Estimation of Distribution Algorithms for Knapsack Problem

Shang Gao
School of Computer Science and Technology, Jiangsu University of Science and Technology, Zhenjiang 212003, China
Email: gao_shang@just.edu.cn

Ling Qiu
Artificial Intelligence of Key Laboratory of Sichuan Province, Sichuan University of Science and Engineering ,Zigong 643000,China

Cungen Cao
Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, China
Email: cgcao@ict.ac.cn

***Abstract*—Estimation of distribution algorithms (EDAs) is a new kind of evolution algorithm. In EDAs , through the statistics of the information of selected individuals in current group, the probability of the individual distribution in next generation is given and the next generation of group is formed by random sampling. A wide range of mathematical model of the knapsack problem are proposed. In this paper, the EDAs is applied to solve the knapsack problem. The influence of several strategies, such as numbers of population and better population selection proportions are analyzed. Simulation results show that the EDAs is reliable and effective for solving the knapsack problem. The Maltab code is given also. It can easily be modified for any combinatorial problem for which we have no good specialized algorithm.**

***Index Terms*—estimation distribution algorithm, knapsack problem, genetic algorithm**

I. Introduction

The knapsack problem or rucksack problem is a problem in combinatorial optimization: Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. It derives its name from the problem faced by someone who is constrained by a fixed-size knapsack and must fill it with the most valuable items. The 0/1 knapsack problem is proven to be NP-complete. It is traditionally solved by the dynamic programming algorithm, which is accepted as the most practical way to solve this problem. With the advent of parallel processors, many researchers concentrated their efforts on development of approximation algorithms for NP-complete problems based on the application of parallel processors. For the 0/1 Knapsack problem, such works were reported by Peters and Rudolf [1], and Gopalakrishnan et al. [2]. Another relevant branch of research was related to design of systolic arrays for dynamic programming problems. This approach was considered in works of Li et al. [3], Lipton et al. [4] and others. A different model for the parallel computation of the Knapsack problem with weights given by real numbers was considered by A. Yao [5]. Currently, the method solving knapsack problem are accurate methods (such as dynamic programming, the recursive method, backtracking, branch and bound method [6]), approximation algorithms (such as the greedy method [6], Lagrange method, etc.) and intelligent optimization algorithms (such as simulated annealing algorithm[7], genetic algorithms [7] genetic annealing evolutionary algorithm [8], ant colony algorithm [9, 10]), particle swarm optimization algorithm [11], DNA [12]）. A new version of MOEA/D with uniform design for solving multiobjective 0/1 knapsack problems is proposed in reference [13].

Estimation of distribution algorithms (EDAs) are stochastic optimization techniques that explore the space of potential solutions by building and sampling explicit probabilistic models of promising candidate solutions. This explicit use of probablistic models in optimization offers some significant advantages over other types of metaheuristics. EDAs were successfully applied to optimization of large spin glass instances in two-dimensional and three-dimensional lattices, military antenna design, multiobjective knapsack, groundwater remediation design, aminoacid alphabet reduction for protein structure prediction, identification of clusters of genes with similar expression profiles, economic dispatch, forest management, portfolio management, cancer chemotherapy optimization, environmental monitoring network design, and others. In this paper, a new method for knapsack problem is put forward based on estimation of distribution algorithms and better population selection proportions are analyzed. Estimation of distribution algorithms (EDAs) is a new area of evolutionary computation. In EDAs there is neither crossover nor mutation operator. New population is generated by sampling the probability distribution, which is estimated from a database containing selected individuals of previous generation.

