Memetic Algorithms

Jin-Kao Hao and Xiangjing Lai

Abstract This chapter is dedicated to memetic algorithms for diversity and dispersion problems. It is organized in two parts. The first part discusses general design issues of memetic algorithms concerning crossover, local search, population management, evaluation function and constraint handling. The second part presents a survey of memetic algorithms applied to three diversity and dispersion problems: maximum diversity, the max-mean dispersion, and generalized max-mean dispersion.

1 Introduction

Diversity and dispersion problems are typical combinatorial optimization problems whose search spaces increase exponentially with the size of the input. Seeking the best or a good-enough solution in such a combinatorial space requires a suitable balance between "exploitation" and "exploration" of the search process. Indeed, the dual concept of exploitation and exploration covers two fundamental and complementary aspects of an effective search procedure. This concept is also known under the term "intensification" and "diversification" introduced within the Tabu Search methodology (Glover and Laguna, 1997).

Exploitation emphasizes the ability of the search procedure to examine intensively and in depth specific search areas while exploration is the capacity to lead the search to new promising areas. As such, a search procedure focusing only on exploitation will confine itself in a limited area and may be trapped in poor optima (i.e., missing good solutions that are not in the area under examination). On the other hand, a method

Jin-Kao Hao
LERIA, Université d'Angers, 2 Boulevard Lavoisier, 49045 Angers, France, e-mail: jin-kao.hao@univ-angers.fr

Xiangjing Lai
Institute of Advanced Technology, Nanjing University of Posts and Telecommunications, Nanjing 210023, China, e-mail: laixiangjing@gmail.com

focusing heavily on exploration and overlooking exploitation will lack capacity to examine in depth a given area and pass by solutions of good quality. An effective search method needs thus to appropriately balance exploitation and exploration. Memetic Algorithms (MAs) (Moscato, 1999) constitute a very interesting framework offering a variety of strategies and mechanisms to achieve this general objective.

MAs are hybrid search methods that combine the population-based search framework (Bäck et al., 1989; Eiben and Smith, 2003) and neighborhood-based local search framework (LS) (Hoos and Stützle, 2004). The basic rationale behind MAs is to take advantage of the complementary search strategies offered by these two different methods. Indeed, thanks to the use of a pool of multiple solutions, the population-based framework naturally offers more facilities for exploration while the neighborhood search framework provides more possibilities for exploitation. A suitable hybridization of these two methods can help to ensure a good balance between exploitation and exploration, assuring a high search performance.

MAs are a general optimization framework that can potentially be applied to various combinatorial search problems (Neri et al., 2012). However, a successful MA requires a careful adaptation of its components to the problem under consideration by integrating problem-specific knowledge into the search operators and strategies (Hao, 2012).

The remaining parts of this chapter is organized as follows. In Section 2, we give the general presentation of the memetic algorithms. In Section 3, we show the main considerations needed when designing an efficient memetic algorithm for the combinatorial optimization problems. In Section 4, we review representative memetic algorithms for three diversity and dispersion problems, including the maximum diversity problem, the max-mean dispersion problem, and the generalized max-mean dispersion problem. In the last section, we summarize this chapter.

2 General presentation of memetic algorithms

Memetic Algorithms (Moscato, 1999) are a population-based computational framework and share a number of features with methods like Evolutionary Algorithms and Scatter Search (Glover et al., 2000). MAs maintain a pool of high-quality candidate solutions and use these solutions to create new solutions by applying variation operators such as combinations and local improvements.

A typical MA is composed of the following basic components: a population of individuals to sample the search space, a combination operator (crossover) to create new candidate solutions (offspring) by blending two or more solutions of the population, a local improvement procedure to ameliorate offspring solutions, and a population management strategy to update the population with the offspring. Additionally, the MA uses an evaluation or fitness function to assess the quality of each candidate solution and a selection mechanism to determine the candidate solutions that will survive and undergo variations.

Starting from an initial population (see §3.4), the MA repeats cycles of generations, each generation being composed of four sequential steps.

1. *Selection of parents*: Selection aims to determine the candidate solutions that will be used to create offspring solutions. Selection for reproduction often operates in relation with the fitness (quality) of the candidate solutions; high-quality solutions have thus more chances to be chosen. Well-known examples of selection strategies include roulette-wheel and tournament. Selection can also be done according to other criteria such as diversity. In such a case, only "distanced" individuals are allowed to reproduce. If the solutions of the population are sufficiently diversified, selection can also be carried out randomly. The selection strategy influences the diversity of the population (see also §3.3).

2. *Combination of parents for offspring generation*: Combination (also called crossover) aims to create new *promising* solutions by blending (suitably) existing the chosen parents. An offspring solution is considered to be promising if it can potentially lead the search process to new search areas where better solutions may be found. To achieve this, the combination operator is often designed such that it captures the semantics of the targeted problem to ensure the heritage of good properties from parents to offspring. Additionally, the design of the combination operator should ideally take care of creating *diversified* offspring. From a perspective of exploration and exploitation, such a combination is intended to play a role of strategic diversification with a long-term goal of reinforcing the intensification. A carefully designed combination operator constitutes a driving force of a successful MA.

3. *Local improvement of offspring*: The goal of local improvement is to improve the quality of an offspring solution. Typically, starting from the offspring solution, local improvement iteratively seeks new high-quality candidate solutions within a given neighborhood. This process stops and returns the best solution found when a user-defined stop condition is met. Compared with the combination operator, local improvement plays essentially the role of intensifying the search by exploiting search paths delimited by the underlying neighborhood. Like combination, local improvement is another key component and driving force of a successful MA.

4. *Population updating*: This step decides whether and how the improved offspring solution from local improvement joins the population. Often, these decisions are made according to criteria related to both quality and diversity. Such a strategy is commonly employed in methods like Scatter Search and many Evolutionary Algorithms. For instance, a basic quality-based updating rule replaces the worst solution of the population while a diversity-based rule would substitute for a similar solution according to a distance metric. Other criteria like recency (age) can also be considered. The policy employed for managing the population should maintain an appropriate diversity of the population. This allows the search process

```
Input: p; // Size of population POP
Output: s*; // Best solution found
POP ← PopGeneration(p); //
PopEvaluation(POP); // Fitness evaluation of each individual
s* ← best(POP); // Record the best solution found so far
f* ← f(s*); // Record the fitness of the best solution
while Stop Condition is not verified do
    (s^1,...,s^k) ← ParentsSelection(POP); // k ≥ 2 parents are selected
    s' ← Recombination(s^1,...,s^k); // Offspring generation
    s ← OffspringImprovement(s'); // Improvement of offspring
        solution by local search
    POP ← PopulationUpdate(s,POP); // Population update
        according to a quality-diversity rule
    (s*,f*) ← BestSolutionUpdate(s*,f*,POP); // Best solution and its
        fitness are always recorded
end
return s*
```

Algorithm 1: Memetic Algorithm Template

to avoid premature convergence and helps the algorithm to continually discover new promising search areas.

The general MA template is described in Algorithm 1 where special attention must be paid to the design of particular components. The stopping condition can be a cut-off time, a maximum number of generations, a maximum number of fitness evaluations, a maximum number of generations without improving the best-recorded solution, a solution quality to be reached or a minimum threshold for the population diversity. We deliberately leave out the mutation operator within this MA template. In some sense, local search can be viewed as a guided macro-mutation operator. However, mutation can also be applied to reinforce population diversity (Porumbel et al., 2010). As a lean design principle, only necessary components are included in a MA, while excluding any unjustified and superficial elements.

3 Special design considerations

3.1 Design of local search

Local improvement is one of the most important components of a MA and ensures essentially the role of intensive exploitation of the search space. This is typically achieved either by dedicated local search heuristics (see examples in (Lin, 1965; Kernighan and Lin, 1970; Lin and Kernighan, 1973)) or by tailored general neighborhood search methods. In this part, we focus our discussion on adaptation of local search metaheuristics (Hoos and Stützle, 2004), but a large part of the discussion applies to the design of local improvement procedures based on specific heuristics.

3.1.1 Local search template

Let (S, f) be our search problem where S and f are respectively the search space and optimization objective. A neighborhood N over S is any function that associates to each solution $s \in S$ some other solutions $N(s) \subset S$. Any solution $s' \in N(s)$ is called a neighboring solution or simply a neighbor of s. For a given neighborhood N, a solution s is a *local optimum* with respect to N if s is the best in terms of f among the solutions in $N(s)$.

The notion of neighborhood can be explained in terms of the *move* operator. Typically applying a move mv to a solution s changes slightly s and leads to a neighboring solution s'. This transition from a solution to a neighbor is denoted by $s' = s \oplus mv$. Let $\Gamma(s)$ be the set of all possible moves which can be applied to s, then the neighborhood $N(s)$ of s can be defined by: $N(s) = \{s \oplus mv | mv \in \Gamma(s)\}$.

A typical local search algorithm begins with an initial configuration s in S and proceeds iteratively to visit a series of configurations following the neighborhood. At each iteration, a particular neighbor $s' \in N(s)$ is sought to replace the current configuration and the choice of s' is determined by the underlying metaheuristic and by referring to the quality of the neighboring solution. For instance, a strict Descent algorithm always replaces the current solution s by a *better* neighbor s' while tabu search replaces the current solution by a *best* neighbor s' even if the latter is of inferior quality. Still with simulated annealing, the transition from s to a randomly selected neighbor s' is conditioned by a changing probability.

3.1.2 Neighborhood design

The success of an LS algorithm depends strongly on its neighborhood. The neighborhood defines the subspace of the search problem to be explored by the method. For a given problem, the definition of the neighborhood should structure the search space such that it helps the search process to find its ways to good solutions.

The choice of neighborhood is conditioned by the representation (genotype) used to encode the candidate solutions of the search space (phenotype). It may further depend on the structure and constraints of the problem on hand. Here we briefly review some neighborhoods associated to three conventional representations, which have a variety of applications. For diversity and dispersion problems, the binary representation is most natural encoding technique. With the binary representation, each solution of the search space is coded by a binary string such that each variable represents a candidate item and takes the value of 1 if it is selected or 0 otherwise.

