Iterated maxima search for the maximally diverse grouping problem

Xiangjing Lai [a] and Jin-Kao Hao [a,b,*]

[a] *LERIA, Université d'Angers, 2 bd Lavoisier, 49045 Angers, France*

[b] *Institut Universitaire de France, Paris, France*

Accepted to ***European Journal of Operational Research****, 10 May 2016*

Abstract

The maximally diverse grouping problem (MDGP) is to partition the vertices of an edge-weighted and undirected complete graph into m groups such that the total weight of the groups is maximized subject to some group size constraints. MDGP is a NP-hard combinatorial problem with a number of relevant applications. In this paper, we present an innovative heuristic algorithm called iterated maxima search (IMS) algorithm for solving MDGP. The proposed approach employs a maxima search procedure that integrates organically an efficient local optimization method and a weak perturbation operator to reinforce the intensification of the search and a strong perturbation operator to diversify the search. Extensive experiments on five sets of 500 MDGP benchmark instances of the literature show that IMS competes favorably with the state-of-the-art algorithms. We provide additional experiments to shed light on the rationality of the proposed algorithm and investigate the role of the key ingredients.

Keywords: Graph grouping and clustering problems; iterated search; heuristics.

1 Introduction

Given an edge-weighted and undirected complete graph $G = (V, E, D)$, where $V = \{1, 2, \ldots, n\}$ is the set of n vertices, $E = \{\{i, j\} : i, j \in V, i \neq j\}$ is the set of $n \times (n-1)/2$ edges, and $D = \{d_{ij} \geq 0 : \{i, j\} \in E\}$ represents the set of non-negative edge weights, the maximally diverse grouping problem (MDGP for short) is to partition the vertex set V into m disjoint subsets or groups such

* Corresponding author.
Email addresses: laixiangjing@gmail.com (Xiangjing Lai), hao@info.univ-angers.fr (Jin-Kao Hao).

that the size of each group g lies in a given interval $[a_g, b_g]$ $(g = 1, 2, \ldots, m)$ while maximizing the sum of the edge weights of the m groups. Here, a vertex $v \in V$ is usually called an element, an edge weight $d_{ij} \in D$ is called the diversity between elements i and j, while a_g and b_g are respectively called the lower and upper limits (or bounds) of the size of group g.

MDGP can be expressed as a quadratic integer program with binary variables x_{ig} that take the value of 1 if element i is in group g and 0 otherwise [14,24].

$$\text{Maximize} \quad \sum_{g=1}^{m}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} d_{ij}x_{ig}x_{jg} \tag{1}$$

$$\text{Subject to} \quad \sum_{g=1}^{m} x_{ig} = 1, i = 1, 2, \ldots, n \tag{2}$$

$$a_g \leq \sum_{i=1}^{n} x_{ig} \leq b_g, g = 1, 2, \ldots, m \tag{3}$$

$$x_{ig} \in \{0, 1\}, i = 1, 2, \ldots, n; g = 1, 2, \ldots, m \tag{4}$$

where the set of constraints (2) guarantees that each element is located in exactly one group and the set of constraints (3) forces the size of group g is at least a_g and at most b_g.

MDGP belongs to the category of vertex-capacitated clustering problems which are a type of extensively studied combinatorial search problems and can further be divided into the max-clustering problem and min-clustering problem [11,12,16,20,26]. In short, the max-clustering (min-clustering) problem is to partition the vertices of an undirect graph $G = (V, E)$ with edge and vertex weights into m mutually disjoint subsets (groups or clusters) such that the sum of the vertex weights of the subsets is bounded from below by a and from above by b while maximizing (minimizing) the sum of the weights of the edges inside the subsets [16].

In addition to its theoretical signification as a typical NP-hard problem, MDGP has a variety of real-world applications, like assignment of students to groups [15,17,30], creation of peer review groups [7], VLSI design [28], storage allocation of large programs onto paged memory, and creation of diverse groups in companies so that people from different backgrounds work together [4]. For a review of possible applications of MDGP, the readers are referred to recent papers like [14,22,24,25].

Given the practical importance and high computational complexity of MDGP, a number of exact and heuristic algorithms have been proposed in the literature. One of the most representative exact algorithm for MDGP is the column generation approach presented in [16]. Nevertheless, local search or evolutionary heuristics remain the dominant approach in the literature to find high-quality sub-optimal solutions for large graphs. Examples of local search heuristics includes multistart algorithm [1], Weitz-Jelassi (WJ) algorithm [27], Lotfi-Cerveny-Weitz (LCW) algorithm [29], T-LCW method based on LCW and tabu search [14], tabu search with strategic oscillation (TS-SO) [14], multistart simulated annealing (MSA) [21], variable neighborhood search (VNS) [21], general variable neighborhood search (GVNS) [25], skewed general variable neighborhood search (SGVNS) [6], iterated tabu search (ITS) [22] and several graph theoretical heuristics [10]. The population-based evolutionary approach includes hybrid genetic algorithm (LSGA) [9], hybrid grouping genetic algorithm [7], hybrid steady-state genetic algorithm (HGA) [21], artificial bee colony (ABC) algorithm [24], and constructive genetic algorithm [18]. According to the computational results reported on the MDGP benchmarks, the heuristics T-LCW, TS-SO, HGA, MSA, VNS, ABC, GVNS, ITS, and SGVNS achieved high performances at the time they were published.

In this paper, we propose an effective heuristic called the iterated maxima search (IMS) algorithm for solving MDGP. IMS follows and extends the iterated local search framework [19]. Though IMS shares ideas from breakout local search [2,3] and three-phase local search [13], it distinguishes itself from these approaches by three specific features: its local search procedure (to improve the incumbent solution), the maxima search scheme (to locate other local optima within a limited region of the search space) and its perturbation operator (to modify greatly the incumbent solution). In addition, IMS employs a randomized procedure for initial solution generation. When the proposed algorithm was assessed on five sets of 500 benchmark instances ($120 \leq n \leq 3000$) commonly used in the literature, the computational results showed that the algorithm achieves highly competitive results compared to the state-of-the-art algorithms especially on the large-sized instances.

In Section 2, we describe the components of the proposed algorithm. Section 3 is dedicated to extensive computational assessments based on the commonly used benchmarks with respect to the top performing algorithms from the literature. In Section 4, the important components of the proposed algorithm are analyzed and discussed. Conclusions are provided in the last Section.

2 Iterated Maxima Search Algorithm for MDGP

The proposed iterated maxima search (IMS) algorithm can be considered as an extended iterated local search algorithm [19] and shares ideas from breakout local search [2,3] and three-phase local search [13] (see Section 2.7 for more discussions). IMS is composed of four basic procedures: solution initialization, local search, weak perturbation, and strong perturbation. The basic idea of the IMS algorithm is to provide the search algorithm with a desirable tradeoff between intensification and diversification. This is achieved by iterating the maxima search procedure (local search combined with weak perturbation to explore a limited region around a locally optimal solution) followed by the strong perturbation procedure (to displace the search to a distant region by changing strongly the attained local optimum).

2.1 General Procedure

Algorithm 1: Main framework of iterated maxima search method for MDGP

Input: Instance I, the depth of maxima search (α), the strength of strong perturbation (η_s), the cutoff time (t_{max})
Output: The best solution s^* found

begin
  $s \leftarrow InitialSolution(I)$ `/* Sections 2.3 and 2.4 */`
  $s^* \leftarrow s$
  while *Time()* $\leq t_{max}$ **do**
    $s \leftarrow MaximaSearch(s, \alpha)$ `/* Section 2.5 */`
    if $f(s) > f(s^*)$ **then**
      $s^* \leftarrow s$
    end
    $s \leftarrow StrongPerturbation(s, \eta_s)$ `/* Section 2.6 */`
  end
  return s^*
end

The IMS algorithm (Algorithm 1) starts from a feasible initial solution (Section 2.2) that is obtained with a randomized construction procedure (Section 2.3). Then it repeats a number of iterations until a cutoff time is reached. At each iteration, the incumbent solution s is improved by the maxima search procedure (Sections 2.4 and 2.5) and then perturbed by the strong perturbation procedure (Sections 2.6). The best solution found (s^*) is updated whenever it is needed and finally returned as the output at the end of the IMS algorithm. In the rest of this section, we describe the different components of the proposed algorithm.

2.2 Search Space, Fitness Function and Solution Representation

For a given MDGP instance $G = (V, E)$ with its edge diversity matrix $D = [d_{ij}]_{n \times n}$ and the number m of groups, the search space Ω explored by the IMS algorithm contains all partitions of the elements of V into m groups such that the size of each group lies between its lower and upper limits. In other words, our IMS algorithm visits only feasible solutions.

Formally, let $P : V \to \{1, ..., m\}$ be a partition function of the n elements of V to m groups. For each group $g \in \{1, ..., m\}$ with lower and upper limits a_g and b_g, let $P_g = \{i \in V : P(i) = g\}$ (i.e., P_g is the set of elements of group g). Then the search space is given by:

$$\Omega = \{P : \forall g \in \{1, ..., m\}, a_g \leq |P_g| \leq b_g\}.$$

For any candidate solution $s = \{P_1, P_2, ..., P_m\}$ of Ω, its quality or fitness is given by the objective value $f(s)$:

$$f(s) = \sum_{g=1}^{m} \sum_{i,j \in P_g, i<j} d_{ij} \tag{5}$$

Given a candidate solution $s = \{P_1, P_2, ..., P_m\}$, we use a n-dimensional vector y (element coordinate vector) to indicate the group of each element. In other words, if element i belongs to group P_g, then $y[i] = g$ ($i \in \{1, \ldots, n\}$). Additionally, we use a m-dimensional vector z (group size vector) to indicate the size of each group in solution s, i.e., $z[g] = |P_g|$ ($g \in \{1, \ldots, m\}$). It should be clear that any solution $s \in \Omega$ can be fully characterized by its associated y and z vectors. For this reason, we use the notation $s =< y, z >$ to represent a solution in the rest of this section whenever this is appropriate.

2.3 Initial Solution

In the IMS algorithm, the initial solution is generated as follows. First, we construct β feasible solutions, each solution being generated in two stages. First, we pick an empty group g ($g \in \{1, \ldots, m\}$), randomly select a_g elements among the unassigned elements, assign them to group g, and repeat this process until all groups reach their lower bound a_g in size. Second, the remaining elements are assigned to a random group whose size is less than b_g in an one-by-one way. Finally, for each constructed solution, the local search procedure (see Section 2.4) is applied to improve its quality. The best one among those β solutions is chosen as the initial solution of our IMS algorithm. This initialization procedure is an adaptation of previous approaches used in [6,14,25].

Algorithm 2: Local Search Procedure

Function *LocalSearch*($s =< y, z >$)
Input: Element coordinate vector $y[1:n]$, group size vector $z[1:m]$
Output: The local optimum solution (denoted by $< y, z >$)
Initialize f, move value matrix $\gamma[n, m]$ /* Section 2.4.2 */
$Improve \leftarrow$ **true**
while *Improve*=***true*** **do**
  $Improve \leftarrow$ **false**
  /* Examine the neighborhood N_1 in a lexicographical order */
  for $v \leftarrow 1$ ***to*** n **do**
    for $g \leftarrow 1$ ***to*** m **do**
      if $(y[v] \neq g) \wedge (z[y[v]] > a_{y[v]}) \wedge (z[g] < b_g)$ **then**
        $\Delta_f \leftarrow \gamma[v][g] - \gamma[v][y[v]]$ /* Calculate the move value */
        if $\Delta_f > 0$ **then**
          $z[y[v]] \leftarrow z[y[v]] - 1$
          $z[g] \leftarrow z[g] + 1$
          $y[v] \leftarrow g$ /* Update the incumbent solution */
          $f \leftarrow f + \Delta_f$
          Update matrix γ /* Section 2.4.2 */
          $Improve \leftarrow$ **true**
        end
      end
    end
  end
  /* Examine the neighborhood N_2 in a lexicographical order */
  for $v \leftarrow 1$ ***to*** $n - 1$ **do**
    for $u \leftarrow v + 1$ ***to*** n **do**
      if $y[v] \neq y[u]$ **then**
        $\Delta_f \leftarrow (\gamma[v][y[u]] - \gamma[v][y[v]]) + (\gamma[u][y[v]] - \gamma[u][y[u]]) - 2d_{vu}$
        if $\Delta_f > 0$ **then**
          $Swap(y, v, u)$ /* Update the incumbent solution */
          $f \leftarrow f + \Delta_f$
          Update matrix γ /* Section 2.4.2 */
          $Improve \leftarrow$ **true**
        end
      end
    end
  end
end
return $<y, z>$

2.4 Local Search Method

For local optimization, the IMS algorithm employs a double-neighborhood local search method described in Algorithm 2 (also see Sections 2.4.1 and

2.4.2 for details), where γ and Δ_f are defined in Section 2.4.2. Starting with an input solution, the local search method examines in a deterministic order two complementary neighborhoods N_1 and N_2 (see Section 2.4.1 for the definition of these neighborhoods), and updates the incumbent solution each time an improving neighbor solution is detected. This process terminates when the incumbent solution s can not be further improved by any neighbor solution in both $N_1(s)$ and $N_2(s)$.

Specifically, the local search method examines the neighborhoods N_1 and N_2 in a token-ring way ($N_1 \rightarrow N_2 \rightarrow N_1 \rightarrow N_2 \rightarrow N_1 \rightarrow N_2, \ldots$). When examining a given neighborhood (N_1 or N_2), neighbor solutions are visited in a lexicographical order (Alg. 2, lines 7-21 for N_1 and lines 23-35 for N_2). Each time an improving neighbor solution is found, it becomes the new incumbent solution (first improvement strategy) and the search continues from this point to explore neighbor solutions within the neighborhood of the new incumbent solution (Alg. 2, lines 12-16 for N_1 and lines 28-30 for N_2). According to the algorithm, the search switches from N_1 to N_2 after examining at most $n \times m$ neighbor solutions and from N_2 to N_1 after considering at most $\frac{n\times(n-1)}{2}$ solutions. Note that the 'if' tests in lines 9 and 25 aim at excluding solutions that do not belong to the neighborhoods N_1 and N_2.

The whole local search method stops when no better solution can be found in both N_1 and N_2 and returns the best solution encountered which is a locally optimal solution with respect to N_1 and N_2.

Note that though our local search method uses the same neighborhoods as the variable neighborhood descent (VND) method in [6], our way of exploring the two neighborhoods is quite different. The standard VND method starts with the first neighborhood N_1 and makes a complete exploitation of this neighborhood. It switches to the next neighborhood N_2 when a local optimum is attained with respect to N_1. Moreover, VND switches immediately to the first neighborhood N_1 as soon as an improving solution is encountered in N_2. VND stops when the search process reaches the end of neighborhood N_2, i.e., when no improving solution can be found in N_2.

In the following subsections, we describe in detail the neighborhood structures and the streamlining technique for a fast neighborhood evaluation which is useful to accelerate the local search procedure.

2.4.1 Neighborhood Structures

We now define the two neighborhoods explored by the local search procedure, called the constrained one-move neighborhood (denoted by N_1) and swap neighborhood (N_2) respectively. For MDGP, N_1 and N_2 are two popular neighborhoods that were used in several previous studies [6,14,21,22,24,25].

