European Journal of Operational Research 214 (2011) 31–38

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

Iterated greedy for the maximum diversity problem

M. Lozano [a,*], D. Molina [b], C. García-Martínez [c,1]

[a] Department of Computer Science and Artificial Intelligence, University of Granada, Granada 18071, Spain
[b] Department of Computer Engineering, University of Cádiz, Cádiz 1103, Spain
[c] Department of Computing and Numerical Analysis, University of Córdoba, Córdoba 14071, Spain

ARTICLE INFO

Article history:
Received 24 April 2009
Accepted 20 April 2011
Available online 30 April 2011

Keywords:
Maximum diversity problem
Iterated greedy
Local search
Acceptance criterion

ABSTRACT

The problem of choosing a subset of elements with maximum diversity from a given set is known as the maximum diversity problem. Many algorithms and methods have been proposed for this hard combinatorial problem, including several highly sophisticated procedures. By contrast, in this paper we present a simple iterated greedy metaheuristic that generates a sequence of solutions by iterating over a greedy construction heuristic using destruction and construction phases. Extensive computational experiments reveal that the proposed algorithm is highly effective as compared to the best-so-far metaheuristics for the problem under consideration.

1. Introduction

The *maximum diversity problem* (MDP) consists of determining a subset M of a given cardinality from a set N of elements, in such a way that the sum of the pair-wise distances between the elements of M is the maximum possible. The definition of distance between elements is customized to specific applications. The MDP arises in various contexts, including the location of undesirable or mutually competing facilities, aiding decision analysis with multiple objectives, composing jury panels, genetic engineering, ecological, medical and social sciences, animal and plant genetics, and ethnicity (see [9,18,27,28] for more details on these and some other applications).

More precisely, let $N = \{e_1, \ldots, e_n\}$ be a set of elements and $d(e_i, e_j)$ be the distance between elements e_i and e_j ($d(e_i, e_j)$ will be denoted by d_{ij} as well). Then, the MDP consists in determining a subset $M \subset N$ of given cardinality m ($m < n$), so that the sum of the distances between the elements in M is the maximum possible. The problem may be formulated as the following quadratic zero-one integer program [23]:

$$\text{Maximize } z = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij} \cdot x_i \cdot x_j,$$

$$\text{Subject to } \sum_{i=1}^{n} x_i = m, \quad x_i \in \{0,1\}, \quad i = 1, \ldots, n,$$

where x_i is a binary variable indicating whether an element e_i is selected to be a member of the subset M.

Glover et al. [17] showed that this problem is NP-hard. Although there are exact procedures based on the *branch and bound* method [27] and mathematical programming approaches [30] to provide solutions for this problem, they are unable to solve large MDP instances. Therefore, different metaheuristics were proposed to obtain high quality solutions for this kind of instances. They include memetic algorithms [21], scatter search [13,19], tabu search [4,5,28], estimation of distribution algorithms [39], genetic algorithms [11], and variable neighborhood search [7]. Another approach that has received considerable attention in the solution of the MDP is *greedy randomized adaptive search procedure* (GRASP) [2,3,9,15,36,37].

Although many algorithms and methods have been presented for this hard combinatorial problem, iterated greedy (IG) algorithms have not been applied to it previously to the best of our knowledge. IG [20] is a very simple and effective metaheuristic recently developed for combinatorial optimization problems that follows a very simple principle, is easy to implement and can show excellent performance; in fact, it has exhibited state-of-the-art performances for a considerable number of problems [8,10,12,20,25,38,32,33,42]. IG generates a sequence of solutions by iterating over greedy constructive heuristics using two main phases: *destruction* and *construction*. During the destruction phase, some solution components are removed from a previously constructed complete candidate solution. The construction procedure then applies a greedy constructive heuristic to reconstruct a complete candidate solution. Once a newly re-constructed solution has been obtained, an acceptance criterion is applied to decide whether it will replace the incumbent solution or not. Besides, an optional

* Corresponding author. Tel.: +34 958244258.
E-mail addresses: lozano@decsai.ugr.es (M. Lozano), daniel.molina@uca.es (D. Molina), cgarcia@uco.es (C. García-Martínez).

[1] This work was supported by the Research Projects TIN2008-05854 and P08-TIC-4173.

0377-2217/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2011.04.018

local search phase for improving the initial solution and the re-constructed solution can be added before the main loop and the acceptance test, respectively.

In this paper, we explore the behavior of the IG metaheuristic for the MDP. With the aim of providing a very competitive IG instance, we propose an IG model for this problem and present an extensive computational comparison of different instances that are distinguished by (1) the number of solution components that are removed from the current solution, (2) the local search method, and (3) the acceptance criterion.

The remainder of this paper is structured as follows. In Section 2, we give an overview of the existing research on metaheuristics for the MDP. In Section 3, we describe the proposed IG. In Section 4, we present empirical studies, which are designed to: (1) to analyze the influence of the parameters and settings associated with our proposal, (2) to obtain a tuned IG being able to show a robust operation for test problems with different characteristics, and (3) to compare its results with those of other approaches from the literature. Finally, in Section 5, we discuss conclusions and further work.

2. Metaheuristics for the MDP

Metaheuristics have shown to be very successful in finding high quality solutions to the MDP. Early implementations of simulated annealing and tabu search (TS) were made by Kincaid [22]. Computational results, considering three sets of problem instances of size 25, showed that TS performed somewhat better than simulated annealing. Another TS model, called multiple TS, was designed by Macambira [26] for MDP instances with up to 100 elements. A different implementation of TS for small MDP instances (of a size under 50) was suggested by Alidaee et al. [1]. Their method is centered on the use of *strategic oscillation*, which directly or indirectly underlies many TS implementations [16]. This method alternates between constructive phases that progressively set variables to 1 and destructive phases that increasingly set variables to 0.