Based on this representation, two common neighborhoods are defined by the k-$flip$ and $Swap$ move operators. The k-$flip$ move changes the values of k ($k \geq 1$) variables. So any neighbor $s' \in N(s)$ has a Hamming distance of k to solution s. A larger k induces a larger neighborhood. Nevertheless, whether a larger neighborhood should be preferred in practice depends on the computational cost to evaluate the neighborhood. $Swap$ exchanges the values of two variables that have different values. Note that $Swap$ can be simulated by two successive 1-$flip$ moves.

These neighborhoods can be applied alone or jointly. Moreover, for a given problem, it is often useful to seek meaningful ways to reduce these neighborhoods by considering problem-specific knowledge of the problem.

3.1.3 Neighborhood evaluation

Another design issue that arises is the evaluation of a given neighborhood. Indeed, a local search procedure moves iteratively from the current solution to a new solution chosen within the neighborhood. To make this choice, local search needs to know the cost variation (also called the *move gain* or *move value*) between the current solution s and a candidate neighbor $s' \in N(s)$. The move gain indicates whether the neighbor s' is of better, worse or equal quality relative to s. Let $\Delta f = f(s')$ - $f(s)$ denote this move gain.

- *Incremental evaluation*: Basically, there are two ways to obtain Δf for a neighbor. The trivial way is to calculate $f(s')$ from "scratch" using the objective function[1] f. Doing this way may be expensive if f needs to be evaluated very often or if the evaluation of f itself involves complex calculations. A more efficient alternative aims to derive the value of $f(s')$ from the value $f(s)$ by updating only what is strictly necessary. Indeed, if a neighbor s' is close to its initial solution s, which is true for many neighborhoods, then the evaluation of $f(s')$ can be carried out in this incremental manner. For a number of basic neighborhoods, like those shown previously, such an incremental evaluation is often possible.

- *Full search of neighborhood*: The incremental evaluation can be applied to *all the neighbors* of a given neighborhood relation. In this case, it is generally useful to investigate dedicated data structures (call it Δ-table) to store the move gains for all the neighbors of the current solution. Δ-table provides a convenient way to know the quality of each neighbor and enables an efficient search of the full neighborhood. With such a Δ-table, the local search algorithm can decide easily at each iteration which neighbor to take according to its search strategy. For instance, a best-improvement descent algorithm will take the move that is identified by the most negative value in the Δ-table to minimize the objective function. After each move, the Δ-table (often only a portion of it) is updated accordingly using the incremental evaluation technique to propagate the effect of the move. Δ-table is a very useful technique for local search algorithms. This is particularly the case for descent-based methods like Tabu Search where a best neighbor needs to be identified (see examples in (Hoos and Stützle, 2004)).

- *Approximative evaluation*: The practical usefulness of Δ-table depends on both the complexity and the number of updates needed after each move transition. It may happen that, the move gain can not be incrementally calculated or the Δ

[1] For the reason of simplicity, the term "objective function" is used here. A more precise term is "evaluation function", see §3.4.

updates need to change a large portion of Δ-table. In this case, it would be useful to replace the initial evaluation function by a (fast) approximative evaluation function (Jin, 2005; Ren et al., 2021). More generally, approximate evaluation is useful if the evaluation function is computationally expensive to calculate or if the function is ill-defined.

- *Order of evaluation*: If the neighborhood is not completely searched, one must decide the order in which the neighborhood is explored. For instance, the first-improvement descent technique moves to any improving neighbor. If there are several improving neighbors, the descent search picks the "first" one encountered in the order the neighbors are examined. To allow such a method to increase its search diversity, a random order may be preferred (Papadimitriou and Steiglitz, 1982).

3.1.4 Combination of neighborhoods

When different neighborhoods are available, it is interesting to consider combined use of multiple neighborhoods. As an example, consider two neighborhoods N_1 and N_2. Then one can envisage at least three ways to use them jointly.

First, *neighborhood union* $N_1 \cup N_2$ includes all the neighboring solutions of the two underlying neighborhoods. A local search algorithm using the union neighborhood selects the next neighboring solution among the solutions in $N_1 \cup N_2$. This combination has no sense if one neighborhood is fully included in the other one.

With *Probabilistic neighborhood union* $N_1 \oslash N_2$, a neighboring solution in N_1 (resp. N_2) belongs to $N_1 \oslash N_2$ with probability p (resp. probability 1-p). A local search algorithm using this combined neighborhood selects at each iteration the next neighbor from N_1 with probability p and from N_2 with probability 1-p.

Token-ring combination $N_1 \rightarrow N_2$ is time-dependent and defined alternatively either by N_1 or N_2 according to some pre-defined conditions (Di Gaspero and Schaerf, 2006). A local search algorithm using the token-ring neighborhood alternates between N_1 and N_2. It typically starts with one neighborhood until the search stagnates, then changes to the other neighborhood until the search stagnates again to switch back to the first neighborhood and so on.

The advantage of combined neighborhood was already demonstrated long time ago for solving the Traveling Salesman Problem (Lin and Kernighan, 1973). More generally, the issue of transitioning among alternative neighborhoods was discussed with the Tabu Search framework and strategic oscillation design in (Glover, 1997a). More recent examples of local search methods focusing on multiple neighborhoods include Variable Neighborhood Search (Hansen and Mladenovic, 1999), Neighborhood Portfolio Search (Di Gaspero and Schaerf, 2006) and Progressive Neighborhood Search (Goëffon et al., 2008). Examples of studies on neighborhood combinations can be found in (Hamiez et al., 2009; Jin and Hao, 2015; Li and Hao, 2022; Li et al., 2020).

3.2 Design of combination operator

3.2.1 Solution combination

Combination is another key component of a MA and constitutes one leading force to explore the search space. The basic idea of combination is very appealing since it provides a very general way of generating new solutions by mixing existing solutions. Contrary to local changes of local improvement, combination can bring into new solutions more useful information, that may be beneficial for a healthy evolution of the search process.

As a first step, it would be tempting to consider a blind (random) crossover for solution combination. Doing this has the advantage of ease of application. However, one question should be asked: Is the blind crossover meaningful with respect to the optimization objective? If the answer is negative, the blind crossover is probably not appropriate and plays a simple role of random diversification.

Thus, it is often useful to consider dedicated combination operators that have strong "semantics" with respect to the optimization objective. A semantic crossover aims to pass intrinsic building blocks (interesting patterns or characteristics) from parents to offspring (Holland, 1975; Radcliffe, 1991). The design of such a crossover is far from trivial and represents a challenging issue. Although there is theoretical guidance, the discovery of a semantic crossover in practice relies basically on a deep analysis and understanding of the given problem. Compared to the design of local search procedures, the design of a meaningful crossover constitutes probably one of the most creative parts of an effective MA.

To design a meaningful crossover, it is necessary to identify what are the building blocks of solutions that can be assembled and inherited through the recombination process. Unfortunately, there is no short-cut to this quest and a fine analysis and deep understanding of the given problem are indispensable to find useful clues.

First, one can analyze the samples of optimal or high-quality solutions to possibly identify regular patterns shared by these solutions. Indeed, if such a pattern exists, then the combination operator can conserve the pattern from the parents. Alternatively, the combination operator can also be encouraged to promote the emergence of favorable building blocks. For instance, such an analysis applied to the Traveling Salesman Problem shows that high-quality local optima share sub-tours (Lin, 1965; Lin and Kernighan, 1973). This property has been used by several highly successful crossover operators which conserve common edges or sub-tours in offspring solutions (Whitley et al., 1989; Nagata and Kobayashi, 1997). Similarly, for the graph coloring problem, an analysis of coloring solutions discloses that some nodes are always grouped to the same color class (i.e. colored with the same color). This characteristic has helped to devise powerful combination operators, as shown in (Dorne and Hao, 1998; Galinier and Hao, 1999) and in (Galinier et al., 2008; Malaguti et al., 2008; Lü and Hao, 2010; Porumbel et al., 2010) with multi-parents.

3.2.2 Multi-parent combination

Combination may operate with more than two parents. Multiple parent combination is even a general rule for the Scatter Search metaheuristic which uses, in its original form, linear combinations of several solutions to create new solutions (Glover, 1977). Although there are no theoretical justifications, the practical advantage of multiple parent recombinations was demonstrated in several occasions for discrete optimization. For instance, for the graph k-coloring problem, several recent and top-performing algorithms integrate multiple parent combination (Galinier et al., 2008; Malaguti et al., 2008; Lü and Hao, 2010; Porumbel et al., 2010), where color classes from different solutions are assembled to build offspring colorings. More generally, when multiple solutions are used for creating a new solution, one can define special rules to score the solution components of each parent solution and use strategic voting rules to combine components from different parents solutions.

A question that arises for multi-parent combination is how to determine the number of the parents. By using two parents, the offspring is expected to inherit 50% material from each parent. The contribution of each parent to the new solution decreases with an increasing number of parents. If the building blocks from different parents are independent from one another, taking more parents into account would be interesting to build good and diversified offspring. Otherwise, if a building block from a parent is epistatic with respect to the building blocks of other parents, blending more parents means more disruption, and thus should be avoided.

3.3 Population diversity management

Population diversity is another important issue that should be considered in an effective MA (Galinier and Hao, 1999; Sörensen and Sevaux, 2006; Porumbel et al., 2010). If the population diversity is not properly managed, the algorithm will converge prematurely to poor optima. This is particularly true when a small population is used. The goal of population diversity is to help the search process not only to avoid premature convergence, but also to continually discover new promising areas.

Population diversity can be measured by a similarity (or distance) metric applied to the members of the population. The metric can be defined either on the solution representation level (genotype metric) or solution level (phenotype metric) (Goldberg, 1989). For instance, pair-wise *Hamming distance* can be used as a genotype metric to measure population diversity. Diversity can also be measured in terms of *entropy* (Fleurent and Ferland, 1996) or by the so-called *moment of inertia* (Morrison and Jong, 2002). Genotype metric is usually problem independent, and thus may or may not reflect the intrinsic diversity of a population with respect to the given optimization objective.