The constrained $OneMove$ neighborhood N_1 is defined as follows. Given a solution (or partition) $s = \{P_1, P_2, \ldots, P_m\}$, the $OneMove$ operator transfers a vertex v from its current group P_i to another group P_j such that $|P_i| > a_i$ and $|P_j| < b_j$, where a_i and b_j denote respectively the lower limit of $|P_i|$ and the upper limit of $|P_j|$. Let $< v, P_i, P_j >$ designate such a move and $s \oplus < v, P_i, P_j >$ be the resulting neighboring solution when applying the move to s. Then the neighborhood N_1 of s is composed of all possible solutions that can be obtained by applying the constrained $OneMove$ operator to s. i.e.,

$$N_1(s) = \{s \oplus < v, P_i, P_j > : v \in P_i, i \neq j, |P_i| > a_i, |P_j| < b_j\}$$

One notices that this neighborhood is not applicable if $a_i = b_i$ holds for all the groups. Clearly N_1 is bounded by $O(nm)$ in size.

The $SwapMove$ neighborhood N_2 is defined as follows. Given two vertices v and u which are located in two different groups of solution s, the $SwapMove(v, u)$ move exchanges their groups to produce a neighboring solution. Thus, the neighborhood N_2 of a solution s is composed of all possible neighboring solutions that can be obtained by applying $SwapMove$ to s, i.e.,

$$N_2(s) = \{s \oplus SwapMove(v, u) : v \in P_i, u \in P_j, 1 \leq i < j \leq m\}$$

Clearly the size of N_2 is bounded by $O(n^2)$ and is usually much larger than that of N_1 since often $n >> m$ holds.

2.4.2 Fast Neighborhood Evaluation Technique

Like [6,21,22,24,25], we adopt an incremental technique for fast evaluation of the considered neighborhood. We maintain a $n \times m$ matrix γ to effectively calculate the move value (i.e., the change of objective value) of each candidate move, where the entry $\gamma[v][g]$ represents the sum of weights between the vertex v and vertices in the group g for the current solution.

Given the current solution s, if a $OneMove$ move, $< v, P_i, P_j >$, is performed, the move value can be easily determined as $\Delta_f(< v, P_i, P_j >) = \gamma[v][j] - \gamma[v][i]$, and then the matrix γ is updated accordingly. Specifically, the i-th and j-th columns of matrix γ are updated as follows: $\gamma[u][i] = \gamma[u][i] - d_{vu}$, $\gamma[u][j] = \gamma[u][j] + d_{vu}$, $\forall u \in V, u \neq v$, where d_{vu} is the diversity between vertices v and u. On the other hand, if a $SwapMove(v, u)$ operation, which is composed of two $OneMove$ operations, is performed, its move value can be calculated as $\Delta_f(SwapMove(v, u)) = (\gamma[v][y[u]] - \gamma[v][y[v]]) + (\gamma[u][y[v]] - \gamma[u][y[u]]) - 2d_{vu}$, where $y[v]$ and $y[u]$ are the group of vertex v and u in the current solution $s = < y, z >$ (see Section 2.2).

The matrix γ is initialized at the beginning of each neighborhood search, and is updated after each move, which can respectively be achieved in $O(n^2)$ and $O(n)$ for $OneMove$ and $SwapMove$.

2.5 Maxima Search Procedure with Weak Perturbation

Algorithm 3: Maxima Search Procedure for MDGP

```
1  Function MaximaSearch(s_0, α)
   Input: Initial solution s_0, the depth of maxima search α
   Output: The best solution s_b found
2  begin
3      s ← s_0
4      s_b ← LocalSearch(s)                          /* Section 2.4 */
5      counter ← 0
6      while counter < α do
7          s ← WeakPerturbation(s_b)                 /* Section 2.5 */
8          s ← LocalSearch(s)                        /* Section 2.4 */
9          if f(s) > f(s_b) then
10             s_b ← s
11             counter ← 0
12         else
13             counter ← counter + 1
14         end
15     end
16     return s_b
17 end
```

The maxima search (MS) procedure, as described in Algorithm 3, aims to detect other locally optimal solutions of better quality within a limited search region around a given input local optimum. For this purpose, it alternates between the local search procedure described in Section 2.4 and a weak perturbation procedure explained below. Specifically, starting with the input solution, MS first applies the local search procedure to obtain a local optimum which becomes the incumbent solution (Alg. 3, line 4). Then, it performs the weak perturbation procedure (Alg. 3, line 7) to change slightly the incumbent solution. This is followed by a new round of the local search procedure (Alg. 3, line 8) to reach a new local optimum from the perturbed solution. These two procedures are repeated until the incumbent solution can not be improved for α consecutive perturbations (α is called the depth of the MS procedure). Finally, the best solution found is returned as the result of the maxima search procedure.

Within the above maxima search procedure, the weak perturbation operator aims to jump out of the current local optimum trap by accepting some

Algorithm 4: Weak Perturbation Operator

```
1  Function WeakPerturbation(s_0)
   Input: Input solution s_0, neighborhoods N_1 - N_2 defined in Section 2.4.1,
          strength of weak perturbation \eta_w
   Output: The perturbed solution s_p
2  begin
3      s_p ← s_0
4      for L ← 1 to \eta_w do
5          Randomly pick a neighbor solution s ∈ N_1(s_p) ∪ N_2(s_p)
6          for l ← 1 to |V| do
7              Randomly pick a neighbor solution s' ∈ N_1(s_p) ∪ N_2(s_p)
8              if f(s') > f(s) then
9                  s ← s'
10             end
11         end
12         s_p ← s
13     end
14     return s_p
15 end
```

deteriorating solutions. As shown in Algorithm 4, it realizes η_w controlled *OneMove* or *SwapMove* steps (Section 2.4.1) where η_w (a parameter) is called the strength of weak perturbation. Specifically, for each weak perturbation step (i.e., each iteration of the outer 'for' loop of Alg. 4, line 4), we first identify the best solution among randomly sampled $|V| + 1$ neighbor solutions of the current solution s_p (Alg. 4, lines 5-11) and then use this best solution to replace the current solution s_p (Alg. 4, line 12), which serves as the starting point for the next perturbation step. The generated solution following these η_w weak perturbation steps is returned as the incumbent solution of the next round of the local search procedure. Note that large samples deteriorate less the current solution and we use the problem size $|V|$ to adjust the sample size.

2.6 Strong Perturbation

The maxima search procedure equipped with its weak perturbation procedure helps to discover new and better local optima around an input optimum. However, the search can be trapped into deep local optima and weak perturbations are no more sufficient for the algorithm to continue its search. Hence, our IMS algorithm employs a strong perturbation procedure to jump out of such deep local optimum traps. The strong perturbation procedure consecutively performs η_s randomly selected *OneMove* or *SwapMove* moves regardless of the objective value of each performed move, where η_s is called the strength of

Algorithm 5: Strong Perturbation

Function *StrongPerturbation*(s_0, η_s)
Input: Input solution s_0, neighborhoods $N_1 - N_2$ defined in Section 2.4.1, strength of perturbation η_s
Output: The perturbed solution s_p
begin
  $s_p \leftarrow s_0$
  for $L \leftarrow 1$ ***to*** η_s **do**
    Randomly pick a neighbor solution $s \in N_1(s_p) \cup N_2(s_p)$
    $s_p \leftarrow s$
  end
  return s_p
end

strong perturbation. In this study, η_s is empirically set as $\eta_s = \theta \times \frac{n}{m}$, where θ is a parameter that is used to control the strong perturbation strength, n is the size of problem instance, and m is the number of groups. The pseudo-code of the strong perturbation operator is shown in Algorithm 5.

2.7 Motivations and Related Search Frameworks

Like most heuristic approaches in the literature, the proposed IMS algorithm lacks a strict theoretical basis. On the other hand, the design of the IMS algorithm relies on the well-known basic principle that an efficient algorithm must ensure a suitable tradeoff between intensification and diversification of its search process. To achieve this goal, the proposed algorithm employs the maxima search procedure (Section 2.5 and Alg. 3) to perform an intensified search around a limited search region (i.e., locate the best possible solution within the nearby region around a discovered local optimum). When the maxima search procedure is trapped in a deep local optimum, the IMS algorithm switches to the strong perturbation procedure (Section 2.6) to jump out of the trap and displace the search to a more distant region. The analysis on spatial distribution of high-quality local optima shown in Section 4.3 provides additional evidences about the rationality of the proposed IMS algorithm.

From the perspective of algorithm design, the proposed IMS algorithm shares the same framework as the three-phase search (TPS) method [13] which also uses a weak (or strong) perturbation procedure to explore nearby (or distant) local optima. On the other hand, unlike TPS which applies a directed perturbation procedure based on a tabu list, IMS relies on the best neighbor strategy based on a random sample. IMS also shares ideas with the breakout local search (BLS) method [2,3] where both the perturbation type and perturbation strength are determined in an adaptive manner. BLS has a strong emphasis on an adaptive mechanism to control the application of three well-separated perturbation operators (directed perturbation, random perturbation, and frequency-based perturbation). Unlike BLS, IMS follows a simpler scheme and invokes its weak perturbation procedure (embedded inside the maxima search) and strong perturbation procedure deterministically.

As shown in the next section, the IMS algorithm equipped with its particular features proves to perform very well on many MDGP benchmark instances.

3 Experimental Results and Comparisons

Following the practice of the literature, we assess the performance of our IMS algorithm by presenting computational results on a large number of benchmark instances, and making comparisons with the state-of-the-art MDGP algorithms in the literature.

3.1 Benchmark Instances

Our computational assessments are based on four sets of 460 benchmarks which are extensively used in the literature [1] and a new set of 40 large scale benchmarks which are adapted from the related maximum diversity problem (MDP) [2]. The details of the benchmark instances are described as follows.

- **RanReal set**: This set consists of 40 instances with equal group sizes (EGS) and 40 instances with different group sizes (DGS), where the distances (or diversities) between elements are a real number uniformly and randomly generated in $(0, 100)$. The number of elements n varies from 120 to 960, the number of groups m is between 10 and 24, and the minimum and maximum group sizes a_g and b_g are in the range of $\{2, 3, \ldots, 48\}$. Moreover, for the EGS instances, both a_g and b_g are given by $\lfloor n/m \rfloor$.
- **RanInt set**: This set has the same structure as the RanReal set and consists of 40 EGS instances and 40 DGS instances. The distances between elements are an integer uniformly and randomly generated in the interval $[0, 100]$.
- **Geo set**: As the previous two sets of benchmarks, this set contains 40 EGS instances and 40 DGS instances, but the distances are Euclidean distances between pairs of points with random coordinates from $[0, 10]$, and the number of coordinates for each point is generated randomly in $[2, 21]$.

[1] Available from `http://www.optsicom.es/mdgp/mdgplib.zip`
[2] Available from `http://www.optsicom.es/mdp/#instances`

- **MDG-a set**: This set is composed of 11 subsets adapted from graphs of the related maximum diversity problem [8], where each subset contains 20 instances with $n = 2000$. For the first subset, the number of groups m is set to 50, the lower and upper limits of the group sizes a_g and b_g are respectively fixed to 32 and 48. This subset was first adapted in [21] for MDGP, and then used in [6]. The characteristics of the remaining 10 subsets are given in Table 1. These 10 subsets were first adapted in [24] for MDGP and then used in [6] under the name "Type1_22". For all these 220 instances, the distances between elements are randomly generated between 0 to 10 using an uniform distribution.

Table 1
The number of groups, lower and upper group limits for the instances in the set MDG-a.

		DGS		EGS
n	m	a_g	b_g	$a_g = b_g$
2000	50	32	48	–
2000	10	173	227	200
2000	25	51	109	80
2000	50	26	54	40
2000	100	13	27	20
2000	200	6	14	10

- **MDG-c set**: This new set, introduced in this paper, consists of 20 DGS instances and 20 EGS instances with $n = 3000$ and $m = 50$, which are adapted from graphs of the maximum diversity problem. The lower and upper limits a_g and b_g of group sizes for these instances are respectively fixed to $\lfloor 0.8n/m \rfloor$ and $\lfloor 1.2n/m \rfloor$ for the DGS instances. For the EGS instances, both a_g and b_g are given by $\lfloor n/m \rfloor$. The distances between elements are randomly generated in [0,1000].

3.2 Parameter Setting and Experimental Protocol

Table 2
Setting of parameters

Parameters	Section	Description	Values
β	2.3	number of initial solutions	10
η_w	2.6	strength of weak perturbation	3
α	2.5	depth of maxima search procedure	{3,5}
θ	2.6	coefficient for the strength of strong perturbation	{1.0,1.5}

Before presenting our computational results, we first provide some basic information about our experiments, including the parameter settings used by our IMS algorithm, the reference algorithms, the experimental platform, and the termination conditions of algorithms.

First, Table 2 shows the parameter setting of the IMS algorithm which was achieved by a preliminary experiment. Notice that the parameter α (depth of maxima search) takes 5 for the instances with $n \leq 400$ or $n/m \leq 10$, and 3 for the remaining instances. For the parameter θ, which is used to control the strength of strong perturbation, we also use two values, (i.e., $\theta \in \{1.0, 1.5\}$), where θ is set to 1.0 for small instances with $n \leq 400$ and 1.5 for the remaining instances. These parameter values are used for all the following experiments even though a fine tuning of some parameters would lead to better results.

According to the computational results reported in two most recent papers [6,22], the ITS and SGVNS algorithms can be considered as the state-of-the-art algorithms for MDGP. Specifically, the ITS algorithm dramatically outperformed its reference algorithms, including MSA, HGA, VNS of [21] and TS-SO, T-LCW of [14], and the SGVNS algorithm significantly outperformed its reference algorithms, including HGA, TS-SO, ABC of [24], and GVNS of [25]. Consequently, in this study we use the ITS and SGVNS algorithms as the reference algorithms to assess the performance of our IMS algorithm.

It is worth noting that in this study all the experiments were based on the same computing platform, which makes it possible to perform a rather fair comparison between our IMS algorithm and the two reference algorithms ITS and SGVNS. The source code of the ITS algorithm was kindly provided by the authors and is available at `http://www.proin.ktu.lt/~gintaras/mdgp.html`, and the executable code of the SGVNS algorithm was kindly provided by the corresponding author of [6]. The IMS and ITS codes were compiled using g++ compiler [3], and all experiments were carried out on a computing platform with an Intel Xeon E5440 processor (2.83 GHz CPU and 2Gb RAM), running the Linux operating system. Following the DIMACS machine benchmark procedure, our machine requires respectively 0.23, 1.42, and 5.42 seconds for the graphs r300.5, r400.5, r500.5 [4].

For all algorithms considered in this study, the stopping condition is a fixed cutoff time limit t_{max} which depends on the size of the instances. Specifically, t_{max} is set as follows: $t_{max} = 3$ seconds for $n = 120$, 20 seconds for $n = 240$, 120 seconds for $n = 480$ seconds, 600 seconds for $n = 960$, 1200 seconds for $n = 2000$, and 3000 seconds for $n = 3000$. Finally, due to the stochastic nature of the compared algorithms, each instance was independently solved 20 times by each algorithm.

