Soft Comput (2012) 16:711–728
DOI 10.1007/s00500-011-0780-6

ORIGINAL PAPER

Learnable tabu search guided by estimation of distribution for maximum diversity problems

Jiahai Wang · Ying Zhou · Yiqiao Cai · Jian Yin

Published online: 3 November 2011

Abstract This paper presents a learnable tabu search (TS) guided by estimation of distribution algorithm (EDA), called LTS-EDA, for maximum diversity problem. The LTS-EDA introduces knowledge model and can extract knowledge during the search process of TS, and thus it adopts dual or cooperative evolution/search structure, consisting of probabilistic model space in clustered EDA and solution space searched by TS. The clustered EDA, as a learnable constructive method, is used to create a new starting solution, and the simple TS, as an improvement method, attempts to improve the solution created by the clustered EDA in the LTS-EDA. A distinguishing feature of the LTS-EDA is the usage of the clustered EDA with effective linkage learning to guide TS. In the clustered EDA, different clusters (models) focus on different substructures, and the combination of information from different clusters (models) effectively combines substructures. The LTS-EDA is tested on 50 large size benchmark problems with the size ranging from 2,000 to 5,000. Simulation results show that the LTS-EDA is better than the advanced algorithms proposed recently.

Keywords Tabu search · Estimation of distribution · Linkage learning · Guided mutation · Maximum diversity problem

1 Introduction

Given a set of n elements and a diversity measure or pairwise difference d_{ij} between elements i and j ($d_{ij} = d_{ji}$), with $d_{ij} > 0$ for $i \neq j$ and $d_{ij} = 0$ otherwise, the maximum diversity problem (MDP) consists in selecting a subset of given cardinality m from the n elements, such that the sum of the pairwise differences between the elements of the selected subset is maximized. Let $x_i = 1$ if element i belongs to the subset, $x_i = 0$ otherwise. The MDP can be formulated as follows Kuo et al. (1993):

$$f(x) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_i x_j, \tag{1}$$

subject to

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

This model has lots of different applications, mainly including location, ecological systems, immigration and admissions policies, committee formation, curriculum design, market planning and portfolio selection, medical treatment, genetics, ethnicity, product design fields, and so on (Katayama 2004; Martí et al. 2011). Duarte (2007) proposed an interesting application, where the MDP is solved to help scatter search (SS) obtain a good solution set with a balance between quality and diversity. More details on the applications can be found in Katayama (2004) and Martí et al. (2011).

In the general case, the diversity measure or pairwise difference d_{ij} can be unrestricted in sign. Further, the constraint defined by Eq. (2) can be extended or generalized as Palubecki (2007)

$$m_1 \leq \sum_{i=1}^{n} x_i \leq m_2, \tag{3}$$

where two integers m_1, m_2, $0 \leq m_1 \leq m_2 \leq n$. Hence, in the generalized MDP (gMDP) defined by Eqs. (1) and (3),

J. Wang (✉) · Y. Zhou · Y. Cai · J. Yin
Department of Computer Science, Sun Yat-sen University,
Guangzhou Higher Education Mega Center,
Guangzhou 510006, China
e-mail: wjiahai@hotmail.com

the coefficients d_{ij} can be positive, negative or zero. When all the coefficients d_{ij} are nonnegative, then Eq. (3) can be replaced by Eq. (2) where, obviously, $m = m_1 = m_2$.

The MDP is strong NP-hard (Kuo et al. 1993). Exact algorithm (Martí et al. 2010) proposed recently is able to solve only instances of size less than 100 variables in reasonable computation time. Metaheuristic algorithms have shown to be very successful in solving hard combinatorial optimization problems. Thus, recently lots of metaheuristic approaches, such as tabu search (TS) (Duarte 2007; Palubecki 2007; Aringhieri et al. 2008), variable neighborhood search (VNS) (Brimberg et al. 2009; Aringhieri 2011), greedy randomized adaptive search procedure (GRASP) (Silva et al. 2007; Andrade et al. 2005), Hopfield neural network (HNN) (Wang et al. 2009), SS (Gallego et al. 2009), iterated greedy (IG) algorithm (Lozano et al. 2011) and memetic algorithm (MA) (Katayama 2004), have been successfully applied to the MDP. A well-documented review and comparison of current approaches for the MDP can be found in Martí et al. (2011), Aringhieri (2011), Wang et al. (2009) and Lozano et al. (2011). In Sect. 2, we also briefly review the current advanced approaches for the MDP.

Especially, many TS-based approaches (Duarte 2007; Palubecki 2007; Aringhieri et al. 2008), have achieved superior performance; however, most of these approaches have no explicit learning mechanism to utilize global statistical information, which motivates us to propose a learnable TS. In this paper, a learnable TS guided by estimation of distribution algorithm (EDA), called LTS-EDA, is proposed for the MDP. The LTS-EDA consists of two parts: a clustered EDA and a simple TS. In the clustered EDA, different clusters focus on different substructures and the combination of information from different clusters can effectively combine substructures. Thus, the clustered EDA, as a linkage learnable constructive method, can generate a new promising starting solution to guide the TS. The simple TS attempts to improve the solution generated by the clustered EDA and thus provide more information to update the EDA model. The LTS-EDA can also be seen as a hybrid EDA and one of the efforts to make simpler EDAs more effective for hard optimization problems. Experimental results on benchmark problems show that the LTS-EDA is better than the existing state-of-the-art algorithms for the MDP.

The contributions of this paper are as follows: (1) a novel TS-based algorithm is proposed for the MDP and superior results to most recent advanced algorithms are obtained; (2) a novel learnable TS framework guided by EDA is proposed, which is the contribution to the TS field. The learnable TS framework can be easily instantiated or tailored for solving other 0-1 combinatorial optimization problems. Further, the LTS-EDA can also be seen as one of the efforts to make simpler EDAs more effective for hard optimization problems; and (3) a comprehensive experimental comparison of the proposed LTS-EDA with advanced algorithms proposed most recently is also provided.

The remaining sections of this paper are organized as follows: in the next section, we briefly review some related work, including the current most advanced approaches for the MDP, the current advanced TS algorithms, and the general idea and some issues of EDA. These reviews motivate the design of the proposed LTS-EDA algorithm which is presented in Sect. 3 in detail. In Sect. 4, the experimental benchmark is introduced and simulation results are presented. Finally, in Sect. 5, conclusions and future works are given.

2 Related work

There are three parts in this section. Section 2.1 reviews recent related works for solving the MDP. Current studies of the TS are discussed in Sect. 2.2 so as to motivate the design of learnable TS. In addition, the general idea and current development of the EDA are also briefly reviewed in Sect. 2.3.

2.1 Recent advanced approaches for MDP

Several advanced population-based methods are proposed for the MDP. Katayama and Narihisa (2004) presented a MA. It basically consists of a uniform crossover operator to produce offspring solutions and a k-flip local search to improve the offspring solutions. Gallego et al. (2009) proposed a SS algorithm. The main components of the SS are all based on TS mechanism.

Most of the approaches for the MDP, for example, multistart TS (Duarte 2007; Palubecki 2007; Aringhieri et al. 2008), VNS (Brimberg et al. 2009; Aringhieri 2011), GRASP (Silva et al. 2007; Andrade et al. 2005), and IG (Lozano et al. 2011) belong to a kind of multistart method (Martí et al. 2011). A multistart method generally consists of two main components: a constructive phase to generate a new starting solution and an improvement phase to improve this solution by a local search (Martí et al. 2011). Duarte and Martí (2007) proposed a multistart TS in which memory structures from the TS methodology are incorporated into both construction and improvement phases. Palubeckis (2007) proposed an iterated TS (ITS), which alternates TS with perturbation procedures. The perturbation phase basically consists of selecting (unselecting) a random number of unselected (selected) elements. The procedure TS invoked contains only the main ingredient of TS, namely, a short term memory tabu list without aspiration criterion. If the TS finds a better solution than the previous best solution, a local search is applied to the new solution. Recently, Brimberg et al. (2009) proposed a new VNS, where shaking phase repeats the random vertex swap move several times, and local search phase also adopts swap neighborhood. Simulation results show that this VNS is better than the ITS (Palubecki 2007) and SS (Gallego et al. 2009). Aringhieri and Cordone (2011) proposed several metaheuristics based on the TS and VNS methodologies. Several GRASPs (Silva et al. 2007; Andrade et al. 2005) are proposed. GRASPs often mainly focus on the randomized generation of high-quality starting solutions by very refined construction phases and sophisticated management of the solutions, while the subsequent solution improvement phase is usually performed by a simple local search. To explore an opposite method with respect to the GRASPs, Aringhieri et al. (2008) proposed a hybrid of GRASP and TS algorithm (GRASP-TS), where TS is adopted as improvement method. Wang et al. (2009) proposed a discrete competitive HNN combined with EDA (DCHNN-EDA) for the MDP. Most recently, Lozano et al. (2011) proposed an fine-tuning iterated greedy (TIG) algorithm for the MDP.

This is by no means an exhaustive list of existing approaches for the MDP. More detailed reviews and comparisons of these algorithms can be found in Martí et al. (2011), Aringhieri (2011), Wang et al. (2009) and Lozano et al. (2011). Especially, Martí et al. (2011) presents extensive computational experiments to compare 30 methods, 10 heuristics (including eight construction and two improvement methods), and 20 metaheuristics for the MDP. The experiment comparisons are divided into three groups: in the first one 10 simple heuristics are considered, in the second one 17 metaheuristic methods and in the third one three methods based on populations are considered. The paper finally concludes that the VNS (Brimberg et al. 2009) emerges as the best overall except for the five largest problems where the ITS (Palubecki 2007) is the leader. Note that the comparison conducted in (Martí et al. 2011) does not include the TIG (Lozano et al. 2011) proposed recently. Further, most of approaches can only deal with the MDP defined by (1), (2). So far, only the ITS (Palubecki 2007) can deal with the gMDP defined by (1) and (3).

2.2 Multistart TS and hybrid TS methods

Simple TS mechanism is not enough to prevent the search process from becoming trapped in certain regions of the search space, and thus some diversification strategy is necessary. Misevicius et al. (2006) proposed a multistart TS framework, called iterated TS (ITS) in their paper. The novelty of the ITS is the incorporation of the special kind of solution reconstruction into the classical TS. It is distinguished, in principle, for two main components: reconstruction (or perturbation) of the solutions for diversification and classical TS for intensification. Diversification implemented by reconstruction (perturbation) of the solutions is responsible for escaping from the current local optimum and moving towards new regions in the solution space. By repeating these two main components many times, the ITS tries to seek for near-optimal solutions.