Population diversity can be promoted and managed at several levels of a MA. One evident possibility is to define specific selection rules to favor the selection of *distanced parents* for mating. Another possibility concerns the variation operators

which can be designed in such a way that they favors the generation of divers and varied offspring. More generally, the path-relinking type of combinations typically construct offspring solutions by considering both the solution quality and its distance to its parent solutions (Glover et al., 2000) (see also (Lü et al., 2010) for an example).

Population diversity can also be controlled by the offspring acceptation and replacement strategies. Specifically, this can be done according to both solution diversity and quality. For instance, in Porumbel et al. (2010) a minimum diversity-quality threshold is imposed between the solutions of the population. The acceptation of a new offspring is conditioned not only by its quality, but also by its distance to existing solutions. Similarly, diversity and quality are considered to select the victim solution to be replaced by the offspring.

Other useful ideas for diversity preservation can be found in the areas of Genetic Algorithms. Well-known examples include sharing (Goldberg and Richardson, 1987) and crowding (DeJong, 1975; Mahfoud, 1992).

3.4 Other issues

In addition to the components mentioned until now, the design of an effective Memetic Algorithm should take into account a number of other considerations which are briefly discussed in this Section.

- *Initial population*: There are basically two ways to obtain an initial population: Random generation and constructive elaboration. While random generation is easy to apply, it can hardly generate initial solutions of good quality. An alternative is to use a fast greedy procedure to create the initial population. In this case, the greedy procedure must be randomized such that each application leads to a different solution. Another issue for the initialization stage is to take care of building a diversified population. This can be achieved by controlling the distance between each new solution and the existing solutions of the population.

- *Constraints*: The constraints in the considered problem may influence the design of some MA components. For instance, suppose that the MA algorithm is expected to explore only feasible solutions. Then the local search and combination operators must generate only feasible solutions. If the algorithm explores both feasible and infeasible solutions, it is necessary to define an extended (e.g., pelnalty-based) evaluation function to assess the fitness of an infeasible solution.

- *Connections with Scatter Search and Path Relinking*: The MA framework shares ideas with Scatter Search and Path Relinking (Glover, 1997b; Glover et al., 2000). These latter methods provide unifying principles for joining solutions based on generalized path constructions. In Scatter Search, dispersed new solutions are created from a set of reference solutions by weighted combinations of subsets of the reference solutions that are selected as elite solutions. With Path Relinking, offspring solutions are generated by exploring, within a neighborhood space,

trajectories that connect two or more reference solutions. One notices that the reference solutions or subsets of them can be considered as parent solutions for combination while combination resorts to diverse strategies such as attribute voting and weighting.

4 Memetic algorithms for diversity and dispersion problems

In this section, we review representative memetic algorithms for three diversity and dispersion problems: the maximum diversity problem, the max-mean dispersion problem, and the generalized max-mean dispersion problem.

4.1 Memetic algorithms for the maximum diversity problem

Given a set $V = \{e_1, e_2, \ldots, e_n\}$ of n elements, a distance matrix $[d_{ij}]_{n\times n}$ between the n elements, and a positive integer m ($m < n$), the maximum diversity problem (MDP) consists of selecting a subset M of cardinality m from V, such that the sum of distances between elements in M is maximized. Formally, the MDP can be formulated as a quadratic 0-1 integer program (Kuby, 1987):

$$Maximize \quad f(x) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij} x_i x_j \tag{1}$$

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

where x_i is a binary variable which takes 1 if e_i is selected to be a member of M and 0 otherwise.

The MDP is shown to be an NP-hard problem with a number of practical applications (Ghosh, 1996), such as location, ecological systems, medical treatment, genetics, product design, immigration and admission policies, committee formation, curriculum design (Katayama and Narihisa, 2005; Martí et al., 2013). Due to its practical importance and the NP-hard feature, the MDP has received a lot of attention, and a number of heuristic algorithms have been proposed in the literature, including trajectory-based local search algorithms and population-based memetic algorithms. We focus on the best-performing memetic algorithms.

4.1.1 Memetic algorithms for the maximum diversity problem

Wu and Hao (2013) proposed the first memetic algorithm (denoted by MAMDP) for the maximum diversity problem, whose general procedure is given in Algorithm

Input: The set V of n elements, and distance matrix $[d_{ij}]_{n\times n}$, cardinality of subset m, the population size p;
Output: The best solution s^* found;
Initialize population $POP = \{s^1, \ldots, s^p\}$;
$s^* \leftarrow Best(POP)$;// s^* records the best solution found so far
while *The stopping condition is not met* **do**
  Randomly select two parent solutions s^1 and s^2 from POP;
  $s^o \leftarrow Crossover(s^1, s^2)$;
  $s^o \leftarrow TabuSearch(s^o)$// Local optimization
  if $f(s^o) > f(s^*)$ **then**
    $s^* \leftarrow s^o$; // Update best solution S^*
  end
  $POP \leftarrow PopulationUpdate(s^o, POP)$; // Update population
end
return s^*

Algorithm 2: Memetic Algorithm for the maximum diversity problem

2. The algorithm is composed of a tabu search procedure for local optimization, a crossover for new offspring generation, and a quality-and-distance population updating strategy. Starting from an initial population, the algorithm performs a number of generations until a stopping condition is met. At each generation, two parent solutions s^1 and s^2 are selected from the population and recombined to generate an offspring solution s^o, which is improved by the tabu search procedure and then used to update the population.

The crossover generates, from two parent solutions s^1 and s^2 which have respectively M^1 and M^2 as the sets of selected elements, its offspring solution s^o with M^o as the set of selected elements in two steps. It first retains all common elements of M^1 and M^2 as the elements of offspring solution M^0, i.e., $M^0 \leftarrow M^1 \cap M^2$. Then, the remaining elements of the offspring are alternately selected from $M^1 \setminus M^0$ and $M^2 \setminus M^0$ until the size of M^0 reaches m. This is a cardinality-constrained uniform crossover.

The tabu search based local optimization is the key component of the MAMDP algorithm. It features a constrained swap neighborhood as well as a fast neighborhood evaluation technique. For the MDP, the swap neighborhood N_{swap} defined by the swap move (u, v) is very popular in the literature, which has the advantage of maintaining the feasibility of neighboring solutions. Given a candidate solution M (i.e., the set of selected elements), the swap operation (u, v) exchanges a selected element $e_u \in M$ against a non-selected element $e_v \in V \setminus M$. Let $S \oplus (u, v)$ denote the neighboring solution after applying the swap move (u, v) to solution M. Then the neighborhood $N_{swap}(M)$ is given by $N_{swap}(M) = \{M \oplus (u, v) : e_u \in M, e_v \in V \setminus M\}$, whose size is $m \times (n - m)$.

To reduce this neighborhood, a constrained swap neighborhood CN is introduced, which is a much smaller subset of N_{swap}. The idea of the constrained neighborhood CN is to limit the swap moves to two specific subsets $X \subset M$ and $Y \subset V \setminus M$, while the resulting constrained neighborhood contains the best solution in the neighborhood N_{swap}. To construct X and Y, a "potential" vector $P = (p_1, p_2, \ldots, p_n)$ (i.e., Δ-

table) is defined for the current solution M, where p_i is the potential of element e_i $(1 \leq i \leq n)$ in the current solution and is calculated as $p_i = \sum_{e_j \in M} d_{ij}$. Let $d'_{min} = min\{p_i : e_i \in M\}$ and $d'_{max} = max\{p_i : e_i \in V \setminus M\}$, X is defined as $X = \{e_i \in M : p_i \leq d'_{min} + D_{max}\}$ and Y is defined as $Y = \{e_i \in V \setminus M : p_i \geq d'_{max} - D_{max}\}$, where D_{max} is the maximum distance in the distance matrix $[d_{ij}]_{n \times n}$. Then, the constrained neighborhood $CN(M)$ is defined as:

$$CN(M) = \{M' : M' = M \setminus \{u\} \cup \{v\}, u \in X, v \in Y\} \tag{3}$$

The constrained neighborhood CN is in general much smaller than N_{swap}.

To efficiently evaluate the neighboring solutions, the tabu search method employs a fast neighborhood evaluation technique. Given the current solution M and its potential vector $P = (p_1, p_2, \ldots, p_n)$, the move gain Δ_{uv} of a swap (u, v) (i.e., variation in the objective value) can be calculated by $\Delta_{uv} = p_v - p_u - d_{uv}$ in $O(1)$, and the vector P (i.e., Δ-table) is updated in $O(n)$ after the swap (u, v) as follows:

$$p_i = \begin{cases} p_i + d_{iv}, & \text{for } e_i = e_v; \quad (4) \\ p_i - d_{iu}, & \text{for } e_i = e_u; \quad (5) \\ p_i + d_{iv} - d_{iu}, & otherwise; \quad (6) \end{cases}$$

To avoid premature convergence, the MAMDP algorithm tries to maintain a healthy population during the search. This is achieved by adopting a quality-and-distance updating strategy, which simultaneously considers the quality and the diversity of population, to update the population. Concretely, given an offspring solution M^o and the current population POP, the offspring is first inserted into the population (i.e., $POP \leftarrow POP \cup \{M\}$), and then a quality-and-distance scoring function is used to identify the worst solution M^w. Subsequently, the worst solution M^w is deleted from the population ($POP \leftarrow POP \setminus \{M^w\}$).

Computational experiments on benchmark instances commonly used in the literature show that the MAMDP algorithm is very competitive compared with the previous algorithms.

In addition to the MAMDP algorithm, other memetic algorithms were proposed by further improving the tabu search method and devising other crossover operators and population management strategies. Wang et al. (2014) presented a memetic algorithm (TS/MA) for the MDP, which differs from the MAMDP algorithm mainly by its population updating strategy and the local search method. First, Wang et al. devised a new tabu search method in which the size of neighborhood is controlled by a parameter cls called the candidate list size. By controlling the value of cls, the neighborhood is limited to a high-quality subset of N_{swap}. Moreover, a swap operation (u, v) is performed in a two-phase way. That is, the swap move (u, v) is subdivided into two successive component operations: (1) move e_u from M to $V \setminus M$, (2) move e_v from $V \setminus M$ to M. Second, the algorithm employs a greedy population updating strategy, i.e., the offspring M^o is used to replace the worst solution M^w if M^o differs from any solution in the population and is superior to M^w in the objective value. On the other hand, to enhance its diversification ability, the TS/MA algorithm

rebuilds the population by perturbing randomly the population individuals when the search stagnates, while keeping the best solution found so far M^* in the new population.