[3] Our code will be available at `http://www.info.univ-angers.fr/pub/hao/mdgp.html`

[4] The benchmark program (dmclique.c) and the rxxx.5 graphs are available at `ftp://dimacs.rutgers.edu/pub/challenge/graph/solvers/`

3.3 Computational Results and Comparison on the Medium sized Instances

The RanInt, RanReal and Geo benchmarks have been widely tested to assess the performance of MDGP algorithms in the literature. The computational results of our IMS algorithm and the reference algorithms ITS and SGVNS are summarized in Tables 3–8. Column 1 of the tables gives the names of instances (Instance). Columns 2–4 report the best objective values (f_{best}) obtained over 20 independent runs of the compared algorithms. Columns 5–7 show the average objective values (f_{avg}). The best values among the results of the competing algorithms are indicated in bold. The row '#Best' indicates the number of instances for which an algorithm performs the best among the compared algorithms. To verify whether there exists a significant difference between IMS and the reference algorithms in terms of the best and average objective values, the p-values from the non-parametric Friedman tests are reported in the last row of each table. A p-value smaller than 0.05 indicates a significant difference between two sets of compared results.

Tables 3 and 4 show that for the RanInt instances, IMS outperforms the reference algorithms. For the 40 instances with different group sizes, IMS yields the best outcomes for 29 and 36 instances in terms of the best and average objective values. For the 40 instances with equal group sizes, IMS performs the best on 32 and 26 instances in terms of the best and average objective values. Furthermore, the small p-values confirm the significance of the differences between the results of the IMS algorithm and those of two reference algorithms.

Tables 5 and 6 on the RanReal instances indicate that the IMS algorithm also outperforms the ITS and SGVNS algorithms on this set of benchmarks. For the 40 instances with different group sizes, IMS yields the 25 and 37 best outcomes in term of the best and average values. For the 40 instances with equal group sizes, IMS obtains the largest 'best' and 'average' values for 29 and 26 instances. The Friedman tests (with p-values smaller than 0.05) confirm the statistical significance of the observed differences.

Tables 7 and 8 show that for the Geo instances, IMS performs significantly better than SGVNS, but performs worse than ITS. For the instances with different group sizes, IMS obtains better results than SGVNS for all instances in terms of both the best and average objective values. However, compared to the ITS algorithm, IMS yields a better and worse result respectively for 16 and 24 instances in terms of the best objective value. Concerning the average objective value, IMS obtains a better and worse result respectively for 26 and 14 instances compared to ITS. The p-values (> 0.05) in terms of the best and average objective values do not reveal a significant difference between IMS and ITS. However, for the instances with equal group sizes, ITS clearly dominates IMS and SGVNS.

Table 3
Comparison of the IMS algorithm with two state-of-the-art algorithms (i.e., ITS [22] and SGVNS [6]) on the RanInt instances with different group sizes.

Instance	f_{best}			f_{avg}		
	ITS [22]	**SGVNS** [6]	**IMS**	**ITS** [22]	**SGVNS** [6]	**IMS**
RanInt_n120_ds_01	**51161**	51097	51125	**51003.45**	50855.75	50920.60
RanInt_n120_ds_02	**51400**	51380	51366	51252.10	51204.55	**51259.90**
RanInt_n120_ds_03	50245	**50248**	50233	50144.25	50108.80	**50150.50**
RanInt_n120_ds_04	50394	50387	**50400**	50123.15	50323.30	**50340.30**
RanInt_n120_ds_05	49884	**49996**	49928	49447.30	**49804.00**	49803.50
RanInt_n120_ds_06	**49734**	49678	49701	49532.40	49496.50	**49583.90**
RanInt_n120_ds_07	50184	50282	**50316**	49743.90	50160.05	**50212.90**
RanInt_n120_ds_08	**50459**	50351	50345	50269.20	50262.05	**50272.20**
RanInt_n120_ds_09	**50499**	50429	50445	**50345.05**	50259.80	50322.10
RanInt_n120_ds_10	50337	**50364**	50327	50217.25	**50256.30**	50232.40
RanInt_n240_ds_01	160390	160460	**160498**	159964.70	160203.20	**160308.50**
RanInt_n240_ds_02	160019	**160154**	160095	159638.25	159679.50	**159812.30**
RanInt_n240_ds_03	160235	160211	**160338**	159756.60	159877.35	**160071.50**
RanInt_n240_ds_04	**162728**	162340	162473	161894.35	162001.85	**162280.30**
RanInt_n240_ds_05	160423	160648	**160653**	160177.40	160186.95	**160378.50**
RanInt_n240_ds_06	160944	161049	**161145**	160580.00	160739.70	**160810.40**
RanInt_n240_ds_07	159770	159950	**160156**	159362.70	159506.10	**159734.00**
RanInt_n240_ds_08	**158161**	157953	158087	157748.30	157666.85	**157879.20**
RanInt_n240_ds_09	160463	160409	**160551**	160060.30	160119.85	**160322.00**
RanInt_n240_ds_10	159924	160009	**160278**	159537.75	159659.00	**159934.70**
RanInt_n480_ds_01	389602	390304	**390687**	388476.85	389181.00	**389623.90**
RanInt_n480_ds_02	387757	388585	**388972**	386995.15	387679.55	**388277.70**
RanInt_n480_ds_03	387751	387691	**388054**	386584.75	386844.00	**387213.70**
RanInt_n480_ds_04	391604	391231	**392298**	390641.05	390461.90	**391689.10**
RanInt_n480_ds_05	388757	388818	**389107**	387718.15	387679.15	**388374.30**
RanInt_n480_ds_06	388764	389422	**390029**	387961.70	388465.45	**389168.80**
RanInt_n480_ds_07	388513	389171	**389350**	387879.15	388096.00	**388474.70**
RanInt_n480_ds_08	390230	390473	**391128**	389552.45	389621.65	**390463.60**
RanInt_n480_ds_09	388128	388388	**388704**	387012.65	387309.75	**388001.90**
RanInt_n480_ds_10	392870	393025	**393198**	390827.85	391957.00	**392508.30**
RanInt_n960_ds_01	1238944	1239019	**1243609**	1236212.25	1236760.85	**1241662.80**
RanInt_n960_ds_02	1236221	1236536	**1240503**	1233795.30	1234449.25	**1239303.65**
RanInt_n960_ds_03	1236516	1238062	**1241375**	1234214.00	1235002.35	**1239435.05**
RanInt_n960_ds_04	1237219	1238364	**1241400**	1235456.15	1236642.60	**1239878.15**
RanInt_n960_ds_05	1239017	1236447	**1239833**	1235692.80	1234464.35	**1238354.15**
RanInt_n960_ds_06	1234678	1234548	**1237280**	1233006.60	1232092.50	**1235741.05**
RanInt_n960_ds_07	1238255	1237651	**1241459**	1235368.65	1235281.30	**1240246.50**
RanInt_n960_ds_08	1234648	1234856	**1238195**	1232458.30	1232390.75	**1236536.85**
RanInt_n960_ds_09	1235323	1234310	**1240169**	1233074.55	1232668.80	**1237532.00**
RanInt_n960_ds_10	1237697	1237662	**1240515**	1235317.70	1236135.80	**1239079.30**
#Best	7	4	29	2	2	36
p-value	5.04e-4	9.55e-6		1.26e-8	1.26e-8	

Table 4
Comparison of the IMS algorithm with two state-of-the-art algorithms (i.e., ITS [22] and SGVNS [6]) on the RanInt instances with equal group sizes.

Instance	f_{best}			f_{avg}		
	ITS [22]	**SGVNS** [6]	**IMS**	**ITS** [22]	**SGVNS** [6]	**IMS**
RanInt_n120_ss_01	**47909**	**47909**	**47909**	**47898.35**	47893.25	47884.05
RanInt_n120_ss_02	**47826**	**47826**	**47826**	**47818.60**	47793.90	47794.85
RanInt_n120_ss_03	**47552**	**47552**	**47552**	**47490.20**	47475.25	47449.05
RanInt_n120_ss_04	**47611**	47556	**47611**	**47538.85**	47488.50	47514.45
RanInt_n120_ss_05	47148	**47191**	**47191**	**47122.70**	47106.65	47089.45
RanInt_n120_ss_06	**46647**	46641	46622	**46592.05**	46579.35	46556.05
RanInt_n120_ss_07	**47142**	**47142**	**47142**	**47111.45**	47085.65	47063.75
RanInt_n120_ss_08	**47390**	47356	**47390**	**47369.45**	47323.85	47325.25
RanInt_n120_ss_09	**47660**	**47660**	**47660**	**47636.10**	47628.15	47613.35
RanInt_n120_ss_10	**47807**	**47807**	**47807**	**47801.75**	47780.55	47778.25
RanInt_n240_ss_01	**155566**	155538	155474	155232.15	**155369.20**	155352.50
RanInt_n240_ss_02	155219	**155325**	155252	155088.50	155114.20	**155120.55**
RanInt_n240_ss_03	156325	156387	**156415**	156121.35	156226.85	**156308.30**
RanInt_n240_ss_04	156559	**156643**	156588	156381.25	156408.60	**156430.45**
RanInt_n240_ss_05	156499	**156562**	156479	156171.75	156185.85	**156305.35**
RanInt_n240_ss_06	155465	155516	**155536**	155149.25	**155314.85**	155265.95
RanInt_n240_ss_07	155770	**155729**	155697	155463.15	**155551.10**	155529.05
RanInt_n240_ss_08	155247	**155398**	**155398**	154965.05	**155095.15**	155068.60
RanInt_n240_ss_09	**156013**	155967	156012	155944.15	155800.35	**155991.90**
RanInt_n240_ss_10	155877	155947	**155971**	155670.60	155781.20	**155803.05**
RanInt_n480_ss_01	379455	379879	**380157**	378592.85	379264.40	**379678.05**
RanInt_n480_ss_02	379597	379628	**379992**	378531.60	379026.70	**379482.55**
RanInt_n480_ss_03	378511	379025	**379067**	377643.95	378490.25	**378634.70**
RanInt_n480_ss_04	378395	378936	**379067**	377815.00	378551.90	**378759.80**
RanInt_n480_ss_05	379627	379833	**380147**	378457.50	378946.45	**379316.45**
RanInt_n480_ss_06	379161	379304	**379681**	378067.85	378601.25	**379013.20**
RanInt_n480_ss_07	379325	379544	**380022**	378358.75	379068.55	**379297.20**
RanInt_n480_ss_08	379218	**380115**	380088	378435.25	379222.70	**379466.85**
RanInt_n480_ss_09	378383	378610	**379147**	377686.75	378111.20	**378526.75**
RanInt_n480_ss_10	379409	380291	**380615**	378905.25	379606.65	**379791.65**
RanInt_n960_ss_01	1217365	1217792	**1220724**	1215311.10	1215593.80	**1219004.55**
RanInt_n960_ss_02	1216579	1216227	**1220169**	1214601.85	1215048.10	**1218219.00**
RanInt_n960_ss_03	1217448	1217651	**1220210**	1215203.50	1215957.35	**1218739.85**
RanInt_n960_ss_04	1216950	1217392	**1220242**	1215775.35	1215559.60	**1218289.95**
RanInt_n960_ss_05	1217487	1217823	**1218916**	1214914.15	1215602.55	**1218060.15**
RanInt_n960_ss_06	1217904	1218274	**1220803**	1215522.80	1216142.75	**1218880.75**
RanInt_n960_ss_07	1218376	1217927	**1220234**	1215830.85	1216009.85	**1219180.90**
RanInt_n960_ss_08	1216778	1217721	**1220555**	1215293.95	1215895.80	**1219391.00**
RanInt_n960_ss_09	1215583	1216735	**1218576**	1213553.35	1214226.35	**1217497.95**
RanInt_n960_ss_10	1214982	1216159	**1219384**	1213743.10	1214343.60	**1217076.40**
#Best	11	13	32	10	4	26
p-value	1.01e-4	1.46e-3		1.57e-3	4.43e-3	

Table 5
Comparison of the IMS algorithm with two state-of-the-art algorithms (i.e., ITS [22] and SGVNS [6]) on the RanReal instances with different group sizes.

	f_{best}			f_{avg}		
Instance	**ITS** [22]	**SGVNS** [6]	**IMS**	**ITS** [22]	**SGVNS** [6]	**IMS**
RanReal_n120_ds_01	**50601.64**	50511.77	50471.07	**50405.97**	50233.42	50347.79
RanReal_n120_ds_02	50904.03	**50926.60**	50860.95	50691.85	50653.16	**50710.78**
RanReal_n120_ds_03	49935.33	**50053.22**	49961.99	49818.63	49769.97	**49858.20**
RanReal_n120_ds_04	50349.01	50362.19	**50382.69**	50164.48	50244.15	**50269.68**
RanReal_n120_ds_05	49414.58	49576.69	**49642.54**	49035.99	49389.44	**49426.52**
RanReal_n120_ds_06	**50287.42**	50189.78	50211.93	50120.37	50084.36	**50126.83**
RanReal_n120_ds_07	50074.29	**50300.67**	50107.47	49647.02	**50033.27**	50003.56
RanReal_n120_ds_08	50421.78	**50471.62**	50409.32	**50351.31**	50335.02	50323.86
RanReal_n120_ds_09	**50415.22**	50303.82	50339.42	50256.73	50135.17	**50268.61**
RanReal_n120_ds_10	49726.63	**49738.82**	49718.01	49552.19	49623.88	**49637.11**
RanReal_n240_ds_01	**160122.83**	160100.60	160020.56	159605.36	159658.09	**159838.57**
RanReal_n240_ds_02	**160502.19**	160195.86	160431.14	160056.73	159931.98	**160238.93**
RanReal_n240_ds_03	159389.94	**159489.46**	159404.12	159098.54	159194.52	**159267.93**
RanReal_n240_ds_04	161225.03	161268.00	**161398.07**	160540.42	160725.95	**161021.04**
RanReal_n240_ds_05	**159474.95**	159082.26	159249.91	158702.29	158863.94	**158897.28**
RanReal_n240_ds_06	**161293.41**	161149.24	160935.00	160554.40	160694.18	**160817.23**
RanReal_n240_ds_07	159940.05	**159993.01**	159919.07	159260.64	159256.38	**159689.35**
RanReal_n240_ds_08	**158630.55**	158455.65	158472.42	158194.33	158163.63	**158300.69**
RanReal_n240_ds_09	159707.67	159798.05	**159815.04**	159265.00	159426.01	**159572.93**
RanReal_n240_ds_10	159889.53	160116.32	**160145.72**	159586.36	159674.02	**159897.95**
RanReal_n480_ds_01	387444.56	388123.55	**388632.12**	386581.10	386880.96	**387893.38**
RanReal_n480_ds_02	386295.34	386545.55	**386668.04**	385302.04	385605.13	**386224.71**
RanReal_n480_ds_03	387474.69	387037.52	**388278.66**	386405.68	386385.61	**387672.35**
RanReal_n480_ds_04	389719.77	390468.63	**390477.24**	388910.57	389223.09	**390126.22**
RanReal_n480_ds_05	387556.78	387579.13	**387778.95**	386325.62	386550.27	**387336.34**
RanReal_n480_ds_06	388327.14	388955.10	**389390.05**	387713.26	388062.43	**388652.59**
RanReal_n480_ds_07	387953.20	388220.53	**389460.58**	387364.25	387277.33	**387956.27**
RanReal_n480_ds_08	389271.97	388704.19	**389316.71**	387671.57	387682.22	**388606.65**
RanReal_n480_ds_09	386976.68	386937.27	**387387.51**	385622.96	385829.80	**386709.00**
RanReal_n480_ds_10	391432.16	392583.25	**392665.57**	390233.98	391528.94	**391616.68**
RanReal_n960_ds_01	1237439.17	1236147.11	**1239969.65**	1233118.83	1233890.11	**1238909.72**
RanReal_n960_ds_02	1234310.30	1235681.38	**1240510.37**	1232084.77	1234143.93	**1238715.63**
RanReal_n960_ds_03	1234380.99	1235680.62	**1238336.81**	1232468.76	1232674.97	**1236951.56**
RanReal_n960_ds_04	1234687.12	1236117.09	**1240460.96**	1232447.56	1234544.61	**1238715.10**
RanReal_n960_ds_05	1232616.47	1232701.17	**1237177.90**	1231403.10	1231062.44	**1235398.21**
RanReal_n960_ds_06	1230272.69	1229937.27	**1234363.75**	1228455.18	1228663.53	**1232633.30**
RanReal_n960_ds_07	1233947.58	1235410.78	**1238781.36**	1232238.16	1231707.94	**1237068.09**
RanReal_n960_ds_08	1229287.84	1228957.27	**1233177.59**	1227839.29	1227748.55	**1231635.91**
RanReal_n960_ds_09	1234655.22	1235415.73	**1239566.10**	1232440.67	1233119.91	**1236338.30**
RanReal_n960_ds_10	1236743.27	1237850.16	**1240641.54**	1234957.71	1236045.50	**1239307.80**
#Best	8	7	25	2	1	37
p-value	1.14e-2	1.57e-3		1.26e-8	1.26e-8	