Many different perturbation variants or diversification strategies are proposed to develop different kinds of the multistart TS for optimization problems. Palubeckis (2004) proposed five multistart TSs, called MTS1,…, MTS5, for unconstrained binary quadratic programming problem (UBQP). The MTS1 continues the TS simply by randomly generating a 0–1 vector as a new starting point. The MTS2 applies a constructive heuristic to certain instances of the UBQP obtained by fixing the values of a subset of the variables. The MTS3 invokes a greedy randomized procedure, like the GRASP, to generate 0–1 vector as a new starting point. The MTS4 maintains a set of elite solutions to generate good initial solutions for the TS. The MTS5 applies a problem perturbation technique. Simulation results show that the MTS2 is the best algorithm. Later, Palubeckis (2006) proposed an improved ITS for the UBQP. In this ITS, a solution perturbation mechanism is adopted to enforce search diversification, and the TS procedure invoked contains only the main ingredient of TS, namely, a short term memory tabu list without aspiration criterion. Simulation results show that the improved ITS is better than the MTS2. Palubeckis (2007) also applied this ITS to the MDP and obtained superior results. Glover et al. (2010) proposed an ITS for the UBQP, which introduces a diversification strategy which relies on a memory-based perturbation operator composed of three parts: a flip frequency memory, an elite solution memory, and an elite value frequency memory. These memory structures are used jointly to identify some critical variables. Finally for the perturbation step, the values of the selected critical variables are flipped to perturb the solution. Most recently, James et al. (2009) proposed five multistart TSs for quadratic assignment problem. These MTS variants can be separated into two categories. The first category implements a continuous diversification TS process and contains variants that perturb the search by modifying some algorithm parameters but continue the exploration from the same working solution. The second category implements a discontinuous diversification TS process, which strategically replaces the current working solution with a different solution as a new starting point.

Population-based TS algorithms, in which the TS is combined with EAs, are also proposed (Gallego et al. 2009; Hao 2011; Lv and Hao 2010; Lv 2010). These hybrid TS algorithms are commonly called MA. In these algorithms, the EA is used for global search and the TS is used for local search.

Most of these TS-based methods have no explicit learning mechanism to utilize global statistical information in the search, which motivates us to propose a learnable TS in the Sect. 3.

2.3 EDA

EDAs, also called evolutionary algorithms (EA) based on probabilistic models (EAPMs), have no traditional crossover or mutation, and have been seen as a new computing paradigm in EA field. Recent years have seen a growing interest in the research and application of EDAs. In 2009, IEEE TEC published a special issue on EDAs (Lozano et al. 2009; Emmendorfer 2009; Platel et al. 2009; Hauschild et al. 2009). EDAs explicitly extract global statistical information from their previous search and build a probability distribution model of promising solutions, based on the extracted information. New solutions are then sampled from the built model. Thus, the general EDAs include three steps: solution selection, model building based on selected solutions, and sampling to generate new solutions. EDAs have several merits (Lozano et al. 2009): first, it provides a very natural and systematic way for bridging machine learning and EA, and thus can be seen as a data mining-based metaheuristic algorithm; second, it provides a statistically sound approach for dealing with variable linkages by constructing probabilistic models to capture interactions among variables; and finally, it expresses the knowledge learnt about the search space as a probabilistic model and thus makes the search more effective.

To solve a wide spectrum of optimization problems using EAs, effective building block (BB) mixing is essential. Linkage is the relationship between variables tightly linked to form a BB. Linkage learning attempts to identify the tightly linked genes and bind them together to form BBs (Emmendorfer 2009). From the viewpoint of the capability to capture interactions or linkage among variables, the EDAs can be classified into three classes: no interactions, pairwise interactions and multivariate interactions (Emmendorfer 2009). The first class of EDAs includes well-known population-based incremental learning (PBIL) (Baluja 1994), compact genetic algorithm (cGA) (Harik et al. 1999), and univariate marginal distribution algorithm (UMDA) (Mühlenbein 1996). The second class of EDAs includes mutual information maximization for input clustering (MIMIC) algorithm (de Bonet et al. 1994), combining optimizers with mutual information trees algorithm (COMIT) (Baluja 1997, 1998) and bivariate marginal distribution algorithm (BMDA) (Pelikan 1999). The third class of EDAs includes factorized distribution algorithm (FDA) (Mühlenbein et al. 1999), extended compact genetic algorithm (EcGA) (Hari 1999), Bayesian optimization algorithm (BOA) (Pelikan et al. 2000; Peña et al. 2005) and hierarchical BOA (hBOA) (Hauschild et al. 2009). In general, the EDAs that adopt complex models usually provide excellent performance; however, are computationally expensive, which may hinder these pure EDAs from solving a hard problem (Emmendorfer 2009; Zhang et al. 2005). Complex models themselves may induce spurious variable relationships since the learning of the relationship is based on the biased information of an incomplete sample of the space. If such spurious relationships become an obstacle which prevents the EDAs from solving problems, EDAs with simpler models, i.e., more suitable for the problem structure, should be used to obtain better performance. Several efforts have been performed recently in order to make simpler EDAs more effective. For example, the PBIL with location information of solution (Wang et al. 2009; Zhang et al. 2005, 2007; Zhang 2006; Guturu 2008), called EA with guided mutation (EA/G) (Zhang et al. 2005), and the PBIL with clustering (Emmendorfer 2009), obtain better results than the MIMIC and BOA, respectively. Thus, simple model should not be discredited a priori since the benefit of searching complex variable interactions could, under particular circumstances, be still unclear (Emmendorfer 2009; Platel et al. 2009). In addition, Martí et al. (2009) also pointed out that most modeling building schemes adopted so far by EDAs use off-the-shelf machine learning methods, however, the model-building problem has particular requirements that those machine learning methods do not meet and even have conflicts with.

A single EDA technique is hard for solving complicated optimization problems because the location information of solutions found so far (the actual positions of these solutions in the search space) is not directly used for the generation of offspring in the original EDAs (Zhang et al. 2005). Therefore, how to combine EDAs with other techniques represents an important research direction (Zhang et al. 2007). Recently, some of hybrids of EDAs and EAs, for example, EDA combined with GA (Zhang et al. 2005), EDA combined with PSO (Wang et al. 2009), are proposed for optimization problems.

Recently, Santana et al. (2009) identified several research topics or challengers in discrete EDAs, and pointed out that an open research area is to determine to what extend can hybrid EDAs that use simple probabilistic models improve their performance for hard optimization problems with strong interactions. The LTS-EDA proposed in this paper belongs to this research topic or area, which can be seen as a hybrid EDA to make simpler EDAs more effective for combinatorial optimization problems.

3 LTS-EDA for MDP

This section first describes the framework of the LTS-EDA, then gives the detailed implementation of the LTS-EDA, including a simple TS and a clustered EDA procedure.

3.1 Main framework of LTS-EDA

LTS-EDA introduces knowledge model to TS, and it extracts knowledge during the search process, by means of the evaluation of each point generated. LTS-EDA is made of two main components: the knowledge model and TS search method. The knowledge model, implemented by a clustered EDA (Emmendorfer 2009) in this paper, is the information repository in which the experience of history solutions is stored and knowledge is extracted to guide following TS search. The framework of the LTS-EDA is depicted in Fig. 1. Both components are linked through a communication way, which states the rules about the TS that can contribute to (or update) the knowledge model with its search experiences, and the way the knowledge model can guide following TS search. Thus, the LTS-EDA adopts dual or cooperative evolution/search structure, consisting of probabilistic model space in the clustered EDA and solution space searched by TS.

3.2 Proposed simple TS for MDP

3.2.1 Neighborhood, tabu list and aspiration criteria

A simple TS proposed here is based on a 1-flip move neighborhood. The 1-flip move means to change the value of variable x_i to $1 - x_i$; thus the size of the neighborhood is n. The gain value Δ_i of flipping a single ith variable in a current solution x can be computed by the difference between the objective function values as follows:

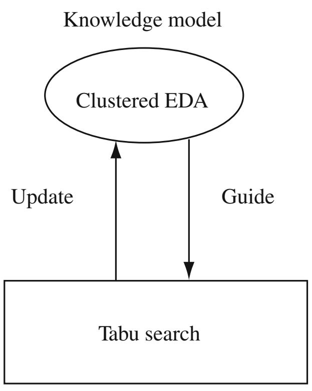

Fig. 1 Framework of the LTS-EDA

$$\Delta_i = f(x') - f(x) = (1 - 2x_i)\sum_{j \neq i} d_{ij}x_j, \tag{4}$$

where x' is the neighboring solution obtained after flipping ith variable of the current solution. This gain can be calculated in $O(n)$. However, the calculation of all gains for the n flipping candidates takes $O(n^2)$ time by Eq. (4). In order to perform efficient search for large size problems, it is crucial to calculate the gains in more efficient way. Using the information of the gains that have already been computed, all of the new gains can be calculated incrementally, instead of recalculating them by Eq. (4). First, Eq. (4) is used to calculate the initial gains Δ_i for all variables. Once jth variable is flipped, the new gain $\Delta_i (i = 1, \ldots, n)$ can be calculated incrementally by

$$\Delta_i = \begin{cases} -\Delta_i & \text{if } i = j \\ \Delta_i - d_{ij}(1 - 2x_i)(1 - 2x_j) & \text{if}(i \neq j) \textit{ and } (d_{ij} \neq 0). \end{cases} \tag{5}$$

Thus, the updating of the gains for the n candidates of the 1-flip neighboring solutions can be performed in linear time $O(n)$. Furthermore, only Δ_i of the variable x_i with $d_{ij} \neq 0$ have to be updated.

Typically, TS manages a tabu list as a memory structure to assure that solutions visited within a certain span of iterations, called tabu tenure, will not be revisited. In our simple TS, each time a variable x_i is flipped, a tabu tenure, T, is assigned to prevent x_i from being flipped again for the next T iterations.

Our TS procedure then restricts consideration to variables not currently tabu, and flips a variable which produces the best (largest) gain value. This strategy is called best improvement strategy. In the case that two or more moves have the same best move value, ties are broken randomly.