Zhou et al. (2017) proposed an opposition-based memetic algorithm (OBMA) for the MDP by introducing the concept of opposition-based learning into the memetic algorithm framework. OBMA brings several improvements into the basic memetic algorithm framework. First, OBMA employs an opposition-based learning (OBL) strategy to reinforce population initialization as well as the evolutionary search process, by simultaneously considering a candidate solution and its an opposite solution. Specifically, in the population initialization, a random solution and its an opposite solution are simultaneously generated and locally optimized, and the best between them is inserted into the population, this process is repeated until the number of individuals reaches the size of population p. Moreover, at each generation of the evolutionary search process, a crossover operator is used to generate an offspring solution as well as its an opposite solution, and the two offspring solutions are then optimized by a local search procedure. Second, OBMA employs an improved tabu search method which relies on a constrained neighborhood as its local search procedure, where the size of the constrained neighborhood is controlled by a parameter α. Third, OBMA employs a rank-based quality-and-distance population updating strategy to maintain a healthy population, where the quality of an individual is measured by its rankings with respect to the objective value and the average distance to the population.

Lai et al. (2018) introduced a diversification-driven memetic algorithm for the MDP. To enhance its diversification ability, the proposed algorithm adds dynamically a number of randomly generated new solutions into the population during the search process. Specifically, at each generation of the evolutionary search process, the algorithm generates randomly an initial solution with a probability of β and improves it by a tabu search method. Then, the improved offspring solution is used to update the population by a quality-and-distance updating strategy. Experimental results show that such a strategy enhances significantly the population diversity and the searching ability of the memetic algorithm.

Liu et al. (2020) proposed a two-phase tabu search based evolutionary algorithm (TPTS/EA) for the MDP, which uses only two individuals in the population, like the HEAD algorithm for graph coloring (Moalic and Gondran, 2018). TPTS/EA uses two path-relinking based combination operators to generate the offspring solutions and a two-phase tabu search method as its local search procedure. To reach a good tradeoff between diversification and intensification, the tabu search method consists of a solution-based tabu search procedure and an attribute-based tabu search procedure. In comparison, the solution-based tabu search has a stronger intensification ability and the attribute-based tabu search has a stronger diversification ability. Moreover, to further enhance its diversification ability, the TPTS/EA algorithm generates randomly new initial solutions as parent solutions when the search stagnates, which shares the same spirit as (Lai et al., 2018).

In summary, these memetic algorithms for the MDP share several common characteristics, including efficient tabu search methods with a small-sized and focused

neighborhood, fast neighborhood evaluation techniques, solution recombination operators, and efficient population management strategies considering both the quality of solutions and the diversity of population. Indeed, these carefully designed algorithmic components ensure together the high performances of these algorithms for the MDP.

4.1.2 Computational results and comparison

Table 1 Comparison between four memetic algorithms in the literature. The computational results of algorithms are taken from the literature (Liu et al., 2020)

		$BKR - f_{best}$				$BKR - f_{avg}$			
Instance	BKR	MAMDP	TS/MA	OBMA	TPTS/EA	MAMDP	TS/MA	OBMA	TPTS/EA
MDG-a_21	114271	**0**	**0**	**0**	**0**	8.1	10.4	5.2	**0.5**
MDG-a_22	114327	**0**	**0**	**0**	**0**	8.8	8.8	0.1	**0**
MDG-a_23	114195	**0**	**0**	**0**	**0**	15.2	8.5	15.2	**7**
MDG-a_24	114093	**0**	**0**	**0**	**0**	15.7	19.9	11.2	**5.3**
MDG-a_25	114196	**0**	**0**	**0**	**0**	42.1	**29.5**	41.5	30.6
MDG-a_26	114265	**0**	**0**	**0**	**0**	10.8	15.6	10.2	**2.6**
MDG-a_27	114361	**0**	**0**	**0**	**0**	**0**	**0**	0.2	**0**
MDG-a_28	114327	**0**	**0**	**0**	**0**	20.9	25.2	18.9	**0.1**
MDG-a_29	114199	**0**	**0**	**0**	**0**	7.6	7.8	4.4	**0.8**
MDG-a_30	114229	**0**	**0**	**0**	**0**	9.3	4.1	8.1	**0**
MDG-a_31	114214	**0**	**0**	**0**	**0**	17.8	24.3	16.7	**9**
MDG-a_32	114214	**0**	**0**	**0**	**0**	26.8	21.5	23.7	**12**
MDG-a_33	114233	**0**	**0**	**0**	**0**	3.6	1.2	2	**0**
MDG-a_34	114216	**0**	**0**	**0**	**0**	3.4	3.6	2.4	**0.1**
MDG-a_35	114240	**0**	**0**	**0**	**0**	**1.2**	1.7	1.6	1.7
MDG-a_36	114335	**0**	**0**	**0**	**0**	8.6	7.3	5.7	**3.6**
MDG-a_37	114255	**0**	**0**	**0**	**0**	6.5	11.7	5.2	**0**
MDG-a_38	114408	**0**	**0**	**0**	**0**	0.7	1	**0.5**	0.6
MDG-a_39	114201	**0**	**0**	**0**	**0**	3.4	4	2	**0**
MDG-a_40	114349	**0**	**0**	**0**	**0**	24.1	**15.6**	23	19.9
b2500-1	1153068	**0**	**0**	**0**	**0**	72.1	**0**	**0**	**0**
b2500-2	1129310	**0**	**0**	**0**	**0**	143.7	73,9	37.9	**30.9**
b2500-3	1115538	**0**	**0**	**0**	**0**	184.5	184.7	**0.4**	92.7
b2500-4	1148012	172	172	172	**0**	324.3	331	237.8	**126.5**
b2500-5	1144756	**0**	**0**	**0**	**0**	10.5	45.2	**5.3**	47.4
b2500-6	1133572	**0**	**0**	**0**	**0**	80.5	54.4	**0**	**0**
b2500-7	1149064	**0**	**0**	**0**	**0**	45	65	**14.1**	25
b2500-8	1142762	**0**	**0**	**0**	**0**	1.7	1.2	1.5	**0**
b2500-9	1138866	**0**	**0**	**0**	**0**	3.7	**0**	1.3	**0**
b2500-10	1153936	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
p3000-1	6502330	**0**	**0**	**0**	**0**	76.7	35.1	24.4	**16.4**
p3000-2	18272568	**0**	**0**	**0**	**0**	146.1	**0**	**0**	7.1
p3000-3	29867138	**0**	**0**	**0**	**0**	527.9	**0**	**0**	**0**
p3000-4	46915044	**0**	**0**	**0**	**0**	399.5	1.2	1.2	**0**
p3000-5	58095467	**0**	**0**	**0**	**0**	210.7	2.27	**0**	18.3
p5000-1	17509433	64	64	64	**0**	229.1	96.4	192.2	**66.8**
p5000-2	50103092	21	21	**0**	**0**	475.5	65	22.8	**19.4**
p5000-3	82040316	176	**0**	**0**	**0**	1419	**171.3**	209.3	235
p5000-4	129413710	279	**0**	**0**	**0**	800.9	198.1	**97.8**	141.7
p5000-5	160598156	136	**0**	**0**	**0**	411.9	139.1	**102.9**	138
#Best		34	37	38	40	3	9	12	28

To illustrate the relative performances of these (best) memetic algorithms for the MDP, Table 1 shows their computational results on 40 representative instances

(extracted from Liu et al. (2020)). Columns 1 and 2 indicate the names and the best-known results (BKR) of the instances, columns 3–6 give the gaps between BKR and the best objective value obtained (f_{best}) by these algorithms, and columns 7–10 present the gaps between BKR and the average objective value (f_{avg}). The last row '#Best' shows the number of instances for which the corresponding algorithm obtained the best result among the compared algorithms in terms of f_{best} and f_{avg}. It is observed that improvements are continually made while the latest two-phase tabu search based memetic algorithm (Liu et al., 2020) holds the current best results.

4.2 Memetic algorithm for the max-mean dispersion problems

Given a set of $V = \{e_1, e_2, \ldots, e_n\}$ of n elements and a distance matrix $[d_{ij}]_{n\times n}$ among elements, the max-mean dispersion problem (MaxMeanDP) consists of selecting a subset M from V, such that the mean dispersion among the selected elements is maximized. Formally, the MaxMeanDP problem can be expressed as a fractional 0–1 programming problem:

$$Maximize \quad f(x) = \frac{\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} d_{ij}x_i x_j}{\sum_{i=1}^{i=n} x_i} \tag{7}$$

$$s.t. \sum_{i=1}^{n} x_i \geq 2 \tag{8}$$

where x_i is a binary variable which takes 1 if e_i is selected to be a member of M and 0 otherwise.

The generalized MaxMeanDP problem (GMaxMeanDP) is a generalization of the MaxMeanDP where each element e_i is associated with a weight w_i. Like MaxMeanDP, the GMaxMeanDP can be expressed as a fractional 0–1 programming problem as follows:

$$Maximize \quad f(x) = \frac{\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} d_{ij}x_i x_j}{\sum_{i=1}^{i=n} w_i x_i} \tag{9}$$

$$s.t. \sum_{i=1}^{n} x_i \geq 2 \tag{10}$$

Thus, for both problems, a candidate solution can be conveniently indicated by a 0-1 vector $x = (x_1, x_2, \ldots, x_n)$.

4.2.1 Memetic algorithm

The first memetic algorithm for the MaxMeanDP (denoted by MAMMDP) was introduced in (Lai and Hao, 2016). Later, this algorithm was adapted to the GMaxMe-

anDP in (Lai et al., 2020), where another perturbation-based evolutionary algorithm (PBEA) was also presented. As an example, we focus on the MAMMDP algorithm, whose general scheme is shown in Algorithm 3.