Since Estimation of distribution algorithms (EDAs) were proposed by Baluja in 1994 [14], EDAs quickly become an important branch of evolutionary algorithms because they have better mathematical foundation than other evolutionary algorithms. On the basis of statistical learning theory, EDAs use some individuals selected from the population at the current evolutionary generation to build a probability model and then produces offspring for the next generation by sampling the probability model in a probabilistic way. A lot of investigations in [15-22] show that EDAs have good optimization performance in both combinatorial problems and numeric optimization problems. Until now there are many studies about EDAs, but EDAs mainly consist of several types: Population based incremental learning (PBIL) [14], univariate marginal distribution algorithm (UMDA), compact genetic algorithm (CGA), mutual-information-maximizing input clustering algorithm (MIMIC) , bivariate marginal distribution algorithm (BMDA), factorized distribution algorithm (FDA), Bayesian optimization algorithm (BOA), extended compact genetic algorithm (ECGA) and estimation of Bayesian network algorithm (EBNA). UMDA works well only in the solution of linear problems with independent variables, so it requires extension as well as application of local heuristics for combinatorial optimizations. PBIL uses vector probabilities instead of population and has good performance for solving problems with independent variables in binary search space. CGA independently deals with each variable and needs less memory than simple genetic algorithm. MIMIC searches the best permutation of the variables at each generation to find the probability distribution through using Kullback-Leibler distance. BMDA is mainly based on the construction of a dependency graph, which is acyclic but does not necessarily have to be a connected graph. FDA integrates evolutionary algorithms with simulated annealing. This method requires additively decomposed function and the factorization of the joint probability distribution remains same for all iterations. BOA applies Bayesian network and Bayesian Dirichlet metric to estimate joint probability distributions, thus, it can take advantage of the prior information about the problem. ECGA factorizes the joint probability distribution as a product of marginal distributions of variable size. EBNA employs Bayesian network for the factorization of the joint probability distribution and BIC score.

II. The Mode of Knapsack Problem

The most common problem being solved is the 0-1 knapsack problem, which restricts the number x_i of copies of each kind of item to zero or one.

Let there be n items, x_1 to x_n where x_i has a value p_i and weight c_i. The maximum weight that we can carry in the bag is C. It is common to assume that all values and weights are nonnegative. To simplify the representation, we also assume that the items are listed in increasing order of weight.

$$\begin{aligned} &\max \sum_{i=1}^{n} p_i x_i \\ &s.t. \quad \sum_{i=1}^{n} c_i x_i \le C \\ &\quad x_i \in \{0,1\}, (i = 1,2,\cdots,n) \end{aligned} \tag{1}$$

Maximize the sum of the values of the items in the knapsack so that the sum of the weights must be less than the knapsack's capacity.

The knapsack problem is one of the most studied problems in combinatorial optimization, with many real-life applications. For this reason, many special cases and generalizations have been examined.

One common variant is that each item can be chosen multiple times. The **bounded knapsack problem** specifies, for each item i, an upper bound u_i (which may be a positive integer, or infinity) on the number of times item i can be selected:

$$\begin{aligned} &\max \sum_{i=1}^{n} p_i x_i \\ &s.t. \quad \sum_{i=1}^{n} c_i x_i \le C \\ &\quad 0 \le x_i \le u_i \end{aligned} \tag{2}$$

x_i integral for all i.

The **unbounded knapsack problem** (sometimes called the **integer knapsack problem**) does not put any upper bounds on the number of times an item may be selected:

$$\begin{aligned} &\max \sum_{i=1}^{n} p_i x_i \\ &s.t. \quad \sum_{i=1}^{n} c_i x_i \le C \\ &\quad x_i \ge 0 \end{aligned} \tag{3}$$

x_i integral for all i.

The unbounded variant was shown to be NP-complete in 1975 by Lueker.

If the items are subdivided into *k* classes denoted N_i, and exactly one item must be taken from each class, we get the **multiple-choice knapsack problem**:

$$\begin{aligned} &\max \sum_{i=1}^{k} \sum_{j\in N_i} p_{ij} x_{ij} \\ &s.t. \quad \sum_{i=1}^{k} \sum_{j\in N_i} c_{ij} x_{ij} \le C \\ &\sum_{j\in N_i} x_{ij} = 1 \qquad 1 \le i \le k \\ &x_{ij} \in \{0,1\} \end{aligned} \tag{4}$$

If for each item the profits and weights are identical, we get the subset sum problem (often the corresponding decision problem is given instead):

$$\begin{aligned} &\max \sum_{i=1}^{n} p_i x_i \\ &s.t. \quad \sum_{i=1}^{n} p_i x_i \le C \\ &x_i \in \{0,1\}, (i = 1,2,\cdots,n) \end{aligned} \tag{5}$$

If we have *n* items and *m* knapsacks with capacities C_i, we get the **multiple knapsack problem**:

$$\begin{aligned} &\max \sum_{i=1}^{m} \sum_{j=1_i}^{n} p_{ij} x_{ij} \\ &s.t. \quad \sum_{i=1}^{n} c_j x_{ij} \le C_i \qquad 1 \le i \le m \\ &\sum_{i=1}^{m} x_{ij} \le 1 \qquad 1 \le j \le n \\ &x_{ij} \in \{0,1\} \end{aligned} \tag{6}$$