A simple aspiration criterion is applied that permits a move to be selected in spite of being tabu if it leads to a solution better than the current best solution.

3.2.2 TS procedure and its characteristics

The proposed complete TS procedure can be described in Algorithm 1. Lines 4–11 are designed for the MDP defined by (1), (2). It eliminates an element which has the lowest contribution to the objective function in the selected subset, and, at the same time, picks an element which will lead the greatest contribution to the object function in the unselected subset. Lines 12–15 are specially designed for the gMDP defined by (1) and (3). It adopts 1-flipping neighborhood operator and thus can easily deal with the constraint of the gMDP. The main characteristics of the proposed TS are that it is very simple, only need to keep one kind of tabu list and can easily deal with the gMDP. Most of TS-based algorithms (Duarte 2007; Aringhieri et al. 2008) and other local search-based algorithms, for example, VNS (Aringhieri 2011; Brimberg et al. 2009) and GRASP (Silva et al. 2007; Andrade et al. 2005), adopt only swap neighborhood move, and thus cannot deal with constraint of the gMDP. The ITS (Palubecki 2007) can also deal with the gMDP because it adopts two kinds of neighborhood operator, 1-fliping and swap. However, two kinds of tabu list have to be kept to control and manage two kinds of neighborhood move. Thus, the ITS is not as simple as our proposed TS.

Algorithm 1 Simple TS procedure for MDP

```
1: $Iter \leftarrow 0$, $curbest \leftarrow f(x)$
2: Initialize all $\Delta_i$ $(i = 1, \ldots, n)$ according to Eq. (4)
3: repeat
4:   if $\sum_{i=1}^{n} x_i \le m_1$ then
5:     $j \leftarrow argmax_k\{\Delta_k | x_k = 0$ and $x_k$ is non-tabu, $k = 1, \ldots, n\}$
6:     $f(x) \leftarrow f(x) + \Delta_j$, $x_j \leftarrow 1$, make $x_j$ tabu for next $T$ times
7:     update all $\Delta_i$ according to Eq. (5)
8:   if $\sum_{i=1}^{n} x_i \ge m_2$ then
9:     $j \leftarrow argmax_k\{\Delta_k | x_k = 1$ and $x_k$ is non-tabu or satisfying aspiration criterion, $k = 1, \ldots, n\}$
10:    $f(x) \leftarrow f(x) + \Delta_j$, $x_j \leftarrow 0$, make $x_j$ tabu for next $T$ times
11:    update all $\Delta_i$ according to Eq. (5)
12:  if $m_1 < \sum_{i=1}^{n} x_i < m_2$ then
13:    $j \leftarrow argmax_k\{\Delta_k | x_k$ is non-tabu, $k = 1, \ldots, n\}$
14:    $f(x) \leftarrow f(x) + \Delta_j$, $x_j \leftarrow 1 - x_j$, make $x_j$ tabu for next $T$ times
15:    update all $\Delta_i$ according to Eq. (5)
16:  if $f(x)$ is better than $curbest$ then
17:    $curbest \leftarrow f(x)$
18:  $Iter \leftarrow Iter + 1$
19: until $Iter > MaxIter$
20: return $curbest$
```

In line 2, the calculation of all gains for the n flipping candidates by Eq. (4) takes $O(n^2)$ time. The main loop described by lines 4–15 takes $O(MaxIter \cdot n)$ time.

3.3 Clustered EDA

The clustered EDA combines a simple low-order EDA and a clustering technique to avoid the high computational cost required by the higher order EDAs such as the BOA (Peña et al. 2005) and can obtain better results (Emmendorfer 2009). This strategy adopts clustering to group similar solutions by similarity of their genotype in a given population of solutions, relying that different clusters focus on different substructures or BBs, and thus the combination of information from different clusters effectively combines substructures or BBs (Emmendorfer 2009). Each cluster forms a probabilistic model and thus multiple interacting models are maintained. The combination mechanism uses an information gain measure when deciding which cluster (or probabilistic model) is more informative for any given gene position, during a pairwise cluster combination (Emmendorfer 2009). The clustered EDA consists of several main procedures: learning clustering concepts; probability modeling of the subpopulation of each cluster; and a probability model combination, called concept-guided combination (cg-combination) operator, for information combination or exchange among different models. Further, the guided mutation (Zhang et al. 2005) is also introduced to generate a new starting solution based on a selected cluster. These procedures are briefly described as follows (Emmendorfer 2009; Zhang et al. 2005):

Clustering given a population of solutions, a simple and fast partitioning clustering algorithm, k-means, is used to group similar solutions by similarity of their genotype.

Probability modeling this procedure computes a probability vector for each cluster extracted by the k-means procedure. If only binary variables are considered then each centroid of the cluster can be directly interpreted as a probability vector of binomial proportions, where each position j of a centroid i corresponds to the proportion of 1s in the corresponding position j for the subpopulation of the cluster i. As in the UMDA and PBIL, the probability vector for cluster i can be computed as follows:

$$\pi_{ij} = \frac{\sum_{c(x)=i} x_i}{n_i}, \tag{6}$$

where $\sum_{c(x)=i} x_i$ is the number of individuals in cluster i possessing the value 1 at gene j and n_i is the number of individuals in cluster i. When all individuals in a cluster have the same value (all 0s or 1s) in a certain locus, this estimator saturates at one of the extremes (0 or 100%). Sampling from those extreme proportions leads to the loss of the chance of the alternative value, 1 or 0, to be generated. Thus, a Wilson estimator, as a mutation operator with probability p_w, is introduced, which changes slightly the binomial proportions estimated, and hence allows for an allele to be generated even if all individuals possess the complementary allele. The Wilson estimator incorporates a degree of uncertainty by estimating binomial proportions as (Emmendorfer 2009)

$$\pi_{ij} = \frac{\sum_{c(x)=i} x_i + 2}{n_i + 4}. \tag{7}$$

The concepts that characterize each cluster are relevant information. The combination of important parts of promising solutions found so far can guide EDAs during the exploration of the search space (Emmendorfer 2009). Identification of substructures is performed by clustering, and combination among substructures can be achieved by model combination described as follows:

Model combination the cg-combination operator combines information from different clusters by building a temporary probability vector π_T from two randomly selected parent probability vectors, π_A and π_B. In the cg-combination, a measure from information theory guides the choice of the best parent for each gene. The cg-combination aims for a careful combination of relevant information from two parent probability vectors, attempting to maintain the most informative part of each parent. Let

$$h_{\cdot j} = -\sum_{q\in\{0,1\}} p_{jq}\log_2(p_{jq}) \tag{8}$$

be the entropy of the distribution of gene j, where $p_{jq} = P(x_j = q)$ is the proportion of individuals possessing the value q for gene j in the whole population. Similarly, let

$$h'_{i,j} = -\sum_{q\in\{0,1\}} p'_{i,jq}\log_2(p'_{i,jq}) \tag{9}$$

be the entropy of the distribution of the same gene j without taking clustering i into account the estimated proportions. The proportion $p'_{i,jq}$ is computed as $P(x_j = q|c(x) \in \{1,2,\ldots,k\} - \{i\})$, where $c(x)$ is the cluster associated with an individual x.

The measure of how informative is a cluster i to a gene j is thus the difference in the entropy of the distribution of the gene j before and after observing cluster i

$$w_{i,j} = h_{\cdot j} - h'_{i,j}. \tag{10}$$

Each position $\pi_{T,j}$ of the temporary probability vector is thus defined as

$$\pi_{T,j} = \begin{cases} \pi_{A,j} & \text{if } w_{A,j} > w_{B,j} \\ \pi_{B,j} & \text{otherwise.} \end{cases} \tag{11}$$

Then a new individual is generated by sampling from each position of π_T independently.

The cg-combination relies on clustering to group together individuals possessing the same substructure and attempts to extract the information about that substructure from $\Pi = (\pi_{ij})$ and $W = (w_{ij})$ (Emmendorfer 2009). A given cluster is expected to be informative for all genes that define a BB if most of the individuals in that cluster possess that BB. Since the cg-combination can combine information from different clusters, it potentially allows a low-order clustered EDA to effectively perform linkage learning (Emmendorfer 2009).

When smaller BBs start blending, new BBs emerge. This causes the loss of older clustering hypothesis, consequently inhibiting those initial BBs in recent individuals (Emmendorfer 2009). A potential solution is the maintenance of some old BBs (hence, old clustering hypothesis) which can be used to generate new solutions. This mechanism works as follows (Emmendorfer 2009): matrices Π and W corresponding to an old clustering hypothesis are maintained as $\{\Pi_{\text{old}}, W_{\text{old}}\}$. Before each combination of probability vectors, one of the two hypotheses is randomly selected. The parameter p_{old} is the probability of the old hypothesis to be selected. The number of individuals generated and accepted represents the performance of each hypothesis. These performance records are g_{old} and g_{new} for the old and current hypotheses, respectively. Whenever the new individual is selected, the corresponding performance record is increased. These records are used to control the replacement of the old hypothesis. If g_{new} overpasses g_{old}, then $\{\Pi_{\text{old}}, W_{\text{old}}\} \leftarrow \{\Pi, W\}, g_{\text{old}} \leftarrow g_{\text{new}}$ and $g_{\text{new}} \leftarrow 0$.

More details about model combination can be found in (Emmendorfer 2009).

Guided mutation this operator mutates the best solution in the current population based on the probability vector π_A of a randomly selected cluster A to generate a new starting solution. This operator is described in Algorithm 2, where rand() produces a random number distributed uniformly on [0,1], and the parameter β controls the strength of the perturbation or mutation.

Algorithm 2 Guided mutation based on selected cluster

```
1: for $i = 1$ to $n$ do
2:   if rand( )< $\beta$ then
3:     gene $i$ of the new solution is sampled from the probability vector $\pi_A$
4:   else
5:     gene $i$ of the new solution is copied from the best solution in the current population
6: end for
```

This operator combines the statistical information in a selected cluster and location information of the best solution in the current population and thus is expected to generate a promising new solution (Zhang et al. 2005).

