Hindawi
Mathematical Problems in Engineering
Volume 2019, Article ID 7104702, 16 pages
https://doi.org/10.1155/2019/7104702

Research Article

A Hybrid Metaheuristic of Integrating Estimation of Distribution Algorithm with Tabu Search for the Max-Mean Dispersion Problem

Dieudonné Nijimbere,[1] Songzheng Zhao,[1] Haichao Liu,[1] Bo Peng,[2] and Aijun Zhang[3]

[1]*School of Management, Northwestern Polytechnical University, 127 Youyi West Road, 710072 Xi'an, China*
[2]*School of Business Administration, Southwestern University of Finance and Economics, 610074 Chengdu, China*
[3]*School of Sciences, Xi'an University of Technology, Xi'an 710054, China*

Correspondence should be addressed to Aijun Zhang; zajmgh@xaut.edu.cn

Received 21 January 2019; Accepted 12 June 2019; Published 1 July 2019

Academic Editor: Piotr Jędrzejowicz

This paper presents a hybrid metaheuristic that combines estimation of distribution algorithm with tabu search (EDA-TS) for solving the max-mean dispersion problem. The proposed EDA-TS algorithm essentially alternates between an EDA procedure for search diversification and a tabu search procedure for search intensification. The designed EDA procedure maintains an elite set of high quality solutions, based on which a conditional preference probability model is built for generating new diversified solutions. The tabu search procedure uses a fast 1-flip move operator for solution improvement. Experimental results on benchmark instances with variables ranging from 500 to 5000 disclose that our EDA-TS algorithm competes favorably with state-of-the-art algorithms in the literature. Additional analysis on the parameter sensitivity and the merit of the EDA procedure as well as the search balance between intensification and diversification sheds light on the effectiveness of the algorithm.

1. Introduction

Dispersion problems are a typical class of combinatorial optimization problems, which can be categorized as efficiency-based objectives and equity-based objectives [1]. Equity-based dispersion problems studied in the literature include the max-mean dispersion problem that maximizes the mean of the aggregate dispersion, the max-min diversity problem that maximizes the minimum dispersion [2, 3], the minimum differential dispersion problem that minimizes the difference between the maximum aggregate dispersion and minimum aggregate dispersion [4, 5], the balanced quadratic optimization problem that minimizes the difference between the maximum dispersion and the minimum dispersion [6], etc. One of the well-known efficiency-based dispersion problems is max-sum dispersion, which requires maximizing the aggregate dispersion [7, 8].

The max-mean dispersion problem (Max-Mean DP) has received extensive attention in recent years [1, 9]. It is proved to be strong NP-hard and has many important applications including architectural space planning, sentiment analysis, social networks, pollution control, and web pages ranks. Let N be a set of n elements and $D = (d_{ij})$ be a $n \times n$ distance matrix. The max-mean dispersion problem (Max-Mean DP) is to find a subset $M \in N$ to maximize the distance function $f(M) = \sum_{i,j\in M, i\neq j} d_{ij}/|M|$, where $|M|$ denotes the size of the set M. It is noteworthy that a solution $M \in N$ is a subset of any cardinality. Let $x_i = 1$ if the element i is in M and $x_i = 0$ otherwise. The max-mean dispersion problem can be formulated as follows:

$$\text{Maximize} \quad \frac{\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} d_{ij}x_i x_j}{\sum_{i=1}^{n} x_i} \tag{1}$$

$$\text{Subject to} \quad \sum_{i=1}^{n} x_i \geq 2 \tag{2}$$

$$x_i \in \{0, 1\}, \quad \text{for } i = 1, 2, \ldots, n. \tag{3}$$

For solving the Max-Mean DP, various approaches have been proposed in the literature. Marti and Sandoya (2013) [10] proposed a GRASP with path relinking algorithm, where GRASP uses a randomized greedy strategy for initial solution construction and a variable neighborhood descent method for solution improvement. Moreover, the path relinking is a postprocessing procedure to control the search diversification. Della Croce et al. (2014) [11] developed a two-phase hybrid heuristic, where the first phase solves a mixed integer programming formulation to obtain an initial set of solutions and the second phase employs a local branch strategy to refine the quality of solutions. An extended version of this algorithm is presented in Della Croce et al. (2016) [12], where a path relinking phase is added to further enhance the quality of solutions. Carrasco et al. (2015) [13] proposed a dynamic tabu search algorithm, where various short-term and long-term memory structures are integrated to enhance the performance of tabu search. Lai and Hao (2016) [9] proposed a tabu search-based memetic algorithm that integrates a tabu search method for discovering good local optima and a random crossover operator to generate well-diversified offspring solutions. Brimberg et al. (2017) [14] developed a variable neighborhood search algorithm that examines multiple neighborhood structures based on add, drop, and swap moves and picks one of them in a probability way to perform a shaking procedure. Garraffa et al. (2017) [15] embed SDP relaxation and cutting planes into a branch and bound framework to reach optimal solutions.

Estimation of distribution algorithms (EDA) are a class of population-based algorithms [16–20], which have been demonstrated to be effective for solving many optimization problems. One of the most distinct features of EDA lies in the use of global statistical information extracted from a set of high-fitness solutions to build a probability model. During the population evolution, EDA continually updates the probability model for producing new offspring solutions. Aickelin et al. [21] proposed an approach based on EDA with ant-miner for solving the nurse rostering problem, where a Bayesian network is constructed for implementing EDA. Peña et al. [22] proposed a hybrid evolutionary algorithm that combines genetic algorithm with EDA for solving multiple optimization problems, where EDA employs a probabilistic graph model to express the dependencies among different variables. Zhang et al. [23] proposed a guided local search with EDA to solve the quadratic assignment problem, in which EDA is used for the mutation operator to generate offspring solutions.

In this paper, we propose the first hybrid metaheuristic that combines EDA with tabu search (EDA-TS) for tackling the Max-Mean DP. Basically, our proposed EDA-TS algorithm follows a multistart search framework, where each start includes an initial solution construction phase and a local search phase. The initial solution construction phase borrows the EDA idea of using statistical information extracted from a set of solutions to construct high-quality and well-diversified solutions. The local search phase uses a simple but effective tabu search method to improve the initial solutions. Experimental results on a total of 140 instances with variables ranging from 500 to 5000 indicate that our proposed algorithm performs significantly better than most reference algorithms in the literature. Furthermore, additional evidence is provided to shed light on the effectiveness of the algorithm.

The rest of the paper is organized as follows. Section 2 describes the proposed EDA-TS algorithm. Section 3 presents experimental comparisons with state-of-the-art algorithms in the literature and additional analysis on the effectiveness of the algorithm. Concluding remarks are given in Section 4.

2. Algorithm

This section presents the proposed EDA-TS algorithm, including the general scheme, the elite set initialiation, the EDA guided solution construction, and the tabu search solution improvement, as well as the elite set updating and rebuilding.

2.1. Main Scheme. The general scheme of our proposed EDA-TS algorithm is shown in Algorithm 1. At first, we create an elite set *ES* of high-quality and well-diversifed solutions and record the best solution x^* in *ES*. Then, we build a conditional preference probability matrix *CPPM* where each entry CPP_{ij} denotes the conditional preference probability of assigning a variable x_j with value 1 under the condition that the variable x_i is assigned with value 1. By referring to the information recorded in *CPPM*, we construct a new solution to launch a new round of search exploitation. For this purpose, we employ a tabu search solution improvement method to refine the quality of the starting solution. If this refined solution satisifies the *ES* updating criterion, an elite set updating method is triggered. Otherwise, we detect if a *ES* rebuilding method is necessary and execute it if it is. The abovementioned procedures are repeated until the elapsed time surpasses the given time limit.

2.2. Elite Set Initialization. Algorithm 2 gives the outline of the elite set initialization procedure. The elite set *ES* consists of a set of high-quality and well-diversified solutions. To obtain each solution in *ES*, we first generate a random initial solution and then submit it to a tabu search procedure for solution improvement. The random initial solution is obtained by assigning each variable with the equal probability of 0.5 to receive value 1 or value 0. The tabu search procedure employs a fast but effective 1-flip move operator to refine the solution within a short computional time (see Section 2.4). If a newly obtained solution after optimization is the same as a solution in *ES*, it is forbidden to join in *ES*. In this way, we guarantee that all the solutions in *ES* are representative.