Input: The set V of n elements, the population size p, the timeout limit t_{max};
Output: The best solution s^* found;
while $time() < t_{max}$ **do**
  $POP = \{s^1, \ldots, s^p\} \leftarrow$ PopInitialization(p);
  if *the loop is not performing its first execution* **then**
    $s^w \leftarrow arg\ min\{f(s^i) : i = 1, \ldots, p\}$;
    $POP \leftarrow POP \cup \{s^*\} \setminus \{s^w\}$;
  end
  $s^* \leftarrow arg\ max\{f(s^i) : i = 1, \ldots, p\}$ `// s* keeps the best solution found`
  $PairSet \leftarrow \{(s^i, s^j) : 1 \le i < j \le p\}$;
  while $PairSet \neq \emptyset$ *and* $time() < t_{max}$ **do**
    Randomly pick a solution pair $(s^i, s^j) \in PairSet$;
    $s^o \leftarrow Crossover(s^i, s^j)$ `// uniform crossover`
    $s^o \leftarrow TabuSearch(s^o)$;
    if $f(s^o) > f(s^*)$ **then**
      $s^* \leftarrow s^o$;
    end
    $PairSet \leftarrow PairSet \setminus \{(s^i, s^j)\}$;
    $s^w \leftarrow arg\ min\{f(s^i) : i = 1, \ldots, np\}$;
    if s^o *dose not exist in* POP *and* $f(s^o) > f(s^w)$ **then**
      $POP \leftarrow POP \cup \{s^o\} \setminus \{s^w\}$;
      $PairSet \leftarrow PairSet \setminus \{(s^w, s^k) : s^k \in POP\}$;
      $PairSet \leftarrow PairSet \cup \{(s^o, s^k) : s^k \in POP\}$;
    end
  end
end
return s^*

Algorithm 3: Memetic algorithm for the MaxMeanDP and GMaxMeanDP

In addition to the population, the algorithm uses a reference set ($PairSet$) to contain the solution pairs of the population for crossover operations. At each generation, a solution pair (s^i, s^j) is randomly picked from $PairSet$ and then the crossover operator is applied to the selected solution pair (s^i, s^j) to generate a new solution s^o. After a crossover, the reference $PairSet$ is accordingly updated as follows: 1) The solution pair (s^i, s^j) is removed from $PairSet$; 2) If the offspring solution s^o replaces the worst solution s^w in the population, all the solution pairs containing s^w are removed from $PairSet$ and all possible solution pairs that can be generated by combining s^o with other solutions in the population are added into $PairSet$. This strategy, inspired by the path relinking method (Glover et al., 2000), ensures that every pair of solutions in the population is combined exactly once, favoring a more intensified search. Second, the algorithm employs an efficient tabu search method as its local optimization method. Third, the population is recreated once the reference

set $PairSet$ becomes empty, while preserving the best solution (s^*) found so far in the new population.

Another key feature of the MAMMDP algorithm concerns the tabu search method which employs the 1-flip neighborhood induced by the 1-flip move operator. Given a candidate solution $s = (x_1, x_2, \ldots, x_n)$, the 1-flip operator changes the value of a single variable x_i to its complementary value $1 - x_i$, while keeping the values of other variables unchanged. Thus, the neighborhood $N_1(s)$ can be written as:

$$N_1(s) = \{s \oplus flip(i) | 1 \leq i \leq n\} \tag{11}$$

Compared to the swap neighborhood N_{swap} in Section 4.1.1, the 1-flip neighborhood is of smaller size and has a stronger exploitation ability that can access any region in the solution space.

For an efficient evaluation of candidate neighboring solutions, the tabu search procedure employs a fast neighborhood evaluation technique. This technique maintains a n-dimensional vector $P = (p_1, p_2, \ldots, p_n)$ (i.e., Δ-table) to incrementally calculate the move gain of each possible 1-flip move applicable to the current solution s, where the entry p_i represents the sum of distances between the element i and the selected elements for the current solution s, i.e., $p_i = \sum_{j \in M; j \neq i} d_{ij}$, where M is the set of selected elements. When a 1-flip move is performed by flipping variable x_i as $x_i \leftarrow (1 - x_i)$, the move gain Δ_i can be incrementally calculated as follows:

$$\Delta_i = \begin{cases} \dfrac{-f(s)}{|M|+1} + \dfrac{p_i}{|M|+1}, & \text{for } x_i = 0; \quad (12) \\ \dfrac{f(s)}{|M|-1} - \dfrac{p_i}{|M|-1}, & \text{for } x_i = 1; \quad (13) \end{cases}$$

where $f(s)$ is the objective value of the current solution s and $|M|$ is the number of the selected elements in s. Subsequently, the vector P is accordingly updated as:

$$p_j = \begin{cases} p_j + d_{ij}, & \text{for } x_i = 0, j \neq i; \quad (14) \\ p_j - d_{ij}, & \text{for } x_i = 1, j \neq i; \quad (15) \\ p_j, & \text{for } j = i; \quad (16) \end{cases}$$

For the GMaxMeanDP, the move gain Δ_i is calculated as follows when variable x_i is flipped:

$$\Delta_i = \begin{cases} \dfrac{-f(s)w_i}{SM + w_i} + \dfrac{p_i}{SM + w_i}, & \text{for } x_i = 0; \quad (17) \\ \dfrac{f(s)w_i}{SM - w_i} - \dfrac{p_i}{SM - w_i}, & \text{for } x_i = 1; \quad (18) \end{cases}$$

where $SM = \sum_{i \in M} w_i$ is the sum of vertex weights of the selected elements in s.

The vector P is initialized at the beginning of each call of the tabu search procedure in $O(n^2)$, and is updated in $O(n)$ at each iteration. With the fast evaluation technique,

the best 1-flip move in $N_1(s)$ can be identified in $O(n)$ and thus the total complexity of each iteration of the tabu search method is bounded by $O(n)$.

The tabu search method uses a specific tabu list management strategy, which is based on a periodic step function defined on the number of iterations to periodically adjust the tabu tenure tt. Denote the current iteration by $iter$ and the tabu tenure of the current move by $tt(iter)$, the tabu tenure function is defined by a sequence of values $(a_1, a_2, \cdots, a_q)$ and a sequence of interval margins $(b_1, b_2, \cdots, b_{q+1})$, such that for any $iter$ in $[b_i, b_{i+1} - 1]$ $tt(iter) = a_i + rand(C)$, where $rand(C)$ denotes a random integer between 0 to C. The value of q is set to 15, and $(a)_{i=1,\cdots,15} = \frac{T_{max}}{8} \times (1, 2, 1, 4, 1, 2, 1, 8, 1, 2, 1, 4, 1, 2, 1)$, where T_{max} is a parameter that is used to control the maximum tabu tenure. The interval margins are then defined by $b_1 = 1$, $b_{i+1} = b_i + 5a_i$ $(i \leq 15)$. In principle, a small tabu tenure leads usually to strong search intensification while a large tabu tenure favors search diversification. Thus, the periodical change of the tabu tenure among several small and large values provides a strategy to reach a desirable balance between intensification and diversification of the search.

Finally, as recombination operator, the uniform crossover is used for offspring generation. Given two selected parent solutions $s^i = (x_1^{i1}, x_2^i, \ldots, x_n^i)$ and $s^2 = (x_1^j, x_2^j, \ldots, x_n^j)$, the value of each component x_i^o $(i = 1, 2, \ldots, n)$ of the offspring solution s^o is randomly chosen from the set $\{x_i^1, x_i^2\}$ with an equal probability of 0.5.

4.2.2 Computational results and comparison

To illustrate the related performance of the MAMMDP algorithm on the MaxMeanDP, Tables 2 - 4 summarize the results of MAMMDP on benchmark instances with $n = 500, 750, 1000, 1500, 2000$ elements compared to other best MaxMeanDP algorithms: a GRASP and path relinking approach (GRASP-PR) (Martí and Sandoya, 2013), a two-phased hybrid heuristic (2PHA) (Della Croce et al., 2014), a three-phase hybrid approach (3PHA) (Della Croce et al., 2016), and a two-phase tabu search (TP-TS) method (Carrasco et al., 2015), and the perturbation-based evolutionary algorithm (PBEA) (Lai et al., 2020), including the best objective value over 20 independent runs (f_{best}), the average objective value (f_{avg}), the success rate (SR), and the average computational time $t(s)$ in seconds. Tables 2 and 3 show that for the instance with $n \leq 1000$, the MAMMDP algorithm dominates other algorithms both in terms of solution quality and computational efficiency. For the largest instances with $n = 1500, 2000$, Table 4 indicates that the two evolutionary algorithms MAMMDP and PBEA dominates the variable neighborhood search algorithm (VNS) in (Brimberg et al., 2017), while MAMMDP and PBEA perform quite similarly on these very large instances.

As to the GMaxMeanDP problem, Table 5 shows the computational results of MAMMDP (Lai and Hao, 2016) and PBEA (Lai et al., 2020) on 40 large-scale instances with $n = 3000$. The results indicate that the MAMMDP and PBEA algorithms have a comparable performance in terms of computational results and efficiency.

Table 2 Computational results of best MaxMeanDP algorithms on the set of 20 representative instances with $n = 500$. Each instance is independently solved 20 times by MAMMDP, and dominating results are indicated in bold compared to the previous best-known results f_{pre} in the literature.