3.4 Complete LTS-EDA for MDP

In the LTS-EDA, the clustered EDA is used to guide the TS. That is, the clustered EDA is used to find more promising starting point for the TS. The LTS-EDA is given in Algorithm 3, where N is the population size, k is the number of clusters, p_c is the probability of interbreeding using cg-combination operator, p_{old} is the probability of an old clustering hypothesis to be used during breeding instead of the current one and p_w is the probability of the Wilson estimator defined by Eq. (7) to be used instead of the sample mean defined by Eq. (6).

Algorithm 3 LTS-EDA for MDP

```
1: Set parameters $N$, $k$, $p_w$, $p_c$, $p_{old}$, $\beta$
2: $POP \leftarrow$ initialize-population ($N$)
3: $CLUSTER \leftarrow$ $k$-means ($POP$, $k$) /*Clustering*/
4: $\Pi_{new} \leftarrow$ compute-probability-vectors ($CLUSTER$, $p_w$) by Eq.(6) or (7)
5: $W_{new} \leftarrow$ compute-information-gain-measures ($CLUSTER$) by Eq.(10)
6: $\Pi_{old} \leftarrow \Pi_{new}$, $W_{old} \leftarrow W_{new}$
7: find the best ($pop_{best}$) and the worst ($pop_{worst}$) individuals of $POP$
8: $gb \leftarrow pop_{best}$ /*Initialize the best-so-far solution*/
9: $count \leftarrow 0$
10: repeat
11:   if rand()$< p_{old}$ then
12:     $\Pi \leftarrow \Pi_{old}$, $W \leftarrow W_{old}$
13:   else
14:     $\Pi \leftarrow \Pi_{new}$, $W \leftarrow W_{new}$
15:   update $\{\Pi_{old}, W_{old}\}$
16:   if rand()$< p_c$ then
17:     $pop_{new} \leftarrow$ cg-combination($\Pi$, $W$)
18:     $pop'_{new} \leftarrow$ TS($pop_{new}$) /*Algorithm 1*/
19:     if $pop'_{new}$ is better than $pop_{worst}$ then /*Replace if better*/
20:       individual $pop_{worst}$ is replaced by $pop'_{new}$
21:   else
22:     $pop_{new} \leftarrow$ guided-mutation($\Pi$, $pop_{best}$) /*Algorithm 2*/
23:     $pop'_{new} \leftarrow$ TS($pop_{new}$) /*Algorithm 1*/
24:     individual $pop_{best}$ is replaced by $pop'_{new}$ /*Random acceptance*/
25:   find the best ($pop_{best}$) and the worst ($pop_{worst}$) individuals of $POP$
26:   if $pop_{best}$ is better than $gb$ then
27:     $gb \leftarrow pop_{best}$, $count \leftarrow 0$
28:   else
29:     $count \leftarrow count + 1$
30:   $CLUSTER \leftarrow$ update-clusters ($POP$)
31:   $\Pi_{new} \leftarrow$ compute-probability-vectors ($CLUSTER$, $p_w$)
32:   $W_{new} \leftarrow$ compute-information-gain-measures ($CLUSTER$)
33:   if (only one cluster left) or ($count \leq 50$) then
34:     $CLUSTER \leftarrow$ $k$-means ($POP$,$k$)
35:     $count \leftarrow 0$
36: until termination conditions met
37: return $gb$
```

At the beginning of the algorithm, line 2 randomly generates a population or pool of solutions. The 1-fipping repair process is performed to obtain a feasible solution from a given infeasible one by flipping some bits as in Algorithm 1. Then lines 3–5 learn k clusters from population using k-means method and compute Π and W matrices for generating new solution.

At each time, the LTS-EDA generates a new starting point for TS improvement procedure, by randomly choosing one from two possible procedures. The first procedure is to generate a new starting point according to the cg-combination operator (lines 16–20). The resulting new solution after the TS procedure replaces the worst individual in the population (if the new solution is better than the one it is replacing). The clustered EDA extracts patterns of bits, BBs or substructures from a pool of suboptimal solutions for the MDP. These patterns, BBs or substructures present characteristics of near-optimal solutions and can be used to guide the following TS iterations in the search through the combinatorial solution space. The second procedure is to generate a new starting point using the guided mutation (lines 22–24). In this procedure, a probability vector or a cluster is randomly chosen, then a new staring point is sampled according to Algorithm 2. After the new starting point is improved by the TS procedure, the best solution of the current population is directly replaced by the resulting solution. Thus, a solution that is worse than the best solution of the current population has chance to enter the current population to provide useful diversity. This strategy is similar to the random walk acceptance criterion in the TIG (Lozano et al. 2011).

The clustered EDA follows an incremental architecture since a single solution is generated at each iteration. The clusters are continuously updated. Subsequently, clustering hypothesis and probabilistic models for each cluster are simply updated (lines 30–32) through a single typical k-means step, and not fully relearned (Emmendorfer 2009). This way, the LTS-EDA may form stable clusters and employ accumulated search experience from previous TS to enhance future ones. Lines 33–35 invoke relearning procedure if all the solutions belong to the same one cluster, or the best-so-far gb cannot be improved within 50 invocation of TS procedures. Relearning means to re-initialize the centroids of the clusters and re-cluster the population.

In summary, the LTS-EDA uses data mining techniques including clustering algorithm and EDA to extract useful information from the search history of the TS in order to guide the TS to interesting or promising search space areas. The search history of the TS is stored in the population of solutions and thus the population of solutions can be seen as a knowledge base or a long-term memory.

The complexity of the LTS-EDA mainly consists in the k-means and TS procedure. The k-means algorithm generally takes $O(t \cdot k \cdot N \cdot n)$ time, where t denotes the number of iterations needed by the algorithm to reach stable clusters. In the LTS-EDA, k-means reaches stable clusters within five iterations. For large size problem, $n \gg N$ and $n \gg k$. Further, the k-means is not often invoked and clusters are updated in an incremental way. Hence, the complexity of the LTS-EDA is mainly dominated by the TS procedure whose complexity is $O(n^2)$. In our implementation, the maximum computation time and the convergence of the algorithm (all solutions in the population are identical) are adopted as the stopping criterion.

3.5 Characteristics and advantages of LTS-EDA

The characteristics and advantages of the LTS-EDA can be identified and summarized as follows:

1. In the LTS-EDA, multiple interacting probability vectors or models, as a long-term memory mechanism, cooperate with the short term memory of the tabu mechanism of TS. This kind of cooperation is expected to provide good performance. Thus, EDA, instead of the commonly used recency or frequency-based long-term memory mechanism in traditional TS, is combined with traditional TS to develop a novel learnable TS. Since the EDA component adopted in the

LTS-EDA is general, the learnable TS, LTS-EDA, can be easily instantiated or tailored for solving other 0–1 combinatorial optimization problems as traditional TS. Hence, the LTS-EDA can be seen as a form of new TS-based metaheuristic algorithm for optimization. In the view of the multistart TS framework, clustered EDA, as a learnable constructive method, is used to create a new starting solution, and a simple TS, as an improvement method, attempts to improve the solution created by the clustered EDA. The LTS-EDA is different from the common TS-based MA (Lv and Hao 2010; Lv et al. 2010). In the former, only one solution is generated at each time by the clustered EDA, and this solution is adopted as a new starting point for the TS. In the latter, a population of solutions are generated at each time. Thus, the LTS-EDA works in a steady-state way.

The LTS-EDA can be seen as a hybrid metaheuristic. Detailed review and taxonomy about hybrid metaheuristics can be referred to (Lozano 2010; Xin et al. 2011). In essence, the hybridization of metaheuristics is a means of adjusting the trade-off between exploitation (intensification) and exploration (diversification) (Lozano 2010; Xin et al. 2011).

2. In the LTS-EDA, clustered EDA is implemented by multiple probability vectors which are built by clustering the pool of solutions. Most of the previous EDAs or hybrid EDAs, for example, GA-EDA (Zhang et al. 2005), DPSO-EDA (Wang et al. 2009), and DCHNN-EDA (Wang et al. 2009), maintain only a singe probability model. The use of only one probabilistic model to represent the whole population will inevitably reduce diversity, thereby degrading the global search abilities of the algorithm. Previous work (Platel et al. 2009) shows that the multimodel EDA can manipulate more complex distribution of solutions than with a single-model approach and thus lead to more efficient optimization of problems with interacting variables. Most importantly, the multiple interacting models guided by the clustering algorithm and cg-combination can further achieve success on the linkage learning task (Emmendorfer 2009). In addition, clustering population effectively extracts relevant local information resulting from similarities in the population of solutions (Emmendorfer 2009).
3. Compared with the original work in Emmendorfer (2009), the contribution of this work consists in applying a new multimodel EDA method with linkage learning to guide TS to more promising search space areas. Inversely, the tabu mechanism in the TS procedure, which memorizes recently visited points and prohibits exploring their neighborhood, prevents the clustered EDA from repeatedly sampling or oversampling in a small region and directs it to explore potentially optimal regions. Furthermore, the guided mutation is also introduced to generate a new solution based on a selected cluster. In addition, the LTS-EDA is used to solve the MDP, a strong NP-hard problem with lots of applications, instead of toy problems tested in Emmendorfer (2009). Experimental comparison of the LTS-EDA with other metaheuristic algorithms is also given.

4 Simulation results and discussion

In order to assess the performance of the LTS-EDA, simulations were implemented in C on a PC (Pentium(R) 2.70 GHz with 2.0 G of RAM). In this section, we first describe the benchmark datasets, the parameter settings and the effectiveness of the clustered EDA and then compare the LTS-EDA with the state-of-the-art algorithms.

4.1 Benchmark datasets

For the MDP, recent works in Martí et al. (2011) and Wang et al. (2009) were tested on 315 instances and 120 instances, respectively. However, part of these test instances are small-scale instances ($n < 1000$ with varying densities). These small-scale instances are not considered in our experiments, since they are solved easily by the advanced heuristics proposed recently in very short time, and thus no good comparisons among advanced algorithms can be provided. Here, the LTS-EDA is tested on 50 large size benchmark problems with the size ranging from 2,000 to 5,000 selected from literature. The details of the instance sets are described as follows (Duarte 2007; Palubecki 2007; Wang et al. 2009):

1. Random Type 1 instances (Type1_22): matrices with real numbers generated from a (0, 10) uniform distribution. 20 instances with $n = 2{,}000$ and $m = 200$ are considered. These instances are first introduced in Duarte (2007) and can be downloaded from http://www.uv.es/~rmarti/paper/mdp.html.
2. Beasely instances (b2500): Instances taken from the OR-Library (Beasle 2000). These test instances were originally introduced for the UBQP. Recently, these instances were adopted for testing algorithms for the MDP by Palubeckis (2007). In the MDP case, the diagonal elements of these instances are ignored. All the instances have 10% density. In each case, the matrix contains both positive and negative number from $[-100,\ 100]$. There are ten instances with $n = 2500$, $m = 1{,}000$ for the MDP defined by (1),

(2). Further, ten instances with $m_1 = 1{,}620$ and $m_2 = 1{,}655$ are set for the gMDP defined by (1) and (3) as in Palubecki (2007).