Table 6
Comparison of the IMS algorithm with two state-of-the-art algorithms (i.e., ITS [22] and SGVNS [6]) on the RanReal instances with equal group sizes.

	f_{best}			f_{avg}		
Instance	**ITS** [22]	**SGVNS** [6]	**IMS**	**ITS** [22]	**SGVNS** [6]	**IMS**
RanReal_n120_ss_01	**47363.21**	**47363.21**	47351.97	**47320.20**	47290.96	47278.28
RanReal_n120_ss_02	**47243.16**	**47243.16**	**47243.16**	**47201.87**	47174.17	47169.16
RanReal_n120_ss_03	**47313.71**	**47313.71**	**47313.71**	47269.91	47252.58	**47270.55**
RanReal_n120_ss_04	**47546.81**	47520.66	**47546.81**	**47500.66**	47460.04	47487.37
RanReal_n120_ss_05	**46930.19**	46896.02	46922.95	**46862.90**	46840.55	46836.60
RanReal_n120_ss_06	**47253.47**	47218.03	47227.14	**47201.03**	47159.19	47155.10
RanReal_n120_ss_07	**47085.87**	**47085.87**	**47085.87**	**47048.75**	47013.89	47016.85
RanReal_n120_ss_08	**47460.13**	**47460.13**	**47460.13**	**47456.74**	47443.28	47447.14
RanReal_n120_ss_09	**47686.34**	**47686.34**	**47686.34**	**47651.17**	47647.16	47623.66
RanReal_n120_ss_10	**47415.35**	**47415.35**	**47415.35**	47364.78	47370.57	**47383.58**
RanReal_n240_ss_01	**155219.12**	155213.84	155214.55	154883.12	154945.19	**155014.44**
RanReal_n240_ss_02	155474.95	**155530.68**	155503.11	155298.19	**155364.94**	155335.96
RanReal_n240_ss_03	155546.38	**155669.29**	155633.92	155285.92	155423.50	**155458.48**
RanReal_n240_ss_04	155275.08	155335.86	**155364.27**	155007.57	**155194.75**	155101.99
RanReal_n240_ss_05	154794.27	**154978.13**	154924.97	154624.80	**154758.73**	154751.91
RanReal_n240_ss_06	155571.81	155498.40	**155671.23**	155244.62	**155344.28**	155305.53
RanReal_n240_ss_07	155585.37	**155689.23**	**155689.23**	155388.71	155386.65	**155476.73**
RanReal_n240_ss_08	155578.35	**155604.41**	155545.42	155427.17	155432.26	**155451.77**
RanReal_n240_ss_09	155089.50	155084.04	**155090.44**	154725.08	**154851.60**	154828.30
RanReal_n240_ss_10	155831.39	**155927.91**	155880.48	155640.99	155715.16	**155720.89**
RanReal_n480_ss_01	377254.93	378091.43	**378207.04**	376577.67	377257.45	**377541.47**
RanReal_n480_ss_02	377084.52	377704.68	**377864.04**	376360.69	376868.37	**377261.07**
RanReal_n480_ss_03	378407.16	378727.11	**379053.79**	377642.01	378210.72	**378523.05**
RanReal_n480_ss_04	377512.67	377859.37	**377964.66**	376680.91	377236.63	**377436.07**
RanReal_n480_ss_05	377637.09	378315.79	**378634.46**	376933.62	377639.49	**378035.78**
RanReal_n480_ss_06	378561.19	**379391.76**	379274.93	377684.89	378435.81	**378634.83**
RanReal_n480_ss_07	378527.49	378960.02	**379542.61**	377868.64	378356.28	**378830.49**
RanReal_n480_ss_08	377521.79	377855.12	**378204.40**	376841.61	377458.17	**377480.99**
RanReal_n480_ss_09	377007.39	377933.93	**378220.84**	376381.06	377123.57	**377425.71**
RanReal_n480_ss_10	379490.57	**379588.71**	379495.84	378242.95	**378988.96**	378904.41
RanReal_n960_ss_01	1213571.81	1214889.59	**1217164.60**	1211655.10	1212393.84	**1215127.66**
RanReal_n960_ss_02	1215220.12	1215624.91	**1218155.24**	1213054.14	1213697.16	**1217047.21**
RanReal_n960_ss_03	1214457.16	1215262.94	**1217349.78**	1213049.09	1213298.63	**1215606.06**
RanReal_n960_ss_04	1214677.73	1215137.53	**1218053.67**	1213221.74	1213708.91	**1217062.32**
RanReal_n960_ss_05	1213355.93	1213915.65	**1216203.18**	1211601.14	1211816.82	**1214833.11**
RanReal_n960_ss_06	1213779.37	1214390.66	**1217570.66**	1211693.78	1212621.07	**1215622.87**
RanReal_n960_ss_07	1215371.11	1214000.10	**1217445.63**	1212882.20	1212855.81	**1215974.81**
RanReal_n960_ss_08	1213216.15	1213071.85	**1216067.45**	1211196.00	1211740.28	**1214937.67**
RanReal_n960_ss_09	1214408.56	1215097.64	**1217955.76**	1212479.70	1213411.80	**1216443.45**
RanReal_n960_ss_10	1216148.98	1215828.57	**1218299.76**	1213816.44	1214373.54	**1217314.64**
#Best	11	15	29	8	6	26
p-value	6.23e-05	3.08e-3		1.48e-4	4.43e-3	

Table 7
Comparison of the IMS algorithm with two state-of-the-art algorithms (i.e., ITS [22] and SGVNS [6]) on the Geo instances with different group sizes.

	f_{best}			f_{avg}		
Instance	**ITS** [22]	**SGVNS** [6]	**IMS**	**ITS** [22]	**SGVNS** [6]	**IMS**
Geo_n120_ds_01	**111939.28**	111873.75	111902.37	**111911.48**	111098.29	111884.65
Geo_n120_ds_02	**61916.95**	61890.78	61909.56	**61903.64**	61338.46	61902.33
Geo_n120_ds_03	**52083.72**	52061.00	52078.00	**52073.17**	51412.27	52069.82
Geo_n120_ds_04	**80789.19**	80759.66	80776.60	**80775.83**	80459.69	80764.28
Geo_n120_ds_05	**121777.06**	121679.46	121739.99	**121735.46**	120417.06	121698.37
Geo_n120_ds_06	**136860.93**	136815.05	136839.56	136712.49	136371.37	**136817.94**
Geo_n120_ds_07	**108557.01**	108489.35	108547.97	108466.72	107518.94	**108511.23**
Geo_n120_ds_08	**88225.79**	88172.62	88187.00	**88206.58**	88148.69	88178.09
Geo_n120_ds_09	**95508.40**	95410.84	95466.84	**95474.05**	94879.69	95445.76
Geo_n120_ds_10	**65560.54**	65521.62	65549.22	**65538.04**	65356.49	65531.18
Geo_n240_ds_01	**200357.13**	200307.82	200336.96	200303.94	199319.63	**200327.31**
Geo_n240_ds_02	**348512.21**	348406.58	348479.43	347585.05	346206.86	**348465.69**
Geo_n240_ds_03	217159.11	217105.66	**217178.64**	217124.09	214955.90	**217155.76**
Geo_n240_ds_04	**263859.76**	263765.00	263835.63	263538.75	261655.07	**263587.65**
Geo_n240_ds_05	**313402.77**	313290.69	313360.73	**313377.51**	312007.31	313336.36
Geo_n240_ds_06	**358632.56**	358456.36	358590.94	**358605.22**	357441.51	358550.91
Geo_n240_ds_07	**341992.17**	341172.78	341980.32	340415.53	339181.31	**341592.84**
Geo_n240_ds_08	131024.86	131024.78	**131026.39**	131020.88	130676.72	**131021.76**
Geo_n240_ds_09	**410563.19**	410379.34	410521.01	409945.98	408441.93	**410464.95**
Geo_n240_ds_10	**355254.36**	353823.23	355198.61	355017.34	352252.56	**355184.75**
Geo_n480_ds_01	580921.73	581315.05	**582479.04**	580805.52	576850.95	**582123.61**
Geo_n480_ds_02	1089043.66	1089409.66	**1090000.26**	1088977.03	1084763.97	**1089716.53**
Geo_n480_ds_03	662588.72	662622.55	**664319.28**	661944.19	658367.03	**663591.45**
Geo_n480_ds_04	**836597.00**	835376.73	836397.62	**836532.13**	831310.73	836358.87
Geo_n480_ds_05	**988491.52**	986287.96	988328.45	986678.57	979722.93	**987742.16**
Geo_n480_ds_06	**1012562.54**	1010237.14	1012489.37	1011750.26	1006038.94	**1012037.76**
Geo_n480_ds_07	**864919.59**	864225.44	864771.60	**864710.73**	860095.40	864281.45
Geo_n480_ds_08	587651.95	587419.70	**587805.09**	587283.15	584076.35	**587506.18**
Geo_n480_ds_09	666297.20	667220.91	**667420.39**	666229.66	663568.54	**667309.42**
Geo_n480_ds_10	932726.34	936532.52	**937476.35**	929792.10	930420.08	**936650.57**
Geo_n960_ds_01	3361972.63	3357246.28	**3364976.99**	3349073.54	3343507.03	**3362203.74**
Geo_n960_ds_02	1719714.70	1720538.84	**1722789.69**	1716091.63	1714377.84	**1721443.28**
Geo_n960_ds_03	3347799.90	3345716.22	**3351291.12**	3347722.82	3335941.16	**3348867.32**
Geo_n960_ds_04	3614983.35	3620711.81	**3623347.13**	3604535.13	3605566.68	**3621476.68**
Geo_n960_ds_05	**2342436.81**	2341962.44	2342220.13	**2342276.35**	2330534.53	2341420.32
Geo_n960_ds_06	**3153302.03**	3148820.77	3152992.29	**3151552.96**	3141553.49	3151442.92
Geo_n960_ds_07	1301825.26	1298267.73	**1301844.91**	1298534.13	1292837.36	**1300216.92**
Geo_n960_ds_08	1721921.20	1720719.54	**1723602.17**	1720286.62	1716296.69	**1723335.84**
Geo_n960_ds_09	1894753.44	1896003.92	**1897535.27**	1891513.59	1889095.78	**1896214.30**
Geo_n960_ds_10	2617039.54	2615485.60	**2618015.72**	2615978.35	2607823.21	**2617741.80**
#Best	24	0	16	14	0	26
p-value	2.06e-01	2.54e-10		5.78e-2	2.54e-10	

3.4 Computational Results and Comparison on the Large Scale Instances

To assess the performance of our IMS algorithm on large-sized instances, we carried out the second experiment based on the MDG-a and MDG-c benchmarks which include 11 MDG-a subsets and 2 MDG-c subsets, where each subset contains 20 instances with $n = 2000$ or 3000. In this experiment, each instance was independently solved 20 times using the IMS, ITS, and SGVNS algorithms respectively. A summary of the results on these 13 benchmark subsets are shown in Table 9, and the detailed computational results for each subset are provided in Tables A.1 – A.13 of the appendix.

In Table 9, the first five columns indicate the characteristics of instances for each subset of benchmarks: the set name (Set), the number of groups (m), the lower and upper limits of group sizes (a_g and b_g), and the type of instances ($Type$). Each row of Table 9 corresponds to one of the 13 tested benchmark subsets and the shown results are based on the averages over all instances of each benchmark set. The row 'Average' indicates the average results of each algorithm over all subsets in terms of the best and average objective values. Once again, the non-parametric Friedman tests were used to verify the statistical significance between our IMS algorithm and the reference algorithms.

Table 8
Comparison of the IMS algorithm with two state-of-the-art algorithms (i.e., ITS [22] and SGVNS [6]) on the Geo instances with equal group sizes.

