Expert Systems with Applications 39 (2012) 5593–5599

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

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

An effective hybrid EDA-based algorithm for solving multidimensional knapsack problem

Ling Wang *, Sheng-yao Wang, Ye Xu

Tsinghua National Laboratory for Information Science and Technology (TNList), Department of Automation, Tsinghua University, Beijing 100084, China

ARTICLE INFO

Keywords:
Multidimensional knapsack problem
Estimation of distribution algorithm
Probability model
Hybrid algorithm
Design of experiment

ABSTRACT

In this paper, an effective hybrid algorithm based on estimation of distribution algorithm (EDA) is proposed to solve the multidimensional knapsack problem (MKP). With the framework of EDA, the probability model is built with the superior population and the new individuals are generated based on probability model. In addition, an updating mechanism of the probability model is proposed and a mechanism for initializing the probability model based on the specific knowledge of the MKP is also proposed to improve the convergence speed. Meanwhile, an adaptive local search is proposed to enhance the exploitation ability. Furthermore, the influences of parameters are investigated based on Taguchi method of design of experiment and the importance of repair operator is also studied via simulation testing and comparisons. Finally, numerical simulation is carried out based on the benchmark instances, and the comparisons with some existing algorithms demonstrate the effectiveness of the proposed algorithm.

1. Introduction

The multidimensional knapsack problem (MKP) is a well-known NP-hard problem with wide engineering backgrounds. Many practical problems can be formulated as the MKP, such as cargo loading, cutting stock problems, resource allocation in computer systems, and economics problems (Martello & Toth, 1990). The study of the MKP has significant importance both academic and engineering fields. Early research on the MKP mainly focused on the exact solution methods, such as branch and bound method (Shih, 1979), dynamic programing algorithm (Toth, 1980), etc. However, the computational effort and memory requirement are tremendous when using the exact solution methods, so that they are insufficient in solving the large-scale problems. In recently years, many heuristic and meta-heuristic algorithms for solving the MKP have been proposed. Chu and Beasley (1998) proposed a heuristic based on genetic algorithm (GA), which required only a modest amount of computational effort. Vasquez and Vimont (2005) obtained some better results by combining linear programming (LP) with an efficient tabu search (TS). Kong, Tian, and Kao (2008) proposed a new ant colony optimization (ACO) approach to solve the MKP, called binary ant system (BAS). Very recently, Ke, Feng, Ren, and Wei (2010) developed an algorithm based on ACO called DMMAS to solve the MKP as an extension of max-min ant system that imposed lower and upper trail limits on pheromone values to avoid stagnation.

As a newly proposed population-based algorithm, Estimation of distribution algorithm (EDA) (Larranaga & Lozano, 2002) has gained an increasing study and wide applications during recent years. Population-based incremental learning (PBIL) (Baluja, 1994) is the earliest model of the EDA. According to the complexity of the model, the EDA can be classified as univariate model, bivariate model or multivariate model. The PBIL, univariate marginal distribution algorithm (UMDA) (MÄuhlenbein & Paass, 1996) and compact GA (CGA) (Harik, Lobo, & Goldberg, 1998) are univariate models, while mutual information maximization for input clustering (MIMIC) (De Bonet, Isbell, & Viola, 1997), combining optimizers with mutual information trees (COMIT) (Baluja & Davies, 1997) and bivariate marginal distribution algorithm (BMDA) (Pelikan & Mühlenbein, 1999) are bivariate models. The factorized distribution algorithms (FDA) (Mühlenbein & Mahnig, 1999), extended compact GA (ECGA) (Harik, 1999) and bayesian optimization algorithm (BOA) (Pelikan, Goldberg, & Cantú-Paz, 1999) are multivariate models. For more details about the EDA, please refer Larranaga and Lozano (2002).

So far the EDA has been applied to a variety of academic and engineering optimization problems, such as feature selection, cancer classification, quadratic assignment problem, machinery structure design, nurse rostering, and etc (Zhou & Sun, 2007). However, to the best of our knowledge, there is almost no research work about the EDA for solving the MKP except the relaxed complemental EDA for the MKP (Yang, Ouyang, & Quan, 2007). In this paper, we take the characteristic of the MKP to propose a hybrid EDA (HEDA) for solving the MKP. A probability model is built and an updating mechanism is proposed. In addition, a mechanism of

* Corresponding author. Tel.: +86 10 62783125; fax: +86 10 62786911.
E-mail address: wangling@tsinghua.edu.cn (L. Wang).