In the recent past, there has been an increasing interest in solving the MDP using the GRASP approach, and, as a result, many algorithms have been designed by combining different construction and local search heuristics. Among the earliest approaches we find the GRASP of Ghosh [15]. The solution construction phase performs m iterations to obtain a solution. In each iteration one element is selected according to an *estimation* of its contribution to the final diversity. The improvement phase implements a straightforward hill-climbing heuristic based on performing the best available exchange (replacing an element in the solution with another that does not belong to the solution). The algorithm was tested on small problem instances with up to 40 elements.

Andrade et al. [3] designed a GRASP whose solution construction phase limits the choice of the new element to a restricted candidate list. It orders the elements with respect to a suitable greedy function, selects the best m ones and removes those whose value presents a difference larger than the average from the following element in the given order. The improvement phase is the same as in the previous algorithm. Andrade et al. showed results for instances randomly created with up to 250 elements. In particular, the algorithm was able to find some solutions better than the ones found by the algorithm by Ghosh. Later, Andrade et al. [2] proposed a variant of GRASP by adding a path relinking post-processing, which allowed better results to be achieved.

Silva et al. [36] implemented three GRASP constructions: KLD, KLDv2, and MDI, and two local search procedures: GhA and SOMA, where GhA is the method proposed by Ghosh [15]. They were extensively tested over a benchmark set of randomly generated instances with up to 500 elements. The computational experiments showed that: (1) when running time is very short, KLD coupled with GhA produces the best quality solutions, (2) when longer running times are admissible, KLDv2 coupled with GhA is the best method, and (3) GRASP instances that use KLD or KLD-v2 construction algorithms and GhA or SOMA local search algorithms significantly improve the average performance of the algorithms by Ghosh [15] and Andrade et al. [3]. Nonetheless, as shown in their experimentation and confirmed in [9], all these methods present extremely long running times when solving medium-size instances. In a later work, Silva et al. [37] presented new construction procedures for the MDP that include existing enhanced features proposed for GRASP: *filtering* of constructed solutions [31] and *reactive* GRASP [29]. In addition, they introduced a path-relinking technique to their GRASP heuristics.

Santos et al. [35] hybridize GRASP with data mining methods. Simply put, after GRASP executes a significant number of iterations, the data mining process extracts patterns from an elite set of solutions that guide the following GRASP iterations. The GRASP used to develop the hybrid strategy is based on the KLD strategy presented in [36].

Duarte and Martí [9] developed four constructive methods based on TS that incorporate memory structures, and four memory-less constructions based on GRASP methodology. They were combined with two new improvement methods: an efficient local search (I_LS), which is a variant of the method of Ghosh, and a short-term TS improvement phase. Overall, experiments with 120 MDP instances (with up to 2000 elements) clearly proved the effectiveness of the use of memory in both construction and improvement phases for solving this problem.

Aringhieri et al. [4] proposed a TS-based GRASP algorithm in which TS is initialized by a trivial constructive procedure, but it adds the tabu mechanism and suitable intensification and diversification devices to enhance the search in the improvement phase. Their simulation results (with instances with up to 500 elements) showed that the TS-based GRASP achieved both better results and much shorter computational time than to those reported for previous GRASP algorithms.

Hybrid evolutionary algorithms have also been a tool of choice for dealing with the MDP. In [21], Katayama et al. presented a memetic algorithm (an evolutionary algorithm incorporating local search) that applies a k-flip local search method based on variable depth search. To investigate the effectiveness of this approach, the authors carried out experiments with problem instances of up to n=2500, and observed that the algorithm is particularly effective for large instances. However, they did not present any comparison with other approaches in the literature. Gallego et al. [13] studied several hybridized procedures within the scatter search framework. The authors found that their algorithm obtained better solutions than the ones achieved by KLD and KLDv2 [36] and Tabu_D2 + LS_TS [9], when tackling MDP instances with $n = 500$.[2]

One of the most appealing contemporary metaheuristic approaches for the MDP is iterated TS (ITS), which was proposed by Palubeckis [28]. This algorithm has two phases: solution perturbation and TS phases. Computational results for problem instances involving up to 5000 elements indicated that its performance surpasses that of other algorithms, such as Tabu_D2 + LS_TS [9], by an impressive margin.

Recently, Brimberg et al. [7] implemented a simple variable neighborhood search approach for this problem that they found to be superior to other metaheuristics, including ITS [28], scatter search [13], KLD and KLD-v2 [36], and Tabu_D2 + LS_TS [9] in solving the set of MDP test instances used by Palubeckis.

[2] The results of a wider comparison between this algorithm and other metaheuristics may be found at the following web site: http://heur.uv.es/optsicom/mdp/.

Finally, Wang et al. [39] developed a discrete competitive Hopfield neural network based on the estimation of distribution algorithms. The algorithm was tested on 120 benchmark problems with the size ranging from 100 to 5000. Simulation results evidenced that the proposal was better than or competitive with other metaheuristic algorithms such as ITS [28], the TS-based GRASP algorithm [4], Tabu_D2 + LS_TS [9], and KLD combined with path-relinking [37].

3. IG for the MDP

In this section, we explore the adaptation of the IG framework to obtain high quality solutions to the MDP. Firstly, in Section 3.1, we describe the greedy constructive and destructive algorithms that play a fundamental role in our IG. Then, in Section 3.2, we provide a general overview of the overall algorithm. Finally, in Section 3.3, we provide details for the improvement methods proposed in the literature for the MDP, which will be employed in the local search phase of our IG.

