m^{0.2}$	0.54620								
$m^{0.3}$	0.00009	0.20682							
$m^{0.4}$	0.00000	0.00069	0.80485						
$m^{0.5}$	0.00000	0.00001	0.27148	0.99846					
$m^{0.6}$	0.00000	0.00400	0.96156	0.99999	0.96747				
$m^{0.7}$	0.00006	0.16582	1.00000	0.85461	0.32774	0.97735			
$m^{0.8}$	0.02148	0.93895	0.96158	0.09351	0.00663	0.25356	0.93893		
$m^{0.9}$	0.36772	1.00000	0.34752	0.00210	0.00003	0.01062	0.28899	0.98470	
$m^{1.0}$	0.99005	0.99004	0.00952	0.00000	0.00000	0.00003	0.00676	0.32694	0.95488

Table 7
Post-hoc test for solution sets obtained by varying β

$\beta =$	$0.1 \cdot m$	$0.2 \cdot m$	$0.3 \cdot m$	$0.4 \cdot m$	$0.5 \cdot m$	$0.6 \cdot m$	$0.7 \cdot m$	$0.8 \cdot m$	$0.9 \cdot m$
$0.2 \cdot m$	1.00000								
$0.3 \cdot m$	0.89398	0.91762							
$0.4 \cdot m$	0.40401	0.44709	0.99885						
$0.5 \cdot m$	0.08163	0.09732	0.89377	0.99942					
$0.6 \cdot m$	0.02051	0.02487	0.63285	0.97663	0.99999				
$0.7 \cdot m$	0.00309	0.00376	0.28305	0.80029	0.99359	1.00000			
$0.8 \cdot m$	0.93734	0.95360	1.00000	0.99617	0.83526	0.54017	0.21556		
$0.9 \cdot m$	0.99999	1.00000	0.95358	0.53992	0.13699	0.03792	0.00602	0.97662	
$1.0 \cdot m$	1.00000	1.00000	0.93739	0.49383	0.11605	0.03077	0.00497	0.96654	1.00000

while fixing the other parameters to their default values from Table 1. We test q (population size) in the range [5, 50], θ (population rebuilding threshold) in the range [10, 50], ρ (tabu search perturbation fraction) in the range [0.1, 1.0], α (tabu search improvement cutoff) in the range $[m, 10 \cdot m]$, cls (candidate list size) in the range $[m^{0.1}, m^{1.0}]$ and β (number of strongly determined variables) in the range $[0.1 \cdot m, m]$. Similarly, for the tabu tenure tt, we try several intervals in the range [1,100]. For each instance and each parameter setting, we conduct experiments under exactly the same conditions.

We use the Friedman test to see whether the performance of TS/MA varies significantly in terms of its average solution values when we vary the value of a single parameter as mentioned above. The Friedman test indicates that the values of ρ do not significantly affect the performance of TS/MA (with p-value=0.2983). This means that TS/MA is not very sensitive to the perturbation fraction when rebuilding the population. However, the Friedman test reveals a statistical difference in performance to the different settings of parameters q, θ, tt, α, cls and β (with p-values of 0.000509, 0.004088, 0.0001017, 1.281e-07, 1.735e-11 and 0.002715, respectively). Hence, we perform the Post-hoc test to examine the statistical difference between each pair of settings of these parameters and show the results in Tables 2 to 7.

Take the parameter q (population size) as an example, the p-value of 0.00057 (smaller than 0.05) of Post-hoc test for the pair of parameter settings $(5, 10)$ indicates a significant difference between these two settings. Figure 1 shows that setting $q = 10$ produces significantly better average results than setting $q = 5$. On the other hand, the Post-hoc result for the pair of parameter settings $(10, 20)$ is 0.98856 (greater than 0.05), the difference between these two settings are thus not statistically significant. By observing Tables 2 to 7, we conclude that although certain pairs of settings present significant differences (with p-value<0.05), there does not exist a determined setting for each parameter that is significantly better than all the other settings.

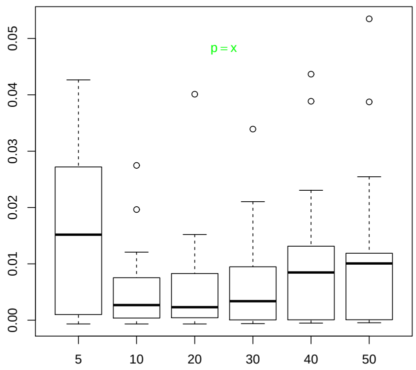
To further investigate the performance of TS/MA with different settings for each parameter, we show in Figure 1 the box and whisker plots which depict the smallest result, lower quartile, median, upper quartile, and the largest result obtained with each parameter value. For the sake of clarity, these results are displayed as the percentage deviation of the average results from the best-known results reported in the literature, computed as $\frac{BKR-Avg.}{BKR} \cdot 100\%$.