0957-4174/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2011.11.058

initializing the probabilistic model is proposed based on the specific knowledge of the MKP in the framework of the EDA. Moreover, the influence of parameter setting and the importance of repair operator are both investigated. Finally, we use a set of benchmark instances to test the performances of the HEDA and to compare it with some existing methods to further demonstrate the effectiveness of the HEDA.

The remainder of the paper is organized as follows: In Section 2, the MKP is formulated. In Section 3, the basic EDA is introduced briefly. Then, the hybrid EDA for the MKP is proposed in Section 4. The influences of parameter setting and repair operator are investigated in Section 5, and computational results and comparisons are also provided. Finally we end the paper with some conclusions in Section 6.

2. Multidimensional knapsack problem

The multidimensional knapsack problem can be formulated as follows:

$$\text{Maximize } \sum_{j=1}^{n} p_j x_j \tag{1}$$

$$\text{Subject to } \sum_{j=1}^{n} r_{ij} x_j \le b_i, \quad i = 1,2,\ldots,m, \tag{2}$$

$$x_j \in \{0,1\}, \quad j = 1,2,\ldots,n \tag{3}$$

where n is the number of items, and m is the number of knapsack constraints with capacities b_i ($i = 1,2,\ldots,m$), associated weights r_{ij} and profits p_j.

The objective is to find a subset of items that yields a maximum profit without exceeding the resource capacities. Without loss of generality, it can be assumed that all the data r_{ij}, b_i and p_j are non-negative integers and $r_{ij} \leqslant b_i$, $\sum_{j=1}^{n} r_{ij} > b_i$ for all $i = 1,2,\ldots,m$ and $j = 1,2,\ldots,n$.

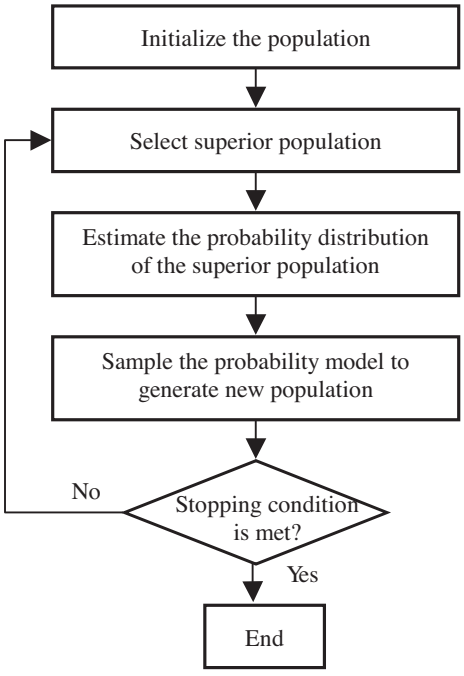

Fig. 1. The general framework of the EDA.

3. Estimation of distribution algorithm

Estimation of distribution algorithms (EDA) is a new paradigm in the field of Evolutionary Computation, which employs explicit probability distributions in optimization (Larranaga & Lozano, 2002). Compared with the genetic algorithms, the EDA reproduces new population implicitly instead of the crossover and mutation operators. In the EDA, a probability model of the most promising area is built by statistical information based on the searching experience, and then the probability model is used for sampling to generate the new individuals. Meanwhile, the probability model is updated in each generation with the potential individuals of the new population. In such an iterative way, the population evolves and finally satisfactory solutions can be obtained.

The general framework of the EDA is illustrated in Fig. 1.

The critical step of the above procedure is to estimate the probability distribution. The EDA makes use of the probability model to describe the distribution of the solution space and the updating process reflects the evolutionary trend of the population. Due to the difference of problem types, a proper probability model and the updating mechanism should be well developed to estimate the underlying probability distribution. Nevertheless, the EDA pays more attention to global exploration while its exploitation capability is limited. So, an effective EDA should balance the exploration and the exploitation abilities.

4. Hybrid EDA for MKP

In this section, we will propose a hybrid EDA (HEDA) for solving the MKP. First, we will introduce the encoding scheme, probability model, update mechanism, repair operator and local search strategy. Then, we will present the framework of proposed HEDA.

4.1. Encoding scheme

Every individual of the population is a solution of the MKP, which is represented by an n-bit binary string. The encoding scheme of the individual is illustrated in Fig. 2, where each bit $x_j = 1$ or 0.

4.2. Probability model and updating mechanism

The HEDA uses an n-dimensional vector $p(X) = [p(x_1), p(x_2), \ldots, p(x_n)]$ to represent the probability model of the solution space, where $p(x_j)$ is the probability of $x_j = 1$. Generally, the vector $p(X)$ is initialized as $p(X) = [0.5, 0.5, \ldots, 0.5]$, which ensures that the whole solution space can be sampled uniformly. To improve the converging speed of the HEDA, we propose an initialization mechanism based on the specific knowledge of the MKP.