2.3. EDA Guided Solution Construction

2.3.1. The CPP Matrix Calculation

Definition 1. Let $SubES_i \in ES$ denote the subset where each solution in $SubES_i$ receives value 1 for the variable x_i, $|SubES_i|$

(1) **Input:** a distance matrix $D = (d_{ij})_{n\times n}$
(2) **Output:** The best solution x^*
(3) $ES = \{x^1, x^2, \cdots, x^{|ES|}\} \longleftarrow$ EliteSetInitialization() (see Section 2.2)
(4) Record the best solution x^* in ES
(5) $CPPM \longleftarrow$ CPPMatrixCalculation(ES) (see Section 2.3.1)
(6) **while** the elapsed time does not surpass the given time limit **do**
(7) $x^0 \longleftarrow$ EDAGuidedSolutionConstruction(ES, $CPPM$) (see Section 2.3.2)
(8) $x^c \longleftarrow$ TabuSearch(x^0) (see Section 2.4)
(9) **if** $f(x^c) > f(x^*)$ **then**
(10) $x^* = x^c$
(11) **end if**
(12) Find the solution x^w getting the worst objective value in ES
(13) $(ES, update\_succ) \longleftarrow$ EliteSetUpdating(x^c, x^w) (see Section 2.5)
(14) **if** $update\_succ = TRUE$ **then**
(15) $update\_fail = 0$
(16) **else**
(17) $update\_fail = update\_fail + 1$
(18) **end if**
(19) **if** $update\_fail \geqslant \eta$ **then**
(20) $ES \longleftarrow$ EliteSetRebuilding(x^*) (see Section 2.5)
(21) **end if**
(22) **if** ES undergoes updates for $|ES|$ times **then**
(23) $CPPM \longleftarrow$ CPPMatrixCalculation(ES) (see Section 2.3.1)
(24) **end if**
(25) **end while**

Algorithm 1: Main scheme of the EDA-TS algorithm.

(1) Input: a distance matrix $D = (d_{ij})_{n\times n}$
(2) Output: an initial elite set ES
(3) $cur\_size = 0$, $ES = \emptyset$
(4) **repeat**
(5) $x \longleftarrow$ RandInitialSol()
(6) $x^c \longleftarrow$ TabuSearch(x)
(7) **if** x^c is not a duplicate of any solutions in ES **then**
(8) $ES = ES \cup \{x^c\}$
(9) $cur\_size = cur\_size + 1$
(10) **end if**
(11) **until** $cur\_size \geqslant |ES|$

Algorithm 2: Outline of the elite set initialization.

denote the size of $SubES_i$, $f(x)$ denote the objective function value of the solution x, and f_{min} denote the minimum objective function value among all the solutions in ES. Then, *the objective preference* op_i of the variable x_i is defined as

$$op_i = \frac{\sum_{x\in SubES_i} f(x) - f_{min}}{|SubES_i|}. \tag{4}$$

Definition 2. Given the sets ES and $SubES_i$, *the selection probability* p_i of the variable x_i is defined as

$$p_i = \frac{|SubES_i|}{|ES|}. \tag{5}$$

Definition 3. Given the selection probability p_i and the objective preference op_i, the *preference probability* q_i of selecting the variable x_i is defined as

$$q_i = p_i \cdot op_i. \tag{6}$$

Definition 4. Let co_{ij} denote the number of solutions in ES where both variables x_i and x_j receive value 1. Then, *the interaction selection probability* cop_{ij} associated with the variables x_i and x_j is defined as

$$cop_{ij} = \frac{co_{ij}}{|ES|}. \tag{7}$$

Definition 5. Given the selection probability p_i and the preference probability q_j as well as the interaction selection probability cop_{ij}, *the conditional preference probability* $q_{j|i}$ of the variable x_j upon having selected the variable x_i is defined as

$$q_{j|i} = q_j \cdot \frac{cop_{ij}}{p_i}. \tag{8}$$

The procedure to calculate the *CPP* matrix is shown in Algorithm 3.

In order to illustrate the CPP matrix construction, Table 1 gives a small instance $D = (d_{ij})_{5\times5}$ to show how to calculate the information stored in the memory structures.

2.3.2. EDA Guided Solution Construction Phase. Algorithm 4 shows the outline of the EDA guided solution construction

(1) Input: an elite set *ES* and the objective function value of each solution $f(x)$ in *ES*
(2) Output: a conditional preference probability matrix *CPPM*
(3) **for** $i = 1$ to n **do**
(4) Calculate the objective preference op_i of the variable x_i according to Eq. (4)
(5) Calculate the selection probability p_i of the variable x_i according to Eq. (5)
(6) Calculate the preference probability q_i of each variable x_i according to Eq. (6)
(7) **end for**
(8) **for** $i = 1$ to n **do**
(9) **for** $j = 1$ to n **do**
(10) Calculate the conditional preference probability $q_{j|i}$ according to Eq. (7) and (8)
(11) Set $CPPM_{ij} = q_{j|i}$
(12) **end for**
(13) **end for**

Algorithm 3: Outline of the *CPP* matrix calculation.

(1) Input: $CPPM_{ij}, i, j = 1, 2, \ldots, n$ and $q_i, i = 1, 2, \ldots, n$
(2) Output: a generated solution
(3) $iter = 0$, $f_{iter} = 0$
(4) **repeat**
(5) $iter = iter + 1$
(6) **if** $iter = 1$ **then**
(7) Construct a preference candidate set $Cnd = \{x_j \mid q_j \geq q_{min} + \alpha \cdot (q_{max} - q_{min})\}$
(8) Randomly choose a variable from *Cnd*
(9) **else**
(10) Build a set of variables getting the maximum conditional preference probability $MaxSet = \{x_j \mid q_{j|i} = q_{j_{max}|i}\}$
(11) Build another set *CndSet* of variables, where each variable x_j satisfies $q_{j_{max}|i} > q_{j|i} > q_{j_{min}|i} + \beta \cdot (q_{j_{max}|i} - q_{j_{min}|i})$
(12) **if** $rand(0, 1) < \gamma$ **then**
(13) Randomly choose a variable from *MaxSet*
(14) **else**
(15) Randomly choose a variable from *CndSet*
(16) **end if**
(17) **end if**
(18) Assign the chosen variable to be value 1 to enlarge the current solution
(19) Calculate the objective value f_{iter} obtained in this iteration
(20) **until** $iter > depth_c$ or $f_{iter} < f_{iter-1}$

Algorithm 4: Outline of the EDA guided solution construction phase.

phase. When creating a new starting solution, the first added variable is determined according to the preference probability information. Specifically, we first construct a preference candidate set *Cnd* in which the preference probability of each variable surpasses the threshold value $q_{min} + \alpha \cdot (q_{max} - q_{min})$, where q_{min} and q_{max} denote the minimum and maximum preference probability among all the variables, respectively. Then, we select a variable from *Cnd* randomly and assign it with value 1. Each time a variable is added in the solution, it is removed from *Cnd* and results in the size of *Cnd* decreased by 1. If *Cnd* is empty, we select variables from its complementary set.

Once a variable x_i is added in the solution, the selection of its next variable x_j for enlarging the solution relies on the conditional preference probability. The conditional preference probability $q_{j|i}$ of selecting a variable x_j upon having selected the variable x_i is obtained from the *CPP* matrix, which is calculated according to Algorithm 3. For each constructive step, we build a set *MaxSet* that includes all variables receiving the maximum conditional preference probability $q_{j_{max}|i}$ and a set *CndSet* that is composed of all the variables satisfying the condition $q_{j_{max}|i} > q_{j|i} > q_{j_{min}|i} + \beta \cdot (q_{j_{max}|i} - q_{j_{min}|i})$. Then, we choose a variable from *MaxSet* with the probability of γ and choose a variable from *CndSet* with the probability of $1 - \gamma$. Finally, the chosen variable is assigned with value 1 to enlarge the current solution. The abovementioned constructive procedure repeats at least $depth_c$ times and terminates when the objective value of the resulting solution begins to deteriorate.