From the box and whisker plots in Figure 1, we obtain the following observations. First, setting $q \in \{10, 20, 30\}$, $\theta \in \{10, 20, 30\}$, $tt \in [15, 25]$, $\alpha \in \{6 \cdot m, 10 \cdot m\}$, $cls \in \{m^{0.4}, m^{0.5}\}$, $\beta \in \{0.6 \cdot m, 0.7 \cdot m\}$ seems preferable in terms of both the solution quality and the variation of solution values. These preferable settings for a parameter are actually obtained with the following steps: (1) the parameter setting in Table 1 is adopted because it produced the best average quality among all the settings; (2) statistical tests are conducted to compare the adopted setting with alternative ones in order to see whether different settings yield statistically different results; (3) the statistical result suggests the existence of a range of values which are acceptably good for this parameter, and in which the choice is partly arbitrary. By the above-mentioned steps, a set of good values for each parameter can be determined. A second observation is that varying values of the parameter cls, i.e., candidate list size of the swap-based neighborhood mostly affects the performance of the TS/MA algorithm, with deviations ranging from [0,0.5%] against deviations ranging from [0,0.05%] with other parameters. Finally, we observe that the performance of TS/MA is less sensitive to the population rebuilding threshold (θ) than to other parameters with deviations less than 0.03% for each setting.

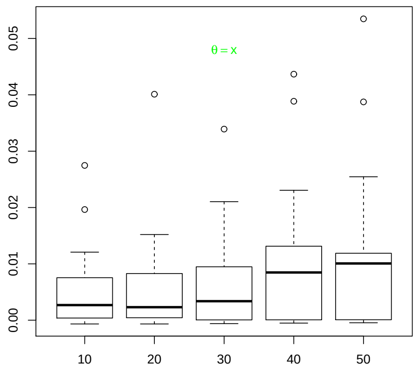
3.3.2 Tabu search analysis

In this section, we provide experiments to demonstrate the successive filter candidate list strategy implemented in our tabu search procedure, denoted as *FastBestImp*, plays an important role to the performance of the TS/MA algorithm. For this purpose, we test the following three other tabu search procedures within our TS/MA algorithm.

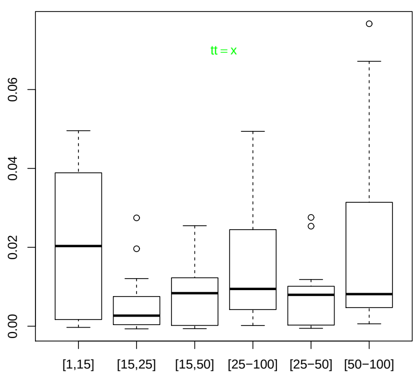
Successive 1-flip based tabu search (*1-flip*): This approach starts from an initial feasible solution x and at each iteration first picks a variable x_i from Z such that flipping x_i to the value of 1 would increase the objective function value of the current solution x by the greatest amount. Next, given the selected first flip, we pick a variable x_j from U such that flipping x_j to the value of 0



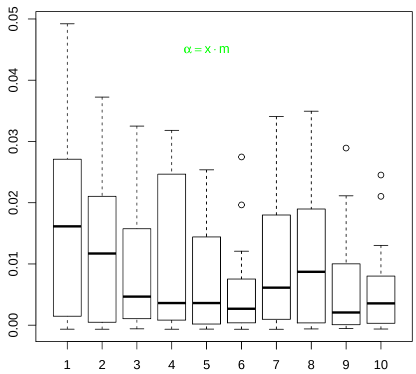
(a) varying population size q



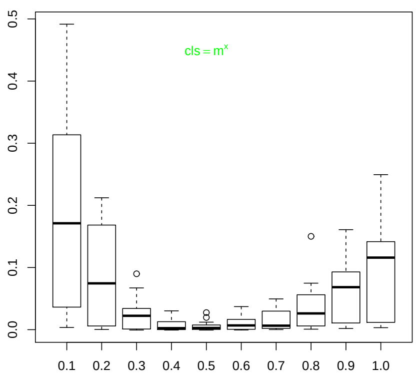
(b) varying population rebuilding threshold θ



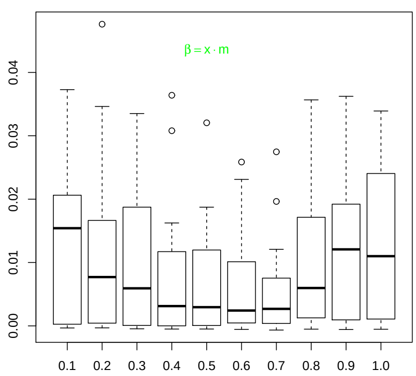
(c) varying tabu tenure tt



(d) varying tabu search improvement cutoff α



(e) varying candidate list size of each component operation cls



(f) varying number of strongly determined variables β

Fig. 1. Box and whisker plot of the results obtained with different settings for each sensitive parameter

creates the least loss in the objective function value of x. These two successive 1-flip moves assure the resulting solution is always feasible with $|U| = m$. In addition, each time a variable is flipped, a tabu tenure is assigned to the variable to prevent it from being flipped again for the next A iterations (where A is drawn randomly from the interval tt; see Table 1). Finally, a move leading to a new solution better than the best solution found so far is always selected even if it is classified tabu. The above procedure repeats until the solution cannot be improved for consecutive $m/4$ iterations. Additional details can be found in [20,29].