As a special case of the multiple knapsack problem, when the profits are equal to weights and all bins have the same capacity, we can have **multiple subset sum problem**: **Quadratic knapsack problem**:

$$\begin{aligned} &\max \sum_{i=1}^{n} p_i x_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} p_{ij} x_i x_j \\ &s.t. \quad \sum_{i=1}^{n} c_i x_i \le C \\ &x_i \in \{0,1\} \end{aligned} \tag{7}$$

If there is more than one constraint (for example, both a volume limit and a weight limit, where the volume and weight of each item are not related), we get the **multiply constrained knapsack problem**, **multi-dimensional knapsack problem**, or ***m*-dimensional knapsack problem**. (Note, "dimension" here does not refer to the shape of any items.) This has 0-1, bounded, and unbounded variants; the unbounded one is shown below.

$$\begin{aligned} &\max \sum_{i=1}^{n} p_i x_i \\ &s.t. \quad \sum_{j=1}^{n} c_{ij} x_j \le C_i \qquad 1 \le i \le m \\ &x_i \ge 0 \qquad 1 \le i \le n \end{aligned} \tag{8}$$

x_i integral for all i.

If all the profits are 1, we can try to minimize the number of items which exactly fill the knapsack:

$$\begin{aligned} &\min \sum_{i=1}^{n} x_i \\ &s.t. \quad \sum_{i=1}^{n} c_i x_i = C \\ &x_i \in \{0,1\}, (i = 1,2,\cdots,n) \end{aligned} \tag{9}$$

We call these problems Knapsack-like problems.

If we have a number of containers (of the same size), and we wish to pack all *n* items in as few containers as possible, we get the bin packing problem, which is modeled by having indicator variables $y_i = 1 \Leftrightarrow$ container *i* is being used:

$$\begin{aligned} &\min \sum_{i=1}^{n} y_i \\ &s.t. \quad \sum_{j=1}^{n} c_j x_{ij} \le C y_i \qquad 1 \le i \le n \\ &\sum_{i=1}^{n} x_{ij} = 1 \qquad 1 \le j \le n \\ &y_i \in \{0,1\} \qquad 1 \le i \le n \\ &x_{ij} \in \{0,1\} \end{aligned} \tag{10}$$

The cutting stock problem is identical to the bin packing problem, but since practical instances usually have far fewer types of items, another formulation is often used. Item j is needed B_j times, each "pattern" of items which fit into a single knapsack have a variable, x_i (there are m patterns), and pattern i uses item j b_{ij} times:

$$\begin{aligned} &\min x\sum_{i=1}^{m} x_i \\ &s.t. \quad \sum_{i=1}^{m} b_{ij}x_i \le B_j \qquad 1\le j\le n \\ &\sum_{i=1}^{n} x_{ij} = 1 \qquad 1\le j\le n \\ &x_i \in \{0,1,\cdots,n\} \qquad 1\le i\le m \end{aligned} \tag{11}$$

If, to the multiple choice knapsack problem, we add the constraint that each subset is of size n and remove the restriction on total weight, we get the assignment problem, which is also the problem of finding a maximal bipartite matching:

$$\begin{aligned} &\max \sum_{i=1}^{k}\sum_{j=1_i}^{n} p_{ij}x_{ij} \\ &s.t. \quad \sum_{i=1}^{n} x_{ij} = \le 1 \qquad 1\le j\le n \\ &\sum_{j=1}^{n} x_{ij} = 1 \qquad 1\le i\le n \\ &x_{ij} \in \{0,1\} \qquad 1\le i\le k, j\in N_i \end{aligned} \tag{12}$$

In the Maximum Density Knapsack variant there is an initial weight c_0 , and we maximize the density of selected of items which do not violate the capacity constraint:

$$\begin{aligned} &\max \frac{\sum_{i=1}^{n} p_i x_i}{c_0 + \sum_{i=1}^{n} c_i x_i} \\ &s.t. \quad \sum_{i=1}^{n} c_i x_i \le C \\ &x_i \in \{0,1\} \end{aligned} \tag{13}$$

III. Basic Estimation of Distribution Algorithms

Estimation of distribution algorithms (EDAs), sometimes called probabilistic model-building genetic algorithms (PMBGAs), are stochastic optimization methods that guide the search for the optimum by building and sampling explicit probabilistic models of promising candidate solutions[12]. Optimization is viewed as a series of incremental updates of a probabilistic model, starting with the model encoding the uniform distribution over admissible solutions and ending with the model that generates only the global optima [13].