Instance	f_{best}			f_{avg}		
	ITS [22]	**SGVNS** [6]	**IMS**	**ITS** [22]	**SGVNS** [6]	**IMS**
Geo_n120_ss_01	**101625.34**	101582.46	101611.27	**101610.59**	101563.00	101595.23
Geo_n120_ss_02	**54852.92**	54831.10	54840.03	**54846.70**	54822.33	54835.58
Geo_n120_ss_03	**47631.33**	47615.71	47621.57	**47626.08**	47610.72	47616.99
Geo_n120_ss_04	**73513.57**	73466.35	73491.50	**73498.55**	73453.53	73483.01
Geo_n120_ss_05	**112657.57**	112621.83	112641.17	**112631.14**	112573.83	112613.01
Geo_n120_ss_06	**125424.91**	125375.10	125399.82	**125404.69**	125345.95	125380.47
Geo_n120_ss_07	**98499.68**	98448.17	98493.37	**98483.28**	98429.09	98466.30
Geo_n120_ss_08	**79982.16**	79925.82	79962.64	**79963.75**	79910.80	79947.92
Geo_n120_ss_09	**87281.56**	87217.23	87263.07	**87260.55**	87206.68	87243.30
Geo_n120_ss_10	**60258.25**	60223.82	60249.31	**60248.61**	60214.08	60233.23
Geo_n240_ss_01	**188872.18**	188820.43	188848.74	**188854.87**	188811.27	188833.16
Geo_n240_ss_02	**330349.92**	330221.98	330285.51	**330303.80**	330194.62	330261.08
Geo_n240_ss_03	**207066.88**	207004.15	207040.96	**207055.28**	206994.19	207028.70
Geo_n240_ss_04	**246387.08**	246293.80	246350.15	**246364.23**	246285.18	246333.52
Geo_n240_ss_05	**298773.97**	298663.11	298729.45	**298738.96**	298631.63	298700.27
Geo_n240_ss_06	**338596.95**	338485.52	338562.44	**338566.51**	338455.23	338526.47
Geo_n240_ss_07	**326061.15**	325947.58	326039.00	**326036.28**	325925.53	325995.21
Geo_n240_ss_08	**126911.48**	126899.34	126903.24	**126908.32**	126897.58	126898.69
Geo_n240_ss_09	**391469.54**	391315.30	391383.98	**391413.51**	391299.48	391365.47
Geo_n240_ss_10	**339533.45**	339426.07	339488.03	**339506.87**	339395.17	339468.43
Geo_n480_ss_01	**552175.76**	552029.26	552104.49	**552155.33**	552012.56	552078.31
Geo_n480_ss_02	**1047450.18**	1047180.27	1047314.11	**1047387.49**	1047128.56	1047241.73
Geo_n480_ss_03	**633760.55**	633556.01	633647.36	**633729.27**	633531.02	633623.07
Geo_n480_ss_04	**789817.46**	789565.85	789709.97	**789778.49**	789536.82	789659.90
Geo_n480_ss_05	**945933.15**	945715.47	945800.11	**945887.14**	945642.66	945743.46
Geo_n480_ss_06	**966654.45**	966380.61	966478.82	**966585.10**	966325.97	966431.60
Geo_n480_ss_07	**827713.64**	827460.21	827558.35	**827658.89**	827399.66	827527.35
Geo_n480_ss_08	**556651.12**	556507.04	556565.16	**556632.18**	556478.77	556538.12
Geo_n480_ss_09	**636342.86**	636150.41	636253.75	**636319.08**	636131.25	636224.49
Geo_n480_ss_10	**883411.86**	883126.77	883313.08	**883359.36**	883097.22	883232.13
Geo_n960_ss_01	**3254400.00**	3253897.94	3254071.56	**3254283.99**	3253844.76	3254022.12
Geo_n960_ss_02	**1663655.10**	1663453.87	1663496.02	**1663632.19**	1663430.65	1663471.31
Geo_n960_ss_03	**3251592.49**	3251133.28	3251368.68	**3251534.31**	3251076.91	3251289.52
Geo_n960_ss_04	**3514369.69**	3513845.78	3513995.60	**3514217.91**	3513760.67	3513948.54
Geo_n960_ss_05	**2264719.61**	2264438.13	2264551.72	**2264687.79**	2264388.84	2264524.29
Geo_n960_ss_06	**3069667.72**	3069224.57	3069457.37	**3069579.72**	3069155.74	3069358.54
Geo_n960_ss_07	**1257746.77**	1257645.60	1257632.82	**1257733.35**	1257629.61	1257624.38
Geo_n960_ss_08	**1674028.54**	1673798.98	1673846.57	**1673989.19**	1673778.03	1673825.81
Geo_n960_ss_09	**1835473.14**	1835246.63	1835324.69	**1835440.88**	1835216.53	1835289.44
Geo_n960_ss_10	**2528974.50**	2528612.67	2528822.78	**2528919.65**	2528571.34	2528761.90
#Best	40	0	0	40	0	0
p-value	2.54e-10	1.87e-9		2.54e-10	1.87e-9	

Table 9
Summary comparison of the IMS algorithm with two state-of-the-art algorithms (ITS [22] and SGVNS [6]) on the large instances with $n = 2000$ (220 MDG-a instances) and $n = 3000$ (40 MDG-c instances). The best results among the compared algorithms are indicated in bold.

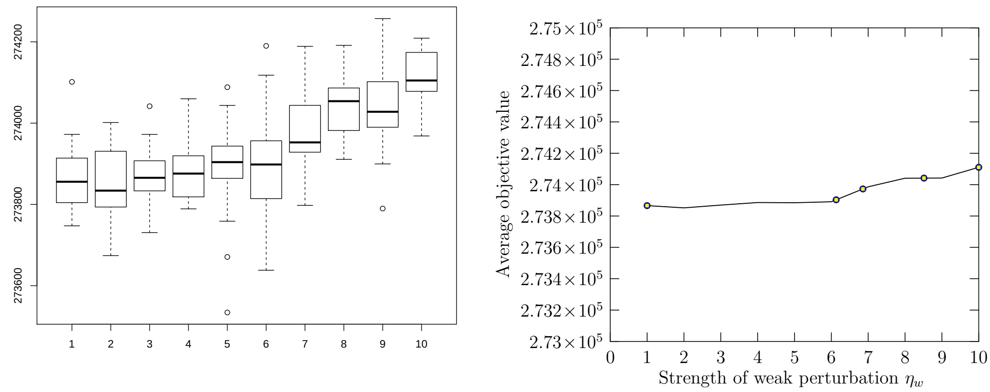
Instance set					f_{best}			f_{avg}		
Set	m	a_g	b_g	*Type*	**ITS** [22]	**SGVNS** [6]	**IMS**	**ITS** [22]	**SGVNS** [6]	**IMS**
MDG-a	10	173	227	DGS	1133961.05	1132509.35	**1135634.25**	1133249.54	1131536.17	**1135113.69**
MDG-a	10	200	200	EGS	1115340.65	1113709.60	**1116975.90**	1114591.49	1112901.98	**1116489.20**
MDG-a	25	51	109	DGS	539418.00	539519.80	**541147.35**	538864.09	538965.23	**540692.18**
MDG-a	25	80	80	EGS	485946.35	485936.70	**487501.50**	485315.72	485465.81	**487151.50**
MDG-a	50	32	48	DGS	272656.00	273440.75	**274262.10**	272184.14	273105.67	**273929.08**
MDG-a	50	26	54	DGS	291116.55	291739.10	**292585.85**	290577.75	291347.52	**292231.91**
MDG-a	50	40	40	EGS	263767.03	264564.90	**265342.55**	263266.11	264211.53	**265009.32**
MDG-a	100	13	27	DGS	158970.90	159544.05	**159905.80**	158552.62	159223.94	**159638.19**
MDG-a	100	20	20	EGS	144118.80	144614.35	**144918.35**	143684.40	144325.44	**144667.81**
MDG-a	200	6	14	DGS	88435.70	88535.80	**88721.50**	88099.37	88262.95	**88549.32**
MDG-a	200	10	10	EGS	77208.50	76587.30	**77242.80**	76907.49	76459.62	**77098.39**
MDG-c	50	48	72	DGS	57908059.10	58034797.30	**58195733.30**	57830101.16	57979531.87	**58149757.61**
MDG-c	50	60	60	EGS	55958533.45	56087667.95	**56245400.90**	55877999.10	56031642.09	**56201493.19**
Average					9110579.39	9130243.61	**9155797.86**	9097953.31	9121306.14	**9148601.65**
p-value					3.12e-4	3.12e-4		3.12e-4	3.12e-4	

Table 9 discloses that for all 13 subsets of large instances, IMS outperforms the reference algorithms ITS and SGVNS. IMS yields better results (indicated in bold) for all subsets in terms of the best and average objective values. Moreover, the p-values (all < 0.05) confirm the significance of the differences between the results of IMS and those of the reference algorithms. This experiment indicates that IMS is very competitive when it is applied to large-sized instances compared to the state-of-the-art algorithms in the literature.

4 Analysis and Discussions

In this section, we investigate some important ingredients of the IMS algorithm to get useful insight into the behavior of the proposed algorithm. This study is based on the first subset of the MDG-a benchmark which is composed of 20 DGS instances with $n = 2000$, $m = 50$, $a_g = 32$, and $b_g = 48$. We also study the spatial distribution of high-quality solutions visited by IMS to shed light on the rationality of the proposed algorithm.

4.1 Influence of the Weak Perturbation Strength

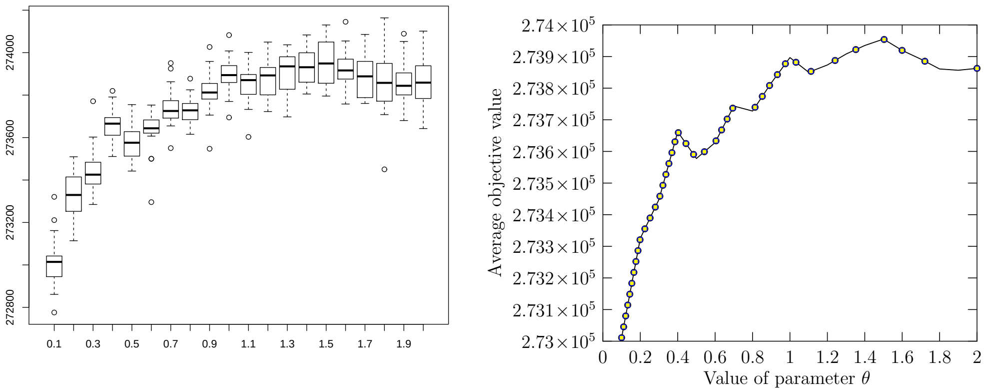


(a) (b)

Fig. 1. Influence of the strength of the weak perturbation procedure

The IMS algorithm uses a weak perturbation procedure in its maxima search (Section 2.5). We carried out an experiment to analyze the influence of this component on the performance of IMS. For this study, we varied the strength (η_w) of weak perturbation within a reasonable range while keeping other parameters with their default values (as shown in Table 2). The box and whisker plots of the computational results are shown in Fig. 1, where the X-axis of Fig. 1(a) indicates the test η_w values and the Y-axis indicates the objective values. As a supplement, we also plot the average objective values over the tested instances in Fig. 1(b), where the X-axis and the Y-axis respectively indicate the η_w values and the average objective values over the benchmark instances. All indicated results were based on the average over 20 independent runs with the cutoff time given in Section 3.2. Fig. 1 discloses that the performance of the IMS algorithm is not sensitive to η_w. For all the values of η_w in the interval $[1, 6]$, the algorithm obtains a similar performance. This explain why we used $\eta_w = 3$ as its default value for our experiments.

4.2 Influence of the Strong Perturbations Strength



(a) (b)

Fig. 2. Influence of the strength of the strong perturbation procedure

We turn now to study the influence of the strength (η_s) of strong perturbation on the IMS algorithm (Section 2.6). Since η_s is proportional to the parameter θ (recall that $\eta_s = \theta \times \frac{n}{m}$), we tested the parameter θ within a reasonable range while keeping other parameters with their default values. The computational results are shown in Fig. 2. In Fig. 2(a), the X-axis and the Y-axis respectively indicate the θ values and the objective values obtained. In Fig. 2(b), the X-axis and the Y-axis respectively indicate the θ values and the average objective values over the benchmark instances. Like in Section 4.1, these results were also based on the average over 20 independent runs.

One observes from Fig. 2 that the performance of the IMS algorithm is significantly influenced by the setting of θ. First, small (large) θ values corresponding to a small (large) perturbation lead generally to a bad (high) performance. Furthermore, when θ reaches 1.5 which corresponds to a strength of $\eta_s = 60$ for the tested instances, the IMS algorithm reaches its best performance. As θ further increases, the performance of the algorithm decreases gradually. This experiment shows that the IMS algorithm is sensitive to the parameter θ, and a large value is more appropriate for the large-sized instances. Therefore, in this study the default value of θ was set to 1.5 for the instances with $n > 400$, and 1.0 for other instances.

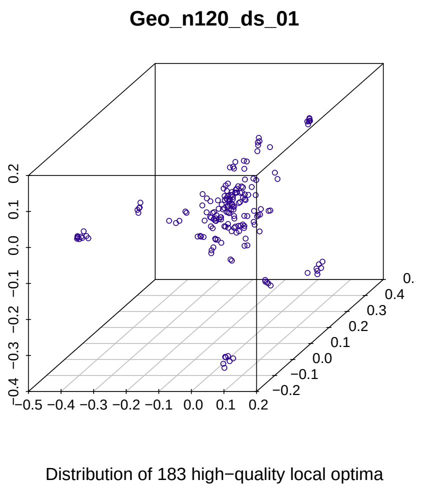
4.3 Spatial Distribution of High-Quality Solutions

To shed insight on the rationality of the proposed IMS algorithm, we carried out an experimental study on spatial distribution of high-quality local optimum solutions. This study, inspired by [23], helps to understand why altering between maxima search and strong perturbation during the IMS process is meaningful. This study was based on 6 representative RanInt, RanReal, and Geo instances with $n = 120$. In this experiment, each instance was solved one time by running our IMS algorithm with a cutoff time of 1 minute, and the distinct high-quality local optimum solutions encountered during the run were recorded. For the 2 RanInt and 2 RanReal instances, the local optimum solutions with an objective value of $f \geq 0.99 \times f_B$ are considered as high-quality, where f_B represents the best objective value found in this work for the given instance. As for two 2 Geo instances, the local optimum solutions with an objective value of $f \geq 0.999 \times f_B$ are considered as high-quality.

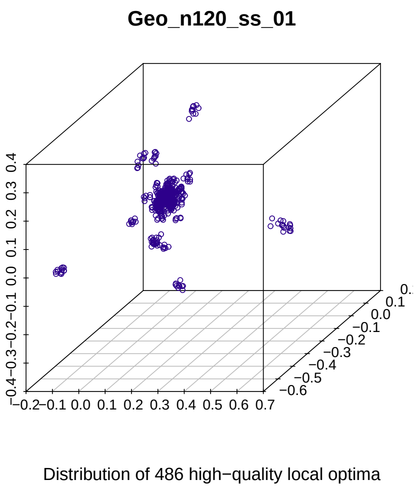
Following [23], to obtain an intuitive image of the spatial distribution of high-quality local optimum solutions, we employ the multidimensional scaling (MDS) procedure to generate the distribution in Euclidean R^3 space, where the MDS procedure is composed of the following two steps.

- Step 1: We first construct the distance matrix $D_{p\times p}$ in the real search space Ω (a n-dimensional space), where the entry $d_{ab} \in D_{p\times p}$ represents the distance between solutions s_a and s_b, and p represents the total number of the local optimum solutions sampled. Given two solutions $s_a = (y^a[1], y^a[2], \ldots, y^a[n])$ and $s_b = (y^b[1], y^b[2], \ldots, y^b[n])$, we adopt the distance function given in [6] to calculate their distance, which is defined as follows:
$$d_{ab} = \frac{|(i,j):((y^a[i]=y^a[j] \wedge y^b[i]\neq y^b[j]) \vee (y^a[i]\neq y^a[j] \wedge y^b[i]=y^b[j]))|}{n^2/m}$$
As shown in [6], d_{ab} estimates the "fraction" of pairs belonging to the same group in one solution, but not to the same group in another solution, and it is suitable to represent the distance between two partitions.
- Step 2: According to the distance matrix $D_{p\times p}$, we then generate p coordinate points in the Euclidean space R^3 by the classic *cmdscale* algorithm implemented in the R language, where each coordinate point corresponds to one of p solutions. The goal of the *cmdscale* algorithm is to map the points in n-dimensional space into Euclidean space R^3 while minimizing the distance distortion. Thus, the distance between two points in R^3 approximately equals to that in the original n-dimensional space. Finally, we plot a scatter graph of the obtained points in R^3.