First Improvement based tabu search (*FirstImp*): Starting from an initial feasible solution, each iteration sequentially fetches a variable x_i from U and then scans each variable x_j from Z. If swapping x_i and x_j improves the current solution, then we perform this move to obtain a new solution. If there is no improved move by interchanging the unit-value of x_i with the zero-value of any variable from Z, we fetch the next variable from U and so on. If no improved move is found by interchanging each variable from U and each variable from Z, the best move among them (which does not improve the current solution) is then performed. The selected variables x_i and x_j become tabu active and thus neither can be involved in a new move during the next B iterations (where B is drawn randomly from tt; see Table 1). However, if a move improves the best solution found so far, it is always performed even if it is tabu active. The method continues until the best solution found so far cannot be improved for α consecutive iterations (see Table 1).

Best Improvement based tabu search (*BestImp*): The only difference between *BestImp* and our *FastBestImp* approach is that *BestImp* identifies a best neighborhood solution within the complete swap neighborhood, without employing the successive filter candidate list strategy described in Section 2.4. Several algorithms in the literature (e.g.,[5,14,25]) are based on *BestImp*.

We carry out experiments for the TS/MA algorithm with *FastBestImp* replaced by *1-flip*, *FirstImp* and *BestImp* while keeping other components unchanged. All the 3 sets of benchmarks with a total of 40 instances (see Section 3.1) are used for each TS/MA variant. The experimental results are shown in Figure 2, in which the left portion and the right portion respectively present the best gap and the average gap, for each tested instance, to the best known result.

As shown in the left portion of Figure 2, *FastBestImp* achieves the best performance with a smaller gap between the best solution value and the best known result than *1-flip*, *FirstImp* and *BestImp* for each instance, except for several Type1_22 instances where both *FastBestImp* and *1-flip* can reach the best known results. In addition, *1-flip* basically outperforms *FirstImp* and *BestImp* for the Type1_22 instances while *BestImp* outperforms *1-flip* and

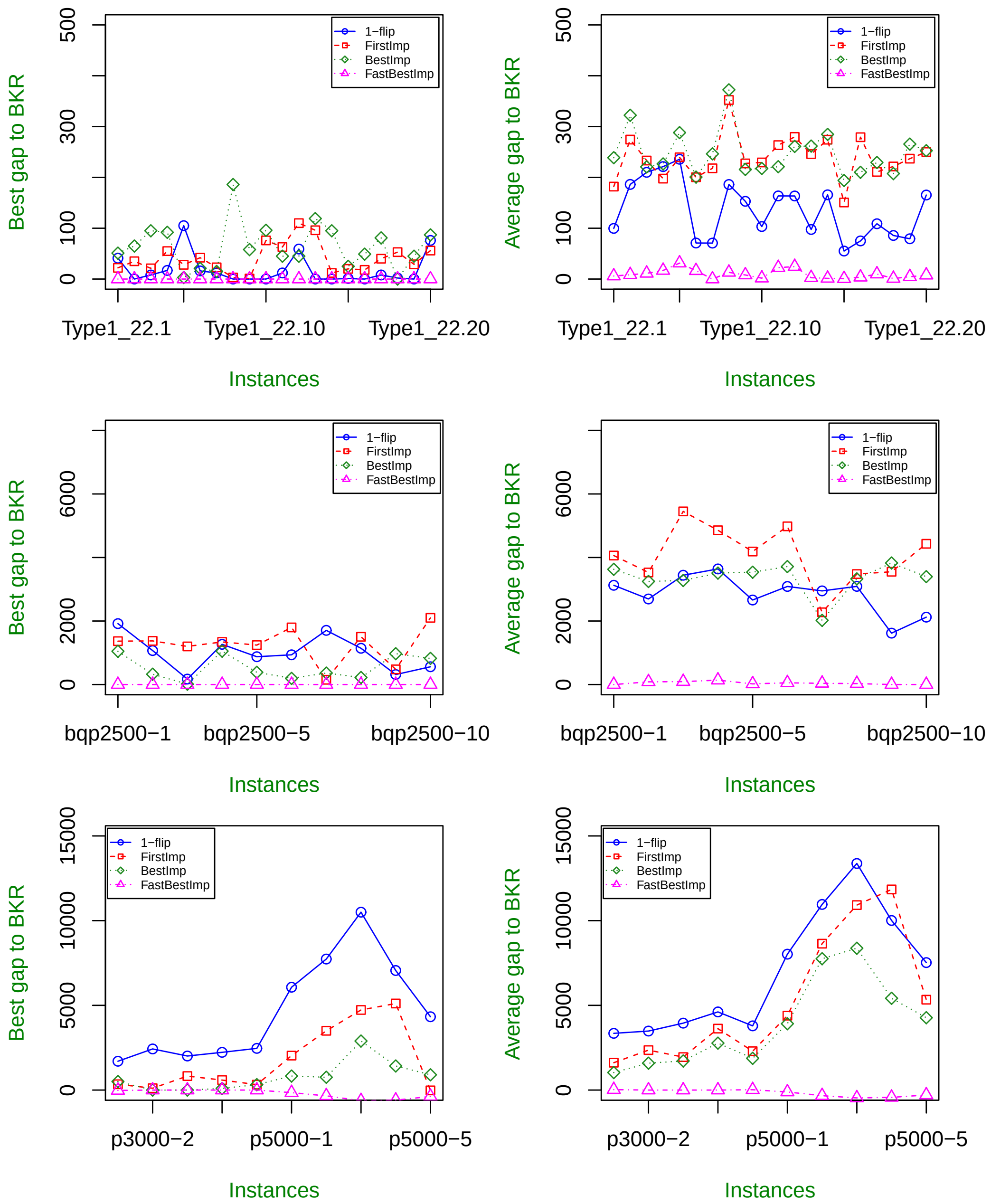


Fig. 2. Best and average solution gaps to the best known result for 3 sets of benchmark instances

FirstImp for the ORLIB and Palubeckis instances.