3. Random larger instances (p3000 and p5000): matrices with integer numbers generated from a [0, 100] uniform distribution. The density of the matrix is 10, 30, 50, 80, 100%. There are five instances with $n = 3{,}000$, $m = 0.5n$, and five instances with $n = 5{,}000$, $m = 0.5n$. These larger instances are firstly introduced in Palubecki (2007). The sources of the generator and input files to replicate these problem instances can be found at http://www.soften.ktu.lt/~gintaras/max_div.html.

4.2 Parameter settings

The LTS-EDA has several parameters to be determined. First, for the parameters of the TS component, we can easily set them empirically since the TS component used in the LTS-EDA is a simple traditional TS procedure, and how to set the parameters in the traditional TS has already been extensively studied in literature. In our TS procedure, tabu tenure T and termination condition *MaxIter* have to be decided. According to the preliminary experiment, we find that $T = \sqrt{m_1}$ for the gMDP (thus $T = \sqrt{m}$ for the MDP), and $MaxIter = n$ can obtain satisfactory solutions. This setting is very simple and adaptive with configuration of the problems.

Next, six parameters in LTS-EDA have to be tuned: N, k, p_c, p_{old}, p_w and β. In this section, we adopt Taguchi method of design of experiment (DOE) used in (Wang 2011, 2012) to determine the suitable parameters. Combinations of different values of these parameters are shown in Table 1. Note that we restrict the range of β to [0.1, 0.5] by the suggestion of Wang et al. (2009) that small mutation strength is adequate for the MDP.

The orthogonal array $L_{25}(5^6)$ is chosen according to the number of parameters and the number of levels. That is, the number of treatments is 25, the number of parameters is 6 and the number of factor levels is 5. The orthogonal arrays are listed in Table 2. We only choose the Type1_22 instances to carry out the DOE test considering time limit. The LTS-EDA under each combination of parameters runs ten times for each instance in Type1_22, and 20 s for each run. "AVG" means the average result obtained by each combination for the whole Type1_22 instances.

According to the orthogonal table, we can illustrate the trends of each factor level in Fig. 2. From Fig. 2, we can figure out that N, k, p_c, p_w and p_{old} have the most significant impact on the LTS-EDA while β has a little influence. Small N and k, and large p_c, p_{old}, and p_w are preferable. Therefore, good combination of parameter values for the LTS-EDA are: $N = 50$, $k = 10$, $p_c = p_{\text{old}} = p_w = 0.7$ and $\beta = 0.5$ according to the factor level trends, and this combination is adopted during the whole experiment.

In the whole simulation, the time limit for all algorithms is set to 20, 300, 600 and 1,800 s for each instance in Type1_22, b2500, p3000 and p5000, respectively. All algorithms run 30 times on each Type1_22 and b2500 instance and 15 times on each p3000 and p5000 instance.

Table 1 Combinations of parameter values

Parameters	Factor level				
	1	2	3	4	5
N	50	100	150	200	250
k	5	10	15	20	25
p_c	0.1	0.3	0.5	0.7	0.9
p_w	0.1	0.3	0.5	0.7	0.9
p_{old}	0.1	0.3	0.5	0.7	0.9
β	0.1	0.2	0.3	0.4	0.5

Table 2 Orthogonal table for the LTS-EDA

Experiment number	Factors						Type1_22 (AVG)
	N	k	p_c	p_{old}	p_w	β	
1	1	1	1	1	1	1	114,145.0
2	1	2	2	2	2	2	114,163.3
3	1	3	3	3	3	3	114,172.4
4	1	4	4	4	4	4	114,178.1
5	1	5	5	5	5	5	114,172.7
6	2	1	2	3	4	5	114,165.7
7	2	2	3	4	5	1	114,170.0
8	2	3	4	5	1	2	114,165.1
9	2	4	5	1	2	3	114,149.9
10	2	5	1	2	3	4	114,118.6
11	3	1	3	5	2	4	114,128.7
12	3	2	4	1	3	5	114,142.4
13	3	3	5	2	4	1	114,136.2
14	3	4	1	3	5	2	114,101.4
15	3	5	2	4	1	3	114,123.2
16	4	1	4	2	5	3	113,972.5
17	4	2	5	3	1	4	113,960.3
18	4	3	1	4	2	5	113,950.6
19	4	4	2	5	3	1	113,951.1
20	4	5	3	1	4	2	113,948.9
21	5	1	5	4	3	2	113,966.4
22	5	2	1	5	4	3	113,950.6
23	5	3	2	1	5	4	113,942.3
24	5	4	3	2	1	5	113,944.7
25	5	5	4	3	2	1	113,951.8

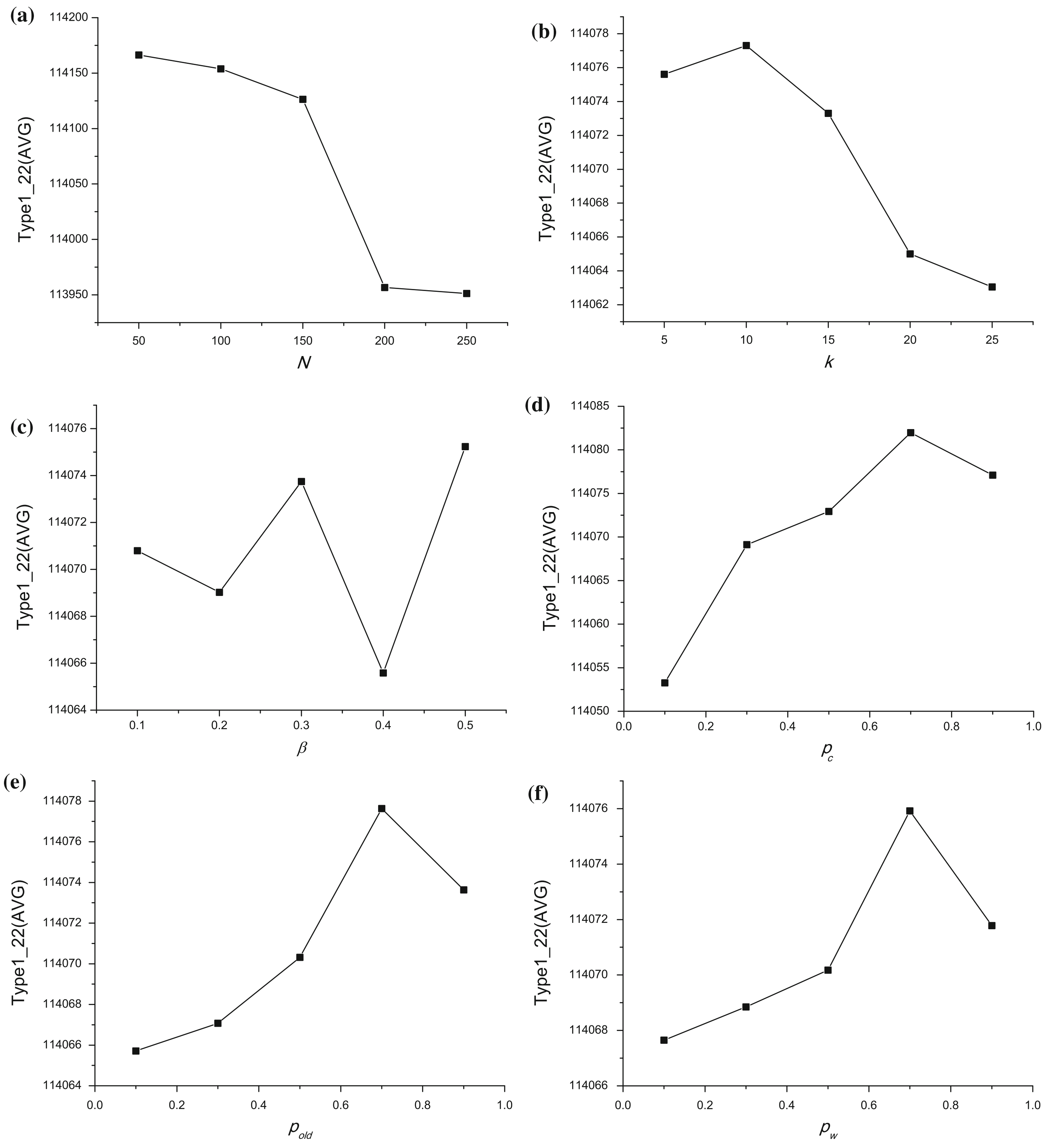

Fig. 2 Factor level trend for each parameter (**a**) N, (**b**) k, (**c**) β, (**d**) p_c, (**e**) p_{old} and (**f**) p_w

4.3 Effectiveness of components in LTS-EDA

To confirm the effectiveness of the components in the LTS-EDA, we conduct our first experiment to compare the LTS-EDA with two variants of the LTS-EDA and a baseline algorithm, random multistart TS (MTS). In the first variant of the LTS-EDA, the guided mutation is removed, and new starting point is generated by directly sampling from the probability vector of the selected cluster as in Emmendorfer (2009). We denote this variant by LTS-EDA w/o GM. In the second variant, the whole clustered EDA component is replaced by the guided mutation. We denote

Table 3 Comparison of the results obtained by the LTS-EDA, two variants of LTS-EDA and MTS for all instances

Algorithm	Average ranking	Type1_22		b2500		p3000, p5000		Holm's test		
		AD-B	*AD-Av*	*AD-B*	*AD-Av*	*AD-B*	*AD-Av*	*p* value	α/*i*	Diff.?
MTS	3.86	431	2,314.57	7,380	1,8968.00	6,215	1,4717.57	1.65E–23	0.017	Yes
LTS-GM	2.56	99	2,204.00	590	7,979.60	**−5,221**	**−1,755.99**	7.14E–7	0.025	Yes
LTS-EDA w/o GM	2.3	335	1,957.70	1,422	7,160.80	**−3,144**	2,787.93	7.8E–5	0.05	Yes
LTS-EDA	1.28	88	1,399.30	480	3,658.60	**−5,275**	**−1,610.70**			

this variant by LTS-GM, where the guided mutation, as a perturbation operator, guides the TS search. In the MTS, the starting point is randomly generated for each invocation of the TS improvement procedure.