During the experiment, we find empirically that bias sampling towards the optimal solution can increase the probability to obtain good solutions. Thus, we choose a solution $S = (s_1, s_2, \ldots, s_n)$ with a similar construction to the optimal solution to guide the initialization of the probability vector as follows:

$$p(x_j) = \begin{cases} 0.5 + \beta, s_j = 1 \\ 0.5 - \beta, s_j = 0 \end{cases}, \quad j = 1,2,\ldots,n \tag{4}$$

where $\beta \in (0, 0.5)$ represents the level of bias to the solution S that can be obtained by a simple procedure.

x_1	x_2	$\cdots$	x_{i-1}	x_i	x_{i+1}	$\cdots$	x_{n-1}	x_n

Fig. 2. Representation of a solution in HEDA.

Let $S' = (s'_1, s'_2, \ldots, s'_n)$ be the solution of the linear programing (LP) relaxation problem (see Section 4.3) of the MKP. Then, S is constructed as follows:

$$s_j = \begin{cases} 1, s'_j = 1 \\ 0, else \end{cases}, \quad j = 1, 2, \ldots, n \tag{5}$$

In each generation of the HDEA, the new individuals are generated via sampling according to the probability vector $p(X)$. For every x_j to be determined, a random number $r \in (0,1)$ is generated. Let $x_j = 1$ if $r < p(x_j)$, otherwise let $x_j = 0$. Once all the items are determined, an individual is constructed. In such a way, P individuals are constructed and their profits are calculated.

Then, the superior population that consists of the best N solutions is determined, and the probability vector $p(X)$ is updated according to the following equation:

$$p_{l+1}(X) = (1-\alpha)p_l(X) + \alpha\frac{1}{N}\sum_{k=1}^{N} X_l^k \tag{6}$$

where $p_l(X)$ is the probability vector, and X_l^k is the kth individual of the superior population in the lth generation, and $\alpha \in (0,1)$ is the learning speed.

4.3. Repair operator

The above way to generate new solutions does not take into account the feasibility of the solutions that are sampled based on the probability model. The individuals could be illegal for violating some of the resource constraints. So, a repair procedure is needed if illegal individuals are constructed. In this sub-section, we provide two repair operators as follows:

Repair Operator 1 (RO1): The general technique is a greedy-like heuristic based on the pseudo-utility ratios that are calculated by the surrogate duality approach (Pirkul, 1987). The general idea of this method is described briefly as follows:

The surrogate relaxation problem of the MKP can be defined as follows:

$$\text{Maximize} \sum_{j=1}^{n} p_j x_j \tag{7}$$

$$\text{Subject to} \sum_{j=1}^{n}\left(\sum_{i=1}^{m}\omega_i r_{ij}\right)x_j \le \sum_{i=1}^{m}\omega_i b_i, \tag{8}$$

$$x_j \in \{0,1\}, \quad j = 1, 2, \ldots, n \tag{9}$$

where $\omega = \{\omega_1, \omega_2, \ldots, \omega_m\}$ is a set of surrogate multipliers (or weights) of some positive real numbers. These weights can be obtained by solving the LP relaxation of the original MKP and the values of the dual variables are viewed as the weights. In other words, the weight $\omega_i (i = 1, 2, \ldots, m)$ can be regarded as the shadow price of the i-th constraint in the LP relaxation of the MKP.

Having obtained the weights, the pseudo-utility ratio for each variable can be computed as follows:

$$u_j = \frac{p_j}{\sum_{i=1}^{m}\omega_i r_{ij}} \tag{10}$$

Before performing the repair procedure, the variables are ordered in the decreasing order according to their pseudo-utility ratios. The repair operator consists of two phases. The first phase examines each bit of the solution string in the increasing order of u_j and changes a bit from one to zero until the solution is legal. The second phase reverses the process by examining each bit in the decreasing order of u_j and changes a bit from zero to one as long as the feasibility is not violated. The pseudo-code of this repair operator is illustrated in Fig. 3.

Procedure *Repair Operator 1*
Let: $I = \{1,2,\cdots,m\}$.