When it comes to the average gap to the best known result, the right portion of Figure 2 clearly shows that once again *FastBestImp* achieves the best performance among the compared strategies for all the tested instances. In addition, the comparison among *1-flip*, *FirstImp* and *BestImp* indicates that *1-flip* generally performs better for the Type1_22 and ORLIB instances while

Table 8
TS/MA$_{cx}$ versus TS/MA$_{ux}$ using Wilcoxon's test (at the 0.05 level)

Problem	R+ TS/MA$_{cx}$	R- TS/MA$_{ux}$	p-value	Diff.?	TS/MA$_{cx}$ AD-B	TS/MA$_{cx}$ AD-Av	TS/MA$_{ux}$ AD-B	TS/MA$_{ux}$ AD-Av
Type1_22	190	0	0.000143	Yes	0	9.57	0.40	27.38
ORLIB	55	0	0.001953	Yes	0	67.21	0	267.39
Palubeckis	55	0	0.001953	Yes	-212.10	-151.51	-194.50	38.48

BestImp performs better for the Palubeckis instances.

3.3.3 Solution combination operator analysis

In order to assess the role of the operator described in Section 2.5 for combining solutions, we conduct additional experiments to compare it with a traditional uniform crossover operator for combining solutions [28]. For the MDP, uniform crossover consists in identifying variables that have the value of 1 in both parents and keeping this value unchanged for these variables in the offspring solution. Then the remaining variables are randomly assigned the value 0 or 1 subject to the cardinality constraint, i.e., the total number of variables with the value of 1 equals m in the offspring solution.

We compare this modified TS/MA algorithm with the uniform crossover, denoted by TS/MA$_{ux}$, and the original TS/MA with the critical variable solution combination operator, denoted by TS/MA$_{cx}$ under the same experimental conditions (see Section 3.2). In order to detect the difference between TS/MA$_{ux}$ and TS/MA$_{cx}$, we also conduct the Wilcoxon nonparametric statistical test and summarize the results in Table 8. In this table, columns 2 to 5 report the results from the Wilcoxon test in terms of the average quality. Column AD-B reports the average gap over each set of benchmark instances of the best solution value to the best known result. Column AD-AV reports the average gap over each set of benchmark instances of the average solution values to the best known results.

The following observations can be made from Table 8. First, the results from the Wilcoxon test indicate that TS/MA$_{cx}$ is significantly better than TS/MA$_{ux}$ for each set of benchmark instances. Second, in terms of AD-B, TS/MA$_{cx}$ performs better than TS/MA$_{ux}$ for both Type1_22 (0 for TS/MA$_{cx}$ versus 0.40 for TS/MA$_{ux}$) and Palubeckis benchmarks (-212.1 for TS/MA$_{cx}$ versus -194.50 for TS/MA$_{ux}$). TS/MA$_{cx}$ performs the same as TS/MA$_{ux}$ for the ORLIB benchmark considering that both can reach the best known results for each instance. Notice that although inferior to TS/MA$_{cx}$, TS/MA$_{ux}$ is still able to improve the best known results over the Palubeckis benchmark. Finally, in terms of AD-Av, TS/MA$_{cx}$ always outperforms TS/MA$_{ux}$.

Table 9
Computational results obtained by TS/MA for Type1_22 instances

Instance	*BKR*	TS/MA					
		Best	*Succ.*	*Avg.*	σ	T_{best}	$T_{avg.}$
Type1_22.1	114271	**114271(0)**	17/30	114260.63(10.37)	15.96	11.64	11.91
Type1_22.2	114327	**114327(0)**	28/30	114318.20(8.80)	32.93	8.89	9.25
Type1_22.3	114195	**114195(0)**	16/30	114186.47(8.53)	13.14	9.34	9.84
Type1_22.4	114093	**114093(0)**	3/30	114073.10(19.90)	17.91	12.68	10.39
Type1_22.5	114196	**114196(0)**	7/30	114166.50(29.50)	33.24	12.48	11.92
Type1_22.6	114265	**114265(0)**	9/30	114249.40(15.60)	12.08	9.81	10.83
Type1_22.7	114361	**114361(0)**	30/30	114361.00(0.00)	0.00	7.16	7.16
Type1_22.8	114327	**114327(0)**	21/30	114301.77(25.23)	39.69	6.88	6.76
Type1_22.9	114199	**114199(0)**	8/30	114191.17(7.83)	11.03	9.07	10.04
Type1_22.10	114229	**114229(0)**	21/30	114224.90(4.10)	12.08	10.16	9.71
Type1_22.11	114214	**114214(0)**	8/30	114189.70(24.30)	18.10	11.85	11.56
Type1_22.12	114214	**114214(0)**	6/30	114192.50(21.50)	18.23	10.10	10.31
Type1_22.13	114233	**114233(0)**	28/30	114231.77(1.23)	5.94	10.37	10.39
Type1_22.14	114216	**114216(0)**	28/30	114212.43(3.57)	19.02	7.70	8.05
Type1_22.15	114240	**114240(0)**	6/30	114238.27(1.73)	2.02	9.72	10.35
Type1_22.16	114335	**114335(0)**	17/30	114327.73(7.27)	10.51	7.64	9.65
Type1_22.17	114255	**114255(0)**	13/30	114243.27(11.73)	12.18	8.69	10.01
Type1_22.18	114408	**114408(0)**	15/30	114407.00(1.00)	1.00	4.41	6.13
Type1_22.19	114201	**114201(0)**	24/30	114197.00(4.00)	8.00	7.10	6.81
Type1_22.20	114349	**114349(0)**	21/30	114333.40(15.60)	28.49	8.76	9.66
Av.		**(0)**	15.3/30	(11.09)	15.58	9.22	9.54