							MAMMDP			
Instance	n	f_{pre}	GRASP-PR	2PHA	3PHA	TP-TS	f_{best}	f_{avg}	SR	$t(s)$
MDPI1_500	500	81.28	78.6050000	81.25	81.28	81.28	81.277044	81.277044	20/20	0.69
MDPI2_500	500	77.79	76.8734667	77.45	77.79	77.60	**78.610216**	**78.610216**	20/20	1.43
MDPI3_500	500	76.30	75.6914063	75.31	76.30	75.65	76.300787	76.300787	20/20	2.71
MDPI4_500	500	82.33	81.8058434	82.28	82.33	81.47	82.332081	82.332081	20/20	0.95
MDPI5_500	500	80.08	78.5695714	80.01	80.08	79.92	**80.354029**	**80.354029**	20/20	2.80
MDPI6_500	500	81.25	79.6426282	81.12	81.25	79.93	81.248553	81.248553	20/20	0.78
MDPI7_500	500	78.16	75.4989726	78.09	78.16	77.71	78.164511	78.164511	20/20	0.92
MDPI8_500	500	79.06	76.9836424	79.01	79.06	78.70	**79.139881**	**79.139881**	20/20	1.27
MDPI9_500	500	77.36	75.7209449	76.98	77.36	77.15	**77.421000**	**77.421000**	20/20	2.37
MDPI10_500	500	81.25	80.3789051	81.24	81.25	81.02	**81.309871**	**81.309871**	20/20	0.91
MDPII1_500	500	109.38	108.152545	109.16	109.38	109.33	**109.610136**	**109.610136**	20/20	0.75
MDPII2_500	500	105.33	103.287851	105.06	105.33	104.81	**105.717536**	**105.717536**	20/20	0.88
MDPII3_500	500	107.79	106.301714	107.64	107.79	107.18	**107.821739**	**107.821739**	20/20	0.89
MDPII4_500	500	106.10	104.618442	105.37	106.10	105.69	106.100071	106.100071	20/20	0.56
MDPII5_500	500	106.59	103.608188	106.37	106.55	106.59	**106.857162**	**106.857162**	20/20	0.99
MDPII6_500	500	106.17	104.813987	105.52	105.77	106.17	**106.297958**	**106.297958**	20/20	0.98
MDPII7_500	500	107.06	104.503378	106.61	107.06	106.92	**107.149379**	**107.149379**	20/20	0.88
MDPII8_500	500	103.78	100.021407	103.41	103.78	103.49	103.779195	103.779195	20/20	0.59
MDPII9_500	500	106.24	104.927769	106.20	106.24	105.97	**106.619793**	**106.619793**	20/20	1.11
MDPII10_500	500	104.15	103.497014	103.79	104.15	103.56	**104.651507**	**104.651507**	20/20	1.01
#Better							13	13		
#Equal							7	7		
#Worse							0	0		

4.3 Discussions

In this section, we discuss the main features of the top-performing memetic algorithms for the three diversity and dispersion problems.

First, like for many combinatorial search problems, tabu search methods with a small-sized neighborhood and a fast neighborhood evaluation technique play the key role of search intensification (Wu and Hao, 2013; Wang et al., 2014; Lai et al., 2018; Lai and Hao, 2016; Lai et al., 2020; Liu et al., 2020). Moreover, a combined use of solution-based and attribute-based tabu searches can further raise the search capacity of the algorithm (Liu et al., 2020). This strategy creates a better balance between diversification and intensification and helps to better solve some very difficult instances.

Second, crossover is another contributing search component. In particular, the uniform crossover (Lai and Hao, 2016; Lai et al., 2020) and the backbone-based crossover (Wu and Hao, 2013; Zhou et al., 2017) allow the offspring solution to inherit the favorable features of elite parent solutions, and contribute to the discovery of promising new regions that are difficult to attain by local search methods.

Finally, as for the population management, the popular quality-based strategy that replaces the worst solution of the population by a better offspring solution has the main advantage of simplicity (Wang et al., 2014; Lai and Hao, 2016; Lai et al., 2020). More advanced methods such as rank-based update (Zhou et al., 2017) and

Table 3 Computational results of MAMMDP (Lai and Hao, 2016) and TS (Carrasco et al., 2015) on the set of 40 representative instances with $750 \leq n \leq 1000$. Each instance is independently solved 20 times by MAMMDP, and improved results are indicated in bold.

			MAMMDP			
Instance	n	TS	f_{best}	f_{avg}	SR	$t(s)$
MDPI1_750	750	95.86	**96.650699**	**96.650699**	20/20	4.31
MDPI2_750	750	97.42	**97.564880**	**97.564880**	20/20	3.82
MDPI3_750	750	96.97	**97.798864**	**97.798864**	20/20	1.81
MDPI4_750	750	95.21	**96.041364**	**96.041364**	20/20	4.38
MDPI5_750	750	96.65	**96.761928**	**96.761928**	20/20	0.65
MDPI6_750	750	99.25	**99.861250**	**99.861250**	20/20	5.55
MDPI7_750	750	96.26	**96.545413**	**96.545413**	20/20	1.01
MDPI8_750	750	96.46	**96.726976**	**96.726976**	20/20	1.73
MDPI9_750	750	96.78	**98.058377**	**98.058377**	20/20	2.18
MDPI10_750	750	99.85	**100.064185**	**100.064185**	20/20	3.42
MDPII1_750	750	127.69	**128.863707**	**128.863707**	20/20	5.66
MDPII2_750	750	130.79	**130.954426**	**130.954426**	20/20	2.31
MDPII3_750	750	129.40	**129.782453**	**129.782453**	20/20	11.64
MDPII4_750	750	125.68	**126.582271**	**126.582271**	20/20	1.48
MDPII5_750	750	128.13	**129.122878**	**129.122878**	20/20	1.32
MDPII6_750	750	128.55	**129.025215**	**129.025215**	20/20	7.98
MDPII7_750	750	124.91	**125.646682**	**125.646682**	20/20	3.38
MDPII8_750	750	130.66	**130.940548**	**130.940548**	20/20	1.91
MDPII9_750	750	128.89	**128.889908**	**128.889908**	20/20	1.30
MDPII10_750	750	132.99	**133.265300**	**133.265300**	20/20	1.81
MDPI1_1000	1000	118.76	**119.174112**	**119.174112**	20/20	8.25
MDPI2_1000	1000	113.22	**113.524795**	**113.524795**	20/20	3.52
MDPI3_1000	1000	114.51	**115.138638**	**115.138638**	20/20	2.32
MDPI4_1000	1000	110.53	**111.150397**	**111.150397**	20/20	3.58
MDPI5_1000	1000	111.24	**112.723188**	**112.723188**	20/20	1.61
MDPI6_1000	1000	112.08	**113.198718**	**113.198718**	20/20	7.72
MDPI7_1000	1000	110.94	**111.555536**	**111.555536**	20/20	1.88
MDPI8_1000	1000	110.29	**111.263194**	**111.263194**	20/20	3.55
MDPI9_1000	1000	115.78	**115.958833**	**115.958833**	20/20	2.38
MDPI10_1000	1000	114.29	**114.731644**	**114.731644**	20/20	2.16
MDPII1_1000	1000	145.46	**147.936175**	**147.936175**	20/20	1.60
MDPII2_1000	1000	150.49	**151.380035**	**151.380035**	20/20	1.78
MDPII3_1000	1000	149.36	**150.788178**	**150.788178**	20/20	4.92
MDPII4_1000	1000	147.91	**149.178006**	**149.178006**	20/20	3.80
MDPII5_1000	1000	150.23	**151.520847**	**151.520847**	20/20	3.28
MDPII6_1000	1000	147.29	**148.343378**	**148.343378**	20/20	3.22
MDPII7_1000	1000	148.41	**148.742375**	**148.742375**	20/20	6.30
MDPII8_1000	1000	145.87	**147.826804**	**147.826804**	20/20	13.52
MDPII9_1000	1000	145.67	**147.083880**	**147.083880**	20/20	3.83
MDPII10_1000	1000	148.40	**150.046137**	**150.046137**	20/20	2.13
#Better			40	40		
#Equal			0	0		
#Worse			0	0		

quality-and-distance based update (Wu and Hao, 2013; Lai et al., 2018) are more likely to better preserve population diversity and prevent premature convergence.

5 Conclusions

Memetic algorithms are a category of hybrid evolutionary algorithms that combine single trajectory local search and population based evolutionary search. Memetic algorithms are quite successful for a number of difficult combinatorial optimiza-

Table 4 Comparison between MAMMDP (Lai and Hao, 2016) and PBEA (Lai et al., 2020) on the 40 MaxMeanDP instances with n = 1500 or 2000 from the literature. Each instance was independently solved 20 times using both algorithms respectively, and the improved results are indicated in bold in terms of f_{best}.