1: Initialize $R_i = \sum_{j=1}^{n} r_{ij}x_j, \forall i \in I$
2: **FOR** $j = n$ to 1 **DO**
3: **IF** ($x_j = 1$) and ($R_i > b_i, \exists i \in I$) **THEN**
4: Set $x_j \leftarrow 0$;
5: Set $R_i \leftarrow R_i - r_{ij}, \forall i \in I$;
6: **END IF**
7: **END FOR**
8: **FOR** $j = 1$ to n **DO**
9: **IF** ($x_j = 0$) and ($R_i + r_{ij} \le b_i, \forall i \in I$) **THEN**
10: Set $x_j \leftarrow 1$;
11: Set $R_i \leftarrow R_i + r_{ij}, \forall i \in I$.
12: **END IF**
13: **END FOR**

Fig. 3. Procedure of RO1.

Repair Operator 2 (RO2): The above repair operator requires solving the LP relaxation of the MKP. So far, most mathematical tools are only able to solve the LP problems with small or medium scale, and it is difficult to implement that operator when the problem scale is large. Thus, we propose a new repair operator based on the specific knowledge of the MKP. With this technique, there is no need to solve the LP relaxation of the MKP in advance.

During the initialization phase, a simple matrix Q and an assistant matrix F are obtained. The element q_{ij} in Q is computed as follows:

$$q_{ij} = \frac{p_j}{r_{ij}} \tag{11}$$

Each row of the assistant matrix F is an arrangement of all the items, and f_{ij} represents the number of the jth item in the descending order of the elements in the ith row of matrix Q. To be specific, the pseudo-code is illustrated in Fig. 4.

Actually, q_{ij} is the jth item's pseudo-utility ratio of the ith constraint. For an illegal solution, this algorithm examines every constraint in the ascending order of q_{ij} and changes one bit from one to

Procedure *Repair Operator 2*
Let: $I = \{1,2,\cdots,m\}$.

1: Initialize $R_i = \sum_{j=1}^{n} r_{ij}x_j, \forall i \in I$
2: **FOR** $i = 1$ to m **DO**
3: **FOR** $j = n$ to 1 **DO**
4: **IF** ($x_{f_{ij}} = 1$) and ($R_i > b_i$) **THEN**
5: Set $x_{f_{ij}} \leftarrow 0$;
6: Set $R_i \leftarrow R_i - r_{i,f_{ij}}, \forall i \in I$;
7: **END IF**
8: **END FOR**
9: **END FOR**
10: Find the constraint i that has the minimum of $b_i - R_i$.
11: **FOR** $j = 1$ to n **DO**
12: **IF** ($x_{f_{ij}} = 0$) and ($R_i + r_{i,f_{ij}} \le b_i, \forall i \in I$) **THEN**
13: Set $x_{f_{ij}} \leftarrow 1$;
14: Set $R_i \leftarrow R_i + r_{i,f_{ij}}, \forall i \in I$.
15: **END IF**
16: **END FOR**

Fig. 4. Procedure of RO2.

zero until that constraint is satisfied. After that, it balances the remaining resources according to the minimum remaining resource. Clearly, this new repair operator is simple to implement, and especially it is not limited to the scale of problem.

4.4. Local search strategy

It is widely accepted that a local search procedure is efficient in improving the solutions generated by the EDA. In the HEDA, we use a simple method as the local search strategy.

In particular, five bits are randomly selected from the solution string, and they are changed form zero to one or from one to zero. When changing the last zero-valued bit to one, if the profit of the current solution is less than that of the best solution obtained so far, the new individual would not be constructed. Then, the new solution is repaired if it is illegal. Such a procedure is applied on the best individual of the current population Y times in every generation, where Y represents the intensity of local search.

Because the solutions obtained in the earlier generations are often of low quality, there is no need to use the local search too intensively. Thus, we design a special adaptive way to control Y to enhance the exploitation ability as follows:

$$Y = \begin{cases} 500, & generation < 50 \\ 800, & 50 \le generation < 100 \\ 1000, & generation \ge 100 \end{cases} \quad (12)$$

4.5. Procedure of HEDA

With the design above, the procedure of the HEDA is illustrated in Fig. 5.

In summary, in the initial stage of evolution, the promising area of the solution space may be found by using the EDA based estimating and sampling. Then, local search strategy is performed in the "good" region to obtain better solutions. The benefits of the EDA and the local search are combined to balance global exploration and local exploitation. Moreover, the repair operator is applied on the illegal solutions if necessary.