Table 10
Computational results obtained by TS/MA for ORLIB instances

Instance	*BKR*	TS/MA					
		Best	*Succ.*	*Avg.*	σ	T_{best}	$T_{avg.}$
b2500-1	1153068	**1153068(0)**	30/30	1153068.00(0.00)	0.00	66.50	66.50
b2500-2	1129310	**1129310(0)**	25/30	1129236.13(73.87)	179.60	109.20	114.68
b2500-3	1115538	**1115538(0)**	22/30	1115353.27(184.73)	306.35	94.70	104.00
b2500-4	1147840	**1147840(0)**	15/30	1147681.00(159.00)	159.11	79.10	87.30
b2500-5	1144756	**1144756(0)**	22/30	1144710.80(45.20)	76.58	51.92	51.00
b2500-6	1133572	**1133572(0)**	24/30	1133517.60(54.40)	108.80	78.90	81.39
b2500-7	1149064	**1149064(0)**	17/30	1148999.00(65.00)	74.33	109.29	89.10
b2500-8	1142762	**1142762(0)**	21/30	1142760.80(1.20)	1.83	96.74	95.28
b2500-9	1138866	**1138866(0)**	30/30	1138866.00(0.00)	0.00	80.08	80.08
b2500-10	1153936	**1153936(0)**	30/30	1153936.00(0.00)	0.00	98.04	98.04
Av.		**(0)**	23.6/30	(58.34)	90.66	86.45	86.74

Table 11
Computational results obtained by TS/MA for Palubeckis instances

Instance	*BKR*	TS/MA					
		Best	*Succ.*	*Avg.*	σ	T_{best}	$T_{avg.}$
p3000-1	6502308	***6502330(-22)***	5/15	6502272.93(35.07)	41.86	243.52	301.49
p3000-2	18272568	**18272568(0)**	15/15	18272568.00(0.00)	0.00	172.12	172.12
p3000-3	29867138	**29867138(0)**	15/15	29867138.00(0.00)	0.00	73.72	73.72
p3000-4	46915044	**46915044(0)**	14/15	46915042.80(1.20)	4.49	289.64	302.48
p3000-5	58095467	**58095467(0)**	13/15	58095464.73(2.27)	5.78	123.13	132.00
p5000-1	17509215	***17509369(-154)***	12/15	17509336.60(-121.60)	95.56	945.58	984.86
p5000-2	50102729	***50103071(-342)***	4/15	50103044.40(-315.40)	23.13	730.26	993.07
p5000-3	82039686	***82040316(-630)***	2/15	82040144.67(-458.67)	69.32	1079.56	965.99
p5000-4	129413112	***129413710(-598)***	5/15	129413511.87(-399.87)	151.40	1063.31	1055.73
p5000-5	160597781	***160598156(-375)***	2/15	160598016.87(-235.87)	82.06	792.20	771.47
Av.		***(-212.1)***	8.7/15	(-149.29)	47.36	551.30	575.29

3.4 Computational results for TS/MA

Tables 9, 10 and 11 respectively show the computational statistics of the TS/MA algorithm on the 20 Type1_22 instances, 10 ORLIB instances and

10 Palubeckis instances. In each table, columns 1 and 2 give the instance names (*Instance*) and the best known results (BKR) reported in the literature [6,19,25,29]. As indicated in [29] (see its tables 4-6 and 11), some of these best known results are obtained with a relaxed time limit, around 20 to 60 times longer than in a typical setting. The columns under the heading TS/MA report the best solution values (*Best*) along with the gap of *Best* to BKR shown in parenthesis (BKR-*Best*), the success rate (*Succ.*) for reaching *Best*, the average solution values (*Avg.*) along with the gap of *Avg.* to BKR shown in parenthesis (BKR-*Avg.*), the standard deviation (σ), the average time (T_{best}) required over the runs which actually reach the value *Best* and the average time ($T_{avg.}$) required to reach the best solution value found in each run (in seconds). To calculate T_{best} and $T_{avg.}$, we use a pair of elements (f_i, t_i) to record the best solution value obtained in the i_{th} run and the time needed to reach this value. Then we sort the pairs obtained over all N runs according to their solution values, say $f_1 \geq \ldots \geq f_u \geq Best > f_v \ldots \geq f_N$. Finally, we set $T_{best} = \sum_{i=1}^{u} t_i/u$ and $T_{avg.} = \sum_{i=1}^{N} t_i/N$. Results marked in bold indicate that TS/MA matches BKR and if also marked in italic indicate that TS/MA improves BKR. Furthermore, the last row *Av.* summarizes TS/MA's average performance over the whole set of benchmark instances. Notice that the reason we show both T_{best} and $T_{avg.}$ lies in the fact that the proposed algorithm does not necessarily lead to the same best value in each run because of its stochastic nature. Only if all the runs for solving a specific instance reach the BKR, the $T_{avg.}$ and T_{best} will be completely the same.

