LIU Wen[1,2]

The School of Computer Science and Technology Dalian University of Technology Dalian,China.(1)
Dept.of Electrical Engineering,Institute of XinJinag Mechano-Electrical Vocational and Technical,Urumqi,China.(2)

Hybrid Particle Swarm Algorithm for Solving Multidimensional Knapsack Problem

Abstract: *In order to effectively solve combinatorial optimization problems, the Estimation of Distribution Algorithm (EDA) and Particle Swarm Optimization (PSO) combine to form a new ED-PSO hybrid algorithm, the algorithm can effectively apply global statistical information and global optimal solution to the solution space search. This algorithm is used to solve the Multidimensional Knapsack Problem (MKP). Experimental results show that when solving multidimensional knapsack problem, ED-PSO algorithm is superior to traditional PSO algorithm, and also better than many heuristic intelligent algorithm. Meanwhile, ED-PSO algorithm uses fewer parameters, and therefore easier to be implemented, and run more stable.*

Streszczenie. *W artykule przedstawiono wykorzystanie algorytmu hybrydowego ED-PSO do rozwiązania wielowymiarowego problem Knapsacka (ang. MKP). Zastosowano tu optymalizację roju cząstek (ang. PSO) oraz algorytmu estymacji EDA. Wyniki eksperymentalne pokazują, że w przypadku MKP proponowany algorytm wykazuje znacznie lepsze możliwości niż klasyczny PSO. Dodatkowo ED-PSO ma mniej parametrów, przez co jest łatwiejszy w implementacji. (**Hybrydowy algorytm roju cząstek w rozwiązywaniu wielowymiarowego problemu Knapsacka**).*

Keywords: Hybird Particle Swarm Optimization; Estimation of Distribution Algorithm; Optimization; Multidimensional Knapsack Problem
Słowa kluczowe: Hybrydowa optymalizacja roju cząstek, algorytm estymacji rozkładu, wielowymiarowy problem Knapsacka.

Introduction

Particle Swarm Optimization (PSO) is based on the iterative optimization algorithm, an optimization algorithm based on swarm intelligence developed in 1995 by Kennedy, Eberhart and etc., behavioral studies derived from artificial life and birds' predation[1]. Due to fast convergence and low parameter settings of this algorithm, it has received extensive academic attention in recent years, and has become an important optimization tool, and has been widely applied in function optimization, neural network training, pattern classification and other engineering fields. Standard PSO algorithm principle is that each particle continuously learns individual experience and the best individual experience of groups; the idea of Estimation of Distribution Algorithm is to generate new solutions based on information probability distribution model of current quality solutions information in each iteration [1,2]. In this paper, these two algorithms are combined to form ED-PSO hybrid algorithm, and using this algorithm to solve the Multidimensional Knapsack Problem (MKP). And prove the validity of solving MKP problem using ED-PSO algorithm by means of experiments. Simulation results show that ED-PSO algorithm has stronger solving abilities than traditional EDA algorithms, PSO algorithm, and a heuristic algorithm.

Particle Swarm Optimization

The basis of PSO algorithm is developed through people's observation of animal social behavior. The initial version of PSO is formed by adding neighboring speed matching, eliminating unnecessary variables, and considering multi-dimensional search as well as acceleration based on distance.

(1) $$v_{id} = wv_{id} + c_1 rand()(p_{id} - x_{id}) + c_2 Rand()(p_{gd} - x_{id})$$

(2) $$x_{id} = x_{id} + v$$

w is the inertia weight,c1 c2 are accelerate constants, rand() and Rand() are two random functions changing in the range of [0,1][1]

Estimation of Distribution Algorithm (EDA) Based on different probability model

Estimation of Distribution Algorithms is an evolutionary algorithm based on probabilistic models[2]. It can describe relationship between variables using probabilistic model, so problems that are difficult to be solved using traditional genetic algorithm can be solved. In the field of EDA research, the simplest case is that there is no relationship between variables. In this case, distribution of solutions can be generally represented by a simple probability vector. Set the problem to be solved as an n-dimensional problem, each variable is a binary value, variable independence makes the probability of any solution can be expressed as

$$P(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} P(x_i)$$

ED-PSO hybrid algorithm

In this algorithm, define a particle swarm, N is the total number of particles in the space, denoted by $X_i(t) = (x_{i1}(t), \ldots x_{id}(t), \ldots, x_{iD}(t)), x_{id}(t) \in \{0,1\}$ indicating the value of i-th particle in the t-th generation, D-th dimension, $x_{id}(t)$ as a candidate solution. When algorithm is working, firstly identify UMD model in space of particle swarm, then base on collected advantage solution set to build a distribution estimated space. UMD uses probability vector $P = (p_1, \ldots, p_d, \ldots, p_D)$ to build estimation of distribution space in the search space, which p_d represents a probability when 1 appears in d-dimension. The search method of probability vector $P = (p_1, \ldots, p_d, \ldots, p_D)$ to guide particles in 0-1 solution space is as follows (i.e. guide to generate particle of next generation) [3]:

rand() < β If < p_d , set $x_{id}(t+1) = 1$, otherwise set $x_{id}(t+1) = 0$; Otherwise $x_{id}(t+1) = gbest_d(t)$.