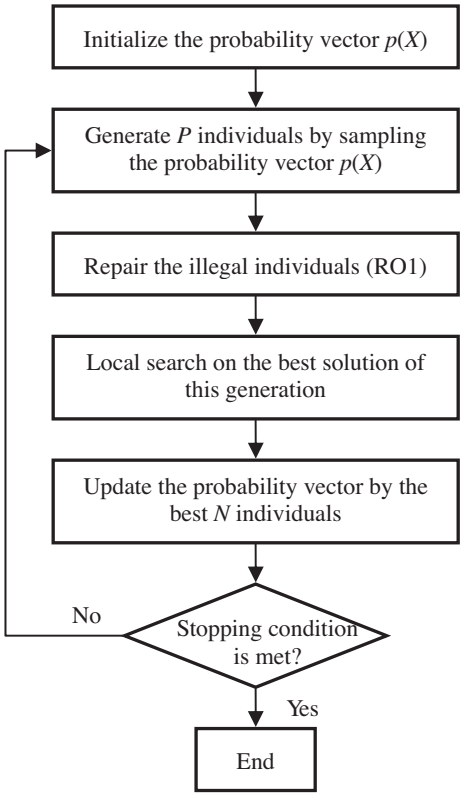

Fig. 5. The framework of the HEDA.

5. Computational results and comparisons

We choose three different sets of widely used benchmark instances to test the HEDA. Test set 1 consists of small size problems (available from ORLIB) with $m = 2$ to 30 and $n = 15$ to 105. Test set 2 consists of 11 large size problems (available from http://hces.bus.olemiss.edu/tools.html) with $m = 15$ to 100 and $n = 100$ to 2500. Test set 3 consists of 20 medium size problems (available from ORLIB) with $m = 5$ or 10 and $n = 100$. The algorithm is coded in C++ and run on a 3.2 GHz Intel Core i5 processor.

5.1. Parameters setting

The proposed HEDA contains several key parameters: the population size of each generation (P), the number of selected individual to update the probability model (N), the learning speed (α) and the bias level of initialization (β). To investigate the influence of those parameters on the performance of the algorithm, we implement the Taguchi method of design of experiment (DOE) (Montgomery, 2005) by using the problem 100.10.05. Combinations of different values of these parameters are shown in Table 1.

For each parameter combination, the HEDA is run 50 times independently with the maximum number 10,000 of the constructed solutions as the stopping criterion. The average response variable (ARV) value is the average of profit obtained by the HEDA in 50 times. According to the number of parameters and the number of factor levels, we choose the orthogonal array $L_{16}(4^4)$. That is, the total number of treatment is 16, the number of parameters is 4, and the number of factor levels is 4. The orthogonal array and the obtained ARV values are listed in Table 2.

Table 1
Combinations of parameter values.

Parameters	Factor level			
	1	2	3	4
P	20	30	40	50
N (%P)	10	20	30	40
α	0.001	0.01	0.1	0.5
β	0	0.05	0.1	0.2

Table 2
Orthogonal array and ARV values.

Experiment number	Factor				ARV
	P	N	α	β	
1	1	1	1	1	22683.5
2	1	2	2	2	22687.4
3	1	3	3	3	22688.2
4	1	4	4	4	22686.7
5	2	1	2	3	22684.9
6	2	2	1	4	22678.4
7	2	3	4	1	22682.8
8	2	4	3	2	22681.5
9	3	1	3	4	22671.6
10	3	2	4	3	22679.8
11	3	3	1	2	22676.9
12	3	4	2	1	22676.9
13	4	1	4	2	22669.7
14	4	2	3	1	22668.4
15	4	3	2	4	22672.4
16	4	4	1	3	22672.0

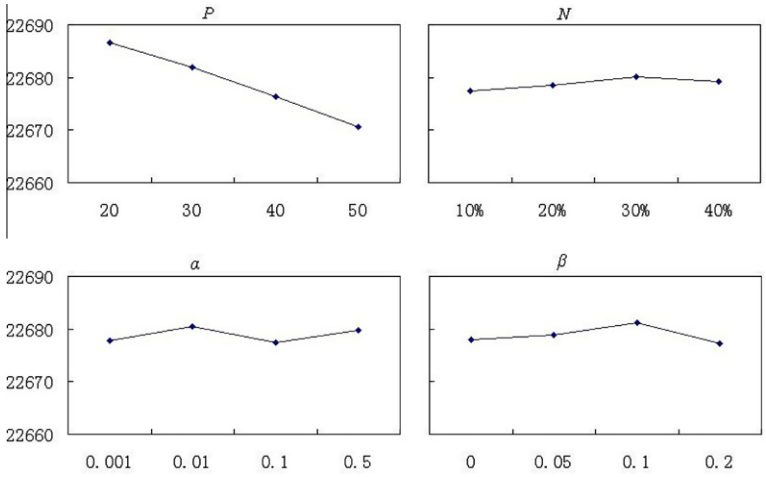

Fig. 6. Factor level trend of the HEDA.