From Tables 9, 10 and 11, we observe that TS/MA can easily reach the best known results for all the tested instances within the given time limit, which none of current state-of-the-art algorithms can compete with. In particular, TS/MA improves the best known results for 6 Palubeckis instances and even its average quality is better than the best known results previously reported in the literature. Finally, we mention that for these 6 Palubeckis instances, similar improved best known results were reported very recently and independently in [31].

3.5 Comparison with state-of-the-art algorithms

In order to further evaluate our TS/MA algorithm, we compare it with four best performing algorithms recently proposed in the literature. These reference methods are Iterated Tabu Search (ITS) [25], Variable Neighborhood Search (VNS) [6], Tuned Iterated Greedy (TIG) [19] and Learnable Tabu Search with Estimation of Distribution Algorithm (LTS-EDA) [29]. The results of these reference algorithms are directly extracted from [29]. Notice that the BKR values are the best values compiled from Tables 4-6 and 11 of [29] which were obtained within the typical and longer time limit. This study is carried out

Table 12
Comparison among TS/MA and other state-of-the-art algorithms for Type1_22 instances

Instance	*BKR*	ITS[2007]		VNS[2009]		TIG[2011]		LTS-EDA[2012]		TS/MA	
		Best	*Avg.*	*Best*	*Avg.*	*Best*	*Avg.*	*Best*	*Avg.*	*Best*	*Avg.*
Type1_22.1	114271	65	209.87	48	150.60	48	101.57	5	60.73	**0**	**10.37**
Type1_22.2	114327	29	262.27	**0**	168.87	**0**	69.90	**0**	89.87	**0**	**8.80**
Type1_22.3	114195	69	201.40	19	110.83	5	117.77	**0**	98.97	**0**	**8.53**
Type1_22.4	114093	22	200.53	70	188.13	58	141.93	**0**	79.87	**0**	**19.90**
Type1_22.5	114196	95	273.27	87	184.10	99	194.70	51	134.47	**0**	**29.50**
Type1_22.6	114265	41	168.17	30	99.30	9	96.20	**0**	40.17	**0**	**15.60**
Type1_22.7	114361	12	167.47	**0**	56.30	**0**	71.27	**0**	18.20	**0**	**0.00**
Type1_22.8	114327	25	256.40	**0**	163.33	**0**	193.60	**0**	159.10	**0**	**25.23**
Type1_22.9	114199	9	139.83	16	78.47	16	80.37	**0**	70.97	**0**	**7.83**
Type1_22.10	114229	24	204.93	7	139.33	35	121.43	**0**	56.20	**0**	**4.10**
Type1_22.11	114214	74	237.77	42	145.13	59	139.57	3	69.87	**0**	**24.30**
Type1_22.12	114214	55	249.53	95	143.30	88	156.00	15	84.93	**0**	**21.50**
Type1_22.13	114233	93	279.87	22	168.07	42	167.40	6	85.30	**0**	**1.23**
Type1_22.14	114216	92	248.50	117	194.30	64	202.80	**0**	81.00	**0**	**3.57**
Type1_22.15	114240	11	117.50	1	62.87	6	80.53	**0**	22.03	**0**	**1.73**
Type1_22.16	114335	11	225.40	42	215.43	35	67.90	**0**	36.47	**0**	**7.27**
Type1_22.17	114255	56	217.53	**0**	170.00	18	144.53	6	57.07	**0**	**11.73**
Type1_22.18	114408	46	169.97	**0**	57.10	2	117.37	2	22.83	**0**	**1.00**
Type1_22.19	114201	34	243.20	**0**	124.60	**0**	144.37	**0**	35.87	**0**	**4.00**
Type1_22.20	114349	151	270.67	65	159.43	45	187.23	**0**	95.40	**0**	**15.60**
Av.		50.7	217.20	33.05	138.97	31.45	129.82	4.40	69.97	**0**	**11.09**

Table 13
Comparison among TS/MA and other state-of-the-art algorithms for ORLIB instances

Instance	*BKR*	ITS[2007]		VNS[2009]		TIG[2011]		LTS-EDA[2012]		TS/MA	
		Best	*Avg.*	*Best*	*Avg.*	*Best*	*Avg.*	*Best*	*Avg.*	*Best*	*Avg.*
b2500-1	1153068	624	3677.33	96	1911.93	42	1960.33	**0**	369.20	**0**	**0.00**
b2500-2	1129310	128	3677.33	88	1034.33	1096	1958.47	154	453.53	**0**	**73.87**
b2500-3	1115538	316	3281.93	332	1503.67	34	2647.87	**0**	290.40	**0**	**184.73**
b2500-4	1147840	870	2547.93	436	1521.07	910	1937.13	**0**	461.73	**0**	**159.00**
b2500-5	1144756	356	1800.27	**0**	749.40	674	1655.87	**0**	286.07	**0**	**45.20**
b2500-6	1133572	250	2173.47	**0**	1283.53	964	1807.60	80	218.00	**0**	**54.40**
b2500-7	1149064	306	1512.60	116	775.47	76	1338.73	44	264.60	**0**	**65.00**
b2500-8	1142762	**0**	247.73	96	862.47	588	1421.53	22	146.47	**0**	**1.20**
b2500-9	1138866	642	2944.67	54	837.07	658	1020.60	6	206.33	**0**	**0.00**
b2500-10	1153936	598	2024.60	278	1069.40	448	1808.73	94	305.27	**0**	**0.00**
Av.		409	2388.79	149.6	1154.83	549	1755.69	40	300.16	**0**	**58.34**