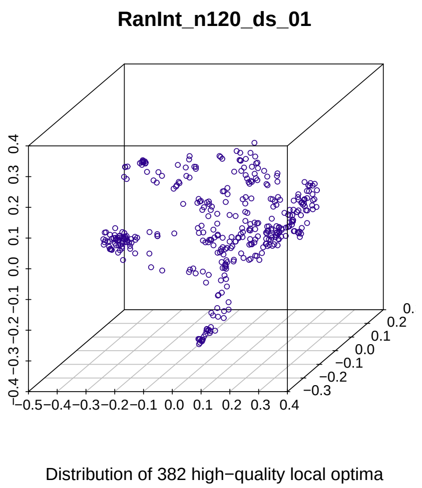
The 3D scatter graph of the high-quality local optimum solutions produced in the experiment is plotted in Fig. 3 for each instance. Interestingly, Fig. 3 discloses that high-quality local optimum solutions tent to be grouped in clusters, which has two relevant implications. On the one hand, the distances between high-quality local optimum solutions within the same cluster are in general small, which implies that it is very helpful for the search algorithm to reinforce its exploitation ability to detect other high-quality local optima in the current region of the search space. On the other hand, the local optimum solutions locating in different clusters are in general separated by a large dis-



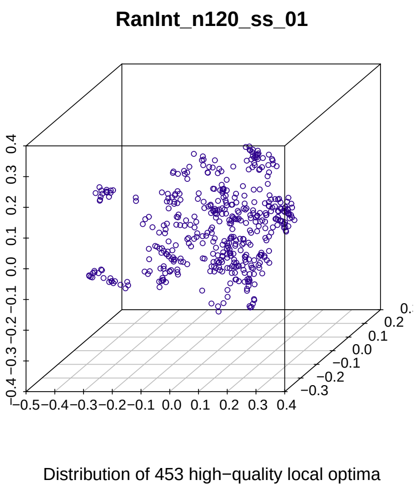
(a)



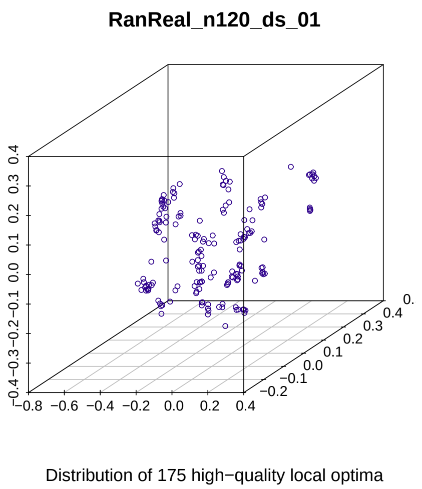
(b)



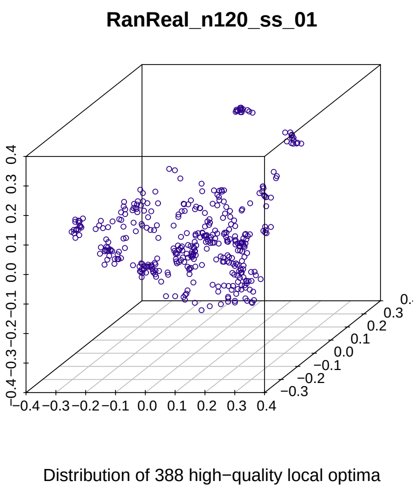
(c)



(d)



(e)



(f)

Fig. 3. Distribution of high-quality solutions produced by the IMS algorithm on the representative instances.

tance. To continue the search effectively, it is useful for a heuristic algorithm to be able to "jump" from from one cluster to another cluster (i.e., to escape from local optimum traps).

Based on the above analysis, the rationality of the proposed IMS algorithm can be explained as follows. On the one hand, the maxima search procedure uses its weak perturbation procedure to visit other (hopefully better) local optima in the nearby areas of an attained local optimum (i.e., corresponding to a cluster). On the other hand, the strong perturbation procedure of the IMS algorithm helps the search to move from one attraction area (corresponding to a cluster) to another one. As a result, the combined use of the maxima search procedure and the strong perturbation procedure ensures a kind of balance between intensification and diversification of its search process, which help to locate high quality solutions.

Finally, one notices that the cluster characteristic of high-quality local optimum solutions was previously observed for other well-known problems like TSP [5] and graph coloring [23].

5 Conclusions

This paper introduced an iterated maxima search (IMS) algorithm for solving the maximally diverse grouping problem (MDGP). IMS effectively integrates a fast local search, a weak perturbation procedure, and a strong perturbation procedure to ensure a suitable trade-off between intensification and diversification of its search process. The performance of the proposed algorithm was evaluated on five sets of 500 benchmark instances with various characteristics. The experimental results showed that IMS is very competitive compared to existing top performing algorithms in the literature and performs particularly well on large-sized instances. To provide some insight about the proposed algorithm, we studied the influence of the weak and strong perturbation procedures. We also investigated the spatial distribution of high-quality local optima with the purpose of shedding light on the rationality of the proposed algorithm.

In addition to the updated best results achieved by the proposed algorithm which are useful to assess new MDGP algorithms, the key ideas of this work could help to design effective heuristics for other related grouping or clustering problems involving dense graphs. Finally, like for most heuristic and metaheuristic algorithms, our current knowledge does not allow us to explain theoretically why the proposed algorithm works. More effort on such fundamental aspects of heuristic search is needed in the long term with the purpose of achieving a better understanding of heuristic algorithms.

Acknowledgments

We are grateful to the reviewers for their insightful comments which helped us to improve the paper. We thank Dr. D. Urošević for providing us with the executable code of their SGVNS algorithm as well as his kind help, Dr. G. Palubeckis for providing the source code of their ITS algorithm, and Dr. F.J. Rodriguez for his help on an early version of this study. The work is partially supported by the LigeRo project (2009-2013, No. 0012, Pays de la Loire region, France), the PGMO project (2014-0024H, Jacques Hadamard Mathematical Foundation) and a post-doc grant from the Pays de la Loire region (France).

References

[1] Arani T., Lofti V., 1989, A three phased approach to final exam scheduling. *IIE Transactions* 21(1), 86–96.

[2] Benlic U., Hao J.K., 2013a, Breakout local search for the vertex separator problem. In F. Rossi (Ed.): Proc. of the 23th Intl. Joint Conference on Artificial Intelligence (IJCAI-13), IJCAI/AAAI Press, pages 461-467, Beijing, China,

[3] Benlic U., Hao J.K., 2013b, Breakout local search for the max-cut problem. Engineering Applications of Artificial Intelligence 26(3), 1162-1173.

[4] Bhadury J., Mighty E., Damar H., 2000, Maximizing workforce diversity in project teams: a network flow approach, *Omega* 28(2), 143–153.

[5] Boese K. D., Kahng A. B., Muddu. S., 1994, A new adaptive multi-start technique for combinatorial global optimizations. *Operations Research Letters*, 16, 101–113.

[6] Brimberg J., Mladenović N., Urošević D., 2015, Solving the maximally diverse grouping problem by skewed general variable neighborhodd search. *Information Sciences* 295, 650–675.

[7] Chen Y., Fan Z.P., Ma J., Zeng S., 2011, A hybrid grouping genetic algorithm for reviewier group construction problem. *Expert Systems with Applications* 38(3), 2401–2411.

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

[9] Fan Z.P., Chen Y., Ma J., Zeng S., 2010, A hybrid genetic algorithmic approach to the maximally diverse grouping problem. *Journal of the Operational Research Society* 62, 92–99.

[10] Feo T., Khellaf M., 1990, A class of bounded approximation algorithms for graph partitioning. *Networks* 20(2) 181–195.

[11] Ferreira C.E., Martin A, Souza C.C., Weismantel R., Wolsey L.A., 1998, The node capacitated graph partitioning problem: a computational study. *Mathematical Programming* 81(2), 229–256.

[12] Ferreira C.E., Martin A, Souza C.C., Weismantel R., Wolsey L.A., 1996, Formulations and valid inequalities for the node capacitated graph partitioning problem. *Mathematical Programming* 74(3), 247–266.

[13] Fu Z.H., Hao J.K., 2015, A three-phase search approach for the quadratic minimum spanning tree problem. *Engineering Applications of Artificial Intelligence* 46: 113-130.

[14] Gallego M., Laguna M., Martí R., Duarte A., 2013, Tabu search with strategic oscillation for the maximally diverse grouping problem. *Journal of the Operational Research Society* 64, 724–734.

[15] Johnes J., 2015, Operational Research in education. *European Journal of Operational Research* 243(3), 683–696.

[16] Johnson E.L., Mehrotra A., Nemhauser G.L., 1993, Min-cut clustering. *Mathematical Programming* 62(1-3), 133–151.

[17] Krass D., Ovchinnikov A., 2010, Constrained group balancing: Why does it work. *European Journal of Operational Research* 206(1), 144–154.

[18] Lorena N., Antonio L., 2001, Constructive genetic algorithm for clustering problems. *Evolutionary Computation* 9(3), 309–327.

[19] Lourenco, H.R., Martin, O., & Stützle, T., 2003, Iterated local search. In Glover F., Kochenberger G. (Eds.), Handbook of Metaheuristics, Kluwer.

[20] Özsoy F.A., Labbé M., 2010, Size-constrained graph partitioning polytopes. *Discrete Mathematics* 310(24), 3473–3493.

[21] Palubeckis G., Karčiauskas E., Riškus A., 2011, Comparative performance of three metaheuristic approaches for the maximally diverse grouping problem. *Information Technology and Control* 40(4), 277–285.

[22] Palubeckis G., Ostreika A., Rubliauskas D., 2015, Maximally diverse grouping: an iterated tabu search approach. *Journal of the Operational Research Society* 66(4), 579–592.

[23] Porumbel D.C., Hao J.K., Kuntz P, 2010, A search space cartography for guiding graph coloring heuristics. *Computers & Operations Research* 37(4): 769-778.

[24] Rodriguez F.J., Lozano M., García-Martínez C., González-Barrera J.D., 2013, An artificial bee colony algorithm for the maximally diverse grouping problem. *Information Sciences* 230(1), 183–196.

[25] Urošević D., 2014, Variable neighborhood search for maximum diverse grouping problem. *Yugoslav Journal of Operations Research* 24(1), 21–33.

[26] Wang H., Alidaee B., Glover F., Kochenberger G., 2006, Solving group technology problems via clique partitioning. *International Journal of Flexible Manufacturing Systems* 18(2), 77–98.

[27] Weitz R., Jelassi M.T., 1992, Assigning students to groups: a multi-criteria decision support system approach. *Decision Sciences* 23(3), 746–757.

[28] Weitz R., Lakshminarayan S., 1997, An empirical comparison of heuristic and graph theoretic methods for creating maximally diverse groups, VLSI design, and exam scheduling. *Omega* 25(4), 473–482.

[29] Weitz R., Lakshminarayan S., 1998, An empirical comparison of heuristic methods for creating maximally diverse groups. *Journal of the Operational Research Society* 49, 635–646.

[30] Yeoh H.K., Nor M.I.M, 2011, An algorithm to form balanced and diverse groups of students. *Computer Applications in Engineering Education* 19(3), 582–590.

A Appendix

This appendix complements the presentation of Section 3.4 and contains the detailed results of the IMS algorithm on the large instances with $n = 2000$ (220 MDG-a instances) and $n = 3000$ (40 MDG-c instances) in comparison with two state-of-the-art algorithms (ITS [22] and SGVNS [6]). Tables A.1–A.13 present respectively the results of the compared algorithms on each of the 11 MDG-a subsets and 2 MDG-c subsets (20 instances per group) in terms of the best and average objective values (based on 20 independent runs per instance, see Section 3.2 for the experimental protocol). The row 'Average' shows the averaged value for each subset of instances. The row '#Best' indicates the number of instances for which an algorithm performs the best among the compared algorithms. The row 'p-value' indicates the outcomes of the non-parametric Friedman tests between a set of results of IMS and those of a reference algorithm. The best results among the compared algorithms are indicated in bold.

Table A.1
Comparison of the IMS algorithm with two state-of-the-art algorithms on the large DGS instances with $n = 2000$, $m = 50$

Instance				f_{best}			f_{avg}		
Graph	m	a_g	b_g	**ITS** [22]	**SGVNS** [6]	**IMS**	**ITS** [22]	**SGVNS** [6]	**IMS**
MDG-a_21	50	32	48	272874	273432	**274075**	272431.05	273124.65	**273762.25**
MDG-a_22	50	32	48	272699	273419	**274335**	272136.20	273107.15	**274045.85**
MDG-a_23	50	32	48	272689	273431	**274181**	272176.50	272987.45	**273948.35**
MDG-a_24	50	32	48	272757	273416	**274075**	272066.20	273069.55	**273762.55**
MDG-a_25	50	32	48	272579	273463	**274069**	272234.85	273155.00	**273740.40**
MDG-a_26	50	32	48	272769	273263	**274466**	272205.45	272991.45	**273929.85**
MDG-a_27	50	32	48	272205	273258	**274133**	272136.60	272949.10	**273835.35**
MDG-a_28	50	32	48	272472	273452	**274228**	272046.55	273076.60	**273859.65**
MDG-a_29	50	32	48	272811	273434	**274396**	272122.60	273148.55	**273999.85**
MDG-a_30	50	32	48	272730	273430	**274316**	272016.35	273107.30	**273897.95**
MDG-a_31	50	32	48	272900	273459	**274342**	272532.40	273137.00	**274117.25**
MDG-a_32	50	32	48	272794	273374	**274303**	272326.35	273080.45	**273946.25**
MDG-a_33	50	32	48	272692	273542	**274389**	272232.50	273083.35	**274055.00**
MDG-a_34	50	32	48	272517	273438	**274379**	272218.70	273177.00	**273990.30**
MDG-a_35	50	32	48	272486	273419	**274097**	272068.70	273028.50	**273794.45**
MDG-a_36	50	32	48	272758	273616	**274172**	272264.55	273068.90	**273889.90**
MDG-a_37	50	32	48	272280	273434	**274457**	271995.35	273163.80	**274039.20**
MDG-a_38	50	32	48	272520	273494	**274382**	272037.40	273228.30	**274033.40**
MDG-a_39	50	32	48	272815	273470	**274131**	272182.85	273204.25	**273883.20**
MDG-a_40	50	32	48	272773	273571	**274316**	272251.65	273225.00	**274050.65**
Average				272656.00	273440.75	274262.10	272184.14	273105.67	273929.08
#Best				0	0	20	0	0	20
p-value				7.74e-6	7.74e-6		7.74e-6	7.74e-6	

Table A.2
Comparison of the IMS algorithm with two state-of-the-art algorithms on the large DGS instances with n=2000 and $m = 10$.

Instance				f_{best}			f_{avg}		
Graph	m	a_g	b_g	**ITS** [22]	**SGVNS** [6]	**IMS**	**ITS** [22]	**SGVNS** [6]	**IMS**
MDG-a_21	10	173	227	1133850	1132437	**1135700**	1133239.35	1131666.00	**1135310.55**
MDG-a_22	10	173	227	1133890	1132473	**1135376**	1133254.25	1131503.85	**1134892.80**
MDG-a_23	10	173	227	1133188	1131686	**1135286**	1132744.00	1130977.10	**1134627.75**
MDG-a_24	10	173	227	1133867	1132812	**1135485**	1133123.65	1131430.65	**1134975.20**
MDG-a_25	10	173	227	1133850	1133050	**1135776**	1133239.35	1131881.80	**1135299.70**
MDG-a_26	10	173	227	1133733	1132438	**1135324**	1133191.00	1131474.25	**1135010.75**
MDG-a_27	10	173	227	1134059	1131955	**1135265**	1133104.75	1130955.80	**1134649.75**
MDG-a_28	10	173	227	1133960	1132356	**1135603**	1133145.65	1131478.80	**1134980.25**
MDG-a_29	10	173	227	1133549	1132086	**1135868**	1132988.25	1131568.75	**1135284.35**
MDG-a_30	10	173	227	1133825	1132401	**1135651**	1133127.05	1131391.45	**1135014.90**
MDG-a_31	10	173	227	1134218	1132994	**1136036**	1133398.80	1132106.35	**1135617.95**
MDG-a_32	10	173	227	1133740	1132278	**1135917**	1133256.50	1131462.05	**1135175.85**
MDG-a_33	10	173	227	1134011	1132321	**1135506**	1133326.30	1131428.45	**1134925.35**
MDG-a_34	10	173	227	1134178	1132913	**1135563**	1133426.75	1131840.65	**1135234.95**
MDG-a_35	10	173	227	1133862	1132279	**1135279**	1132919.85	1131109.90	**1134789.20**
MDG-a_36	10	173	227	1134185	1132075	**1135563**	1133431.80	1131271.55	**1135156.10**
MDG-a_37	10	173	227	1134087	1133144	**1136267**	1133442.25	1132128.65	**1135542.65**
MDG-a_38	10	173	227	1134872	1133027	**1135697**	1133763.00	1131655.55	**1135199.35**
MDG-a_39	10	173	227	1133856	1132120	**1135375**	1133350.10	1131170.55	**1134822.95**
MDG-a_40	10	173	227	1134441	1133342	**1136148**	1133518.15	1132221.20	**1135763.35**
Average				1133961.05	1132509.35	1135634.25	1133249.54	1131536.17	1135113.69
#Best				0	0	20	0	0	20
p-value				7.74e-6	7.74e-6		7.74e-6	7.74e-6	

Table A.3
Comparison of the IMS algorithm with two state-of-the-art algorithms on the large EGS instances with n=2000 and $m = 10$.