where β is control parameter used to control probability vector P to randomly grow.Probability vector is generated according to the following rules[4]:

(3) $$p_d = \frac{\sum_{i=1}^{N} pbest_{id}}{N}$$

where p_d is the probability of value 1 appearing in d-dimension. Probability vector P will be updated according to the case of solution in each generation. In order to maintain the diversity of particles and keep particles effective convergence, learning parameter λ can be set, and generating rules of probability vector P are as follows:

(4) $$p_d = (1-\lambda) p_d + \lambda \frac{\sum_{i=1}^{n} pbest_{id}}{N}$$

where $\lambda \in (0,1]$ is learning parameter. ED-PSO algorithm process is shown in Fig.1.

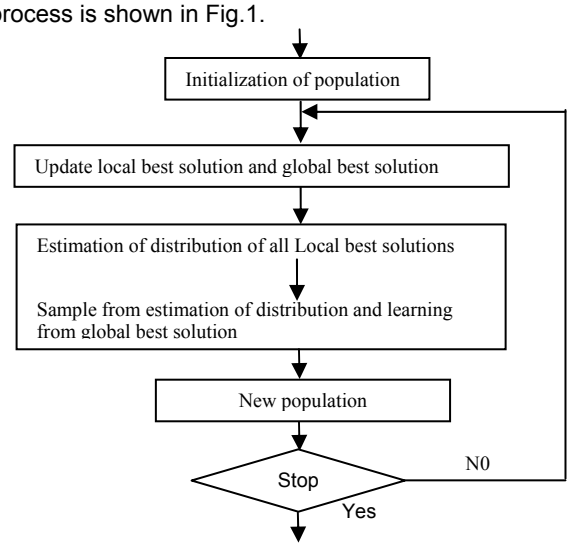

Fig.1. ED-PSO algorithm flowchart

It can be seen from Figure.1. that ED-PSO algorithm is different from traditional EDA. Traditional EDA algorithm obtains global statistics from previous search results, and builds a probabilistic model using global statistics. The ED-PSO inherited this characteristic of traditional EDA algorithm. In traditional EDA algorithm, global optimal solution is not used directly in subsequent search, while in ED-PSO, the global optimal solution is directly used. Therefore, particles can not only access to other particles historical optimal solution, but also get optimal solution in the group. Traditional PSO algorithm does not have such advantages.

Multidimensional Knapsack Problem

Multidimensional Knapsack Problem is the expansion of 0-1 knapsack problem. Assuming there are n items, the value of item j is $p_j > 0$, and there are m resources and constraint $b_i > 0$, the amount of resources consumed i of term j is $w_{ij} \geq 0$. Variable x_j is 0 means item j not putting into package, x_j is 1 means item j putting into package. Multidimensional Knapsack Problem (MKP) can be described by the following expression:

$$\text{(5)} \quad Maximize \sum_{j=1}^{n} p_j x_j$$

$$\text{(6)} \quad Subject \quad to \sum_{j=1}^{n} w_{ij} x_j \leq b_j, i = 1, \ldots, m$$

$$\text{(7)} \quad x_j \in (0,1),\ j = 1, \ldots, n$$

Multidimensional Knapsack Problem (MKP) is a combination NP-hard problem. Theoretically combination used can be enumerated to solve, but there will be a combinatorial explosion problem. Therefore, it is necessary to reduce search space through computing evolutionary methods in order to let Multidimensional Knapsack Problem get optimal solution in limited time and space.

ED-PSO applied to Multidimensional Knapsack Problem

Implementation Methods Set N objects and M backpacks in MKP, each backpack has a maximum capacity $c_j (j = 1, \ldots, m)$. Load conditions of backpack are represented by a vector $x_i (i = 1, \ldots, n)$. Load the object when x_i is 1, not load the object when x_i is 0. Each object has a value $p_i (i = 1, \ldots, n)$ and a weight w_{ij}. The goal of MKP is to maximize total value of objects in backpack in the case of not exceeding the backpack capacity.