3.1. Greedy constructive and destructive algorithms

In this section, we describe the greedy constructive and destructive algorithms, C2 and D2, proposed by Glover et al. [18], which may be used to build feasible solutions for the MDP (i.e., a solution containing m elements). In order to do this, we firstly define the *contribution* of an element $e_i \in N$ to a set $S \subseteq N$ as:

$$D(e_i, S) = \sum_{e_j \in S} d(e_i, e_j), \quad i = 1, \ldots, n.$$

The C2 algorithm selects one element at a time and adds it to the current partial solution, S_p, until m elements are included. Specifically, C2 adds, at each step, the element in $N \backslash S_p$ with the maximum contribution value across all elements, D_{max}. By contrast, starting with all the elements selected ($S_p = N$), the D2 algorithm removes, at each step, the component of S_p with the minimum contribution value, D_{min}. In both cases, we can easily update $D(e_i, S_p)$ for each e_i, $i = 1, \ldots, n$ by adding the value D_{max} to it or subtracting the D_{min} value from it, respectively. We should point out that given a feasible solution S, the objective function value z can be obtained from the D values with the expression:

$$z = \frac{1}{2} \sum_{e_i \in S} D(e_i, S).$$

In the following, we denote by `Greedy-Add` (S, n) the greedy construction procedure that introduces n elements in a set S, by performing n iterations of the C2 algorithm.

3.2. General scheme of the proposed IG

An outline of our proposal, called IG-MDP, is depicted in Fig. 1. IG-MDP starts from some complete initial solution S (Steps 1 and 2) and then iterates through a main loop in which first a partial candidate solution S_d is obtained by removing a fixed number of solution components from a complete candidate solution S (destruction phase, Step 5) and next a complete solution S_c is reconstructed starting with S_d (construction phase, Step 6). A local search phase is added for improving the re-constructed solution (Step 7). Before continuing with the next loop, an acceptance criterion decides whether the solution returned by the local search procedure, S_{ls}, becomes the new incumbent solution (Step 10). The process iterates through these two phases until some termination conditions (e.g. maximum number of iterations, or maximum computation time allowed) have been met. The best solution, S_b, generated during the iterative process is kept as the overall result.

```
Procedure IG-MDP
1:   S' ← D2 ()
2:   S ← Local-Search (S')
3:   S_b ← S
4:   repeat
5:     S_d ← Random-Drop (S, n_d)
6:     S_c ← Greedy-Add (S_d, n_d)
7:     S_ls ← Local-Search (S_c)
8:     if (S_ls is better than S_b) then
9:       S_b ← S_ls
10:    S ← Acceptance-Criterion (S, S_ls)
11:  until (termination condition met)
12:  return S_b
```

Fig. 1. Pseudocode algorithm for IG-MDP.

The specific features of IG-MDP are:

- It starts with the solution resulting from the application of a local search procedure on the solution returned by the D2 algorithm (Section 3.1). Duarte and Martí [9] observed that this algorithm is very suitable to generate initial solutions for the MDP.
- The procedure `Random-Drop` (S, n_d) removes n_d (a parameter of the algorithm) components from the current solution. These components are selected at random. Most IG instances in the literature follow this strategy for the destruction phase [33,34,40].
- The procedure `Greedy-Add` (Section 3.1) is employed for the construction phase.
- We have explored the following acceptance criteria:
  - *'Replace if better' acceptance criterion* (RB). The new solution is accepted only if it provides a better objective function value [41].
  - *Random walk acceptance criterion* (RW). An IG algorithm using the RB acceptance criterion may lead to stagnation situations of the search due to insufficient diversification [33]. At the opposite extreme is the random walk acceptance criterion, which always applies the destruction phase to the most recently visited solution, irrespective of its objective function value. This criterion clearly favors diversification over intensification, because it promotes a stochastic search in the space of local optima.
- For the local search phase, we have investigated different improvement methods proposed in the literature for the MDP. They are described in Section 3.3.

3.3. Improvement methods

Next, we describe the most representative improvement methods proposed in the literature for the MDP:

- *Best-improvement local search* (Best) [15]. It is based on the *1-interchange move*. Given a solution S, this move exchanges a single element $s \in S$ with a single element $t \in N \backslash S$.
  Thus, the result of this move is $S' = S \cup \{t\} \backslash \{s\}$. It is possible to efficiently evaluate such a move without recomputing the objective function from scratch. Let z be the value of the objective function for solution S. The value z' of the new solution S' is obtained by subtracting the total contribution of the old element s (that is $D(s,S)$) and adding the total contribution of the new element t (that is $D(t,S) - d(s,t)$), that is, more formally:

  $$z' = z - D(s, S) = D(t, S) - d(s, t).$$

The move yielding the largest positive improvement in the objective function is selected and applied. After each move, the values $D(s_i,S')$, $i = 1,\ldots,n$, may be calculated as follows:

$$D(e_i,S') = D(e_i,S) - d(e_i,s) = d(e_i,t).$$

The method performs moves while the objective value increases until no further improvement is possible. We should point out that this local search was the preferred improvement method for many GRASP instances proposed in the literature for the MDP [2,3,9,36].

- *First-improvement local search* (First) [7]. In this local search, a first improvement strategy is considered, that is, the first 1-interchange move yielding an improvement in the objective function is selected and applied.
- *Improved local search* (I_LS) [7]. Duarte and Martí [9] modified the Best improvement technique to increase its efficiency. First, I_LS selects the element $s_{lowest} \in S$ with the lowest contribution to S, i.e.,

$$D(s_{lowest},S) = \min_{i=1,\ldots,n} D(s_i,S).$$

Then, instead of scanning the whole set $N\backslash S$ searching for the best exchange associated with s_{lowest}, the authors restrict their method to performing the first improving move (without examining the remaining elements in $N\backslash S$). If there is no improving move associated with s_{lowest}, we resort to the next element with the lowest contribution to S and so on. I_LS performs iterations until no further improvement is possible.

- *Fast Lin–Kernighan algorithm for the MDP* (FastLK). Previous improvement methods are based on the N_1 neighborhood. Larger sized neighborhoods, such as the N_k neighborhood ($1 < k < n$), may yield better local optima, but the effort needed to search the neighborhood is computationally too expensive. *Lin-Kernighan* algorithms are based on efficiently searching a small fraction of the large neighborhood [24]. A basic implementation of this algorithm for the MDP is described as follows. Given a starting solution S, a sequence of k solutions ($k = \min(m,n-m)$) is generated by these two steps:

1. Perform the *1-interchange move* with the highest contribution to the objective function.
2. Repeat Step 1 k times so that the elements cannot be chosen to be inserted in S or removed from S if they have been used at one of the previous iterations of Step 1.