Instance				f_{best}			f_{avg}		
Graph	m	a_g	b_g	**ITS** [22]	**SGVNS** [6]	**IMS**	**ITS** [22]	**SGVNS** [6]	**IMS**
MDG-a_21	10	200	200	1115022	1114270	**1117343**	1114442.75	1113004.65	**1116684.40**
MDG-a_22	10	200	200	1114794	1113468	**1117097**	1114409.70	1112711.05	**1116442.00**
MDG-a_23	10	200	200	1114892	1112976	**1116331**	1113981.75	1112453.60	**1115880.90**
MDG-a_24	10	200	200	1115312	1113919	**1116863**	1114676.95	1112947.40	**1116410.05**
MDG-a_25	10	200	200	1115809	1114269	**1117108**	1114749.00	1113427.15	**1116743.85**
MDG-a_26	10	200	200	1115590	1113571	**1116942**	1114765.90	1112838.80	**1116368.10**
MDG-a_27	10	200	200	1115132	1113272	**1116468**	1114553.75	1112405.70	**1115947.35**
MDG-a_28	10	200	200	1115335	1113925	**1116740**	1114421.70	1112842.45	**1116460.90**
MDG-a_29	10	200	200	1116073	1113747	**1117064**	1114900.85	1112835.30	**1116585.35**
MDG-a_30	10	200	200	1114993	1113583	**1116748**	1114373.75	1112662.90	**1116396.95**
MDG-a_31	10	200	200	1116037	1114170	**1117636**	1115106.35	1113322.30	**1117179.15**
MDG-a_32	10	200	200	1115944	1113406	**1116908**	1114746.20	1112626.25	**1116441.30**
MDG-a_33	10	200	200	1114739	1113326	**1116815**	1114292.15	1112568.10	**1116290.25**
MDG-a_34	10	200	200	1115094	1113611	**1117125**	1114697.30	1113039.85	**1116608.10**
MDG-a_35	10	200	200	1114836	1114084	**1116561**	1114283.50	1112689.40	**1116093.60**
MDG-a_36	10	200	200	1114858	1113480	**1116789**	1114249.55	1112746.90	**1116379.10**
MDG-a_37	10	200	200	1115425	1113931	**1117783**	1114776.40	1113387.60	**1117045.70**
MDG-a_38	10	200	200	1115548	1113837	**1117110**	1114912.15	1113541.35	**1116615.75**
MDG-a_39	10	200	200	1114931	1113255	**1116610**	1113943.30	1112574.00	**1116142.15**
MDG-a_40	10	200	200	1116449	1114092	**1117477**	1115546.75	1113414.85	**1117069.00**
Average				1115340.65	1113709.60	1116975.90	1114591.49	1112901.98	1116489.20
#Best				0	0	20	0	0	20
p-value				7.74e-6	7.74e-6		7.74e-6	7.74e-6	

Table A.4
Comparison of the IMS algorithm with two state-of-the-art algorithms on the large DGS instances with n=2000 and $m = 25$.

Instance				f_{best}			f_{avg}		
Graph	m	a_g	b_g	**ITS** [22]	**SGVNS** [6]	**IMS**	**ITS** [22]	**SGVNS** [6]	**IMS**
MDG-a_21	25	51	109	539304	539616	**541130**	538762.40	539050.10	**540694.30**
MDG-a_22	25	51	109	539732	539554	**541013**	539199.75	538882.55	**540636.20**
MDG-a_23	25	51	109	539571	539030	**541352**	538867.15	538663.70	**540553.45**
MDG-a_24	25	51	109	539549	539238	**540794**	538784.55	538911.85	**540574.50**
MDG-a_25	25	51	109	539481	539872	**541283**	538876.90	539156.75	**540781.90**
MDG-a_26	25	51	109	539321	539696	**541099**	538978.25	538840.85	**540601.95**
MDG-a_27	25	51	109	539028	539365	**541084**	538475.60	538794.80	**540623.70**
MDG-a_28	25	51	109	539560	539458	**541179**	539090.75	538852.30	**540636.25**
MDG-a_29	25	51	109	539356	539458	**541492**	539043.55	539005.30	**540881.65**
MDG-a_30	25	51	109	539467	539482	**541224**	538600.30	538898.80	**540769.85**
MDG-a_31	25	51	109	539592	539808	**541215**	539220.20	539184.65	**540829.25**
MDG-a_32	25	51	109	539738	539484	**540939**	539135.95	538982.75	**540533.85**
MDG-a_33	25	51	109	539268	539719	**540958**	538685.75	539054.15	**540656.25**
MDG-a_34	25	51	109	539567	539726	**541136**	538968.95	538980.60	**540707.70**
MDG-a_35	25	51	109	539225	539294	**541054**	538632.90	538743.00	**540568.35**
MDG-a_36	25	51	109	539375	539416	**541469**	538797.05	538920.90	**540742.60**
MDG-a_37	25	51	109	539576	539492	**541183**	538805.95	539108.85	**540742.45**
MDG-a_38	25	51	109	539367	539363	**541150**	538962.70	539168.40	**540801.00**
MDG-a_39	25	51	109	539075	539423	**540920**	538523.20	538810.10	**540524.70**
MDG-a_40	25	51	109	539208	539902	**541273**	538869.90	539294.25	**540983.60**
Average				539418.00	539519.80	541147.35	538864.09	538965.23	540692.18
#Best				0	0	20	0	0	20
p-value				7.74e-6	7.74e-6		7.74e-6	7.74e-6	

Table A.5
Comparison of the IMS algorithm with two state-of-the-art algorithms on the large EGS instances with n=2000 and $m = 25$.

Instance				f_{best}			f_{avg}		
Graph	m	a_g	b_g	**ITS** [22]	**SGVNS** [6]	**IMS**	**ITS** [22]	**SGVNS** [6]	**IMS**
MDG-a_21	25	80	80	486046	485997	**487776**	485271.85	485574.30	**487360.10**
MDG-a_22	25	80	80	486509	486171	**487472**	485676.00	485437.50	**487041.00**
MDG-a_23	25	80	80	485690	485669	**487391**	485125.40	485310.00	**487078.55**
MDG-a_24	25	80	80	485848	485853	**487382**	485157.20	485532.90	**487086.95**
MDG-a_25	25	80	80	485819	486062	**487700**	485355.85	485639.90	**487270.35**
MDG-a_26	25	80	80	485712	486000	**487443**	485164.55	485368.25	**487191.90**
MDG-a_27	25	80	80	485532	485688	**487186**	484994.95	485327.50	**486913.65**
MDG-a_28	25	80	80	485922	485600	**487340**	485424.50	485336.40	**487079.05**
MDG-a_29	25	80	80	486329	485907	**487624**	485606.05	485550.20	**487346.40**
MDG-a_30	25	80	80	485702	486040	**487319**	484916.00	485510.60	**486990.20**
MDG-a_31	25	80	80	486403	486147	**487883**	485924.95	485516.45	**487432.55**
MDG-a_32	25	80	80	486174	485956	**487254**	485169.60	485478.00	**486995.40**
MDG-a_33	25	80	80	485676	486056	**487442**	485156.65	485369.50	**487050.05**
MDG-a_34	25	80	80	486631	486017	**487620**	485533.65	485594.65	**487131.35**
MDG-a_35	25	80	80	485440	485647	**487339**	485035.05	485394.20	**487024.85**
MDG-a_36	25	80	80	485524	485988	**487417**	485159.30	485475.20	**487090.90**
MDG-a_37	25	80	80	486137	486084	**487675**	485567.85	485676.05	**487307.95**
MDG-a_38	25	80	80	485976	485928	**487711**	485153.10	485225.10	**487341.50**
MDG-a_39	25	80	80	486005	485898	**487422**	485529.45	485325.75	**486955.75**
MDG-a_40	25	80	80	485852	486026	**487634**	485392.50	485673.65	**487341.45**
Average				485946.35	485936.70	487501.50	485315.72	485465.81	487151.50
#Best				0	0	20	0	0	20
p-value				7.74e-6	7.74e-6		7.74e-6	7.74e-6	

Table A.6
Comparison of the IMS algorithm with two state-of-the-art algorithms on the large DGS instances with $n = 2000$, $m = 50$, $a_g = 26$, and $b_g = 54$.

Instance				f_{best}			f_{avg}		
Graph	m	a_g	b_g	**ITS** [22]	**SGVNS** [6]	**IMS**	**ITS** [22]	**SGVNS** [6]	**IMS**
MDG-a_21	50	26	54	291149	291749	**292702**	290802.40	291384.75	**292343.30**
MDG-a_22	50	26	54	291098	291457	**292564**	290700.10	291219.20	**292104.55**
MDG-a_23	50	26	54	290579	291553	**292344**	290236.45	291259.25	**292023.75**
MDG-a_24	50	26	54	290934	291709	**292651**	290386.75	291378.35	**292271.05**
MDG-a_25	50	26	54	291081	291826	**292727**	290633.45	291337.60	**292415.10**
MDG-a_26	50	26	54	291348	291812	**292723**	290380.70	291344.05	**292289.20**
MDG-a_27	50	26	54	291009	291530	**292139**	290506.30	291211.70	**291842.00**
MDG-a_28	50	26	54	290997	291685	**292574**	290511.25	291320.40	**292159.20**
MDG-a_29	50	26	54	291008	291731	**292634**	290484.05	291384.20	**292220.40**
MDG-a_30	50	26	54	291069	291838	**292654**	290428.00	291415.45	**292283.10**
MDG-a_31	50	26	54	291321	292093	**292507**	291090.65	291607.55	**292273.25**
MDG-a_32	50	26	54	291478	291753	**292651**	290647.00	291344.35	**292303.15**
MDG-a_33	50	26	54	291093	291682	**292480**	290618.00	291273.10	**292194.45**
MDG-a_34	50	26	54	290986	291702	**292813**	290638.25	291283.35	**292401.75**
MDG-a_35	50	26	54	290840	291689	**292619**	290427.75	291301.15	**292252.05**
MDG-a_36	50	26	54	291476	291714	**292401**	290579.15	291351.00	**292250.85**
MDG-a_37	50	26	54	291160	291781	**292815**	290639.15	291413.00	**292425.85**
MDG-a_38	50	26	54	291271	291840	**292445**	290676.85	291380.40	**292148.60**
MDG-a_39	50	26	54	291199	291642	**292390**	290479.05	291247.05	**291960.90**
MDG-a_40	50	26	54	291235	291996	**292884**	290689.75	291494.40	**292475.75**
Average				291116.55	291739.10	292585.85	290577.75	291347.52	292231.91
#Best				0	0	20	0	0	20
p-value				7.74e-6	7.74e-6		7.74e-6	7.74e-6	

Table A.7
Comparison of the IMS algorithm with two state-of-the-art algorithms on the large EGS instances with n=2000 and $m = 50$.

Instance				f_{best}			f_{avg}		
Graph	m	a_g	b_g	**ITS** [22]	**SGVNS** [6]	**IMS**	**ITS** [22]	**SGVNS** [6]	**IMS**
MDG-a_21	50	40	40	263680	264582	**265422**	263184.70	264259.00	**265155.60**
MDG-a_22	50	40	40	263929	264461	**265250**	263296.00	264121.60	**264885.10**
MDG-a_23	50	40	40	263731	264489	**265146**	263221.95	264092.20	**264789.15**
MDG-a_24	50	40	40	263675	264567	**265455**	263126.30	264177.40	**265122.20**
MDG-a_25	50	40	40	263844	264571	**265363**	263152.10	264225.30	**265101.95**
MDG-a_26	50	40	40	263837	264639	**265341**	263340.45	264216.75	**265106.80**
MDG-a_27	50	40	40	263641	264387	**265383**	262982.40	264095.05	**265016.35**
MDG-a_28	50	40	40	263628	264419	**265406**	263210.05	264165.35	**264907.00**
MDG-a_29	50	40	40	264015	264763	**265375**	263270.45	264246.25	**264950.80**
MDG-a_30	50	40	40	263457	264536	**265564**	263067.10	264236.85	**264933.00**
MDG-a_31	50	40	40	264085	264713	**265452**	263577.85	264323.10	**265262.35**
MDG-a_32	50	40	40	263893	264570	**265235**	263488.65	264267.60	**264884.90**
MDG-a_33	50	40	40	263682	264569	**265285**	263253.50	264215.00	**264863.10**
MDG-a_34	50	40	40	263564	264622	**265372**	263158.95	264227.50	**265022.75**
MDG-a_35	50	40	40	263876	264572	**265304**	263456.40	264149.25	**265031.40**
MDG-a_36	50	40	40	263674	264575	**265385**	263147.05	264213.35	**265109.45**
MDG-a_37	50	40	40	263798	264585	**265378**	263196.25	264324.05	**265171.30**
MDG-a_38	50	40	40	263879	264514	**265100**	263337.00	264213.25	**264816.35**
MDG-a_39	50	40	40	263612	264561	**265133**	263347.35	264136.90	**264831.90**
MDG-a_40	50	40	40	263841	264603	**265502**	263507.70	264324.90	**265224.95**
Average				263767.03	264564.90	265342.55	263266.11	264211.53	265009.32
#Best				0	0	20	0	0	20
p-value				7.74e-6	7.74e-6		7.74e-6	7.74e-6	

Table A.8
Comparison of the IMS algorithm with two state-of-the-art algorithms on the large DGS instances with n=2000 and $m = 100$.