It is noteworthy that we do not calculate a new *CPP* matrix each time the elite set *ES* is updated but wait until the *ES* updating reaches a specified number of times setting to be

Table 1: The CPP matrix construction procedure for a small instance $D = (d_{ij})_{5\times5}$.

(a) Step 0: Given the following problem instance D = $(d_{ij})_{5\times5}$

d_{ij}	x_1	x_2	x_3	x_4	x_5
x_1	0	-8.17	4.48	-8.73	-7.53
x_2	-8.17	0	7.18	-6.12	3.74
x_3	4.48	7.18	0	-1.75	-2.81
x_4	-8.73	-6.12	-1.75	0	1.8
x_5	-7.53	3.74	-2.81	1.8	0

(b) Step l: Produce three high quality solutions to form the elite set

ES	x_1	x_2	x_3	x_4	x_5	f(x)
x^1	0	1	1	0	0	3.59
x^2	1	0	1	0	0	2.24
x^3	0	1	1	0	1	2.70
$\lvert SubES_i\rvert$	1	2	3	0	1	

(c) Step 2: According to Eq. (4), Eq. (5), Eq. (6), calculate the objective preference op_i, the selection probability p_i and the preference probability q_i

	x_1	x_2	x_3	x_4	x_5
op_i	0	0.91	0.60	0	0.46
p_i	0.33	0.67	1	0	0.33
q_i	0	0.61	0.60	0	0.15

(d) Step 3: According to Eq. (7), Eq. (8), calculate the interaction selection probability cop_{ij} and the conditional preference probability $q_{j|i}$ which composes the CPP matrix

cop_{ij}					
			i		
j	x_1	x_2	x_3	x_4	x_5
x_1		0	0.33	0	0
x_2	0		0.67	0	0.33
x_3	0.33	0.67		0	0.33
x_4	0	0	0		0
x_5	0	0.33	0.33	0	
$q_{j\|i}$					
			i		
j	x_1	x_2	x_3	x_4	x_5
x_1		0	0	0	0
x_2	0		0.41	0	0.61
x_3	0.60	0.60		0	0.60
x_4	0	0	0		0
x_5	0	0.07	0.05	0	

the size of *ES* because of the following reasons. First, it is time consuming to calculate a *CPP* matrix and performing fewer calculation will save computational efforts. Second, the *CPP* matrix calculation based on the old *ES* is demonstrated to be effective to provide statistical information for promising solutions construction according to preliminary experiments.

2.4. Tabu Search. Tabu search is an advanced local search-based metaheuristic, which has been demonstrated to be highly effective for solving the Max-Mean DP [9]. Since the EDA phase needs high-quality solutions to maintain the elite set, we use a tabu search phase to improve solutions to local optimum. The general components of tabu search include move operators, an evaluation function, and a solution selection strategy, as well as tabu and aspiration rules.

We use a popular 1-flip move operator in solving many binary quadratic programming problems, which consists in changing the value of a variable to its complement when transforming the current solution into its neighbor solutions in the search space. The evaluation function is the same as the objective function defined in (1) to measure the solution quality. For each iteration, we use the best improvement strategy that picks the one among all the neighborhood solutions

with the best evaluation function value to perform the move. Given that each iteration has a total of n neighborhood solutions (where n denotes the number of the variables) and the identification of the best solution is time consuming if we use (1) to calculate the objective value, hence, we adopt a quick evaluation technique to reduce the computational efforts.

Specifically, we use an additional vector to record the potential p_i of each variable x_i, defined as

$$p_i = \sum_{i=1, i \neq j}^{n} d_{ij} x_j. \tag{9}$$

Then, the move gain of flipping the variable x_i can be represented as

$$\Delta_i = \begin{cases} \dfrac{mf(x) + p_i}{m+1} - f(x), & x_i = 0 \\ \dfrac{mf(x) - p_i}{m-1} - f(x), & x_i = 1, \end{cases} \tag{10}$$

where $f(x)$ denotes the objective value of the solution x and m denotes the number of the selected elements in the current solution, i.e., $m = \sum_{i=1}^{n} x_i$.

It can also be written as

$$\Delta_i = \begin{cases} \dfrac{-f(x)}{m+1} + \dfrac{p_i}{m+1}, & x_i = 0 \\ \dfrac{f(x)}{m-1} - \dfrac{p_i}{m-1}, & x_i = 1. \end{cases} \tag{11}$$

After flipping a variable x_i to be its complement $1 - x_i$, the potential p_j is updated as

$$p_j = \begin{cases} p_j + d_{ij}, & x_i = 0,\ i \neq j \\ p_j - d_{ij}, & x_i = 1,\ i \neq j \\ p_j, & i = j. \end{cases} \tag{12}$$

Using the abovementioned method, the computational complexity of each iteration is reduced to $O(n)$.

Tabu rule requires that each time a move is performed, its reverse move is prohibited from being performed during the following specified number of iterations (called tabu tenure). A dynamic tabu tenure management strategy that divides the iterations into different intervals and uses a specified order of different tabu tenures for these intervals is employed. This strategy was first proposed in [24] for solving the graph partitioning problem and demonstrated to be effective in [9] for the Max-Mean DP. Furthermore, an aspiration rule overrides the tabu rule if performing a tabu move leads to a better solution than the best solution found so far.

The proposed tabu search phase repeatedly carries out the following iterations until the best solution can not be improved for a consecutive number of iterations λ, called tabu search depth. To be specific, all the moves are first categorized as tabu and nontabu according to the tabu and aspiration rules. Then, the objective function value of each neighborhood solution caused by the 1-flip move is quickly evaluated according to (11) and (12). Finally, either the nontabu move with the best objective value or a tabu move that satisfies the aspiration rule is chosen to be performed to move to the next solution.

2.5. Elite Set Updating and Rebuilding. We employ a popular quality-based elite set updating strategy, where the elite set updating happens whenever the newly generated solution gets a better objective value than the worst solution in *ES*. In this case, the worst solution is replaced by this solution to execute the *ES* updating procedure. Otherwise, this new solution is disregared and *ES* updating fails. When *ES* does not undergo successful updates for continuous η times, we rebuild *ES* to restart the search. The rebuilding strategy we use is to keep the best solution found so far in *ES* and generate the other solutions in the same way as initializing *ES*.

3. Computational Results and Comparisons

3.1. Benchmarks and Experimental Protocols. In order to evaluate our proposed EDA-TS algorithm, we use 7 different sizes of benchmarks n = 500, 750, 1000, 1500, 2000, 3000, and 5000 with a total of 140 instances, where each size of benchmarks includes 20 instances from two groups (TypeI and TypeII). These instances are widely used in the literature for performance comparisons among algorithms and can be downloaded from the websites (http://www.optsicom.es/edp, http://www.mi.sanu.ac.rs/~nenad/edp, http://www.info.univ-angers.fr/pub/hao/maxmeandp.html).

Similar to the other reference algorithms, we use time limits as the stopping condition, which are set to be 100 seconds, 1000 seconds, and 2000 seconds for problem instances with $n \leq 1000$, $1000 < n < 5000$, and $n = 5000$, respectively. Our EDA-TS algorithm is programmed in C++ and compiled using GNU g++ with −O3 flag on an Intel Xeon E5440@2.83GHz CPU. Given the random nature of our EDA-TS algorithm, we give 20 independent runs for each instance.

3.2. Parameter Sensitivity Analysis. Our EDA-TS algorithm includes seven parameters, which are the size of the elite set $|ES|$, the threshold coefficient α in determining the preference candidate set, the threshold coefficient β in determining the conditional preference candidate set, the probability γ to choose a variable from the maximum conditional preference set, the minimum number $depth_c$ of added variables when constructing a solution, the tabu search depth λ, and the number of continuous unsuccessful elite set updates η. For most algorithms that maintain an elite set of solutions, a small value between 10 and 20 is usually a preferable setting [5, 8, 9] for $|ES|$. The settings of these parameters are listed in Table 2, where the parameters in the tabu search phase use the recommended settings in [9] and the parameters α, β, and γ in the EDA guided solution construction phase are determined based on the following preliminary experiments.

Specifically, we choose 20 most challenging problem instances and vary $\alpha \in [0.2, 0.5]$ with a step size of 0.05,

Table 2: Parameter settings for the proposed EDA-TS algorithm.