		MAMMDP				PBEA			
Instance	VNS	f_{best}	f_{avg}	SR	$t(s)$	f_{best}	f_{avg}	SR	$t(s)$
MDPI1_1500	136.26	**136.535222**	136.535222	20/20	14.21	**136.535222**	136.535222	20/20	18.75
MDPI2_1500	138.00	**138.341482**	138.341482	20/20	5.38	**138.341482**	138.341482	20/20	8.04
MDPI3_1500	138.91	**139.200599**	139.200599	20/20	3.17	**139.200599**	139.200599	20/20	3.28
MDPI4_1500	139.81	**140.166920**	140.166920	20/20	5.65	**140.166920**	140.166920	20/20	4.67
MDPI5_1500	136.47	**137.129630**	137.129630	20/20	7.73	**137.129630**	137.129630	20/20	12.65
MDPI6_1500	136.22	**136.508768**	136.508768	20/20	7.05	**136.508768**	136.508768	20/20	10.13
MDPI7_1500	137.65	**137.971032**	137.971032	20/20	2.49	**137.971032**	137.971032	20/20	3.20
MDPI8_1500	138.02	**138.728444**	138.728444	20/20	13.56	**138.728444**	138.728444	20/20	13.95
MDPI9_1500	136.30	**136.495674**	136.495674	20/20	21.39	**136.495674**	136.495674	20/20	28.95
MDPI10_1500	140.33	140.333159	140.333159	20/20	3.47	140.333159	140.333159	20/20	3.90
MDPI1_2000	158.03	**158.588217**	158.588217	20/20	10.40	**158.588217**	158.588217	20/20	11.79
MDPI2_2000	162.91	**163.939616**	163.939616	20/20	19.11	**163.939616**	163.939616	20/20	31.71
MDPI3_2000	158.98	**159.570786**	159.545090	13/20	39.86	**159.570786**	159.528479	6/20	38.94
MDPI4_2000	159.14	**160.185217**	160.185217	20/20	28.46	**160.185217**	160.184761	17/20	54.41
MDPI5_2000	156.11	**156.805331**	156.758147	10/20	41.25	**156.805331**	156.776147	13/20	55.30
MDPI6_2000	161.61	**161.839100**	161.839100	20/20	11.30	**161.839100**	161.839100	20/20	13.72
MDPI7_2000	157.58	**158.336131**	158.336131	20/20	9.79	**158.336131**	158.336131	20/20	7.93
MDPI8_2000	161.43	**161.446931**	161.446931	20/20	20.03	**161.446931**	161.446931	20/20	22.30
MDPI9_2000	159.15	**160.190374**	160.190374	20/20	29.21	**160.190374**	160.187769	17/20	44.28
MDPI10_2000	160.90	**161.638099**	161.638099	20/20	7.60	**161.638099**	161.638099	20/20	4.99
MDPII1_1500	181.67	**182.089413**	182.089413	20/20	6.33	**182.089413**	182.089413	20/20	8.23
MDPII2_1500	185.48	**186.243869**	186.243869	20/20	6.78	**186.243869**	186.243869	20/20	4.66
MDPII3_1500	181.55	**182.142902**	182.142902	20/20	3.13	**182.142902**	182.142902	20/20	4.76
MDPII4_1500	184.91	**185.557302**	185.500190	8/20	42.93	**185.557302**	185.514675	9/20	35.93
MDPII5_1500	190.15	**190.860529**	190.860529	20/20	2.25	**190.860529**	190.860529	20/20	1.65
MDPII6_1500	183.14	**183.575336**	183.575336	20/20	3.05	**183.575336**	183.575336	20/20	1.90
MDPII7_1500	179.34	**179.820242**	179.820242	20/20	13.93	**179.820242**	179.820242	20/20	18.43
MDPII8_1500	186.60	186.602804	186.602804	20/20	2.74	186.602804	186.602804	20/20	3.30
MDPII9_1500	181.43	**181.918814**	181.918814	20/20	17.85	**181.918814**	181.918814	20/20	14.75
MDPII10_1500	182.70	**183.384692**	183.384692	20/20	32.37	**183.384692**	183.384692	20/20	26.01
MDPII1_2000	208.85	**209.845273**	209.845273	20/20	8.13	**209.845273**	209.845273	20/20	11.24
MDPII2_2000	218.19	**218.404860**	218.404860	20/20	16.03	**218.404860**	218.404860	20/20	22.40
MDPII3_2000	209.57	**210.819147**	210.807415	19/20	15.52	**210.819147**	210.819147	20/20	18.05
MDPII4_2000	211.99	**212.424859**	212.424859	20/20	16.15	**212.424859**	212.424859	20/20	25.81
MDPII5_2000	215.33	**216.088722**	216.088722	20/20	9.90	**216.088722**	216.088722	20/20	8.96
MDPII6_2000	210.61	**211.769151**	211.769151	20/20	10.88	**211.769151**	211.769151	20/20	6.88
MDPII7_2000	209.65	**209.780651**	209.780651	20/20	19.95	**209.780651**	209.780651	20/20	25.74
MDPII8_2000	212.43	**212.575432**	212.575432	20/20	17.03	**212.575432**	212.575432	20/20	27.75
MDPII9_2000	214.61	**215.007759**	215.007759	20/20	15.87	**215.007759**	215.007759	20/20	12.92
MDPII10_2000	210.06	**210.735749**	210.735436	15/20	28.22	**210.735749**	210.735561	17/20	28.19
#Improved		38	38			38	38		
#Better		0	3			0	4		
#Equal		40	33			40	33		
#Worse		0	4			0	3		

tion problems, including diversity and dispersion problems. This chapter illustrates some representative and effective memetic algorithms designed for the maximum diversity problem, the max-mean dispersion problem, and the generalized max-mean dispersion problem.

These memetic algorithms generally integrate three complementary features. An effective tabu search procedure is used to intensively examine a given search area. This is achieved by exploring constrained (small-sized) 1-flip and swap neighborhoods and adopting fast neighborhood evaluation techniques. A crossover operator is

Table 5 Comparison between MAMMDP (Lai and Hao, 2016) and PBEA (Lai et al., 2020) on the 40 GMaxMeanDP instances with $n = 3000$. Each instance was independently solved 20 times by each algorithm, and better results between the two algorithms are indicated in bold both in terms of f_{best} and f_{avg}.

	MAMMDP				PBEA			
Instance	f_{best}	f_{avg}	SR	$t(s)$	f_{best}	f_{avg}	SR	$t(s)$
I_3000_1	80.743467	80.743467	20/20	44.45	80.743467	80.743467	20/20	92.53
I_3000_2	84.201027	84.201027	20/20	17.82	84.201027	84.201027	20/20	46.71
I_3000_3	81.630082	81.630082	20/20	6.62	81.630082	81.630082	20/20	9.34
I_3000_4	80.234334	80.234334	20/20	29.61	80.234334	80.234334	20/20	28.07
I_3000_5	81.218062	81.218043	19/20	108.59	81.218062	**81.218062**	20/20	138.28
I_3000_6	83.197618	83.197618	20/20	37.99	83.197618	83.197618	20/20	64.05
I_3000_7	81.732080	81.732080	20/20	2.73	81.732080	81.732080	20/20	4.50
I_3000_8	80.624273	80.624273	20/20	79.61	80.624273	80.624273	20/20	83.53
I_3000_9	80.574438	80.574438	20/20	7.59	80.574438	80.574438	20/20	10.76
I_3000_10	83.397670	83.397670	20/20	48.63	83.397670	83.397670	20/20	138.84
II_3000_1	99.055143	99.055143	20/20	15.04	99.055143	99.055143	20/20	12.66
II_3000_2	105.574146	105.574146	20/20	29.08	105.574146	105.574146	20/20	63.27
II_3000_3	101.299271	101.299271	20/20	3.31	101.299271	101.299271	20/20	6.71
II_3000_4	101.079824	101.079824	20/20	8.41	101.079824	101.079824	20/20	8.03
II_3000_5	100.029225	**100.029225**	20/20	84.01	100.029225	100.028322	18/20	241.64
II_3000_6	101.978783	101.978783	20/20	5.80	101.978783	101.978783	20/20	4.56
II_3000_7	100.189718	100.189718	20/20	6.43	100.189718	100.189718	20/20	17.36
II_3000_8	101.160428	101.160428	20/20	3.36	101.160428	101.160428	20/20	4.52
II_3000_9	98.665034	98.665034	20/20	39.15	98.665034	98.665034	20/20	59.96
II_3000_10	104.896612	104.896612	20/20	4.40	104.896612	104.896612	20/20	11.86
III_3000_1	27.847334	27.847334	20/20	102.65	27.847334	27.847334	20/20	108.70
III_3000_2	27.776796	**27.774430**	7/20	214.29	27.776796	27.774272	4/20	120.08
III_3000_3	27.946519	27.944592	17/20	147.23	27.946519	**27.946519**	20/20	157.85
III_3000_4	27.816272	27.816272	20/20	81.41	27.816272	27.816272	20/20	70.66
III_3000_5	27.727167	27.727167	20/20	115.40	27.727167	27.727167	20/20	160.51
III_3000_6	27.686986	27.677719	8/20	136.73	**27.691682**	**27.686631**	4/20	131.25
III_3000_7	27.642060	27.642060	20/20	74.29	27.642060	27.642060	20/20	158.84
III_3000_8	27.736643	27.733842	5/20	287.29	27.736643	**27.734079**	6/20	184.58
III_3000_9	27.745820	27.744637	19/20	139.88	27.745820	**27.745820**	20/20	77.88
III_3000_10	27.561083	27.560295	19/20	157.43	27.561083	**27.561083**	20/20	92.74
IV_3000_1	278.039443	**278.037117**	19/20	137.79	278.039443	278.027811	15/20	151.80
IV_3000_2	276.539877	276.530847	18/20	216.82	276.539877	**276.539691**	18/20	238.37
IV_3000_3	277.334878	277.334878	20/20	31.02	277.334878	277.334878	20/20	40.40
IV_3000_4	278.956422	278.956422	20/20	42.08	278.956422	278.956422	20/20	61.36
IV_3000_5	276.595238	276.595238	20/20	152.28	276.595238	276.595238	20/20	108.00
IV_3000_6	280.721533	280.721533	20/20	55.32	280.721533	280.721533	20/20	60.47
IV_3000_7	273.653396	273.653396	20/20	84.47	273.653396	273.653396	20/20	169.85
IV_3000_8	276.358447	276.358447	20/20	70.96	276.358447	276.358447	20/20	81.56
IV_3000_9	274.864865	274.821773	17/20	159.03	274.864865	**274.838571**	18/20	241.92
IV_3000_10	276.428571	**276.411918**	17/20	220.16	276.428571	276.407810	16/20	151.95
#Better	0	4			1	8		
#Equal	39	28			39	28		
#Worse	1	8			0	4		

employed to discover promising new search regions guided by elite parent solutions. This is achieved by inheriting the common elements from the parents. A certain level of population diversity is maintained during the search, which is warranted by adopting a suitable population updating strategy, often relying on both solution quality and distance to the population solutions.

Finally, it is worth indicating that for some other dispersion problems such as the minimum difference dispersion problem, the current best performing method is based on an intensification-driven solution-based tabu search (IDTS) (Lai et al., 2019), which outperforms significantly other algorithms including a previous memetic al-

gorithm (Wang et al., 2017). It would be interesting to combine these two approaches to create a still better algorithm for this problem by adopting the IDTS algorithm within the memetic framework. Using the best local optimization is part of the best practices for designing effective memetic algorithms. This approach can be applied to better solve other diversity and dispersion problems.

References

Bäck T., Fogel D. B., (Eds) M. Z. (1989) Handbook of evolutionary computation. Springer.

Brimberg J., Mladenović N., Todosijević R., Urošević D. (2017) Less is more: solving the max-mean diversity problem with variable neighborhood search. Information Sciences, 382:179–200.