Table 3
Response value.

Level	P	N	α	β
1	22686.5	22677.4	22677.7	22677.9
2	22681.9	22678.5	22680.4	22678.9
3	22676.3	22680.1	22677.4	22681.2
4	22670.6	22679.3	22679.7	22677.3
Delta	15.9	2.7	3.0	3.9
Rank	1	4	3	2

According to the orthogonal table, we illustrate the trend of each factor level in Fig. 6. Then, we figure out the response value of each parameter to analyze the significance rank of each parameter. The results are listed in Table 3.

From Table 3 it can be seen that P is the most significant parameter among the four parameters. In condition of a fixed maximum number of constructed solutions, small population size allows more generations so that it may provide deeper search. In addition, the significant rank of β is the second and the performance of the HEDA is better when β is more than zero, which demonstrates that the new initialization mechanism of the probability model is efficient. A better choose of the parameter combination is: $P = 20$, $N = 6$, $\alpha = 0.01$, and $\beta = 0.1$.

5.2. Influence of repair operator

Next, we study the influence of the repair operator in the HEDA. Denote HEDA1 as the HEDA with RO1 and HEDA2 as the HEDA with RO2. We run both HEDA1 and HEDA2 20 times independently by using test set 2. Since test set 2 has medium or large scale, we set $P = 200$, $N = 10$, $\alpha = 0.01$, $\beta = 0.1$, and set the maximum number 100,000 of the constructed solutions as the stopping criterion.

The statistical results are summarized in Table 4, where Min.Dev and Ave.Dev represent the minimum and average percentage deviations from the best-known solutions, and Var. Dev denotes the variance of the deviations.

From Table 4, it can be seen that the performance of HEDA1 is better than that of HEDA2 on the first 8 problems. However, since RO1 depends on solving the LP relaxation problem of the MKP, it is very difficult to solve the problem with LP when the number of $m+n$ is very large. So, HEDA2 can solve another 3 problems with larger scale while HEDA1 cannot solve those problems. The conclusion is that the repair operator is important to improve the performance of the algorithm. Comparing RO2, RO1 is superior to achieve better results but it can only applied to the problems with small or medium size.

5.3. Comparisons of HEDA with RCEDA

The relaxed complemental estimation of distribution algorithm (RCEDA) is the only existing EDA for solving the MKP (Yang et al., 2007). In this sub-section, we compare the HEDA with the RCEDA by using test set 1. For the HEDA, we set $P = 20$, $N = 6$, $\alpha = 0.01$, and $\beta = 0.1$. Both the algorithms are run 100 times independently. The maximum numbers of solutions constructed in the HEDA and the RCEDA are 5,000 and 50,000, respectively. That is, the HEDA uses less computational effort. The statistical results are listed in Table 5,

Table 4
Comparisons of HEDA1 with HEDA2.

Problem	$n \times m$	Best known	HEDA1			HEDA2		
			Min. Dev	Ave. Dev	Var. Dev	Min. Dev	Ave. Dev	Var. Dev
Mk_gk01	100 × 15	3766	**0.2655**	**0.3173**	**0.0783**	0.6107	0.9360	0.6555
Mk_gk02	100 × 25	3958	**0.0758**	**0.1983**	**0.1194**	0.7580	0.9790	0.4039
Mk_gk03	150 × 25	5650	**0.1239**	**0.2018**	**0.0998**	0.9381	1.1531	0.7828
Mk_gk04	150 × 50	5764	**0.0867**	**0.3348**	**0.2916**	1.0930	1.2673	0.7069
Mk_gk05	200 × 25	7557	**0.1323**	**0.2772**	**0.2150**	1.2439	1.4960	0.7483
Mk_gk06	200 × 50	7672	**0.2998**	**0.4438**	**0.2339**	1.1210	1.4436	1.1573
Mk_gk07	500 × 25	19215	**0.5204**	**0.7203**	**0.8240**	1.7018	1.8460	0.9873
Mk_gk08	500 × 50	18801	**0.6117**	**0.7680**	1.1028	1.6595	1.7260	**0.2880**
Mk_gk09	1500 × 25	58085	–	–	–	2.4585	2.5461	0.6813
Mk_gk10	1500 × 50	57292	–	–	–	2.0177	2.1170	0.7807
Mk_gk11	2500 × 100	95231	–	–	–	1.7043	1.7348	0.2184

Note: the bold values mean better results.

Table 5
Comparisons of HEDA with RCEDA.