Parameters	Description	Section	Value
$\|ES\|$	the size of the elite set	2.2	10
α	the threshold coefficient in determining the preference candidate set *Cnd*	2.3.2	0.35
β	the threshold coefficient in determining the conditional preference set *MaxSet*	2.3.2	0.5
γ	the probability to choose a variable from *MaxSet*	2.3.2	0.1
$depth_c$	the minimum number of added variables when constructing a solution	2.3.2	$n/4$
λ	the tabu search depth	2.4	50000
η	the number of continuous unsuccessful *ES* updates	2.5	50

$\beta \in [0.2, 0, 5]$ with a step size of 0.05, and $\gamma \in [0.1, 0.5]$ with a step size of 0.1. During each run, we only change the setting of one parameter while keeping the settings of the other parameters unchanged. For each chosen instance, we give 20 independent runs and record the average objective values.

In order to avoid many tabulated results, we report the results of the Friedman statistical testings and examine if different parameter settings present significant differences. Our experimental findings disclose that different settings of the parameter α present significant difference with the p-value of 0.01639 and setting $\alpha = 0.35$ yields better results. In addition, the parameters β and γ are not sensitive with the p-values of 0.3369 and 0.0686, respectively.

3.3. Experimental Results. In this section, we report experimental comparisons between our EDA-TS algorithm and the best algorithms in the literature on solving different sizes of benchmarks.

Table 3 shows results of different algorithms on solving the instances with 500 variables, including a path relinking algorithm (PR) [10], a hybrid heuristic (HH) [11], a tabu search algorithm (TPTS) [13], a hybrid three-phase approach (HTP) [12], a tabu search-based memetic algorithm (TSMA) [9], a variable neighborhood search approach (GVNS) [14], and our EDA-TS algorithm. The results of other algorithms are directly extracted from [9, 14]. The first column represents the name of each instance. Columns 2 to 5 report the best solution values *Bst* found by the algorithms PR, HH, HTP, and TPTS, respectively. The average solution values and computational time of these algorithms are not available in the literature. Columns 6 to 14 report computational results of the algorithms TSMA, GVNS, and EDA-TS, where the three columns under each algorithm report the best solution values *Bst*, the average solution values *Avg*, and the computational time *Time*, respectively.

To make the computational times of TSMA, GVNS, and EDA-TS comparable, we use the PassMark performance test results to evaluate the performance differences between different computing platforms. We find that the mark of the computer where GVNS runs on (Intel(R) Core(TM) i7-2600@3.4GHz) is 8203 and the mark of the computer (Intel Xeon E5440@2.83GHz) where TSMA and EDA-TS run is 3976. Hence, the time column of GVNS is multiplied by 2.06 when comparing with TSMA and EDA-TS.

As can be seen from Table 3, EDA-TS and TSMA perform the same best in terms of the best solution values, performing better than the other algorithms PR, HH, HTP, TPTS, and GVNS. In particular, the average solution values found by both EDA-TS and TSMA are the same as the best solution value for each instance, performing better than GVNS. The wilcoxon statistical testings performed for EDA-TS and each other algorithm indicate that EDA-TS is significantly better than PR, HH, TS, and HTP with the p-values less than 0.05 in terms of the best solution values. Although EDA-TS is not significantly better than GNVS with respect to the best solution values, it performs significantly better than GVNS with respect to the average solution values by obtaining the p-value of 0.001656. Finally, the normalized average CPU time 0.71 seconds of EDA-TS is shorter against 52.08 seconds of GVNS and 1.17 seconds of TSMA.

Since PR, HH, and HTP do not report results for instances with more than 500 variables, we are not able to include comparisons with these algorithms in Tables 4–9. Table 4 compares EDA-TS with TPTS, GVNS, and TSMA for solving instances with 750 variables. From Table 4, we observe that EDA-TS and TSMA perform much better than GVNS and TPTS in terms of the best solution values. In addition, both EDA-TS and TSMA are able to reach the same results for all the 20 independent runs, dominating GVNS and TPTS in terms of the average solution values. Wilcoxon statistical testings in terms of the best solution values indicate that EDA-TS is significantly better than GVNS and TPTS by obtaining the p-values of 0.0001428 and 0.009152, respectively. In terms of the average solutions, EDA-TS is also significantly better than GVNS by obtaining the p-values of 0.000213. The normalized average CPU time 2.09 seconds of EDA-TS is shorter compared to 250.15 seconds of GVNS and 3.38 seconds of TSMA.

Table 5 compares EDA-TS with TPTS, GVNS, and TSMA for solving instances with 1000 variables. We find that EDA-TS and TSMA perform much better than GVNS and TPTS, as observed in Table 4. Statistical testing indicates that EDA-TS is significantly better than TPTS by obtaining the p-values of 0.00009556. Moreover, EDA-TS is significantly better than GVNS in terms of both best and average results, by obtaining the p-values of 0.03603 and 0.00009502, respectively. The normalized average CPU time 2.46 seconds of EDA-TS is shorter compared to 435.30 seconds of GVNS and 4.07 seconds of TSMA.

Table 3: Computational comparisons between EDA-TS and other state-of-the-art algorithms on instances with 500 variables.

Instances	PR[10]	HH[11]	HTP[12]	TPTS[13]	TSMA[9]			GVNS[14]			EDA-TS		
	Bst	Bst	Bst	Bst	Bst	Avg	Time	Bst	Avg	Time	Bst	Avg	Time
MDPI1_500	78.61	81.25	81.28	81.28	81.28	81.28	0.69	81.28	81.28	0.75	81.28	81.28	0.33
MDPI2_500	76.87	77.45	77.79	77.60	78.61	78.61	1.43	78.61	78.28	76.86	78.61	78.61	1.04
MDPI3_500	75.69	75.31	76.30	75.65	76.30	76.30	2.71	76.30	76.09	87.62	76.30	76.30	1.59
MDPI4_500	81.81	82.28	82.33	81.47	82.33	82.33	0.95	82.33	82.31	3.54	82.33	82.33	0.53
MDPI5_500	78.57	80.01	80.08	79.92	80.35	80.35	2.80	80.35	80.31	73.02	80.35	80.35	1.59
MDPI6_500	79.64	81.12	81.25	79.93	81.25	81.25	0.78	81.25	81.08	46.50	81.25	81.25	0.40
MDPI7_500	75.50	78.09	78.16	77.71	78.16	78.16	0.92	78.16	78.02	80.44	78.16	78.16	0.81
MDPI8_500	76.98	79.01	79.06	78.70	79.14	79.14	1.27	79.14	79.14	3.54	79.14	79.14	0.66
MDPI9_500	75.72	76.98	77.36	77.15	77.42	77.42	2.37	77.37	77.36	59.46	77.42	77.42	1.74
MDPI10_500	80.38	81.24	81.25	81.02	81.31	81.31	0.91	81.31	81.31	1.49	81.31	81.31	0.49
MDPII1_500	108.15	109.16	109.38	109.33	109.61	109.61	0.75	109.61	109.61	1.03	109.61	109.61	0.46
MDPII2_500	103.29	105.06	105.33	104.81	105.72	105.72	0.88	105.72	105.58	13.34	105.72	105.72	0.45
MDPII3_500	106.30	107.64	107.79	107.18	107.82	107.82	0.89	107.82	107.77	13.45	107.82	107.82	0.46
MDPII4_500	104.62	105.37	106.10	105.69	106.10	106.10	0.56	106.10	106.10	5.05	106.10	106.10	0.35
MDPII5_500	103.61	106.37	106.55	106.59	106.86	106.86	0.99	106.75	106.74	0.52	106.86	106.86	0.76
MDPII6_500	104.81	105.52	105.77	106.17	106.30	106.30	0.98	106.30	106.18	14.91	106.30	106.30	0.54
MDPII7_500	104.50	106.61	107.06	106.92	107.15	107.15	0.88	107.15	107.15	11.24	107.15	107.15	0.44
MDPII8_500	100.02	103.41	103.78	103.49	103.78	103.78	0.59	103.78	103.78	0.35	103.78	103.78	0.39
MDPII9_500	104.93	106.20	106.24	105.97	106.62	106.62	1.11	106.62	106.52	2.80	106.62	106.62	0.58
MDPII10_500	103.50	103.79	104.15	103.56	104.65	104.65	1.01	104.65	104.23	9.75	104.65	104.65	0.65

Table 4: Computational comparisons between EDA-TS and other state-of-the-art algorithms on instances with 750 variables.