EDAs belong to the class of evolutionary algorithms. The main difference between EDAs and most conventional evolutionary algorithms is that evolutionary algorithms generate new candidate solutions using an implicit distribution defined by one or more variation operators, whereas EDAs use an explicit probability distribution encoded by a Bayesian network, a multivariate normal distribution, or another model class. In EDAs the new population of individuals is generated without using neither crossover nor mutation operators. Instead, the new individuals are sampled starting from a probability distribution estimated from the database containing only selected individuals from the previous generation. Figure 1 illustrates the flowchart of EDA.

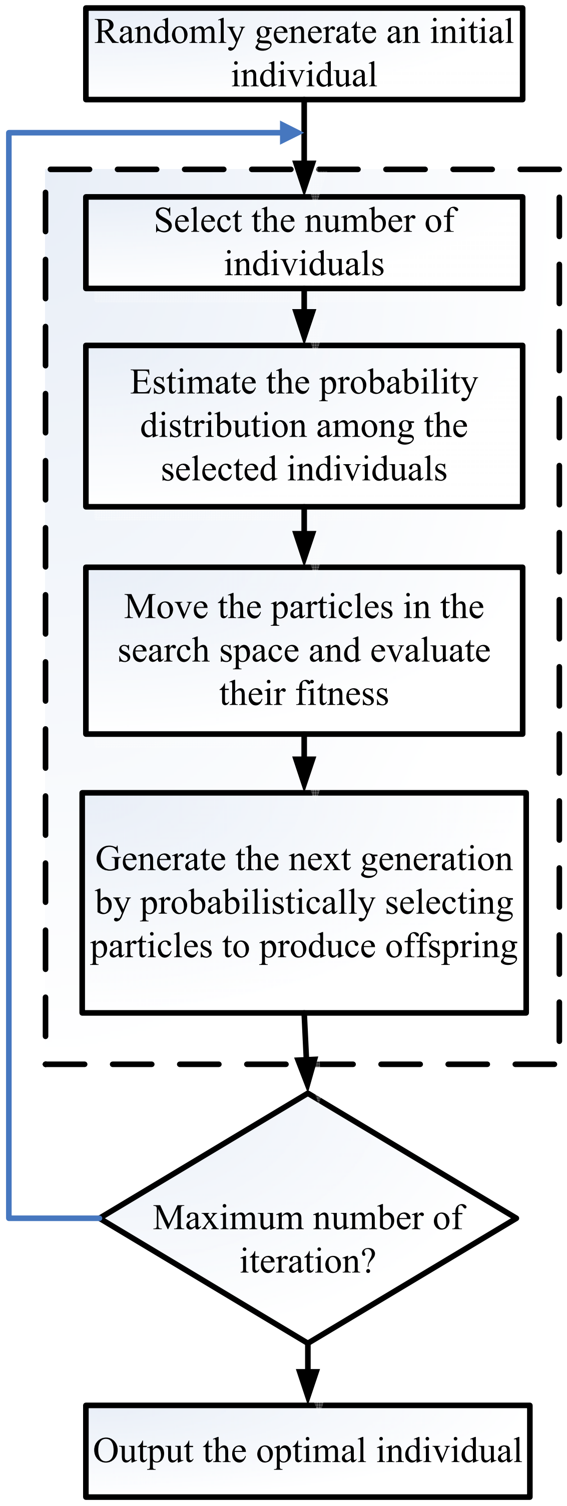
Randomly generate an initial individual
Select the number of individuals
Estimate the probability distribution among the selected individuals
Move the particles in the search space and evaluate their fitness
Generate the next generation by probabilistically selecting particles to produce offspring
Maximum number of iteration?
Output the optimal individual

Figure 1. Illustrates the flowchart of EDA.