Instance				f_{best}			f_{avg}		
Graph	m	a_g	b_g	**ITS** [22]	**SGVNS** [6]	**IMS**	**ITS** [22]	**SGVNS** [6]	**IMS**
MDG-a_21	100	13	27	159084	159614	**160129**	158583.45	159143.20	**159840.15**
MDG-a_22	100	13	27	159062	159462	**159845**	158669.95	159200.50	**159630.75**
MDG-a_23	100	13	27	158786	159406	**159829**	158413.80	159200.10	**159558.05**
MDG-a_24	100	13	27	158850	159477	**159882**	158440.15	159212.50	**159671.90**
MDG-a_25	100	13	27	158750	159474	**159835**	158437.20	159197.60	**159606.05**
MDG-a_26	100	13	27	158994	159538	**160219**	158759.90	159258.60	**159795.30**
MDG-a_27	100	13	27	158825	159459	**160040**	158487.05	159169.60	**159759.30**
MDG-a_28	100	13	27	159165	159442	**159789**	158548.45	159141.65	**159448.45**
MDG-a_29	100	13	27	159165	159720	**159790**	158570.55	159222.65	**159466.30**
MDG-a_30	100	13	27	158787	159446	**159739**	158435.35	159222.60	**159468.30**
MDG-a_31	100	13	27	158881	**159837**	159810	158514.30	159368.00	**159516.70**
MDG-a_32	100	13	27	159017	159525	**160096**	158499.25	159257.10	**159815.00**
MDG-a_33	100	13	27	158840	159503	**159619**	158470.70	159236.40	**159426.20**
MDG-a_34	100	13	27	158993	**159680**	159608	158558.45	159195.85	**159405.05**
MDG-a_35	100	13	27	159101	159623	**159875**	158625.55	159200.50	**159625.15**
MDG-a_36	100	13	27	159063	159445	**159869**	158705.10	159211.50	**159627.75**
MDG-a_37	100	13	27	158899	159564	**160054**	158554.15	159249.80	**159790.70**
MDG-a_38	100	13	27	158748	159456	**160130**	158357.90	159251.15	**159828.85**
MDG-a_39	100	13	27	159343	159610	**159999**	158802.65	159228.55	**159777.65**
MDG-a_40	100	13	27	159065	159600	**159959**	158618.50	159310.90	**159706.25**
Average				158970.90	159544.05	159905.80	158552.62	159223.94	159638.19
#Best				0	2	18	0	0	20
p-value				7.74e-6	3.47e-6		7.74e-6	7.74e-6	

Table A.9
Comparison of the IMS algorithm with two state-of-the-art algorithms on the large EGS instances with n=2000 and $m = 100$.

Instance				f_{best}			f_{avg}		
Graph	m	a_g	b_g	**ITS** [22]	**SGVNS** [6]	**IMS**	**ITS** [22]	**SGVNS** [6]	**IMS**
MDG-a_21	100	20	20	144020	144879	**144918**	143619.25	144426.40	**144592.80**
MDG-a_22	100	20	20	143938	144536	**144756**	143554.95	144288.90	**144549.90**
MDG-a_23	100	20	20	143915	144577	**144927**	143505.35	144342.90	**144696.95**
MDG-a_24	100	20	20	144071	144794	**144962**	143690.95	144304.35	**144707.35**
MDG-a_25	100	20	20	144341	144648	**145043**	143635.05	144299.65	**144766.50**
MDG-a_26	100	20	20	144148	144601	**145032**	143691.65	144331.80	**144626.60**
MDG-a_27	100	20	20	143928	144458	**144935**	143617.35	144231.30	**144747.70**
MDG-a_28	100	20	20	144140	144557	**144848**	143765.70	144265.15	**144644.90**
MDG-a_29	100	20	20	144099	144567	**144937**	143831.25	144255.30	**144636.80**
MDG-a_30	100	20	20	144263	144628	**144784**	143744.45	144354.60	**144500.70**
MDG-a_31	100	20	20	144184	144655	**144881**	143852.55	144385.10	**144666.35**
MDG-a_32	100	20	20	143846	144693	**144934**	143641.50	144315.35	**144701.00**
MDG-a_33	100	20	20	144071	144532	**144611**	143667.75	144321.20	**144447.00**
MDG-a_34	100	20	20	144511	144573	**144803**	143799.00	144383.80	**144576.65**
MDG-a_35	100	20	20	144022	144627	**144999**	143682.45	144357.50	**144763.65**
MDG-a_36	100	20	20	143829	144543	**144998**	143369.80	144311.30	**144788.15**
MDG-a_37	100	20	20	144332	144632	**144803**	143779.55	144371.25	**144645.50**
MDG-a_38	100	20	20	144051	144585	**144958**	143701.05	144314.35	**144656.15**
MDG-a_39	100	20	20	144344	144579	**145009**	143794.55	144253.20	**144797.55**
MDG-a_40	100	20	20	144323	144623	**145229**	143743.75	144395.45	**144843.90**
Average				144118.80	144614.35	144918.35	143684.40	144325.44	144667.81
#Best				0	0	20	0	0	20
p-value				7.74e-6	7.74e-6		7.74e-6	7.74e-6	

Table A.10
Comparison of the IMS algorithm with two state-of-the-art algorithms on the large DGS instances with n=2000 and $m = 200$.

Instance				f_{best}			f_{avg}		
Graph	m	a_g	b_g	**ITS** [22]	**SGVNS** [6]	**IMS**	**ITS** [22]	**SGVNS** [6]	**IMS**
MDG-a_21	200	6	14	88464	88548	**88949**	88139.95	88276.40	**88718.70**
MDG-a_22	200	6	14	88243	**88536**	88459	87968.15	88223.90	**88318.15**
MDG-a_23	200	6	14	88131	88483	**88973**	87800.10	88200.25	**88796.70**
MDG-a_24	200	6	14	88343	88464	**88760**	87915.65	88202.35	**88597.55**
MDG-a_25	200	6	14	88435	88571	**89031**	88030.90	88276.05	**88814.00**
MDG-a_26	200	6	14	88538	**88642**	88476	**88324.75**	88312.55	88322.00
MDG-a_27	200	6	14	**88644**	88469	88522	88126.45	88221.35	**88272.05**
MDG-a_28	200	6	14	88565	88454	**88794**	88060.65	88296.80	**88656.85**
MDG-a_29	200	6	14	88361	88532	**89005**	88001.85	88259.90	**88839.35**
MDG-a_30	200	6	14	88543	88560	**89017**	88270.80	88287.55	**88811.00**
MDG-a_31	200	6	14	**88623**	88439	88601	88363.20	88260.65	**88459.75**
MDG-a_32	200	6	14	88445	88502	**88639**	88269.85	88262.40	**88456.90**
MDG-a_33	200	6	14	88424	**88775**	88486	88147.25	88306.55	**88341.10**
MDG-a_34	200	6	14	88368	88490	**88961**	88063.85	88264.40	**88790.35**
MDG-a_35	200	6	14	88426	**88572**	88437	88070.35	88228.15	**88273.80**
MDG-a_36	200	6	14	88542	**88590**	88492	88129.15	88296.75	**88340.65**
MDG-a_37	200	6	14	88457	88439	**88558**	87943.10	88191.85	**88395.35**
MDG-a_38	200	6	14	**88544**	88442	88520	88304.75	88266.35	**88325.40**
MDG-a_39	200	6	14	88038	88643	**88947**	87750.55	88326.45	**88831.10**
MDG-a_40	200	6	14	88580	88565	**88803**	88306.05	88298.30	**88625.65**
Average				88435.70	88535.80	88721.50	88099.37	88262.95	88549.32
#Best				3	5	12	1	0	19
p-value				2.54e-2	2.54e-2		5.70e-5	7.74e-6	

Table A.11
Comparison of the IMS algorithm with two state-of-the-art algorithms on the large EGS instances with n=2000 and $m = 200$.

Instance				f_{best}			f_{avg}		
Graph	m	a_g	b_g	**ITS** [22]	**SGVNS** [6]	**IMS**	**ITS** [22]	**SGVNS** [6]	**IMS**
MDG-a_21	200	10	10	**77249**	76543	77175	76842.90	76438.70	**76994.40**
MDG-a_22	200	10	10	**77214**	76546	**77214**	76895.75	76449.15	**77107.05**
MDG-a_23	200	10	10	77033	76564	**77234**	76824.15	76464.70	**77022.85**
MDG-a_24	200	10	10	77092	76535	**77197**	76873.65	76463.75	**77058.50**
MDG-a_25	200	10	10	77198	76578	**77229**	76962.55	76461.45	**77121.40**
MDG-a_26	200	10	10	**77234**	76567	77121	76822.05	76467.90	**77008.70**
MDG-a_27	200	10	10	76875	76521	**77131**	76608.15	76441.75	**77001.80**
MDG-a_28	200	10	10	77216	76633	**77249**	76940.85	76465.05	**77122.85**
MDG-a_29	200	10	10	77171	76627	**77339**	76870.25	76463.50	**77137.85**
MDG-a_30	200	10	10	**77470**	76592	77214	77011.35	76490.35	**77121.60**
MDG-a_31	200	10	10	77245	76600	**77302**	76924.15	76452.55	**77146.90**
MDG-a_32	200	10	10	77168	76638	**77198**	76935.05	76464.00	**77123.30**
MDG-a_33	200	10	10	77352	76621	**77396**	76975.65	76478.70	**77148.10**
MDG-a_34	200	10	10	77316	76615	**77366**	76960.65	76456.60	**77154.80**
MDG-a_35	200	10	10	77233	76685	**77236**	76908.75	76470.25	**77146.05**
MDG-a_36	200	10	10	77150	76553	**77270**	76918.05	76458.15	**77148.45**
MDG-a_37	200	10	10	77196	76586	**77302**	76980.40	76452.00	**77154.85**
MDG-a_38	200	10	10	**77226**	76523	77145	76914.30	76431.70	**77010.85**
MDG-a_39	200	10	10	**77311**	76618	77261	77000.70	76441.60	**77142.25**
MDG-a_40	200	10	10	77221	76601	**77277**	76980.45	76480.55	**77095.15**
Average				77208.50	76587.30	77242.80	76907.49	76459.62	77098.39
#Best				6	0	15	0	0	20
p-value				3.90e-2	7.74e-6		7.74e-6	7.74e-6	

Table A.12
Comparison of the IMS algorithm with two state-of-the-art algorithms on the large DGS instances with n=3000 and $m = 50$.

Instance				f_{best}			f_{avg}		
Graph	m	a_g	b_g	**ITS** [22]	**SGVNS** [6]	**IMS**	**ITS** [22]	**SGVNS** [6]	**IMS**
MDG-c_1	50	48	72	57945495	58002162	**58256069**	57866205.40	57969360.45	**58206836.25**
MDG-c_2	50	48	72	57941604	58058009	**58222189**	57847954.50	57978365.60	**58176353.85**
MDG-c_3	50	48	72	57934848	58042369	**58183020**	57853662.10	57981548.00	**58139867.15**
MDG-c_4	50	48	72	57927582	58040996	**58203033**	57856933.15	57986655.35	**58160671.30**
MDG-c_5	50	48	72	57923652	58026970	**58195892**	57837851.55	57973993.50	**58138551.15**
MDG-c_6	50	48	72	57912572	58051277	**58170930**	57835981.70	57995787.85	**58115476.40**
MDG-c_7	50	48	72	57920362	58006501	**58184196**	57832628.50	57965788.60	**58145525.85**
MDG-c_8	50	48	72	57917641	58018105	**58175358**	57828244.60	57958586.70	**58137072.20**
MDG-c_9	50	48	72	57852060	58053174	**58148355**	57804909.10	57996436.85	**58108503.85**
MDG-c_10	50	48	72	57855618	58032526	**58177707**	57807867.05	57988658.75	**58126674.90**
MDG-c_11	50	48	72	57927899	58057511	**58202140**	57823896.80	57996449.35	**58166369.45**
MDG-c_12	50	48	72	57879753	58043186	**58211896**	57810309.90	58009313.20	**58171859.25**
MDG-c_13	50	48	72	57899195	58033030	**58231541**	57820288.55	57981936.95	**58173176.70**
MDG-c_14	50	48	72	57893447	58050347	**58170717**	57819670.60	58000079.35	**58128947.60**
MDG-c_15	50	48	72	57936099	58006905	**58219534**	57842744.00	57971202.50	**58181767.65**
MDG-c_16	50	48	72	57899782	58017384	**58213319**	57841066.40	57977613.10	**58153544.70**
MDG-c_17	50	48	72	57912576	58080720	**58192484**	57822019.25	57983021.95	**58146759.40**
MDG-c_18	50	48	72	57884836	58033798	**58152922**	57824357.45	57968806.20	**58112703.35**
MDG-c_19	50	48	72	57918318	58018351	**58212035**	57831674.95	57956907.60	**58163660.60**
MDG-c_20	50	48	72	57877843	58022625	**58191329**	57793757.60	57950125.50	**58140830.60**
Average				57908059.10	58034797.30	58195733.30	57830101.16	57979531.87	58149757.61
#Best				0	0	20	0	0	20
p-value				7.74e-6	7.74e-6		7.74e-6	7.74e-6	

Table A.13
Comparison of the IMS algorithm with two state-of-the-art algorithms on the large EGS instances with n=3000 and $m = 50$.

Instance				f_{best}			f_{avg}		
Graph	m	a_g	b_g	**ITS** [22]	**SGVNS** [6]	**IMS**	**ITS** [22]	**SGVNS** [6]	**IMS**
MDG-c_1	50	60	60	55977693	56095305	**56253625**	55910079.05	56056762.55	**56218194.70**
MDG-c_2	50	60	60	55969792	56090550	**56272017**	55890036.80	56045634.60	**56226594.65**
MDG-c_3	50	60	60	55997944	56130490	**56262455**	55867986.95	56042121.65	**56208888.85**
MDG-c_4	50	60	60	55959548	56076697	**56257907**	55901255.80	56045542.35	**56219344.25**
MDG-c_5	50	60	60	55966544	56082421	**56234604**	55866485.35	56028107.75	**56200190.95**
MDG-c_6	50	60	60	55942210	56117081	**56243850**	55881502.00	56028119.45	**56202899.40**
MDG-c_7	50	60	60	55944042	56086355	**56236720**	55872630.10	56026724.60	**56183765.50**
MDG-c_8	50	60	60	55935072	56055008	**56249049**	55884553.95	56024573.75	**56218073.10**
MDG-c_9	50	60	60	55906493	56057439	**56235250**	55851227.75	55997916.45	**56188505.85**
MDG-c_10	50	60	60	55920804	56107377	**56179602**	55812256.00	56025439.20	**56157919.35**
MDG-c_11	50	60	60	56010877	56095642	**56238432**	55875876.35	56030011.90	**56197694.85**
MDG-c_12	50	60	60	55952824	56067675	**56267216**	55872773.75	56021610.00	**56196066.80**
MDG-c_13	50	60	60	55954309	56068977	**56273597**	55868322.80	56030276.70	**56201739.90**
MDG-c_14	50	60	60	55990777	56097074	**56270275**	55892913.50	56038050.25	**56211709.95**
MDG-c_15	50	60	60	55945742	56076707	**56252183**	55891242.55	56032771.00	**56215053.55**
MDG-c_16	50	60	60	55960705	56140977	**56249512**	55877242.10	56060959.00	**56215024.25**
MDG-c_17	50	60	60	55958700	56079252	**56226673**	55891889.30	56021554.10	**56181329.00**
MDG-c_18	50	60	60	55947581	56061668	**56240125**	55879358.00	56004267.15	**56175311.65**
MDG-c_19	50	60	60	55949205	56063226	**56225414**	55875167.65	56037554.50	**56204567.35**
MDG-c_20	50	60	60	55979807	56103438	**56239512**	55897182.15	56034844.75	**56206989.95**
Average				55958533.45	56087667.95	56245400.90	55877999.10	56031642.09	56201493.19
#Best				0	0	20	0	0	20
p-value				7.74e-6	7.74e-6		7.74e-6	7.74e-6	