Instances	TPTS[13]	TSMA[9]			GVNS[14]			EDA-TS		
	Bst	Bst	Avg	Time	Bst	Avg	Time	Bst	Avg	Time
MDPI1_750	95.86	96.65	96.65	4.31	96.63	96.38	58.67	96.65	96.65	4.89
MDPI2_750	97.42	97.56	97.56	3.82	97.56	97.54	106.10	97.56	97.56	2.16
MDPI3_750	96.97	97.80	97.80	1.81	97.80	97.54	119.97	97.80	97.80	0.64
MDPI4_750	95.21	96.04	96.04	4.38	95.88	95.75	108.79	96.04	96.04	4.16
MDPI5_750	96.65	96.76	96.76	0.65	96.76	96.76	52.16	96.76	96.76	0.86
MDPI6_750	99.25	99.86	99.86	5.55	99.72	99.45	89.43	99.86	99.86	4.92
MDPI7_750	96.26	96.55	96.55	1.01	96.55	96.07	219.43	96.55	96.55	0.62
MDPI8_750	96.46	96.73	96.73	1.73	96.73	96.49	202.35	96.73	96.73	1.09
MDPI9_750	96.78	98.06	98.06	2.18	98.06	97.77	227.97	98.06	98.06	1.04
MDPI10_750	99.85	100.06	100.06	3.42	100.06	99.91	19.62	100.06	100.06	1.41
MDPII1_750	127.69	128.86	128.86	5.66	128.72	128.49	206.66	128.86	128.86	3.71
MDPII2_750	130.79	130.95	130.95	2.31	130.95	130.52	30.88	130.95	130.95	1.08
MDPII3_750	129.40	129.78	129.78	11.64	129.72	129.50	169.83	129.78	129.78	4.95
MDPII4_750	125.68	126.58	126.58	1.48	126.58	126.58	28.89	126.58	126.58	0.80
MDPII5_750	128.13	129.12	129.12	1.32	129.05	128.80	96.16	129.12	129.12	0.99
MDPII6_750	128.55	129.03	129.03	7.98	129.02	128.82	87.80	129.03	129.03	4.41
MDPII7_750	124.91	125.65	125.65	3.38	125.65	125.37	267.80	125.65	125.65	1.85
MDPII8_750	130.66	130.94	130.94	1.91	130.94	130.88	68.90	130.94	130.94	0.83
MDPII9_750	128.89	128.89	128.89	1.30	128.67	127.32	108.51	128.89	128.89	0.59
MDPII10_750	132.99	133.27	133.27	1.81	133.17	133.03	158.44	133.27	133.27	0.88

Table 5: Computational comparisons between EDA-TS and other state-of-the-art algorithms on instances with 1000 variables.

Instances	TPTS[13]	TSMA[9]			GVNS[14]			EDA-TS		
	Bst	Bst	Avg	Time	Bst	Avg	Time	Bst	Avg	Time
MDPI1_1000	118.76	119.17	119.17	8.25	119.17	118.90	335.60	119.17	119.17	3.91
MDPI2_1000	113.22	113.52	113.52	3.52	113.52	113.41	199.62	113.52	113.52	2.18
MDPI3_1000	114.51	115.14	115.14	2.32	114.83	114.62	254.93	115.14	115.14	0.93
MDPI4_1000	110.53	111.15	111.15	3.58	111.15	110.76	304.17	111.15	111.15	3.08
MDPI5_1000	111.24	112.72	112.72	1.61	112.72	112.02	67.02	112.72	112.72	0.95
MDPI6_1000	112.08	113.20	113.20	7.72	113.18	112.83	382.65	113.20	113.20	6.93
MDPI7_1000	110.94	111.56	111.56	1.88	111.56	111.16	284.62	111.56	111.56	1.67
MDPI8_1000	110.29	111.26	111.26	3.55	111.07	110.64	331.39	111.26	111.26	1.57
MDPI9_1000	115.78	115.96	115.96	2.38	115.96	115.87	105.09	115.96	115.96	1.61
MDPI10_1000	114.29	114.73	114.73	2.16	114.73	114.34	91.50	114.73	114.73	1.48
MDPII1_1000	145.46	147.94	147.94	1.60	147.94	147.20	155.02	147.94	147.94	1.14
MDPII2_1000	150.49	151.38	151.38	1.78	151.38	150.98	237.94	151.38	151.38	1.20
MDPII3_1000	149.36	150.79	150.79	4.92	150.79	149.65	227.01	150.79	150.79	2.24
MDPII4_1000	147.91	149.18	149.18	3.80	149.18	148.57	180.51	149.18	149.18	1.60
MDPII5_1000	150.23	151.52	151.52	3.28	151.49	150.97	96.20	151.52	151.52	1.46
MDPII6_1000	147.29	148.34	148.34	3.22	148.25	147.20	94.32	148.34	148.34	1.48
MDPII7_1000	148.41	148.74	148.74	6.30	148.73	148.34	404.66	148.74	148.74	2.82
MDPII8_1000	145.87	147.83	147.83	13.52	147.83	147.08	125.13	147.83	147.83	6.73
MDPII9_1000	145.67	147.08	147.08	3.83	147.08	146.18	144.99	147.08	147.08	5.04
MDPII10_1000	148.40	150.05	150.05	2.13	150.05	149.57	203.82	150.05	150.05	1.12

Table 6: Computational comparisons between EDA-TS and GVNS on instances with 1500 variables.

Instances	GVNS[14]			EDA-TS		
	Bst	Avg	Time	Bst	Avg	Time
MDPI1_1500	136.26	135.65	117.73	136.54	136.54	72.57
MDPI2_1500	138.00	136.71	397.15	138.34	138.34	12.49
MDPI3_1500	138.91	138.32	471.60	139.20	139.20	5.17
MDPI4_1500	139.81	139.21	141.11	140.17	140.17	14.28
MDPI5_1500	136.47	136.20	362.80	137.13	137.13	61.25
MDPI6_1500	136.22	135.13	283.58	136.51	136.51	15.87
MDPI7_1500	137.65	136.84	185.13	137.97	137.97	3.72
MDPI8_1500	138.02	137.58	222.33	138.73	138.73	76.34
MDPI9_1500	136.30	135.25	325.65	136.50	136.50	201.98
MDPI10_1500	140.33	139.80	226.43	140.33	140.33	5.86
MDPII1_1500	181.67	181.08	202.58	182.09	182.09	58.87
MDPII2_1500	185.48	185.01	331.15	186.24	186.24	12.89
MDPII3_1500	181.55	180.57	264.69	182.14	182.14	7.91
MDPII4_1500	184.91	184.34	113.97	185.56	185.55	608.96
MDPII5_1500	190.15	189.76	195.12	190.86	190.86	7.08
MDPII6_1500	183.14	182.32	107.77	183.58	183.58	6.00
MDPII7_1500	179.34	178.34	244.34	179.82	179.82	33.76
MDPII8_1500	186.60	185.63	219.73	186.60	186.60	5.85
MDPII9_1500	181.43	180.39	303.44	181.92	181.92	53.99
MDPII10_1500	182.70	181.98	385.14	183.38	183.38	65.83

Table 7: Computational comparisons between EDA-TS and GVNS on instances with 2000 variables.