In the experimental comparison of LTS-EDA, LTS-EDA w/o GM, LTS-GM and MTS, we use all instances described in Sect. 4.1. The experiment results are summarized in Table 3. In column *AD-B* we show the average difference (over all instances in the set) between the best-known values reported by Palubeckis (2007) and the best solutions found by each algorithm, and in column *AD-Av*, we include the average difference between the best-known values and the average values produced by each algorithm. One algorithm is preferable if it has lower values for these two measures. Bold figures indicate the best-known values or new best solutions found by an algorithm. From Table 3, the MTS seems to perform the worst for both measures. The LTS-EDA outperforms the competitors except that it is outperformed slightly by the LTS-GM on p3000 and p5000 in terms of *AD-Av*.

To show the significant differences between LTS-EDA and other algorithms, nonparametric statistical tests, i.e., Iman-Davenport, Holm and Wilcoxon's test (García et al. 2009, 2010; Derrac et al. 2011), are carried out as in (Lozano et al. 2011). The average rankings of algorithms calculated according to Friedman method are given. Smaller ranking means better performance. We can find that the LTS-EDA is the best-ranked algorithm among all algorithms, followed by LTS-EDA w/o GM, LTS-GM and MTS. In order to analyze these results, first, we apply the Iman-Davenports test (the level of significance considered was 0.05), and significant differences among the rankings of algorithms can be observed (p-value: 7.82E–36). Then, we compare the best-ranked algorithm, LTS-EDA as the control algorithm, with the variants and MTS by Holm's test ($p = 0.05$). In Holm's test column, if the corresponding p-value is smaller than the adjusted α, there is significant differences between the control algorithm and the corresponding algorithm. From the last column of Table 3, we can find that there are significant differences between LTS-EDA and other algorithms, which confirms that the LTS-EDA is the best algorithm.

We point out the following observations from Table 3:

1. The performance difference between the LTS-EDA and LTS-EDA w/o GM shows the positive contribution of the guided mutation to the algorithm.
2. The performance difference between the LTS-EDA and LTS-GM shows the superiority of the clustered EDA, a multimodel EDA, over the guided mutation, a single-model EDA.
3. The MTS is outperformed by all EDA-based TS algorithms, which shows that the EDA plays an important role in the whole search. The EDA model is built to characterize the distribution of promising solutions in the search space, and thus it can extract useful information from the search history of the TS and further guide the TS to interesting or promising search space areas.

4.4 Comparison with recent advanced approaches

The LTS-EDA is compared with several recent advanced approaches, including the ITS (Palubecki 2007), VNS (Brimberg et al. 2009) and TIG (Lozano et al. 2011). The reasons why we choose these algorithms for comparison are

1. The ITS (Palubecki 2007) and VNS (Brimberg et al. 2009) are the best algorithms for the MDP among 30 heuristic algorithms reviewed and compared in the most recent review (Martí et al. 2011). Further, both the ITS and LTS-EDA use the TS component.
2. The TIG (Lozano et al. 2011) is recently proposed and is not included in the recent review (Martí et al. 2011). Simulation results show that the TIG obtains better or competitive performance than VNS and ITS.

Therefore, we select these algorithms as the state-of-the-art algorithms for comparison. For fair comparison, the LTS-EDA and competitors are run under the same computational conditions. We use the source code of the ITS kindly provided by the authors (Palubecki 2007), executable file of the VNS kindly provided by the authors (Brimberg et al. 2009), and implemented the TIG in C.

Table 4 Comparison of the results obtained by the LTS-EDA and other metaheuristic algorithms for Type1_22 instances ($n = 2000$, $m = 200$)

Instance	Best known values	ITS		VNS		TIG		LTS-EDA	
		Best	Av.	Best	Av.	Best	Av.	Best	Av.
Type1_22.1	114,271	65	209.87	48	150.6	48	101.57	5	60.73
Type1_22.2	114,327	29	262.27	**0**	168.87	**0**	69.90	**0**	89.87
Type1_22.3	114,195	69	201.40	19	110.83	5	117.77	**0**	98.97
Type1_22.4	114,093	22	200.53	70	188.13	58	141.93	**0**	79.87
Type1_22.5	114,196	95	273.27	87	184.10	99	194.70	51	134.47
Type1_22.6	114,265	41	168.17	30	99.30	9	96.20	**0**	40.17
Type1_22.7	114,361	12	167.47	**0**	56.30	**0**	71.27	**0**	18.20
Type1_22.8	114,327	25	256.40	**0**	163.33	**0**	193.60	**0**	159.10
Type1_22.9	114,199	9	139.83	16	78.47	16	80.37	**0**	70.97
Type1_22.10	114,229	24	204.93	7	139.33	35	121.43	**0**	56.20
Type1_22.11	114,214	74	237.77	42	145.13	59	139.57	3	69.87
Type1_22.12	114,214	55	249.53	95	143.30	88	156.00	15	84.93
Type1_22.13	114,233	93	279.87	22	168.07	42	167.40	6	85.30
Type1_22.14	114,216	92	248.50	117	194.30	64	202.80	**0**	81.00
Type1_22.15	114,240	11	117.50	1	62.87	6	80.53	**0**	22.03
Type1_22.16	114,335	11	225.40	42	215.43	35	167.90	**0**	36.47
Type1_22.17	114,255	56	217.53	**0**	170.00	18	144.53	6	57.07
Type1_22.18	114,408	46	169.97	**0**	57.10	2	117.37	2	22.83
Type1_22.19	114,201	34	243.20	**0**	124.60	**0**	144.37	**0**	35.87
Type1_22.20	114,349	151	270.67	65	159.43	45	187.23	**0**	95.40

Table 5 Comparison of the results obtained by the LTS-EDA and other metaheuristic algorithms for Beasley instances ($n = 2500$, $m = 1000$)

Instance	Best known values	ITS		VNS		TIG		LTS-EDA	
		Best	Av.	Best	Av.	Best	Av.	Best	Av.
b2500-1	1153,068	624	3,677.33	96	1,911.93	42	1,960.33	**0**	369.20
b2500-2	1129,310	128	1,855.33	88	1,034.33	1,096	1,958.47	154	454.53
b2500-3	1115,538	316	3,281.93	332	1,503.67	34	2,647.87	**0**	290.40
b2500-4	1147,840	870	2,547.93	436	1,521.07	910	1,937.13	**0**	461.73
b2500-5	1144,756	356	1,800.27	**0**	749.4	674	1,655.87	**0**	286.07
b2500-6	1133,572	250	2,173.47	**0**	1,283.53	964	1,807.60	80	218.00
b2500-7	1149,064	306	1,512.60	116	775.47	76	1,338.73	44	264.60
b2500-8	1142,762	**0**	2,467.73	96	862.47	588	1,421.53	22	146.47
b2500-9	1138,866	642	2,944.67	54	837.07	658	1,020.60	6	206.33
b2500-10	1153,936	598	2,024.60	278	1,069.40	448	1,808.73	94	305.27

The LTS-EDA is first tested on 40 MDP instances. Tables 4, 5 and 6 display the name of the instances, the best-known solution value, the results including the best and average values produced by the ITS, VNS, TIG and the LTS-EDA. For larger size instances, the absolute values of the solutions are very large; therefore, we report the relative values of the solutions like in (Palubecki 2007; Brimberg et al. 2009; Lozano et al. 2011), that is, "Best" means the deviation from the best-known solution value of the best solution value found by the algorithms, and "Av." means the deviation from the best-known solution value of the average solution value found by the algorithms. Bold figures indicate the best-known values obtained or the new best solutions found by the corresponding algorithms. In terms of the best solution, the LTS-EDA finds the best-known values on 18 instances and **new** best solutions on 8 out of 40 instances. The ITS, VNS and TIG find the best-known solutions on 2, 9, 5 instances and **new** best solutions on 2, 5, 5 instances. In Table 6, although all algorithms can refresh some best-known solutions reported by Palubeckis

Table 6 Comparison of the results obtained by the LTS-EDA and other metaheuristic algorithms for random larger problems (n = 3000, 5000, m = 0.5n)

Instance	Best known values	ITS		VNS		TIG		LTS-EDA	
		Best	Av.	Best	Av.	Best	Av.	Best	Av.
p3000-1	6501,999	135	1,156.53	**−58**	578.80	**−195**	383.67	**−235**	**−36.93**
p3000-2	18272,568	**0**	1,321.60	**0**	924.20	**0**	991.07	140	387.00
p3000-3	29867,138	1,442	2,214.73	328	963.53	820	1,166.13	**0**	304.33
p3000-4	46914,817	1,084	2,016.93	27	841.47	199	2,255.20	**−97**	90.07
p3000-5	58095,034	**−10**	1,088.60	**−433**	230.00	**−155**	920.27	**−433**	**−62.60**
p5000-1	17508,071	902	2,266.93	**−296**	673.27	**−144**	1247.80	**−1,107**	**−727.00**
p5000-2	50101,514	1,353	3,229.80	**−79**	1,062.00	**−1,029**	954.73	**−1,031**	**−664.20**
p5000-3	82038,723	3,859	6,649.33	321	2,101.40	563	4,414.13	**−889**	**−134.47**
p5000-4	129411,337	**−743**	2,703.90	**−860**	592.90	**−677**	1501.80	**−1,515**	**−1,097.80**
p5000-5	160597,469	1,370	3,746.90	504	1,591.30	602	1,441.90	**−108**	330.90

Table 7 Comparison of the results obtained by the LTS-EDA and ITS for Beasley instances (m_1 = 1,620, m_2 = 1,655)