Problem	$n \times m$	Best known	RCEDA		HEDA	
			AVG	SR	AVG	SR
Sento1	60 × 30	7772	7771.5	95	**7772**	**100**
Sento2	60 × 30	8722	8720.9	93	**8772**	**100**
Weing7	105 × 2	1095445	1095422.6	90	**1095445**	**100**
Weing8	105 × 2	624319	624316.8	80	**624319**	**100**
Knap15	15 × 10	4015	4012.7	95	**4015**	**100**
Knap20	20 × 10	6120	6117.9	97	**6120**	**100**
Knap39	39 × 5	10618	10615.3	95	**10618**	**100**
Knap28	28 × 10	12400	12389.4	98	**12400**	**100**

Note: the bold values mean better results.

Table 6
Comparisons of HEDA with DMMAS.

Problem	Best known	DMMAS			HEDA		
		Best	AVG	STD	Best	AVG	STD
5.100.00	24381	**24381**	24362	23.8	**24381**	**24381**	0.0
5.100.01	24274	**24274**	24273	6.2	**24274**	**24274**	0.0
5.100.02	23551	**23551**	23540	7.2	**23551**	**22541**	5.7
5.100.03	23534	**23534**	23482	14.9	**23534**	**23524**	11.0
5.100.04	23991	**23991**	23954	10.8	**23991**	**23976**	16.3
5.100.05	24613	**24613**	24608	6.3	**24613**	**24612**	5.7
5.100.06	25591	**25591**	**25591**	0.0	**25591**	**25591**	0.0
5.100.07	23410	**23410**	**23404**	13.3	**23410**	23375	7.1
5.100.08	24216	**24216**	**24211**	5.9	**24216**	**24211**	5.9
5.100.09	24411	**24411**	24406	13.8	**24411**	**24411**	0.0
10.100.00	23064	**23064**	23045	19.6	**23064**	**23051**	2.4
10.100.01	22801	**22801**	**22742**	40.8	**22801**	22739	25.2
10.100.02	22131	**22131**	22091	29.8	**22131**	**22131**	0.0
10.100.03	22772	**22772**	22710	37.6	**22772**	**22772**	0.0
10.100.04	22751	**22751**	22617	43.9	**22751**	**22620**	24.7
10.100.05	22777	**22777**	22663	40.3	**22777**	**22683**	30.0
10.100.06	21875	**21875**	**21826**	28.4	**21875**	21823	9.0
10.100.07	22635	**22635**	**22557**	31.0	**22635**	22552	27.7
10.100.08	22511	22438	22409	17.3	**22511**	**22424**	5.0
10.100.09	22702	**22702**	22696	32.7	**22702**	**22702**	0.0

Note: the bold values mean better results.

where SR denotes the successful ratio of the run that obtains the best-known value among the 100 independent runs.

From Table 5, it can be seen that the successful ratio and average profit obtained by the HEDA are better than those obtained by the RCEDA, even the maximum number of constructed solutions by the HEDA is only one tenth of that by the RCEDA. So, the conclusion is that our HEDA is superior to the existing EDA based algorithm for solving the MKP. Note that, it has been shown that the performance of the RCEDA is better than the hybrid GA (Yang et al., 2007). So, the performance of our HEDA is definitely better than that of the hybrid GA in the literature.

5.4. Comparisons of HEDA with DMMAS

Finally, we compare the HEDA with the DMMAS by using test set 3. For the HEDA, we set $P = 20$, $N = 6$, $\alpha = 0.01$, and $\beta = 0.1$. We set the maximum numbers of constructed solution 10,000 as the stopping criterion for the two algorithms. For every instance, the two algorithms are run 50 times independently. The statistical results are summarized in Table 6, including the best profit, average profit and the standard derivation.

Form Table 6, it can be seen that the HEDA can find all the best solutions of the twenty instances while the DMMAS can find the best solutions of nineteen instances. As for the average profit, the results of the HEDA are better than those of the DMMAS in 14 instances, the results of the HEDA are worse than those of the DMMAS in only 4 instances, and the results are the same in 2 instances. Especially, the HEDA can find the best-known solutions consistently in 7 instances. As for the STD, the results of the HEDA are better than that of the DMMAS in most instances. So, the conclusion is that our HEDA is superior to the DMMAS in solving the MKP.