The general procedure of an EDA is outlined in the following[16]:

```
Step 1 t = 0;
Step 2 initialize model M(0) to represent uniform distribution over admissible solutions
Step 3  while (termination criteria not met)
    Step 3.1 P = generate N>0 candidate solutions by sampling M(t)
    Step 3.2 F = evaluate all candidate solutions in P
    Step 3.3 M(t+1) = adjust_model(P,F,M(t))
    Step 3.4 t = t + 1
```

Using explicit probabilistic models in optimization allowed EDAs to feasibly solve optimization problems that were notoriously difficult for most conventional evolutionary algorithms and traditional optimization techniques, such as problems with high levels of epistasis. Nonetheless, the advantage of EDAs is also that these algorithms provide an optimization practitioner with a series of probabilistic models that reveal a lot of information about the problem being solved. This information can in turn be used to design problem-specific neighborhood operators for local search, to bias future runs of EDAs on a similar problem, or to create an efficient computational model of the problem.

IV. Solving 0/1 Knapsack Problem by EDAs

Firstly, we transform (1)(constrained problem) into a single unconstrained problem.

$$\min f = -\sum_{i=1}^{n} p_i x_i + M\left\{\min\left\{0, \left[C - \sum_{i=1}^{n} c_i x_i\right]\right\}\right\}^2 \qquad (14)$$

where $M > 0$ is a large number.

The other knapsack problem models can also transform. For example, we transform (13)(constrained problem) into a single unconstrained problem.

$$\min f = -\frac{\sum_{i=1}^{n} p_i x_i}{c_0 + \sum_{i=1}^{n} c_i x_i} + M\left\{\min\left\{0, \left[C - \sum_{i=1}^{n} c_i x_i\right]\right\}\right\}^2 \qquad (15)$$

The estimation of distribution algorithms for 0/1knapsack problem is as follows:

Step 1 Using the uniform design technique, for each variable are the probability of random values within $(p_1, p_2, \cdots, p_n)^T = (0.5, 0.5, \cdots, 0.5)^T$. Generate N individuals constitute the initial population.

Step 2 Assess the fitness of all individuals in the initial population, and retain the best solution.

Step 3 Order the population by fitness in descending sorting, and choose the optimal m individuals (m ≤ N).

Step 4 Build a probability vector $(p_1, p_2, \cdots, p_n)^T$ based on the statistical information extracted from the selected m solutions in the current population.

Step 5 Sample N new solutions from this build probability models $(p_1, p_2, \cdots, p_n)^T$.

Step 6 If the given stopping condition (up to the required number of iterations nmax) is not met, go to step 2.

The estimation of distribution algorithms' time complexity is estimated as follows: The time to calculate the fitness operation is the longest, so the time complexity of algorithm is about $O(N.n_{\max})$.

The estimation of distribution algorithms for other knapsack problem models is similar to above algorithm.

V. Numerical Example

We solve a typical knapsack problem of literature [9]. $n = 10$, $C = 269$ g, $\{p_1, p_2, \ldots, p_{10}\}=\{55,10,47,5,4,50,8,61,85,87\}$, and $\{c_1, c_2, \ldots, c_{10}\}=\{95,4,60,32,23,72,80,62,65,46\}$.

The program of EDAs is implemented by MATLAB. The MATLAB implementation is given below:

```
%EDA_Knapsack.m
%EDAs for Knapsack Problem
 clear all
 n=10;
 p=[55 10 47 5 4 50 8 61 85 87]';
 c=[95 4 60 32 23 72 80 62 65 46]';
 G=269;
 M=1;
 N=1000;
 m=0.4*N;
 r=[0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5 0.5]';
 for nn=1:20
    for j=1:N
       X(j,:)=Xrand(r,n);
    end
    for j=1:N
       fknapsack(j)=objknapsack(n,c,p,X(j,:),G,M);
       ffknapsack(j)=X(j,:)*p;
    end
    SX=X;
    SX(:,n+1)=fknapsack';
    B=sortrows(SX,n+1);
    fmin=B(1,n+1);
    xmin=B(1,1:n);
    for k=1:m
       SelectX(k,1:n)= B(k,1:n);
    end
    r=sum(SelectX)/m;
    for i=1:N
       if ffknapsack(i)>295
          ffknapsack(i)=0;
       end
    end
    opf(nn)=max(ffknapsack);
    meanf(nn)=mean(ffknapsack);
end
opf
meanf
plot(1:20,opf,'-.',1:20,meanf,'-')
legend('Best values','Average values');
xlabel('The time of iteration')
ylabel('The value of knapsack')
```

Xrand.m is is given below:

```
function y=Xrand(r,n)
for i=1:n
    if rand<=r(i)
        y(i)=1;
    else
        y(i)=0;
    end
end
```

Objknapsack.m is is given below:

```
function f=objknapsack(n,c,p,x,G,M)
f=-x*p+M*(min(0,G-x*c))^2;
```

When N = 100, m = 0.4 * N, the procession of value is shown in Figure 1. The main parameters affecting the performance of the EDA are the number N of the population and selected population number m.