Table 14
Comparison among TS/MA and other state-of-the-art algorithms for Palubeckis instances

Instance	*BKR*	ITS[2007]		VNS[2009]		TIG[2011]		LTS-EDA[2012]		TS/MA	
		Best	*Avg.*	*Best*	*Avg.*	*Best*	*Avg.*	*Best*	*Avg.*	*Best*	*Avg.*
p3000-1	6502308	466	1487.53	273	909.80	136	714.67	96	294.07	**-22**	**35.07**
p3000-2	18272568	**0**	1321.60	**0**	924.20	**0**	991.07	140	387.00	**0**	**0.00**
p3000-3	29867138	1442	2214.73	328	963.53	820	1166.13	**0**	304.33	**0**	**0.00**
p3000-4	46915044	1311	2243.93	254	1068.47	426	2482.20	130	317.07	**0**	**1.20**
p3000-5	58095467	423	1521.60	**0**	663.00	278	1353.27	**0**	370.40	**0**	**2.27**
p5000-1	17509215	2200	3564.93	1002	1971.27	1154	2545.80	191	571.00	**-154**	**-121.60**
p5000-2	50102729	2910	4786.80	1478	2619.00	528	2511.73	526	892.80	**-342**	**-315.40**
p5000-3	82039686	5452	8242.33	1914	3694.40	2156	6007.13	704	1458.53	**-630**	**-458.67**
p5000-4	129413112	1630	5076.90	1513	2965.90	1696	3874.80	858	1275.20	**-598**	**-399.87**
p5000-5	160597781	2057	4433.90	1191	2278.30	1289	2128.90	579	1017.90	**-375**	**-235.87**
Av.		1789.1	3489.43	795.3	1805.79	848.3	2377.57	322.4	688.83	**-212.1**	**-149.29**

Table 15
TS/MA versus ITS, VNS, TIG and LTS-EDA (Wilcoxon's test at the 0.05 level)

	Type1_22		ORLIB		Palubeckis	
	p-value	Diff.?	*p*-value	Diff.?	*p*-value	Diff.?
ITS	1.91e-06	Yes	0.002	Yes	0.002	Yes
VNS	1.91e-06	Yes	0.002	Yes	0.002	Yes
TIG	1.91e-06	Yes	0.002	Yes	0.002	Yes
LTS-EDA	1.91e-06	Yes	0.002	Yes	0.002	Yes

under the same time condition as that used in [29] (see Section 3.2).

Tables 12, 13 and 14 display the best and average solution values obtained by ITS, VNS, TIG, LTS-EDA and our TS/MA algorithm. Since the absolute solution values are very large, we report the gap of best and average solution values to the best known results. Smaller gaps indicate better performances. Negative gaps represent improved results. The best performances among the 5 compared algorithms are highlighted in bold. In addition, the averaged results over the whole set of instances are presented in the last row.

As we can observe from Tables 12, 13 and 14, our TS/MA algorithm outperforms the four reference algorithms in terms of both the best and average solution values. Specifically, TS/MA is able to match or surpass the best known results for all the 40 instances, while ITS, VNS TIG and LTS-EDA can only match 2, 10, 5 and 19 out of 40 instances, respectively. Furthermore, the average gap to the best known results of TS/MA is much smaller than that of each reference algorithm.

We also conduct nonparametric statistical tests to verify the observed differences between TS/MA and the reference algorithms in terms of solution quality are statistically significant. Table 15 summarizes the results by means of the Wilcoxon signed-ranked test, where p-value<0.05 indicates that there is significant difference between our TS/MA algorithm and a reference algorithm. We observe that TS/MA is significantly better than all these reference algorithms for each set of benchmark.

In sum, this comparison demonstrates the efficacy of our TS/MA algorithm in attaining the best and average solution values.

4 Conclusion

In this paper, we have proposed an effective memetic algorithm for the maximum diversity problem based on tabu search. The tabu search component utilizes successive filter candidate list strategy and is joined with a solution combination strategy based on identifying strongly determined and consistent variables.

Computational experiments on three sets of 40 popular benchmark instances have demonstrated that the proposed TS/MA algorithm is capable of easily attaining all the previous best known results and improving the best known results for 6 instances. Moreover, statistical tests have confirmed that our proposed algorithm performs significantly better than several recently proposed state-of-the-art algorithms.

In addition to a parameter sensitivity analysis, we have studied the effects of the dedicated tabu search procedure based on the swap move combined with the successive filter candidate list strategy and the specific combination operator based on the concept of strongly determined and consistent variables. These studies have confirmed the importance of these two key components for the high performance of the proposed algorithm.

Finally, even if some best-known results could further be improved for the tested benchmark instances, unfortunately it is unknown how far these results are away from the optimal solutions given that these instances are too large to be solved to optimality by the existing exact methods. An interesting issue would be to devise methods which are able to deliver tight upper bounds. Another research direction is to characterize the hardness of the existing instances and design a parameterable model for generating new benchmarks whose difficulty could be controlled.