Instances	GVNS[14]			EDA-TS		
	Bst	Avg	Time	Bst	Avg	Time
MDPI1_2000	158.03	157.17	240.74	158.59	158.59	17.78
MDPI2_2000	162.91	162.34	401.78	163.94	163.94	48.77
MDPI3_2000	158.98	157.81	251.86	159.57	159.57	316.30
MDPI4_2000	159.14	158.46	249.16	160.19	160.19	80.73
MDPI5_2000	156.11	155.38	314.34	156.81	156.80	223.70
MDPI6_2000	161.61	160.29	485.13	161.84	161.84	27.10
MDPI7_2000	157.58	157.13	303.16	158.34	158.34	13.89
MDPI8_2000	161.43	160.02	367.14	161.45	161.45	52.22
MDPI9_2000	159.15	158.15	533.79	160.19	160.19	158.81
MDPI10_2000	160.90	159.77	224.99	161.64	161.64	5.54
MDPII1_2000	208.85	208.19	303.51	209.85	209.85	27.10
MDPII2_2000	218.19	217.53	210.86	218.40	218.40	66.31
MDPII3_2000	209.57	208.55	184.25	210.82	210.82	76.56
MDPII4_2000	211.99	211.30	370.77	212.42	212.42	38.41
MDPII5_2000	215.33	214.39	364.76	216.09	216.09	9.59
MDPII6_2000	210.61	208.96	371.52	211.77	211.77	7.46
MDPII7_2000	209.65	207.50	186.52	209.78	209.78	82.86
MDPII8_2000	212.43	211.23	548.82	212.58	212.58	86.94
MDPII9_2000	214.61	213.64	302.15	215.01	215.01	40.83
MDPII10_2000	210.06	208.76	327.43	210.74	210.74	232.23

Table 8: Computational comparisons between EDA-TS and other state-of-the-art algorithms on instances with 3000 variables.

Instances	TPTS[13]	TSMA[9]			EDA-TS		
		Bst	Avg	Time	Bst	Avg	Time
MDPI1_3000	188.10	189.05	189.05	88.36	189.05	189.05	69.35
MDPI2_3000	186.47	187.39	187.39	60.71	187.39	187.39	53.64
MDPI3_3000	184.34	185.67	185.66	352.85	185.67	185.65	399.44
MDPI4_3000	185.59	186.16	186.15	300.37	186.16	186.16	240.45
MDPI5_3000	186.23	187.55	187.55	61.29	187.55	187.55	94.16
MDPI6_3000	189.09	189.43	189.43	51.99	189.43	189.43	48.21
MDPI7_3000	187.45	188.24	188.24	86.57	188.24	188.24	120.95
MDPI8_3000	185.74	186.80	186.80	48.04	186.80	186.80	38.70
MDPI9_3000	187.11	188.23	188.23	151.78	188.23	188.23	82.66
MDPI10_3000	184.69	185.68	185.62	228.72	185.68	185.67	510.41
MDPII1_3000	252.18	252.32	252.32	59.70	252.32	252.32	42.42
MDPII2_3000	248.70	250.06	250.06	220.10	250.06	250.06	248.10
MDPII3_3000	250.53	251.91	251.91	146.32	251.91	251.91	139.36
MDPII4_3000	253.10	253.94	253.94	370.76	253.94	253.94	352.17
MDPII5_3000	252.56	253.26	253.26	374.00	253.26	253.26	308.80
MDPII6_3000	249.72	250.68	250.68	55.35	250.68	250.68	55.90
MDPII7_3000	249.59	251.13	251.13	74.72	251.13	251.13	88.61
MDPII8_3000	252.06	253.00	253.00	79.82	253.00	253.00	123.56
MDPII9_3000	251.36	252.43	252.43	90.27	252.43	252.43	58.50
MDPII10_3000	251.12	252.40	252.40	13.18	252.40	252.40	8.73

Table 9: Computational comparisons between EDA-TS and other state-of-the-art algorithms on instances with 5000 variables.

Instances	TPTS[13]	TSMA[9]			EDA-TS		
		Bst	Avg	Time	Bst	Avg	Time
MDPI1_5000	236.33	240.16	240.10	312.13	240.16	240.04	905.58
MDPI2_5000	239.01	241.83	241.79	1244.36	241.83	241.77	958.60
MDPI3_5000	238.47	240.89	240.89	810.48	240.89	240.85	992.57
MDPI4_5000	237.40	241.00	240.98	653.64	241.00	240.93	1240.33
MDPI5_5000	240.04	242.48	242.48	735.16	242.48	242.44	1104.64
MDPI6_5000	238.00	240.32	240.31	976.02	**240.38**	240.27	992.36
MDPI7_5000	239.74	242.81	242.77	259.50	**242.82**	242.76	965.79
MDPI8_5000	237.92	241.19	241.16	1148.60	241.19	241.13	909.39
MDPI9_5000	235.91	239.76	239.67	1219.71	239.76	239.54	1001.40
MDPI10_5000	241.80	243.47	243.37	457.28	243.47	243.34	981.99
MDPII1_5000	316.75	322.24	322.18	1519.05	322.24	322.16	1114.21
MDPII2_5000	323.68	327.30	327.01	1103.13	327.30	327.10	731.65
MDPII3_5000	321.93	324.81	324.80	955.81	324.81	324.78	1043.24
MDPII4_5000	317.68	322.24	322.20	664.10	322.23	322.12	1059.47
MDPII5_5000	317.75	322.49	322.38	1014.90	**322.50**	322.37	971.06
MDPII6_5000	319.39	322.95	322.70	352.88	322.95	322.69	1405.72
MDPII7_5000	319.98	322.85	322.79	714.31	322.85	322.76	1164.19
MDPII8_5000	318.85	323.11	323.05	879.48	323.11	322.88	1168.86
MDPII9_5000	320.44	323.54	323.34	569.73	323.54	323.30	952.74
MDPII10_5000	320.85	324.52	324.41	752.95	324.52	324.42	960.22

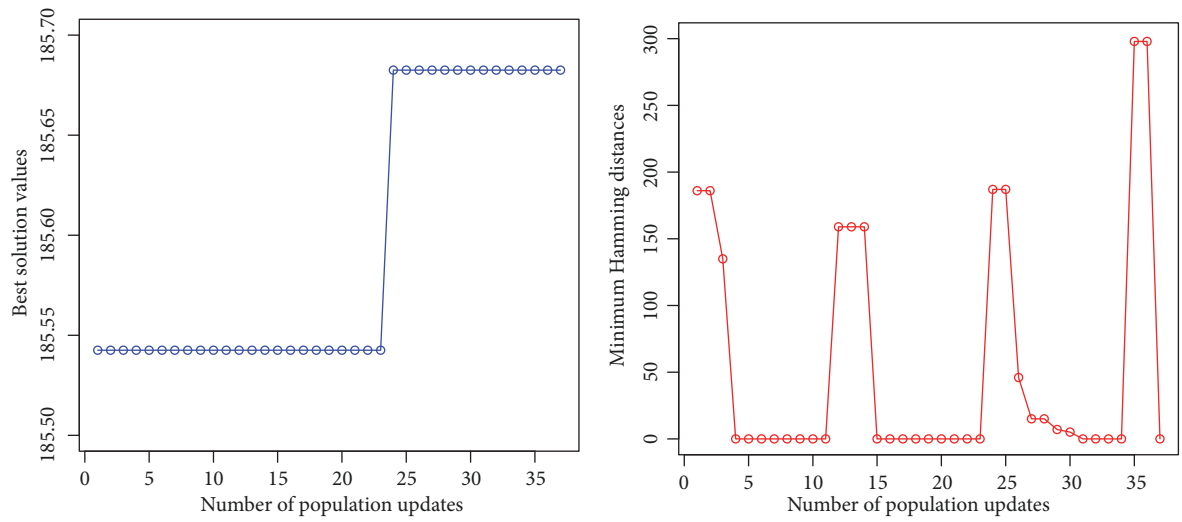

Figure 1: Best solution values and minimum Hamming distances as the population evolves: Instance MDPI10_3000.

Tables 6 and 7 compare EDA-TS with GVNS for solving instances with 1500 and 2000 variables since GVNS is the only algorithm that reports results on these instances. From these two tables, we find that EDA-TS is able to find better solution values for 38 out of 40 instances. Moreover, the average solution values obtaind by EDA-TS are much better than GVNS. Further statistical testings indicate that EDA-TS is significantly better than GVNS in terms of best and average results. For solving instances with 1500 variables, the normalized average CPU time of EDA-TS and GVNS is 66.53 seconds against 525.44 seconds. For solving instances with 2000 variables, the normalized average CPU time of EDA-TS and GVNS is 80.66 seconds against 673.89 seconds. These results indicate that EDA-TS performs better than GVNS in terms of both solution quality and computational time.

Tables 8 and 9 present results of EDA-TS, TSMA, and TPTS for the most challenging instances. From Table 8, we find that EDA-TS performs as well as TSMA in terms of the best and average solution values, which are better than TPTS. Further statistical testings indicate that EDA-TS is significantly better than TPTS by obtaining the p-values of 0.000001907 but is not statistically comparable to TSMA. The normalized average CPU time of EDA-TS and TSMA is 154.21 seconds against 145.75 seconds.