The best solution in the sequence is adopted as a new initial solution for the next iteration. Such a process is repeated until no better solution is found. We should point out that an instance of the Lin–Kernighan algorithm for the MDP may be found in [21]. In this case, the authors suggested a strategy to reduce the running time of the algorithm by stopping the process for generating the sequence of solutions when the best solution found so far in the sequence has not been updated for more than n_{it} iterations. They proposed the adoption of a parameter value $n_{it} = \frac{m}{5}$ for large MDP instances. We have incorporated this strategy to the Lin–Kernighan algorithm described above, obtaining the FastLK improvement algorithm.

4. Computational experiments

In this section, we describe the experiments carried out in order to study the behavior of the IG-MDP model presented in the previous section. Firstly, we detail the experimental setup and statistical method applied (Section 4.1), then, we analyze the results obtained from different experimental studies carried out with IG-MDP. Our aim is: (1) to analyse the influence of the parameters and settings associated with the algorithm (Section 4.2), (2) to obtain a tuned IG-MDP instance being able to show a robust operation for test problems with different characteristics, and 3) to compare the results of the tuned IG-MDP with ones of other metaheuristic approaches for the MDP from the literature (Section 4.4).

4.1. Experimental setup

The codes of all the studied algorithms were compiled with Dev-C++ 4.9.8.0. All runs were made on a 2.5 GHz Pentium Dual-Core with 3 GB of RAM running Windows Vista Business. To evaluate the performance of IG-MDP, 76 medium-large MDP instances were selected from the literature. Table 1 shows their names, references where a detailed description may be found, associated n and m values, number of instances, and the maximum CPU time limits allotted for each run of the studied algorithms (they were taken from [28], with the exception of the ones for the *mdp* instances). All algorithms were run 10 times on each problem instance.

Non-parametric tests have been used to compare the results of different search algorithms [14]. Given that the non-parametric tests do not require explicit conditions to be conducted, it is recommended that the sample of results should be obtained following the same criterion, which is, to compute the same aggregation (we have considered the average of the best objective function value found at the end of each run) over the same number of runs for each algorithm and problem. Specifically, we have considered two alternative methods based on nonparametric tests to analyze the experimental results:

- The first method is the application of the *Iman and Davenport* test and the *Holm* method as a post hoc procedure. The first test may be used to see whether there are significant statistical differences among the algorithms in a certain group of test algorithms. If differences are detected, then Holm's test is employed to compare the best algorithm (control algorithm) against the remaining ones.
- The second method is the utilization of the *Wilcoxon matched-pairs signed-ranks* test. With this test, the results of two algorithms may be directly compared.

Table 1
MDP instances considered for the experiments.

Name	Ref.	n	m	Number inst.	Time limit (s)
Type2	[9]	500	50	20	20
Type1_22	[9]	2000	200	20	20
mdp0500	[21]	500	50, 100, 150, 200	4	20
mdp0750	[21]	750	75, 150, 225, 300	4	20
mdp1000	[21]	1000	100, 200, 300, 400	4	20
mdp2500	[21]	2500	250, 500, 750, 1000	4	1200
b2500	[6,28]	2500	1000	10	1200
p3000	[28]	3000	1500	5	1200
p5000	[28]	5000	2500	5	1200

4.2. Study of IG-MDP with different parameters

The aim of this section is to investigate the effect of different parameters and strategies used in IG-MDP. Particularly, we attempt to find the combinations of acceptance criteria, improvement method, and n_d (number of solution components that are removed from the current solution) that result in high performance. (Note that only some few possible combinations are explored, so performance may be further improved by examining more possible combinations/ values).

In Table 2 we have summarized the results obtained by these algorithms following the methodology suggested by Palubeckis [28]. For each algorithm A and set of problem instances, in column *AD-B* we include the average difference (over all instances in the set) between the best known values (reported by Palubeckis [28]) and the best solutions found by the algorithm A, and in the case of the column *AD-Av*, we outline the average difference between the best known values and the average values of 10 runs of the algorithm A (clearly, low values for these two measures indicate that results of an algorithm are close to the best ones). In addition, the table shows the average *rankings* obtained by the IG-MDP versions. This measure is obtained by computing, for each problem instance, the ranking r_j of the observed results for algorithm version j assigning to the best of them the ranking 1, and to the worst the ranking n_v (n_v is the number of versions of the algorithm). Then, an average measure is obtained from the rankings of this algorithm version for all test problems. For example, if a certain algorithm achieves rankings 1, 3, 1, 4, and 2, on five problem instances, the average ranking is $\frac{1+3+1+4+2}{5} = \frac{11}{5}$. In order to analyze these results, we have applied the Iman-Davenport's test (the level of significance considered was 0.05). We have observed the existence of significant differences among the rankings (the statistical value is 154.63 and the critical one 1.5488). Then, we have compared the best ranked algorithm (RW, I_LS, $0.75 \cdot m$) with the other IG-MDP versions, by means of Holm's test. Table 2 contains all the computations associated with this procedure with $p = 0.05$. The last column indicates whether the Holm's test finds statistical differences between the control algorithm and the corresponding algorithm. If the corresponding p-value is smaller than the adjusted α, the test detects significant differences between them, which means that the control algorithm is better.