Instance	Best known values	ITS		LTS-EDA	
		Best	Av.	Best	Av.
b2500-1	1514,640	**0**	459.53	**0**	3.67
b2500-2	1470,970	6	259.00	**0**	101.47
b2500-3	1412,726	48	927.53	80	348.93
b2500-4	1508,654	**0**	157.00	**0**	2.80
b2500-5	1490,366	18	102.27	**0**	42.93
b2500-6	1470,200	**0**	**0.00**	**0**	0.80
b2500-7	1478,924	**0**	106.13	**0**	9.67
b2500-8	1483,440	**0**	312.47	**0**	6.67
b2500-9	1482,384	**0**	1.13	**0**	30.20
b2500-10	1484,416	132	165.27	**0**	108.87

Table 8 LTS-EDA versus ITS using Wilcoxon's test (at the 0.05 level)

Problem	$R+$ (LTS-EDA)	$R-$ (ITS)	p value	Diff.?
Type1_22	210	0	1.91E–6	Yes
b2500	55	0	0.002	Yes
b2500(gMDP)	52	3	0.01	Yes
p3000, p5000	55	0	0.002	Yes

Table 9 LTS-EDA versus VNS using Wilcoxon's test (at the 0.05 level)

Problem	$R+$ (LTS-EDA)	$R-$ (VNS)	p value	Diff.?
Type1_22	210	0	1.91E–6	Yes
b2500	55	0	0.002	Yes
p3000, p5000	55	0	0.002	Yes

Table 10 LTS-EDA versus TIG using Wilcoxon's test (at the 0.05 level)

Problem	$R+$ (LTS-EDA)	$R-$ (TIG)	p value	Diff.?
Type1_22	207	3	9.54E–6	Yes
b2500	55	0	0.002	Yes
p3000, p5000	55	0	0.002	Yes

(2007), the LTS-EDA can find the best **new** solutions than the competitors. In terms of average value, the LTS-EDA still outperforms the competitors. For random larger problems shown in Table 6, even some average solutions found by the LTS-EDA are better than the best-known values.

As mentioned before, only the LTS-EDA and ITS can deal with the gMDP. Table 7 displays the results produced by the ITS and the LTS-EDA for the gMDP instances with m_1 = 1,620, and m_2 = 1,655. From Table 7, we can find that the LTS-EDA obtains the best-known values on nine instances, while the ITS finds the best-known values on just 6 instances. The average performance of the LTS-EDA is also better than that of the ITS except for b2500-6 and b2500-9 instances where the LTS-EDA is slightly worse.

We also carry out the nonparametric statistical tests to see whether there are significant performance differences between the LTS-EDA and other competitors. Tables 8, 9 and 10 show the results of the comparison of the LTS-EDA, ITS, VNS and TIG, by means of Wilcoxon's test (the $R+$ are associated with the LTS-EDA). These tables show that the LTS-EDA is always significantly better than other algorithms, which shows the LTS-EDA is currently the best algorithm for solving the MDP.

Next, we run the LTS-EDA, ITS, VNS and TIG for long CPU time to figure out the capability of each algorithm to find best values. The time limit for a run is 1,200 s for instances in Type1_22, 5h for instances in b2500 and

Table 11 Comparison of the results obtained by the LTS-EDA, ITS, VNS and TIG for long time limit

Instance	Best known values	ITS	VNS	TIG	LTS-EDA	Instance	Best known values	ITS	VNS	TIG	LTS-EDA
Type1_22.1	114,271	14	**0**	35	**0**	b2500-1	1153,068	**0**	**0**	46	**0**
Type1_22.2	114,327	7	126	**0**	**0**	b2500-2	1129,310	354	82	1,150	248
Type1_22.3	114,195	13	19	**0**	**0**	b2500-3	1115,538	**0**	**0**	12	**0**
Type1_22.4	114,093	17	77	**0**	10	b2500-4	1147,840	80	**0**	486	**0**
Type1_22.5	114,196	52	69	**0**	16	b2500-5	1144,756	**0**	**0**	430	**0**
Type1_22.6	114,265	43	9	27	**0**	b2500-6	1133,572	276	88	596	**0**
Type1_22.7	114,361	6	**0**	**0**	**0**	b2500-7	1149,064	**0**	**0**	44	**0**
Type1_22.8	114,327	**0**	**0**	**0**	**0**	b2500-8	1142,762	102	4	218	**0**
Type1_22.9	114,199	3	16	**0**	**0**	b2500-9	1138,866	**0**	**0**	170	**0**
Type1_22.10	114,229	24	**0**	**0**	**0**	b2500-10	1153,936	**0**	56	372	**0**
Type1_22.11	114,214	68	139	15	**0**	p3000-1	6501,999	**−74**	50	565	**−309**
Type1_22.12	114,214	28	27	15	**0**	p3000-2	18272,568	565	160	356	107
Type1_22.13	114,233	11	**0**	**0**	**0**	p3000-3	29867,138	304	285	**0**	**0**
Type1_22.14	114,216	23	**0**	**0**	**0**	p3000-4	46914,817	81	**−227**	530	**−133**
Type1_22.15	114,240	24	1	7	**0**	p3000-5	58095,034	51	**−191**	487	**−392**
Type1_22.16	114,335	10	**0**	8	**0**	p5000-1	17508,071	**−30**	**−511**	92	**−1,144**
Type1_22.17	114,255	13	88	17	**0**	p5000-2	50101,514	**−494**	**−516**	771	**−1,215**
Type1_22.18	114,408	**0**	17	**0**	**0**	p5000-3	82038,723	630	**−93**	1,872	**−963**
Type1_22.19	114,201	23	34	**0**	**0**	p5000-4	129411,337	570	**−579**	**−1,665**	**−1,775**
Type1_22.20	114,349	77	76	6	**0**	p5000-5	160597,469	722	75	1,746	**−312**

p3000, and 10h for p5000 as in Lozano et al. (2011), respectively. Each algorithm on each instance runs only once.

Tables 11–12 show the results found by the LTS-EDA, ITS, VNS and TIG under long time limit. It is not surprising that some best values found are worse than those found under short time limit since each stochastic algorithm runs only once on each instance. The LTS-EDA, ITS, VNS and TIG find the best-known solutions on 28, 8, 13, 13 instances and new best values on 8, 3, 6 and 1 instances out of 40 instances in Table 11. In particular, the new best solutions found by the LTS-EDA are much better than the others found by competitors, except a little worse than the VNS on p3000-4. On ten gMDP instances in Table 12, the LTS-EDA finds nine best-known values while ITS finds seven best-known values. From the results in Tables 11, 12, the LTS-EDA shows the best capability of finding high-quality solutions for the MDP.

Table 12 Comparison of the best values obtained by the LTS-EDA and ITS on Beasley gMDP instances ($m_1 = 1{,}620$, $m_2 = 1{,}655$) for long time limit

Instance	Best known values	ITS	LTS-EDA
b2500-1	1514,640	**0**	**0**
b2500-2	1470,970	**0**	**0**
b2500-3	1412,726	106	80
b2500-4	1508,654	**0**	**0**
b2500-5	1490,366	36	**0**
b2500-6	1470,200	**0**	**0**
b2500-7	1478,924	**0**	**0**
b2500-8	1483,440	**0**	**0**
b2500-9	1482,384	**0**	**0**
b2500-10	1484,416	156	**0**

From all the results mentioned above, we can conclude that the LTS-EDA can obtain better results than other metaheuristic algorithms. In particular, the LTS-EDA obtains remarkable improvement for larger size problems.

4.5 A bit more discussion on parameter settings

Tuning parameters for large size instances, like b2500, p3000 and p5000 instances, will spend large amount of time and thus is a burdensome task. Due to the time limit and for simplicity, we directly use the best combination of parameters for Type1_22 in the whole experiment. One may point out that the best combination of parameters found on Type1_22 instances may not be the best choice for other instances. In this section we will make a little change to the parameter setting to test whether there exists better parameters for other instances. The parameter setting of p_c, p_{old}, p_w and β are directly adopted from Emmendorfer (2009) and Wang et al. (2009), that is, $p_c = p_{\text{old}} = p_w = 0.5$ by default,

Table 13 Comparison of LTS-EDA with original parameters and other parameters

Instance	Best known values	LTS-EDA		LTS-EDA/M	
		Best	Av.	Best	Av.
b2500-2	1129,310	154	454.53	**0**	543.93
b2500-6	1133,572	80	218.00	**0**	140.53
b2500-7	1149,064	44	264.60	**0**	186.33
b2500-8	1142,762	22	146.47	**0**	97.47
b2500-9	1138,866	6	206.33	**0**	190.73
b2500-10	1153,936	94	305.27	**0**	265.00
b2500-3 (gMDP)	1412,726	80	348.93	**0**	101.93
p3000-2	18272,568	140	387.00	**0**	249.60

and $\beta = 0.2$. We choose a subset of instances, which the best-known solution cannot be obtained by the LTS-EDA under the original parameter setting for short time limit.

Table 13 shows the comparison between the LTS-EDA with original parameter setting and modified parameter setting. The LTS-EDA with modified parameter setting (LTS-EDA/M) outperforms the original one on both best solutions and average solutions. Further, it can find the best-known values on all instances. The experiment results show that a better combination of parameters for b2500, p3000 and p5000 instances may exist. In general, each type of instances should have its own best combination of parameters. Hence, in the future, we will make more efforts on developing adaptive parameter setting method to fit for each type of instances.

5 Conclusion

By introducing a knowledge model to TS, this paper presents a learnable TS guided by EDA, named LTS-EDA, for the MDP. The clustered EDA, as a learnable constructive method, is used to create a new starting solution, and the simple TS, as an improvement method, attempts to improve the solution created by the clustered EDA in the LTS-EDA. A distinguishing feature of the LTS-EDA is the usage of the clustered EDA with effective linkage learning to guide TS. The LTS-EDA is tested on 50 large size benchmark problems, and simulation results show that the LTS-EDA is better than the recent advanced algorithms, including the ITS, VNS and TIG. The clustered EDA component, as a multimodel EDA, is also compared with the guided mutation as a single-model EDA. This work proposes a learnable TS, which also can be seen as one of the efforts to make simpler EDAs more effective for hard optimization problems.