Acknowledgment

We are grateful to the reviewers whose comments have helped to improve the quality of the paper. The work is partially supported by the Pays de la Loire Region (France) within the RaDaPop (2009-2013) and LigeRO (2010-2013) projects.

References

[1] Andrade PMF, Plastino A, Ochi LS, Martins SL (2003) GRASP for the maximum diversity problem. In proceedings of the Fifth Metaheuristics International Conference (MIC2003), Paper 15

[2] Andrade MRQ, Andrade PMF, Martins SL, Plastino A (2005) Grasp with path-relinking for the maximum diversity problem. In: Nikoletseas SE (Ed.), Experimental and Efficient Algorithms, LNCS 3503: 558–569, Springer Berlin

[3] Aringhieri R, Bruglieri M, Cordone R (2009) Optimal results and tight bounds for the maximum diversity problem. Foundations of Computing and Decision Sciences 34(2):73–85

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

[5] Aringhieri R, Cordone R (2011) Comparing local search metaheuristics for the maximum diversity problem. Journal of the Operational Research Society 62(2):266–280

[6] Brimberg J, Mladenović N, Urošević D, Ngai E (2009) Variable neighborhood search for the heaviest-subgraph. Computers & Operations Research 36(11):2885–2891

[7] Duarte A, Marti R (2007) Tabu search and GRASP for the maximum diversity problem. European Journal of Operational Research 178:71–84

[8] Gallego M, Duarte A, Laguna M, Martí R (2009) Hybrid heuristics for the maximum diversity problem. Computational Optimization and Applications 200(1):36–44

[9] Glover F (1977) Heuristics for integer programming using surrogate constraints. Decision Sciences 8(1):156–166

[10] Glover F (1994) Tabu search for nonlinear and parametric optimization (with links to genetic algorithms). Discrete Applied Mathematics 49:231–255

[11] Glover F (1997) A Template for scatter search and path relinking. In Hao JK, Lutton E, Ronald E, Schoenauer M and Snyers D (Eds.), Artificial Evolution, Lecture Notes in Computer Science, vol.1363, Springer, pp.13–54.

[12] Glover F, Laguna M (1997) Tabu search. Kluwer Academic Publishers

[13] Hao JK (2012) Memetic algorithms in discrete optimization. In Neri F, Cotta C, Moscato P (Eds.) Handbook of Memetic Algorithms. Studies in Computational Intelligence 379, Chapter 6, pp.73–94

[14] Ghosh JB (1996) Computational aspects of the maximum diversity problem. Operation research letters 19:175–181

[15] Jones T, Forrest S (1995) Fitness distance correlation as a measure of problem difficulty for genetic algorithms. Proceedings of the 6th International Conference on Genetic Algorithms, Morgan Kaufmann, pp.184–192

[16] Katayama K, Narihisa H (2005) An evolutionary approach for the maximum diversity problem. In: Hart W, Krasnogor N, Smith JE (Eds.), Recent Advances in Memetic Algorithms. Studies in Fuzziness and Soft Computing 166:31–47

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

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

[19] Lozano M, Molina D, García-Martínez C (2011) Iterated greedy for the maximum diversity problem. European Journal of Operational Research 214(1):31–38

[20] Lü Z, Glover F, Hao JK (2010) A Hybrid Metaheuristic Approach to Solving the UBQP Problem. European Journal of Operational Research 207(3):1254–1262

[21] Lü Z, Glover F, Hao JK (2013) Neighborhood combination for unconstrained binary quadratic problems. In M. Caserta and S. Voss (Eds.): MIC-2009 Post-Conference Book, Chapter 4, pp.49-61 (to appear)

[22] Martí R, Gallego M, Duarte A (2010) A branch and bound algorithm for the maximum diversity problem. European Journal of Operational Research 200(1):36–44

[23] Martí R, Gallego M, Duarte A, Pardo EG (2013) Heuristics and metaheuristics for the maximum diversity problem. Journal of Heuristics. 19(4):591–615

[24] Neri F, Cotta C, Moscato P (Eds.) (2011) Handbook of Memetic Algorithms. Studies in Computational Intelligence 379, Springer.

[25] Palubeckis G(2007) Iterated tabu search for the maximum diversity problem. Applied Mathematics and Computation 189(1):371–383

[26] Silva GC, Ochi LS, Martins SL (2004) Experimental comparison of greedy randomized adaptive search procedures for the maximum diversity problem. In: C. Ribeiro C, Martins C, Simone L (Eds.), Experimental and Efficient Algorithms, Lecture Notes in Computer Science, Volume 3059, Springer, Berlin, pp.498–512

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

[28] Syswerda G (1989) Uniform crossover in genetic algorithms. Proceedings of the 3rd International Conference on Genetic Algorithms, Morgan Kaufmann Publishers Inc. San Francisco, CA, pp.2–9

[29] 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

[30] Wang Y, Lü Z, Glover F, Hao JK (2012) Path relinking for unconstrained binary quadratic programming. European Journal of Operational Research 223(3):595–604

[31] Wu Q, Hao JK (2013) A hybrid metaheuristic method for the maximum diversity problem. European Journal of Operational Research 231(2): 452-464