We point out the following from Table 2:

- Quite surprisingly, the three best algorithms use the RW acceptance criterion. As a matter of fact, it is unusual to find IG models in the literature employing this strategy (most of them show a bias towards maintaining high quality solutions). Thus, we may remark that the combination of the diversification of RW and the intensification of the improvement procedure produces beneficial effects on the IG-MDP performance.
- IG-MDP instances based on the fastest improvement algorithms, I_LS and First, have obtained significant performance results. When a maximum CPU time limit is imposed, the use of an efficient improvement algorithm may allow a high number of cycles to be accomplished, favoring the exploration of the search space.
- The combinations (RW, I_LS, $0.75 \cdot m$) and (RW, I_LS, $0.5 \cdot m$) offer the best *AD-Av* and *AD-B* values for the two sets of problem instances. Both are based on the RW acceptance criterion and the I_LS improvement method and differ on the n_d value. Particularly, meaningful advantage is achieved with the first combination for the *Type1_22* instances (in fact, it was the best ranked IG variation). The combination with $n_d = 0.5 \cdot m$ obtained surprisingly good results for the *p3000* instances. Particularly, the *AD-B* measure is negative, which means that this algorithm found solutions for these instances that are better than the best known solutions reported in [28].

4.3. Tuned IG-MDP algorithm

Regarding the results in Table 2, and with the aim of producing a robust operation for the IG-MDP algorithm, we have built an instance based on RW and I_LS that varies randomly the value of n_d in $[0.5 \cdot m, 0.75 \cdot m]$ at each iteration. It will be called tuned IG-MDP (TIG).

Now, we attempt to detect the differences among TIG and the best IG-MDP algorithms (the IG algorithms for which Holm's test did not detect significant differences in Table 2) by means of

Table 2
Average rankings, results, and comparison (Holm's test) between the IG-MDP instances.

IG-MDP	Av. Ran.	Type1_22		p3000		Holm's test			
(Acc. crit. - LS - n_d)		*AD-B*	*AD-Av*	*AD-B*	*AD-Av*	*z*	*p*-value	α/i	*Diff.?*
RB Best $0.25 \cdot m$	22.64	418.9	569.74	4619.4	7897.4	10.04	1.02E−23	0.002	yes
RB I_LS $0.25 \cdot m$	22.16	425.7	560.49	4902.8	7321.3	9.8	1.13E−22	0.002	yes
RB First $0.25 \cdot m$	22.14	417.6	562.85	4377.6	7321.4	9.79	1.24E−22	0.002	yes
RB FastLK $0.25 \cdot m$	20.12	411	498.52	5007	6632.88	8.78	1.63E−18	0.002	yes
RW FastLK $0.25 \cdot m$	19.82	418.4	493.42	4833	6483.7	8.63	6.14E−18	0.002	yes
RW Best $0.25 \cdot m$	18.72	326.2	479.32	3458.8	5374.14	8.08	6.48E−16	0.002	yes
RW I_LS $0.25 \cdot m$	17.96	353.7	483.345	3061.8	4612.08	7.7	1.36E−14	0.002	yes
RW First $0.25 \cdot m$	17.8	332.75	469.57	3502.4	5349.52	7.62	2.54E−14	0.003	yes
RB FastLK $0.5 \cdot m$	14.52	290	372.845	2987.6	4092.82	5.98	2.23E−09	0.003	yes
RB Best $0.5 \cdot m$	14	216.8	373.75	2279	3654.72	5.72	1.07E−08	0.003	yes
RW FastLK $0.5 \cdot m$	13.16	252.55	355.045	2791.8	3742.74	5.3	1.16E−07	0.003	yes
RB I_LS $0.5 \cdot m$	12.72	227.9	359.215	2150.4	3458.34	5.08	3.77E−07	0.004	yes
RB FastLK $0.75 \cdot m$	12.68	155.3	308.015	2540.2	4573.28	5.06	4.19E−07	0.004	yes
RB First $0.5 \cdot m$	12.55	239.1	357.03	1857	3120.86	4.999	5.73E−07	0.005	yes
RB Best $0.75 \cdot m$	9.44	128.35	262.25	1938.4	3518.16	3.44	5.82E−04	0.005	yes
RW Best $0.75 \cdot m$	9.12	91.1	217.89	3416.4	4838.22	3.279	0.001	0.006	yes
RW FastLK $0.75 \cdot m$	8.48	136.65	252.955	1619.2	2954.96	2.96	0.003	0.007	yes
RW Best $0.5 \cdot m$	6.44	140.4	236.07	148.4	1169.14	1.94	0.052	0.008	no
RW First $0.5 \cdot m$	5.92	121.7	234.2	151.2	840.82	1.68	0.092	0.01	no
RB First $0.75 \cdot m$	5.16	95.15	207.735	14.6	1228.38	1.3	0.193	0.012	no
RB I_LS $0.75 \cdot m$	5	82.9	200.78	24.8	1665.14	1.22	0.222	0.016	no
RW I_LS $0.5 \cdot m$	4.24	108.55	204.66	−155.8	408.48	0.84	0.4	0.025	no
RW First $0.75 \cdot m$	2.64	31.15	115.65	1012.4	2012.68	0.04	0.968	0.05	no
RW I_LS $0.75 \cdot m$	2.56	27.9	97.87	1712.6	2435.18				

Table 3
TIG vs. best standard IG-MDP algorithms (Wilcoxon's test with p-value = 0.05 and critical value = 89).

Standard IG	Wilcoxon's test			Type1_22		p3000	
(Acc. criterion - LS - n_d^f)	$R+$	$R-$	*Diff.?*	*AD-B*	*AD-Av*	*AD-B*	*AD-Av*
RW Best 0.5·m	322.0	3.0	yes	140.4	236.07	148.4	1169.14
RW First 0.5·m	321.0	4.0	yes	121.7	234.2	151.2	840.82
RB First 0.75·m	325.0	0.0	yes	95.15	207.735	14.6	1228.38
RB I_LS 0.75·m	325.0	0.0	yes	82.9	200.78	24.8	1665.14
RW I_LS 0.5·m	302.0	23.0	yes	108.55	204.66	−155.8	408.48
RW First 0.75·m	294.0	31.0	yes	31.15	115.65	1012.4	2012.68
RW I_LS 0.75·m	243.0	82.0	yes	27.9	97.87	1712.6	2435.18
TIG				20.1	92.39	191.4	631.38

Wilcoxon's test. Table 3 summarizes the results of this procedure for $p = 0.05$, where the values of $R+$ (associated to TIG) and $R-$ of the test are specified. The fourth column indicates whether Wilcoxon's test found statistical differences between these algorithms. If $min(R^+,R^-)$ is less than or equal to the critical value, this test detects significant differences between the algorithms, which means that an algorithm outperforms its opposite. Particularly, if this occurs and $R^- = min(R^+,R^-)$, then TIG is statistically better than the other algorithm.