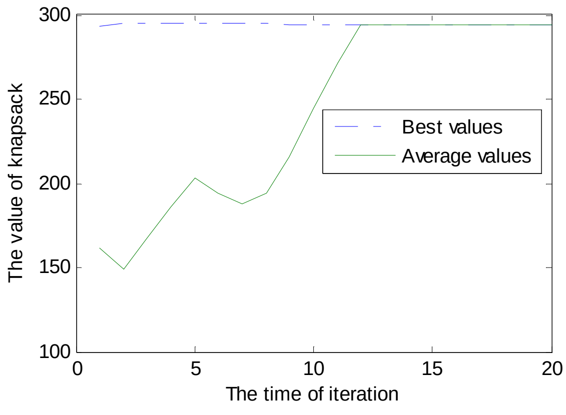

Figure 2. Figure2. The iterative process of the best values and the average values

When m=N/2, it test 100 times, and the statistics are shown in Table 1. When N = 100, sometimes the algorithm goes into the local optimal solution and not to reach the global optimum value 295, so we can't give statistics. Seen from Table 1, N is smaller, the effect is not good. N is greater, the effect is better. Of course, the greater the time is needed. We set N to moderate, such as N of 800.

TABLE I.
COMPARISON RESULTS OF N

N	Average number of iterations	Minimum number of iterations	Maximum number of iterations
100	-	-	-
200	3.3	1	10
300	2.82	1	6
400	2.53	1	6
500	2.13	1	5
600	1.99	1	5
700	1.71	1	5
800	1.69	1	5
900	1.59	1	4
1000	1.56	1	4

When N = 800, it test 100 times, and the statistics are shown in Table 2. From Table 2, if the ratio of m/N is the greater, the effect is the worse. Of course, the ratio m/N is too small, it is easy to fall into local minima. So the ratio of m/ N is 10% -30%, the results were quite good.

TABLE II.
COMPARISON RESULTS OF M/N

N	Average number of iterations	Minimum number of iterations	Maximum number of iterations
2.5%	—	—	—
5%	1.42	1	2
10%	1.37	1	2
20%	1.53	1	3
30%	1.56	1	3
40%	1.60	1	4
50%	1.69	1	5
60%	1.71	1	5

VI. CONCLUSIONS

The estimation of distribution algorithms can not only solve the knapsack problem, but also the algorithm can be applied for integer programming problem. Estimation of distribution algorithms can be slightly modified to solve similar nonlinear mixed integer programming problem. The estimation of distribution algorithms can be further improved, such as adding the crossover operators and mutation operators, so the performance may be better.

ACKNOWLEDGMENT

This work was supported by the Open Project Program of Key Laboratory of Intelligent Computing & Information Processing (Xiangtan University), Ministry of Education (No. 2011ICIP05), Artificial Intelligence of Key Laboratory of Sichuan Province(2012RYJ04), Jiangsu 333 Project , Qing Lan Project. and the National Natural Science Foundation of China under Grant 51008143.

REFERENCES

[1] J. Peters, L. Rudolph. Parallel Approximation Schemes for Subset Sum and Knapsack Problems. In 22nd Annual