Carrasco R., Pham A., Gallego M., Gortázar F., Martí R., Duarte A. (2015) Tabu search for the max–mean dispersion problem. Knowledge-Based Systems, 85:256–264.

DeJong K. A. (1975) An analysis of the behavoir of a class of genetic adaptive systems. PhD thesis, University of Michigan, Ann Arborn, MI, USA.

Della Croce F., Garraffa M., Salassa F. (2014) A hybrid heuristic approach based on a quadratic knapsack formulation for the max-mean dispersion problem. In: International Symposium on Combinatorial Optimization, Springer, pp. 186–197.

Della Croce F., Garraffa M., Salassa F. (2016) A hybrid three-phase approach for the max-mean dispersion problem. Computers & Operations Research, 71:16–22.

Di Gaspero L., Schaerf A. (2006) Neighborhood portfolio approach for local search applied to timetabling problems. Journal of Mathematical Modeling and Algorithms, 5(1):65–89.

Dorne R., Hao J.-K. (1998) A new genetic local search algorithm for graph coloring. In: 5th International Conference on Parallel Problem Solving from Nature (PPSN), Springer, *Lecture Notes in Computer Science*, vol 1498, pp. 745–754.

Eiben A. E., Smith J. E. (2003) Introduction to Evolutionary Computing. Springer.

Fleurent C., Ferland J. A. (1996) Object-oriented implementation of heuristic search methods for graph coloring. In: Johnson D. S., Trick M. A. (eds) Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol 26, American Mathematical Society, pp. 619–652.

Galinier P., Hao J.-K. (1999) Hybrid evolutionary algorithms for graph coloring. Journal of Combinatorial Optimization, 3(4):379–397.

Galinier P., Hertz A., Zufferey N. (2008) An adaptive memory algorithm for the k-coloring problem. Discrete Applied Mathematics, 156(2):267–279.

Ghosh J. B. (1996) Computational aspects of the maximum diversity problem. Operations Research Letters, 19(4):175–181.

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

Glover F. (1997a) Tabu search and adaptive memory programming - advances, applications and challenges. In: R.S. Barr J. K. R.V. Helgason (ed) Interfaces in Computer Science and Operations Research, vol 7, Springer, pp. 1–75.

Glover F. (1997b) A template for scatter search and path relinking. In: et al. J.-K. H. (ed) Selected Papers from the Third European Conference on Artificial Evolution, Nîmes, France, October 1997, Springer, *Lecture Notes in Computer Science*, vol 1363, pp. 1–51.

Glover F., Laguna M. (1997) Tabu Search. Kluwer Academic Publishers.

Glover F., Laguna M., Martí R. (2000) Fundamentals of scatter search and path-relinking. Control and Cybernetics, 39:654–684.

Goëffon A., Richer J.-M., Hao J.-K. (2008) Progressive tree neighborhood applied to the maximum parsimony problem. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 5(1):136–145.

Goldberg D. E. (1989) Genetic algorithms in search, optimization and machine learning. Addison-Wesley.

Goldberg D. E., Richardson J. (1987) Genetic algorithms with sharing for multimodal function optimization. In: Proceedings of the Second International Conference on Genetic algorithms and their application, L. Erlbaum Associates Inc., Cambridge, Massachusetts, USA, pp. 41–49.

Hamiez J.-P., Robet J., Hao J.-K. (2009) A tabu search algorithm with direct representation for strip packing. In: Cotta C., Cowling P. I. (eds) Proceedings of the 9th European Conference on Evolutionary Computation in Combinatorial Optimization (EvoCOP), Tübingen, Germany, April 15-17, 2009, Springer, *Lecture Notes in Computer Science*, vol 5482, pp. 61–72.

Hansen P., Mladenovic N. (1999) An introduction to variable neighborhood search. In: et al. S. V. (ed) Metaheuristics, Advances and Trends in Local Search Paradigms for Optimization, Kluwer Academic Publishers, pp. 433–458.

Hao J. (2012) Memetic algorithms in discrete optimization. In: Neri F., Cotta C., Moscato P. (eds) Handbook of Memetic Algorithms, *Studies in Computational Intelligence*, vol 379, Springer, pp. 73–94.

Holland J. H. (1975) Adaptation and artificial systems. University of Michigan Press.

Hoos H. H., Stützle T. (2004) Stochastic local search: foundations and applications. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Jin Y. (2005) A comprehensive survey of fitness approximation in evolutionary computation. Soft Computing - A Fusion of Foundations, Methodologies and Applications, 9(1):3–12.

Jin Y., Hao J.-K. (2015) General swap-based multiple neighborhood tabu search for the maximum independent set problem. Engineering Applications of Artificial Intelligence, 37:20–33.

Katayama K., Narihisa H. (2005) An evolutionary approach for the maximum diversity problem. In: Recent advances in memetic algorithms, Springer, pp. 31–47.

Kernighan B. W., Lin S. (1970) An efficient heuristic procedure for partitioning graphs. The Bell System Technical Journal, 49(2):291–307.

Kuby M. J. (1987) Programming models for facility dispersion: The p-dispersion and maxisum dispersion problems. Geographical Analysis, 19(4):315–329.

Lai X., Hao J.-K. (2016) A tabu search based memetic algorithm for the max-mean dispersion problem. Computers & Operations Research, 72:118–127.

Lai X., Hao J., Yue D., Gao H. (2018) Diversification-driven memetic algorithm for the maximum diversity problem. In: 2018 5th IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS), IEEE, pp. 310–314.

Lai X., Hao J.-K., Glover F., Yue D. (2019) Intensification-driven tabu search for the minimum differential dispersion problem. Knowledge-Based Systems, 167:68–86.

Lai X., Hao J.-K., Glover F. (2020) A study of two evolutionary/tabu search approaches for the generalized max-mean dispersion problem. Expert Systems with Applications, 139:112856.

Li M., Hao J., Wu Q. (2020) General swap-based multiple neighborhood adaptive search for the maximum balanced biclique problem. Computers & Operations Research, 119:104922.

Li Y., Hao J.-K. (2022) Multi-neighborhood simulated annealing for personalized user project planning. Applied Soft Computing, 119:108566.

Lin S. (1965) Computer solutions of the traveling salesman problem. The Bell System Technical Journal, 44(10):2245–2269.

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

Liu X., Chen J., Wang M., Wang Y., Su Z., Lü Z. (2020) A two-phase tabu search based evolutionary algorithm for the maximum diversity problem. Discrete Optimization, p. 100613.

Lü Z., Hao J.-K. (2010) A memetic algorithm for graph coloring. European Journal of Operational Research, 200(1):235–244.

Lü Z., Glover F., Hao J.-K. (2010) A hybrid metaheuristic approach to solving the ubqp problem. European Journal of Operational Research, Published online: 30 June 2010.

Mahfoud S. W. (1992) Crowding and preselection revisited. In: Männer R., Manderick B. (eds) Parallel Problem Solving from Nature 2, PPSN-II, Brussels, Belgium, September 28-30, 1992, Elsevier Science, pp. 27–36.

Malaguti E., Monaci M., Toth P. (2008) A metaheuristic approach for the vertex coloring problem. INFORMS Journal on Computing, 20(2):302–316.

Martí R., Sandoya F. (2013) Grasp and path relinking for the equitable dispersion problem. Computers & Operations Research, 40(12):3091–3099.

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

Moalic L., Gondran A. (2018) Variations on memetic algorithms for graph coloring problems. Journal of Heuristics, 24(1):1–24.

Morrison R. W., Jong K. A. D. (2002) Measurement of population diversity. In: et al. P. C. (ed) Selected Papers from the 5th International Conference on Artificial Evolution, Le Creusot, France, October 29-31, 2001, Springer, *Lecture Notes in Computer Science*, vol 2310, pp. 31–41.

Moscato P. (1999) Memetic algorithms: a short introduction. In: Corne D., Dorigo M., Glover F. (eds) New Ideas in Optimization, McGraw-Hill Ltd., Maidenhead, UK., pp. 219–234.

Nagata Y., Kobayashi S. (1997) Edge assembly crossover: a high-power genetic algorithm for the travelling salesman problem. In: Bäck T. (ed) Proceedings of the 7th International Conference on Genetic Algorithms, Morgan Kaufmann, pp. 450–457.

Neri F., Cotta C., Moscato P. (eds) (2012) Handbook of Memetic Algorithms, *Studies in Computational Intelligence*, vol 379. Springer.

Papadimitriou C. H., Steiglitz K. (1982) Combinatorial optimization: algorithms and complexity. Prentice-Hall.

Porumbel D. C., Hao J.-K., Kuntz P. (2010) An evolutionary approach with diversity guarantee and well-informed grouping recombination for graph coloring. Computers and Operations Research, 37(10):1822–1832.

Radcliffe N. J. (1991) Forma analysis and random respectful recombination. In: R.K. Belew, L.B. Booker (Eds.), Proceedings of the 4th International Conference on Genetic Algorithms, Morgan Kaufmann, pp. 222–229.

Ren J., Hao J.-K., Wu F., Fu Z. (2021) An effective hybrid search algorithm for the multiple traveling repairman problem with profits. CoRR, abs/2111.05017, URL https://arxiv.org/abs/2111.05017, 2111.05017.

Sörensen K., Sevaux M. (2006) Ma|pm: memetic algorithms with population management. Computers and Operations Research, 33(5):1214–1225.

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

Wang Y., Wu Q., Glover F. (2017) Effective metaheuristic algorithms for the minimum differential dispersion problem. European Journal of Operational Research, 258(3):829–843.

Whitley D., Starkweather T., Fuquay D. (1989) Scheduling problems and traveling salesmen: the genetic edge recombination operator. In: Proceedings of the 3rd International Conference on Genetic Algorithms, George Mason University, USA, Morgan Kaufmann Publishers Inc., pp. 33 – 140.

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

Zhou Y., Hao J.-K., Duval B. (2017) Opposition-based memetic search for the maximum diversity problem. IEEE Transactions on Evolutionary Computation, 21(5):731–745.