TIG has the upper hand in the statistical comparison over all other IG instances. Thus, we may conclude that this fine-tuned IG-MDP algorithm may produce a robust operation for test problems with different characteristics.

4.4. TIG vs. state-of-the-art metaheuristics for the MDP

In a recent publication, Brimberg et al. [7] present a simple *variable neighborhood search* (VNS) approach to the MDP that was found to be superior to other metaheuristics (see Section 2). Thus, we assume that VNS is, up to now, the best performing metaheuristic for the MDP. Since in [7] ITS was found to be the best competitor for VNS and, it is considered to be one of the best algorithms for this problem [39], we also compared our TIG to it. The objective of this section is to pit TIG against VNS and ITS. Firstly, we describe, in depth, the general scheme of these two algorithms.

VNS is a metaheuristic that systematically exploits the idea of neighborhood change (from a given set of neighborhood structures N_k, $k = 1,\ldots,k_{max}$), both in the descent to local minima and in the escape from the valleys, which contain them. Brimberg et al. [7] propose a simple VNS algorithm specifically developed to tackle the MDP. It mainly consists of the following three phases, which work on a single candidate solution, the current solution, S_c:

- *Generation phase:* Firstly, a method is executed to generate an initial solution S_c. It is a random procedure sampling uniformly the search space.
- *Improvement phase:* S_c undergoes a refinement process by the *first-improvement* local search (Section 3.3).
- *Shaking phase:* It is performed to escape from the valley where S_c lies. This method applies k times a 1-interchange move (at random) on S_c. For the algorithm to have a reactive behavior, the parameter k is adjusted dynamically depending on the quality of the solutions obtained. At the beginning of the run, and every time the improvement process outperforms the best solution found, k is set to one; otherwise, k is set to $k + 1$. When k exceeds k_{max}, a new iteration starts, and k gets back to 1. The authors recommended $k_{max} = \min\{m, n - m\}$.

In this VNS, the generation phase is performed once, at the beginning of the optimization process, then, shaking and improvement phases are iterated until an imposed time limit is reached. The current solution is updated whenever the improvement phase obtains a better solution.

ITS [28] was designed to address the need to increase the efficiency of TS for the MDP by incorporating a powerful search diversification mechanism. ITS alternates between two phases: solution perturbation and TS. Each time, the second of them is applied to the solution from the first phase. The employed TS is a short-term memory tabu list without aspiration criterion. In addition, in the case of finding, during the run of TS, a better solution than the previous best solution, a local search procedure (a standard routine performing an ascent from the given point to a local optimum) is applied to this new solution. The method to perturb a solution S consists of generating, at random, a point up to a maximum distance k_{max} from S (a parameter), which becomes a new initial solution for TS. It is interesting to point out that ITS uses different neighborhoods in the perturbation, not systematically (as VNS does) but at random.

We should point out that ITS, VNS, and TIG were run under the same computational conditions (machine, programming language,

Table 4
TIG versus VNS using Wilcoxon's test (p-value = 0.05).

Problem Inst.	$R+$ (TIG)	$R-$ (VNS)	Critical value	Diff.?
Type2	177	33	52	yes
Type1_22	210	0	52	yes
mdp	133	3	29	yes
b2500, p3000, p5000	210	0	52	yes

Table 5
TIG versus ITS using Wilcoxon's test (p-value = 0.05).

Problem Inst.	$R+$ (TIG)	$R-$ (ITS)	Critical value	Diff.?
Type2	171	39	52	yes
Type1_22	210	0	52	yes
mdp	134.5	1.5	29	yes
b2500, p3000, p5000	195	15	52	yes

Table 6
Results of TIG, ITS, and VNS on all problem instances.

Problem	TIG		ITS		VNS	
	AD-B	*AD-Av*	*AD-B*	*AD-Av*	*AD-B*	*AD-Av*
Type2	0	4.72	0	81.9	0	46.02
Type1_22	20.1	92.39	111.6	275.6	160.9	356.95
mdp	5.18	70.79	61.13	225.67	134.38	421.54
b2500	465	993.32	473.2	1532.56	2535	4254.3
p3000	191.4	631.38	321.2	1370.36	1624.4	3406.92
p5000	648.8	1876.2	2990.8	6183.04	4662.8	7825

Table 7
Comparison between best known results.

Ins.	ITS	TIG	VNS	Ins.	ITS	TIG	VNS
Type1_22.1	114271	**114271**	114211	b2500–1	1153068	1153014	1151144
Type1_22.2	114327	**114327**	114064	b2500–2	1129310	1128492	1124394
Type1_22.3	114195	**114195**	114131	b2500–3	1115538	1115142	1112346
Type1_22.4	114093	**114093**	114000	b2500–4	1147840	1147260	1145192
Type1_22.5	114196	**114196**	114054	b2500–5	1144756	1144420	1143016
Type1_22.6	114265	**114265**	114238	b2500–6	1133572	1133296	1131062
Type1_22.7	114361	**114361**	**114361**	b2500–7	1149064	1148616	1147138
Type1_22.8	114327	**114327**	114115	b2500–8	1142762	1142616	1138750
Type1_22.9	114199	**114199**	114183	b2500–9	1138866	1138592	1135812
Type1_22.10	114229	**114229**	114205	b2500–10	1153936	1153578	1153118
Type1_22.11	114214	114199	114123	p3000–1	6501999	6501569	6500376
Type1_22.12	114214	114199	114093	p3000–2	18272568	18272043	18270504
Type1_22.13	114233	**114233**	114120	p3000–3	29867138	**29867138**	29864823
Type1_22.14	114216	**114216**	114065	p3000–4	46914817	46914806	46912779
Type1_22.15	114240	114239	114239	p3000–5	58095034	58095120	58093171
Type1_22.16	114335	**114335**	114168	p5000–1	17508071	17507722	17505292
Type1_22.17	114255	**114255**	114126	p5000–2	50101514	**50102477**	50098434
Type1_22.18	114408	**114408**	114391	p5000–3	82038723	82038177	82030273
Type1_22.19	114201	**114201**	113897	p5000–4	129411337	129411237	129406255
Type1_22.20	114349	**114349**	114179	p5000–5	160597469	160597077	160596072