Allerton Conference on Communication, Control and Computing, 1984, pp.671-680.
[2] P.S. Gopalakrishnan, I.V. Ramakrishnan, L.N. Kanal. Parallel Approximate Algorithms for the 0-1 Knapsack Problem. In Proceedings of the International Conference on Parallel Processing, 1986, pp 444-451.
[3] Guo-jie Li, Benjamin W.Wah. Systolic Processing for Dynamic Programming Problems. In Proceedings of the International Conference on Parallel Processing, 1985, pp.434-441.
[4] Richard J. Lipton, Daniel Lopresti. Delta Transformations to Simplify VLSI Processor Arrays for Serial Dynamic Programming. In Proceedings of the International Conference on Parallel Processing, 1986, pp.917-920.
[5] Andrew Chi-Chin Yao. On the Parallel Computation for the Knapsack Problem. In 13th Annual ACM Symposium on Theory of Computing, 1981, pp. 123-127.
[6] Wang Xiaodong. Algorithm design and analysis. Beijing: Electronic Industry Press,2001, pp.92-168.(In Chinese)
[7] Wang Ling. Intelligent optimization algorithm and its application, Beijing: Tsinghua University Press,2001:17-59. (In Chinese)
[8] Jin huimin,Ma Liang. Genetic annealing evolutionary algorithm applied to the knapsack problem. Journal of University Of Shanghai for Science And Technology, 2004, vol.26, no.6, pp.561-564. (In Chinese)
[9] Ma Liang,Wang Longde. Ant optimization algorithm for knapsack problem, Computer Applications . 2001, 21(8), pp. 4-5. (In Chinese)
[10] Yu Yongxin,Zhang Xinrong. Optimization algorithm for multiple-choice knapsack problem based on ant colony system. Computer engineering, 2003, 29(20), pp.75-76, 84. (In Chinese)
[11] Gao Shang, Yang Jingyu. Solving Knapsack Problem by Hybrid Particle Swarm Optimization Algorithm. Engineering science,2006,.8(11), pp. 94-98. (In Chinese)
[12] Ye Lian, Zhang Min. Solution to the 0-1 knapsack problem based on DNA encoding and computing method. Journal of Computers, 2013, 8(3), p 669-675.
[13] Tan Yan-Yan, Jiao Yong-Chang. MOEA/D with uniform design for solving multiobjective knapsack problems. Journal of Computers, 2013, 8(2),pp.302-307.
[14] Shumeet Baluja. Population based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical Report, No. CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, Pennsylvania, 1994.
[15] Zhou shude, Sun zengqi. A Survey on Estimation of Distribution Algorithm. Acta Automatica Ainica, 2007,33(2), pp.113-124. (In Chinese)
[16] H. Muhliebe, G. Paass. From recombination of genes to the estimation of distributions I. binary parameters. In Lecture notes in computer science. Berlin, Germany: Springer Verlag, 1996 ,vol.1141, pp.178-187.
[17] M. Pelikan, D. E. Godberg, E. C. Paz. Linkage problem, distribution estimation, and Bayesian networks. Evolutionary Computation. 2000, 8(3), pp.311-340.
[18] T K Paul, H. Iba. Linear and combinatorial optimizations by estimation of distribution algorithms. In 9th MPS Symposium on Evolutionary Computation, IPSJ, Japan, 2002.
[19] Haina Rong, Yuquan Li. A Novel Estimation of Distribution Algorithm with Multiple Probability Models. AISS: Advances in Information Sciences and Service Sciences, 4(17), pp. 308- 315, 2012.
[20] Guolin Yu. Multi-objective estimation of Estimation of Distribution Algorithm based on the Simulated binary Crossover. JCIT: Journal of Convergence Information Technology, 7(3), pp. 110-116, 2012.
[21] Rui Zhang. A Rule-Based Estimation of Distribution Algorithm for Solving Job Shop. JCIT: Journal of Convergence Information Technology, 6(8), pp. 220-227, 2011.
[22] Rong Haina, Cheng Jixiang, Li Yuquan. Radar emitter signal analysis with estimation of distribution algorithms. Journal of Networks, 2013, 8(1), p 108-115.

Shang Gao was born in 1972, and received his M.S. degree in 1996 and Ph.D degree in 2006. He now works in school of computer science and technology, Jiangsu University of Science and Technology. He is a professor and He is engage mainly in systems engineering and soft computing.

Ling Qiu was born in 1980, and received his M.S. degree in mathematics in 2008. She now works in Artificial Intelligence of Key Laboratory of Sichuan Province, Sichuan University of Science and Engineering. She is engage mainly in soft computing.

Cungen Cao was born in 1964, and received his M.S. degree in 1989 and Ph.D. degree in 1993 both in mathematics from the Institute of Mathematics, the Chinese Academy of Sciences. Now he is a professor of the Institute of Computing Technology, the Chinese Academy of Sciences. His research area is large scale knowledge processing.