From Table 9, we first observe that EDA-TS finds better results than TSMA for the instances MDPI6_5000, MDPI7_5000, and MDPII5_5000 (marked in bold) and matches results for the other instances. Statistical testings indicate that EDA-TS is significantly better than TPTS with the p-value of 0.000001907 but not better than TSMA in a significant level with the p-value of 0.3447. However, EDA-TS is significantly outperformed by TSMA in terms of the average solution values with the p-value of 0.001592. The normalized average CPU time of EDA-TS and TSMA is 1031.20 seconds against 817.16 seconds.

To conclude, these experimental comparisons demonstrate the effectiveness and efficacy of our proposed EDA-TS algorithm.

3.4. The Balance between Intensification and Diversification. Our proposed EDA-TS algorithm achieves a good balance between intensification and diversification during the search. For characterizing search intensification, we record how the best solution value changes as the population (elite set) evolves. For characterizing search diversification, we record the minimum Hamming distances (i.e., we calculate the Hamming distance for each pair of solutions in this elite set and use the minimum value of the Hamming distances to evaluate the diversification.) We perform experiments on two large instances MDPI10_3000 and MDPI3_5000. The experimental results are shown in Figures 1 and 2.

Figures 1(a) and 2(a) disclose that the best solution values are constantly improving as the population updates, which indicates good intensification of the EDA-TS algorithm during the search. Figures 1(b) and 2(b) show that the minimum Hamming distance fluctuates as the population updates. In some search periods, the observation that the distances between solutions in the population are decreased indicates that the EDA-TS algorithm focuses on intensification since high-quality solutions are usually closer to each other. When the algorithm has already explored for certain periods, the search is switched to perform diversification and the distances between solutions are increased. Hence, this experiment reveals that our EDA-TS algorithm achieves a good balance between intensification and diversification.

3.5. Effectiveness of the EDA Component for Search Diversification. In order to verify the role of the EDA component for achieving good search diversification, we produce two

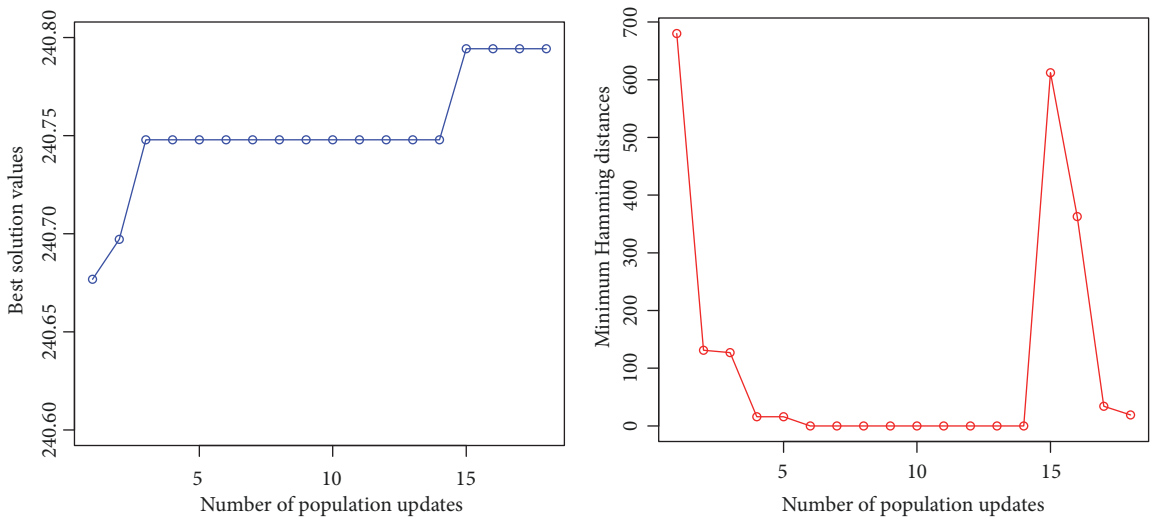

Figure 2: Best solution values and minimum Hamming distances as the population evolves: MDPI3_5000.

EDA-TS variants by using popular diversification strategies in the literature to replace EDA. The first variant called RG-TS uses a randomized greedy strategy and the second variant called RP-TS uses a randomized perturbation strategy. In RG-TS, the solution is iteratively constructed, where each iteration first builds a restricted candidate list of elements that produces a large objective gain and then randomly chooses an element from this list to join in the partial solution. Repeat this procedure until no objective gain is positive. In RP-TS, the best solution in the population is perturbed, i.e., randomly changing a certain number of elements, to produce a new solution.

We perform additional experiments for RG-TS and RP-TS on a set of 20 most challenging instances with 3000 and 5000 variables (whose best objective values are difficult to be obtained comparing to the other instances), in which the experimental settings are the same as the previous EDA-TS algorithm. For each instance, we record the percent deviations of the best and average solution values obtained by RG-TS and RP-TS from those found by EDA-TS and show the experimental results in Figure 1.

Figures 3(a) and 3(b) show the best percent deviations of RG-TS and RP-TS from EDA-TS for each of the 20 chosen instances, respectively. From Figure 3(a), we find that the best percent deviations of RG-TS from EDA-TS are nonpositive for all the instances except for 1 instance. From Figure 3(b), we find that the best percent deviations of RP-TS from EDA-TS are nonpositive for all the instances. These observations disclose that EDA-TS finds better or equal best objective values than RG-TS and RP-TS.

Figures 3(c) and 3(d) show the average percent deviations of RG-TS and RP-TS from EDA-TS for each of the 20 chosen instances, respectively. From Figure 3(c), we find that the average percent deviations of RG-TS from EDA-TS are negative for all the instances. From Figure 3(d), we find that the average percent deviations of RP-TS from EDA-TS are negative for all the instances except for 1 instance. These observations disclose that EDA-TS finds better average objective values than RG-TS and RP-TS.

In conclusion, this experiment demonstrates that the incorporated EDA component plays an essential role to the effectiveness of the algorithm.

4. Conclusion

In this paper, we have presented for the first time a hybrid metaheuristic that combines estimation of distribution algorithm with tabu search (EDA-TS) for solving the max-mean dispersion problem. The proposed algorithm mainly relies on a dedicated EDA guided solution construction method for diversification and a tabu search optimization method for intensification. Extensive experiments disclose that our EDA-TS algorithm outperforms or competes favorably with state-of-the-art algorithms in the literature. Additional analysis on the parameter sensitivity and the merit of the EDA procedure as well as the search balance between intensification and diversification sheds light on the effectiveness of the algorithm. Our findings motivate further investigation of hybrid metaheuristics of combining local search with other learning strategies.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