compiler, and time limits; see Section 4.1) in order to enable a fair comparison between them. We have used the source code of ITS provided by the author[3] and implemented the other two algorithms in C.

For the different problem instances, Tables 4 and 5 have the results of the comparison of TIG and these two algorithms, by means of Wilcoxon's test (the $R+$ are associated with our algorithm). The *AD-B* and *AD-Av* measures for these algorithms may be found in Table 6. From these Tables, we clearly notice that TIG obtained improvements with respect to the other algorithms, which are always statistically significant (because all $R-$ values are lower than both $R+$ ones and critical values). In summary, this experimental analysis shows that TIG is currently a state-of-the-art method for solving the MDP, which evidences the great potential of the IG metaheuristic as a search algorithm for this problem.

Other performance criteria, different from *AD-B* and *AD-Av*, have been suggested in the literature to assess the performance of metaheuristics for the MDP. One of them consists of quantifying how many best known results in the literature can be equaled or improved by the algorithms [4,7,28,37]. In [28], the ITS algorithm was executed for long CPU times in order to improve as much as possible the best known values (in fact, it was able to find new best solutions for 69 test problems appearing in the literature). The imposed time limit for a run was 1200 s for instances in *Type1_22*, 5 h for instances in *b2500* and *p3000*, and 10 h for *p5000*. We ran TIG and VNS under these time conditions, in order to compare their best results with the ones of ITS (reported in [28]). They are displayed in Table 7 (the values that are equal or better than the ones of ITS are presented in boldface).

We may point out the following facts:

- VNS could not find any new best known value for the considered problem instances.
- In the case of the *Type1_22* problem instances, our algorithm equaled the results of ITS in 17 out of 20 instances.
- For the *p3000* instances, TIG returned the same best solution as ITS did in one out of 5 cases. For the largest size instances, *p5000*, TIG was able to achieve the best known value in one out of 5 instances. For Beasley instances (*b2500*), TIG was not able to improve the results offered by ITS.
- We should recall that, in a previous experiment in Section 4.2 (10 runs with a time limit of 1200 seconds), IG-MDP with $(RW, I\_LS, 0.5 \cdot m)$ was able to return new best known solutions for the five *p3000* instances[4]. Given that the IG with this parameter setting was so high performing on the p3000 instances, we decided to run for the longer computation time limits this version of IG on the *p3000* and *p5000* instances (5 hours and 10 hours, respectively). Under this experimental scenario, it was able to find new best known solutions for the five *p5000* instances[5]. These facts arise as another sign of the ability of our proposed IG metaheuristic to obtain high quality solutions for the MDP.

5. Conclusions

In this paper, we proposed an IG scheme for the MDP and investigated the influence of applying different n_d values, local search methods, and acceptance criteria. From the analysis of the obtained results, we have derived a fine-tuning IG that has proved to be a very high performing algorithm for the problem, resulting very competitive with other state-of-the-art algorithms. This result along with the simplicity and flexibility of the IG approach allow us to conclude that this metaheuristic arises as a tool of choice to tackle this problem.

We believe that the IG framework presented in this paper is a significant contribution, worthy of future study. We will intend to explore two interesting avenues of research: (1) adapt the IG approach for its application to other challenging combinatorial problems, such as the max–min diversity problem and the max-cut problem, and (2) build hybrid metaheuristics combining the proposed IG with other salient metaheuristics for the MDP (e.g., ITS and VNS).

Acknowledgments

The authors thank the associate editor and anonymous reviewers for the valuable suggestions and constructive comments.

[3] ITS is publicly available at http://www.soften.ktu.lt/~gintaras/max_div.html.

[4] The new best known results are 6502255, 18272570, 29867140, 46914900, and 58095470.

[5] The new best known results are 17509352, 50103059, 82040100, 129413344, and 160598020.

References

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

[2] M. Andrade, P. Andrade, S. Martins, A. Plastino, GRASP with path-relinking for the maximum diversity problem, in: S. Nikoletseas (Ed.), Experimental and Efficient Algorithms, vol. 3503, Springer, Berlin, 2005, pp. 558–569.

[3] P.M.D. Andrade, A. Plastino, L.S. Ochi, S.L. Martins, GRASP for the maximum diversity problem. in: Proceedings of the Fifth Metaheuristics International Conference (MIC 2003), 2003, CD-ROM Paper: MIC03 15.

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

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

[6] J.E. Beasley, Obtaining test problems via internet, Journal of Global Optimization 8 (1996) 429–433.

[7] J. Brimberg, N. Mladenović, D. Urošević, Variable neighborhood search for the heaviest ksubgraph, Computers & Operations Research 36 (11) (2009) 2885–2891.

[8] J. Culberson, F. Luo, Exploring the k-colorable landscape with iterated greedy, in: D.S. Johnson, M.A. Trick (Eds.), Cliques, coloring, and satisfiability: second DIMACS implementation challenge, vol. 26, American Mathematical Society, 1996, pp. 245–284.

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

[10] L. Fanjul-Peyroa, R. Ruiz, Iterated greedy local search methods for unrelated parallel machine scheduling, European Journal of Operational Research 207 (2010) 55–69.