In the future, several research lines can be followed. On the one hand, since the LTS-EDA is a framework, some components in the framework can be refined to further improve the performance. For the clustering procedure, an adaptive clustering algorithm without a predefined cluster number can be adopted to cluster the solution pool automatically. For the TS procedure, an adaptive TS with adaptive tabu tenure and search depth can be designed. For the updating procedure, we can further consider two features of the candidate solution to be included into the solution pool: a measure of the contribution of diversity to the population and the fitness function. Thus, we can try to replace an individual in the solution pool with worse values for these two features, not just fitness. In this way, high levels of useful diversity can be preserved. In addition, the adaptive parameter setting method is also worthwhile studying. On the other hand, the LTS-EDA can be applied to solve other hard 0–1 combinatorial optimization problems (Guturu 2008; Wang et al. 2009; Kochenberger et al. 2004), such as UBQP, maximum satisfiability problem (MAX-SAT), polygonal approximation problem, and problems in graph and set theory including set covering (partitioning) problem, maximum clique problem, and maximum cut problem.

Acknowledgments This work was supported in part by the National Natural Science Foundation of China (60805026, 60905038, 61070076, 61033010), the Zhujiang New Star of Science and Technology in Guangzhou City (2011), and the Fundamental Research Funds for the Central Universities (10lgpy32).

References

Andrade PMD, Ochi LS, Martins SL (2005) GRASP with path relinking for the maximum diversity problem. In: Nikoletseas S (ed) Proceedings 4th International Workshop on Efficient Experimental Algorithms WEA, vol 3539. Springer, Berlin, pp 558–569

Aringhieri R, Cordone R (2011) Comparing local search metaheuristics for the maximum diversity problem. J Oper Res Soc 62(2):266–280

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

Baluja S (1994) Population-based incremental learning: a method for integrating genetic search based function optimization and competitive learning. Carnegie Mellon Univ., Pittsburgh, PA Tech. Rep. CMU-CS-94-163

Baluja S, Davies S (1997) Using optimal dependency-trees for combinatorial optimization: learning the structure of the search space. In: Proceedings of 1997 International Conference on Machine and Learning, pp 30–38

Baluja S, Davies S (1998) Fast probabilistic modeling for combinatorial optimization. In: Proceedings of 15th National Conference Artificial Intelligence (AAAI-98), pp 469–476

Beasley JE (2000) OR-Library: Unconstrained binary quadratic programming. http://people.brunel.ac.uk/~mastjjb/jeb/orlib/bqpinfo.htm de Bonet JS, Isbell CL Jr, Viola P (1994) MIMIC: finding optima by estimating probability densities. In: Mozer MC, Jordan MI, Petsche T (eds) Advances in neural information processing systems, vol 9. MIT Press, Cambridge, pp 424–430

Brimberg J, Mladenović N, Urošević D, Ngai E (2009) Variable neighborhood search for the heaviest k-subgraph. Comput Oper Res 36(11):2885–2891

Derrac J, García S, Molina D, Herrera F (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol Comput 1(1):3–18

Duarte A, Martí R (2007) Tabu search and GRASP for the maximum diversity problem. Eur J Oper Res 178(1):71–84

Emmendorfer LR, Pozo ATR (2009) Effective linkage learning using low-order statistics and clustering. IEEE Trans Evol Comput 13(6):1233–1246

Gallego M, Duarte A, Laguna M, Martí R (2009) Hybrid heuristics for the maximum diversity problem. Comput Optim Appl 44(3): 411–426

García S, Molina D, Lozano M, Herrera F (2009) A study on the use of nonparametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization. J Heurist 15(6):617–644

García S, Fernández A, Luengo J, Herrera F (2010) Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power. Inform Sci 180(10):2044–2064

Glover F, Lv Z, Hao J-K (2010) Diversification-driven tabu search for unconstrained binary quadratic problems. 4OR Q J Oper Res 8(3): 239–253

Guturu P, Dantu R (2008) An impatient evolutionary algorithm with probabilistic Tabu search for unified solution of some NP-Hard problems in graph and set theory via clique finding. IEEE Trans Syst Man Cybern Part B 38(3):645–666

Hao J-K (2011) Memetic algorithms for discrete optimization. In: Neri F, Cotta C, Moscato P (eds) Handbook of memetic algorithms. Springer, Berlin (in press)

Harik G (1999) Linkage learning via probabilistic modeling in the ECGA. Springer, Berlin

Harik GR, Lobo FG, Goldberg DE (1999) The compact genetic algorithm. IEEE Trans Evol Comput 3(4):287–297

Hauschild M, Pelikan M, Sastry K, Lima C (2009) Analyzing probabilistic models in hierarchical BOA. IEEE Trans Evol Comput 13(6):1199–1217

James T, Rego C, Glover F (2009) Multistart tabu search and diversification strategies for the quadratic assignment problem. IEEE Trans Syst Man Cybern Part A Syst Hum 39(3):579–596

Katayama K, Narihisa H (2004) An evolutionary approach for the maximum diversity problem. In: Hart W, Krasnogor N, Smith JE (eds) Recent advances in memetic algorithms. Spinger, Berlin, pp 31–47

Kochenberger G, Glover F, Alidaee B, Rego C (2004) A unified modeling and solution framework for combinatorial optimization problems. OR Spectr 26(2):1–14

Kuo CC, Glover F, Dhir KS (1993) Analyzing and modeling the maximum diversity problem by zero-one programming. Decis Sci 24:1171–1185

Lozano M, Garca-Martnez C (2010) Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: overview and progress report. Comput Oper Res 37(3):481–497

Lozano JA, Zhang Q, Larrañaga P (2009) Guest Editorial: Special issue on evolutionary algorithms based on probabilistic model. IEEE Trans Evol Comput 13(6):1197–1198

Lozano M, Molina D, García-Martínez C (2011) Iterated greedy for the maximum diversity problem. Eur J Oper Res 214(1):31–38

Lv Z, Hao J-K (2010) A memetic algorithm for graph colorin. Eur J Oper Res 203(1):241–250

Lv Z, Glover F, Hao J-K (2010) A hybrid metaheuristic approach to solving the UBQP problem. Eur J Oper Res 207(3): 1254–1262

Martí R, Moreno-Vega JM, Duarte A (2011) Advanced multi-start methods. In: Gendreau M, Potvin J-Y (eds) Handbook of Metaheuristics, International series in operations research & management Science, vol 146, pp 265–281

Martí R, Gallego M, Duarte A (2011) Heuristics and metaheuristics for the maximum diversity problem. J Heurist (in press). doi: 10.1007/s10732-011-9172-4

Martí L, García J, Berlanga A, Molina JM (2009) On the model-building issue of multi-objective estimation of distribution algorithms. In: Corchado E et al (eds) Proceeding of the 4th international conference on hybrid artificial intelligence system, LNAI, vol. 5572. Springer, Berlin, pp 293–300

Martí R, Gallego M, Duarte A (2010) A branch and bound algorithm for the maximum diversity problem. Eur J Oper Res 200(1): 36–44

Misevicius A, Lenkevicius A, Rubliauskas D (2006) Iterated tabu search: an improvement to standard tabu search. Inform Technol Control 35(3):187–197

Mühlenbein H, Paass G (1996) From recombination of genes to the estimation of distributions I. binary parameters. In: Proceedings of International Conference on Evolution and Computers, PPSN IV, pp 178–187

Mühlenbein H, Mahnig T, Rodriguez A (1999) Schemata, distributions and graphical models in evolutionary optimization. J Heurist 5:215–247

Palubeckis G (2004) Multistart tabu search strategies for the unconstrained binary quadratic optimization problem. Ann Oper Res 131(1–4):259–282

Palubeckis G (2006) Iterated tabu search strategies for the unconstrained binary quadratic optimization problem. Informatica 17(2):279–296

Palubeckis G (2007) Iterated tabu search for the maximum diversity problem. Appl Math Comput 189(1):371–383

Peña J, Lozano J, Larrañaga P (2005) Globally multimodal problem optimization via an estimation of distribution algorithm based on unsupervised learning of Bayesian networks. Evol Comput 13(1):43–66

Pelikan M, Mühlenbein H (1999) The bivariate marginal distribution algorithm. In: Roy R, Furuhashi T, Chawdhry PK (eds) Advances in soft computing–engineering design and manufacturing. Springer, London pp 521–535

Pelikan M, Goldberg DE, Cantú-paz EE (2000) Linkage problem, distribution estimation, and Bayesian networks. Evol Comput 8(3):311–340

Platel MD, Schliebs S, Kasabov N (2009) Quantum-inspired evolutionary algorithm: a multimodel EDA. IEEE Trans Evol Comput 13(6):1218–1232

Santana R, Larrañaga P, Lozano JA (2009) Research topics in discrete estimation of distribution algorithms based on factorizations. Memet Comput 1(1):35–54

Silva GC, de Andrade MRQ, Ochi LS, Martins SL, Plastino A (2007) New heuristics for the maximum diversity problem. J Heurist 13(4):315–336

Wang L, Fang C (2010) An effective estimation of distribution algorithm for the multi-mode resource-constrained project scheduling problem. Comput Oper Res 39(2):449–460

Wang L, Fang C (2011) An effective shuffled frog-leaping algorithm for multi-mode resource-constrained project scheduling problem. Inform Sci 181(20):4804–4822

Wang J, Zhou Y, Yin J, Zhang Y (2009) Competitive Hopfield network combined with estimation of distribution for maximum diversity problems. IEEE Trans Syst Man Cybern Part B Cybern 39(4):1048–1066

Wang J, Kuang Z, Xu X, Zhou Y (2009) Discrete particle swarm optimization based on estimation of distribution for polygonal approximation problems. Expert Syst Appl 36(5):9398–9408

Xin B, Chen J, Zhang J, Fang H, Peng Z-H (2011) Hybridizing differential evolution and particle swarm optimization to design powerful optimizers: a review and taxonomy. IEEE Trans Systems Man Cybern Part C Appl Rev (in press)

Zhang Q, Sun J (2006) Iterated local search with guided mutation. In: Proceedings of IEEE Congress on Evolutionary Computation, pp 924–929

Zhang Q, Sun J, Tsang E (2005) Evolutionary algorithm with guided mutation for the maximum clique problem. IEEE Trans Evol Comput 9(2):192–200

Zhang Q, Sun J, Xiao G, Tsang E (2007) Evolutionary algorithms refining a heuristic: a hybrid method for shared-path protections in WDM networks under SRLG constraints. IEEE Trans Syst Man Cybern Part B 37(1):51–61

Zhang Q, Sun J, Tsang E (2007) Combinations of estimation of distribution algorithms and other techniques. Int J Autom Comput 4(3):273–280