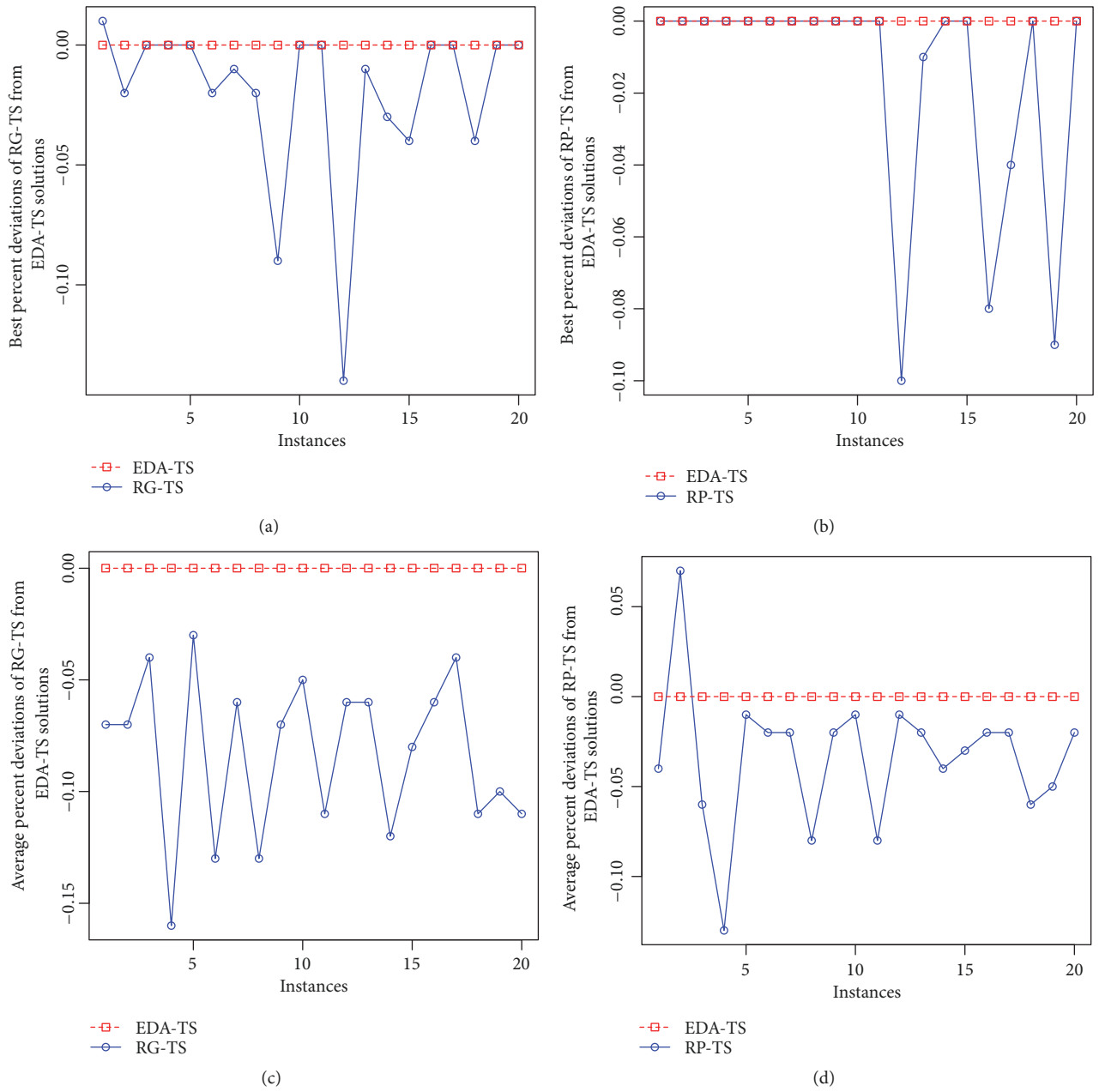

Figure 3: Effectiveness of the incorporated EDA component in the proposed algorithm.

Acknowledgments

This research has been supported by the National Natural Science Foundation of China (NSFC) under Grant 71501157.

References

[1] O. A. Prokopyev, N. Kong, and D. L. Martinez-Torres, "The equitable dispersion problem," *European Journal of Operational Research*, vol. 197, no. 1, pp. 59–67, 2009.

[2] F. Della Croce, A. Grosso, and M. Locatelli, "A heuristic approach for the max-min diversity problem based on max-clique," *Computers & Operations Research*, vol. 36, no. 8, pp. 2429–2433, 2009.

[3] D. C. Porumbel, J. Hao, and F. Glover, "A simple and effective algorithm for the MaxMin diversity problem," *Annals of Operations Research*, vol. 186, no. 1, pp. 275–293, 2011.

[4] R. Aringhieri, R. Cordone, and A. Grosso, "Construction and improvement algorithms for dispersion problems," *European Journal of Operational Research*, vol. 242, no. 1, pp. 1–13, 2015.

[5] Y. Wang, Q. Wu, and F. Glover, "Effective metaheuristic algorithms for the minimum differential dispersion problem," *European Journal of Operational Research*, vol. 258, no. 3, pp. 829–843, 2017.

[6] A. P. Punnen, S. Taghipour, D. Karapetyan, and B. Bhattacharyya, "The quadratic balanced optimization problem," *Discrete Optimization*, vol. 12, pp. 47–60, 2014.

[7] R. Aringhieri and R. Cordone, "Comparing local search metaheuristics for the maximum diversity problem," *Journal of the Operational Research Society*, vol. 62, no. 2, pp. 266–280, 2011.

[8] Y. Wang, J.-K. Hao, F. Glover, and Z. Lü, "A tabu search based memetic search for the maximum diversity problem," *Engineering Applications of Artificial Intelligence*, vol. 27, pp. 103–114, 2014.

[9] X. Lai and J.-K. Hao, "A tabu search based memetic algorithm for the max-mean dispersion problem," *Computers & Operations Research*, vol. 72, pp. 118–127, 2016.

[10] R. Martí and F. Sandoya, "GRASP and path relinking for the equitable dispersion problem," *Computers & Operations Research*, vol. 40, no. 12, pp. 3091–3099, 2013.

[11] F. Della Croce, M. Garraffa, and F. Salassa, "A hybrid heuristic approach based on a quadratic knapsack formulation for the max-mean dispersion problem," in *Combinatorial Optimization*, Lecture Notes in Computer Science, pp. 186–197, 2014.

[12] F. Della Croce, M. Garraffa, and F. Salassa, "A hybrid three-phase approach for the Max-Mean Dispersion Problem," *Computers & Operations Research*, vol. 71, pp. 16–22, 2016.

[13] R. Carrasco, A. Pham, M. Gallego, F. Gortázar, R. Martí, and A. Duarte, "Tabu search for the max–mean dispersion problem," *Knowledge-Based Systems*, vol. 85, pp. 256–264, 2015.

[14] J. Brimberg, N. Mladenović, R. Todosijević, and D. Urošević, "Less is more: Solving the Max-Mean diversity problem with variable neighborhood search," *Information Sciences*, vol. 382-383, pp. 179–200, 2017.

[15] M. Garraffa, F. Della Croce, and F. Salassa, "An exact semidefinite programming approach for the max-mean dispersion problem," *Journal of Combinatorial Optimization*, vol. 34, no. 1, pp. 71–93, 2017.

[16] A. Arin and G. Rabadi, "Integrating estimation of distribution algorithms versus Q-learning into Meta-RaPS for solving the 0-1 multidimensional knapsack problem," *Computers & Industrial Engineering*, vol. 112, pp. 706–720, 2017.

[17] P. A. N. Bosman and D. Thierens, "Expanding from discrete to continuous estimation of distribution algorithms: The IDEA," in *Parallel Problem Solving from Nature PPSN VI*, M. Schoenauer, K. Deb, and G. Rudolph, Eds., vol. 1917 of *Lecture Notes in Computer Science*, pp. 767–776, Springer, 2000.

[18] P. Larrañaga, "A review on estimation of distribution algorithms," in *Estimation of Distribution Algorithms*, P. Larranaga and J. A. Lozano, Eds., vol. 2, pp. 57–100, Kluwer Academic Publishers, 2002.

[19] P. Larranaga and J. A. Lozano, *Estimation of Distribution Algorithms*, Kluwer Academic Publishers, 2002.

[20] M. Pelikan, D. E. Goldberg, and F. G. Lobo, "A survey of optimization by building and using probabilistic models," *Computational Optimization and Applications*, vol. 21, no. 1, pp. 5–20, 2002.

[21] U. Aickelin, E. K. Burke, and J. Li, "An estimation of distribution algorithm with intelligent local search for rule-based nurse rostering," *Journal of the Operational Research Society*, vol. 58, no. 12, pp. 1574–1585, 2017.

[22] J. M. Peña, V. Robles, P. Larrañaga, V. Herves, F. Rosales, and M. S. Pérez, "GA-EDA: hybrid evolutionary algorithm using genetic and estimation of distribution algorithms," in *Proceedings of the International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE 2004)*, R. Orchard, Ed., Lecture Notes in Artificial Intelligence, pp. 361–371, 2004.

[23] Q. Zhang, J. Sun, E. Tsang, and J. Ford, "Estimation of distribution algorithm with 2-opt local search for the quadratic assignment problem," *Towards a New Evolutionary Computation*, pp. 281–292, 2006.

[24] P. Galinier, Z. Boujbel, and M. Coutinho Fernandes, "An efficient memetic algorithm for the graph partitioning problem," *Annals of Operations Research*, vol. 191, no. 1, pp. 1–22, 2011.