[11] B. Feng, Z-Z. Jiang, Z-P. Fan, N. Fu, A method for member selection of cross-functional teams using the individual and collaborative performances, European Journal of Operational Research 203 (3) (2010) 652–661.

[12] J.M. Framinan, R. Leisten, A multi-objective iterated greedy search for flowshop scheduling with makespan and flowtime criteria, OR Spectrum 30 (4) (2008) 787–804.

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

[14] S. García, D. Molina, M. Lozano, F. Herrera, A study on the use of nonparametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization, Journal of Heuristics 15 (2009) 617–644.

[15] J.B. Ghosh, Computational aspects of the maximum diversity problem, Operations Research Letters 19 (1996) 175–181.

[16] F. Glover, Multi-start and strategic oscillation methods - principles to exploit adaptive memory - a tutorial on unexplored opportunities, in: M. Laguna, J.L. Gonzales-Valarde (Eds.), Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research, Kluwer Academic Publishers., 2000, pp. 1–24.

[17] F. Glover, C.C. Kuo, K.S. Dhir, Analyzing and modeling the maximum diversity problem by zero–one programming, Decision Sciences 24 (1993) 1171–1185.

[18] F. Glover, C.C. Kuo, K.S. Dhir, Heuristic algorithms for the maximum diversity problem, Journal of Information and Optimization Sciences 19 (1998) 109–132.

[19] F. Gortázar, A. Duarte, M. Laguna, R. Martí, Black box scatter search for general classes of binary optimization problems, Computers & Operations Research 37 (11) (2010) 1977–1986.

[20] L.W. Jacobs, M.J. Brusco, A local-search heuristic for large set-covering problems, Naval Research Logistics 42 (1995) 1129–1140.

[21] K. Katayama, H. Narihisa, An evolutionary approach for the maximum diversity problem, in: W. Hart, N. Krasnogor, J.E. Smith (Eds.), Recent Advances in Memetic Algorithms, Springer, Berlin, 2004, pp. 31–47.

[22] R.K. Kincaid, Good solutions to discrete noxious location problems via metaheuristics, Annals of Operations Research 40 (1-4) (1992) 265–281.

[23] C.C. Kuo, F. Glover, K.S. Dhir, Analyzing and modeling the maximum diversity problem by zero-one programming, Decision Sciences 24 (1993) 1171–1185.

[24] S. Lin, B. Kernighan, An effective heuristic algorithm for the traveling salesman problem, Operations Research 21 (1973) 498–516.

[25] S.W. Lin, Z.J. Lee, K.C. Ying, C.C. Lu, Minimization of maximum lateness on parallel machines with sequence-dependent setup times and job release dates, Computers & Operations Research 38 (5) (2011) 809–815.

[26] E.M. Macambira, An application of tabu search heuristic for the maximum edge-weighted subgraph problem, Annals of Operations Research 117 (1-4) (2002) 175–190.

[27] R. Martí, M. Gallego, A. Duarte, A branch and bound algorithm for the maximum diversity problem, European Journal of Operational Research 200 (1) (2009) 36–44.

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

[29] M. Prais, C.C. Ribeiro, Reactive GRASP: an application to a matrix decomposition problem in TDMA traffic assignment, INFORMS Journal on Computing 12 (2000) 164–176.

[30] O.A. Prokopyev, N.K. Dayna, L. Martínez-Torres, The equitable dispersion problem, European Journal of Operational Research 197 (2009) 59–67.

[31] M.G.C. Resende, C.C. Ribeiro, Greedy randomized adaptive search procedures, in: Glover, F., Kochenberger, G. (Eds.), Handbook of Metaheuristics, 2003, pp. 219–249.

[32] I. Ribas, R. Companys, X. Tort-Martorell, An iterated greedy algorithm for the flowshop scheduling problem with blocking, Omega 39 (3) (2011) 293–301.

[33] R. Ruiz, T. Stützle, A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem, European Journal of Operational Research 177 (2007) 2033–2049.

[34] R. Ruiz, T. Stützle, An Iterated Greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives, European Journal of Operational Research 187 (2008) 1143–1159.

[35] L.F. Santos, M.H. Ribeiro, A. Plastino, S.L. Martins, A hybrid GRASP with data mining for the maximum diversity problem, in: M. Blesa, C. Blum, A. Roli, M. Sampels (Eds.), Hybrid Metaheuristics, vol. 3636, Springer, Berlin, 2005, pp. 116–127.

[36] G.C. Silva, L.S. Ochi, S.L. Martins, Experimental comparison of greedy randomized adaptive search procedures for the maximum diversity problem, in: C. Ribeiro, C. Martins, L. Simone (Eds.), Experimental and Efficient Algorithms, vol. 3059, Springer, Berlin, 2004, pp. 498–512.

[37] G.C. Silva, M.R.Q. De Andrade, L.S. Ochi, S.L. Martins, A. Plastino, New heuristics for the maximum diversity problem, Journal of Heuristics 13 (4) (2007) 315–336.

[38] T. Urlings, R. Ruiz, T. Stützle, Shifting representation search for hybrid flexible flowline problems, European Journal of Operational Research 207 (2010) 1086–1095.

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

[40] K.-C. Ying, Solving non-permutation flowshop scheduling problems by an effective iterated greedy heuristic, International Journal of Advanced Manufacturing Technology 38 (3–4) (2008) 348–354.

[41] K.-C. Kuo-Ching Ying, H.-M. Cheng, Dynamic parallel machine scheduling with sequence-dependent setup times using an iterated greedy heuristic, Expert Systems with Applications 37 (4) (2010) 2848–2852.

[42] Z. Yuan, A. Fügenschuh, H. Homfeld, P. Balaprakash, T. Stützle, M. Schoch, Iterated greedy algorithms for a real-world cyclic train scheduling problem, in: Blesa et al. (Eds.), Hybrid Metaheuristics, vol. 5296, Springer, Berlin, 2008, pp. 102–116.