6. Conclusions

In this paper, a hybrid estimation of distribution algorithm was proposed for solving the MKP. We designed a probability model with the superior population for the EDA to solve the MKP by generating the new individuals via sampling based on the probability model. We also designed an updating mechanism for the probability model and an initialization mechanism for probability vector to increase the convergence speed. To avoid the difficulty of the existing repair operator in solving large-scale problems, we presented a special repair operator. Moreover, an adaptive local search was designed to enhance the exploitation of the algorithm. The influence of parameter setting was investigated by using DOE based testing, and the two repair operators were also compared through simulation testing. We also compared our HEDA with the existing RCEDA and DMMAS, which showed that the HEDA was more effective in solving the MKP. The future work is to design more effective EDA for the MKP including the model and the repair operator as well as the local search and to study the EDA for solving other kinds of combinatorial optimization problems, such as scheduling problems.

Acknowledgments

This research is partially supported by National Science Foundation of China (61174189 and 60834004), Doctoral Program Foundation of Institutions of Higher Education of China (20100002110014), the National Key Basic Research and Development Program of China (No. 2009CB320602) and National Science and Technology Major Project of China (No. 2011ZX02504-008).

References

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

Baluja, S., & Davies, S. (1997). Using optimal dependency-trees for combinatorial optimization: Learning the structure of the search space. In: *Proceedings of the 14th international conference on machine learning*, San Francisco. pp. 30–38.

Chu, P. C., & Beasley, J. E. (1998). A genetic algorithm for the multidimensional knapsack problem. *Journal of Heuristics, 4*, 63–86.

De Bonet, J. S., Isbell, C. L. Jr., & Viola, P. (1997). MIMIC: Finding optima by estimating probability densities. *Advances in Neural Information Processing Systems*, Cambridge: MIT Press. pp. 424–430..

Harik, G. (1999). Linkage learning via probabilistic modeling in the ECGA. Illigal Report NO. 99010, Illinois Genetic Algorithms Lboratory, Illinois: University of Illinois at Urbana-Champaign.

Harik, G. R., Lobo, F. G., & Goldberg, D. E. (1998). The compact genetic algorithm. In: *Proceedings of the IEEE conference on evolutionary computation*, Indianapolis. pp. 523–528.

Ke, L. J., Feng, Z. R., Ren, Z. G., & Wei, X. L. (2010). An ant colony optimization approach for the multidimensional knapsack problem. *Journal of Heuristics, 16*, 65–83.

Kong, M., Tian, P., & Kao, Y. C. (2008). A new ant colony optimization algorithm for the multidimensional Knapsack problem. *Computers & Operations Research, 35*, 2672–2683.

Larranaga, P., & Lozano, J. A. (2002). *Estimation of distribution algorithms: A new tool for evolutionary computation*. Netherlands: Springer.

Martello, S., & Toth, P. (1990). *Knapsack problems: Algorithms and computer implementations*. New York: Wiley.

MÄuhlenbein, H., & Paass, G. (1996). From recombination of genes to the estimation of distributions I: Binary parameters. *Lecture Notes in Computer Science, 1141*, 178–187.

Montgomery, D. C. (2005). *Design and analysis of experiments*. Arizona: John Wiley & Sons.

Mühlenbein, H., & Mahnig, T. (1999). Convergence theory and applications of the factorized distribution algorithm. *Jounal of Computing and Information Technology, 7*, 19–32.

Pelikan, M., Goldberg, D. E., & Cantú-Paz, E. (1999). BOA: The bayesian optimization algorithm. In: *Proceedings of the genetic and evolutionary computation*, San Francisco. 525–532.

Pelikan, M., & Mühlenbein, H. (1999). The bivariate marginal distribution algorithm. In J. M. Benítez (Ed.), *Advances in Soft Computing: Engineering Design and Manufacturing*. London: Springer-Verlag.

Pirkul, H. (1987). A heuristic solution procedure for the multiconstraint zero-one knapsack problem. *Naval Research Logistics, 34*, 161–172.

Shih, W. (1979). A branch and bound method for the multiconstraint zero-one knapsack problem. *Journal of the Operational Research Society, 30*, 369–378.

Toth, P. (1980). Dynamic programing algorithms for the zero-one knapsack problem. *Computing, 25*, 29–45.

Vasquez, M., & Vimont, Y. (2005). Improved results on the 0-1 multidimensional knapsack problem. *European Journal of Operational Research, 165*, 70–81.

Yang, G. Y., Ouyang, Z. M., & Quan, H. Y. (2007). Relaxed complemental estimation of distribution algorithm for the multidimensional knapsack problem. *Computer Engineering and Applications, 43*, 77–80.

Zhou, S. D., & Sun, Z. Q. (2007). A survey on estimation of distribution algorithms. *Acta Automatica Sinica, 33*, 113–124.