Therefore, when implemented, there is a vector number [DIMENSION] in each particle's data structure representing the object selection. In this way, each particle is a solution of the knapsack problem. The length of vector number[DIMENSION] equals to the number of objects N to be equivalent to search in N-dimension space. Vector shown as (8):

$$\text{(8)} \quad number = [x_1, x_2, \ldots, x_n]$$

However, this search may produce infeasible solutions that solution may be in violation of the knapsack constraints, shaped like:

$$\text{(9)} \quad \sum_{j=1}^{n} w_{ij} number[j] > c_j, i = 1, \ldots, m$$

In this case, infeasible solutions should be repaired. Repairing method is according to pseudo-utility ratios u, and use greedy algorithm to repair the infeasible solutions. Pseudo utility ratio is similar to the value density of single knapsack problem. In a single knapsack problem the value of density of Item j is equal to the ratio of item value p_j and resources that item occupied w_j, p_j / w_j .The greater the value density, the better the item, the greater the possibility to be selected. For Multidimensional Knapsack Problem, the most simple pseudo utility ratio is derived from the value density of the single knapsack problem, i.e. the ratio of value of the item j, and the sum of resources item occupied according to different constraints $\sum_{i=1}^{m} r_{ij}$, $p_j / \sum_{i=1}^{m} r_{ij}$ .Repairing operator is as follows:

$$\text{(10)} \quad u_j = \frac{p_j}{\sum_{i=1}^{m} r_i w_{ij}} \qquad \text{(11)} \quad r_j = \frac{\sum_{j=1}^{n} w_{ij} - c_i}{\sum_{j=1}^{n} w_{ij}}$$

After u is obtained, u is sorted in descending sequence. Repairing operator considers whether to remove the item or join it in on the basis of the value of u. First operation (remove): If the new individual is an infeasible solution or the same individual is existed in the current population, then remove items with smallest value from the package. If the individual is an infeasible solution or the same individual is existed in the current population, then repeat this operation until it is a feasible solution and no same individual is in current population. Second operation (join): after getting the feasible solution from first step, consider all the items that are not in the package according to u value in descending sequence. Looking for the existence of such items that the individual is still a feasible solution and there is no same individual in current population when it is added in package. If existed, then add it in.

Fitness Function and Parameter Setting Due to the special nature of the knapsack problem, define the fitness function as $f(X) = \sum_{i=1}^{n} p_i x_i$, where p_t is the value of each item. When $x_i = 1$ the item is selected, and $x_i = 0$ shows that the item is not selected. The choice of backpack satisfies $\sum_{i=1}^{n} w_i x_i \leq C$, where w_i is the weight of each item,

C is the upper limit of each resource. The randomly selected parameter β is set to be 0.95 [5] in the experiment.

UMD Algorithm UMD algorithm probability model is used in the experiment. The UMD algorithm is proposed by German scholar Muhlenbein in 1996. The UMD algorithm process is described as follows [3,6].

Operation and Test

Test Cases The test case selected more than 270 multidimensional knapsack problems provided by Beasley and Chu. Constraint number DIMENSION includes 5, 10 and 30, the Item Quantity BAGNUMBER includes the 100, 250 and 500, each group DIMENSION-BAGNUMBER has 30 issues. The generating methods of these 270 cases are as follows:

The amount of resource i consumed by item j, is an integer uniformly distributed in the interval (0, 1000). For every combination of DIMENSION-BAGNUMBER, each resource constraints $b_i = \alpha \sum_{j=1}^{n} w_{ij}$, α is the issue close ratio, α of first ten issues is 0.25, α of next ten issues is 0.50, and α of last ten issues is 0.75. The item value p has relationship with, $p_j = \sum_{i=1}^{m} w_{ij} / m + 500 q_i$, $j = 1, ..., n$. q_i is a randomly generated real number in the range (0,1).

Calculation Results and Analysis Since the optimal solutions of many problems are not known, the value of 100(relaxation LP optimal solution - optimal solution derived) / (relaxation LP optimal solution) is used to assess solutions, denoted as % gap. Obviously, the smaller the % gap, the closer the solution approaching optimal solution.

ED-PSO Calculation Time Statistics When testing, the size of particle swarm POPSIZE is set to be 50, randomly run 50 times, each time 500 generations is run. Table 1. is come up depending on the computing time of MKP by different DIMENSION, BAGNUMBER and α . From Table 1., the optimal solution of MKP can be found efficiently by using ED-PSO algorithm. With the increasing size of problem, advantages of ED-PSO algorithm are manifested better.

Where DMENSION is the number of items, BAGNUMBER is the number of backpack, that is, the number of constraints. α is the problem close ratio, a.z.t is the average time spent in ED-PSO finding the best solution, a.s.t is the average time in ED-PSO calculating, other form parameters are the same as Table 1.

Table 1. MKP Calculation Time Tables (unit: seconds)

Problem			a.z.t	a.s.t
DIMENSION	BAGNUMBER	α		
5	100	0.25	7.21	132.77
		0.50	11.35	130.73
		0.75	4.27	122.34
5	250	0.25	116.82	281.88
		0.50	166.93	278.68
		0.75	142.31	301.20
10	100	0.25	16.17	118.97
		0.50	22.21	112.86
		0.75	7.69	124.34
5	500	0.25	437.11	567.13
		0.50	303.19	491.49
		0.75	367.55	619.86
10	250	0.25	138.52	283.87
		0.50	137.32	286.30
		0.75	135.36	273.81
10	500	0.25	292.00	527.55
		0.50	279.38	431.66
		0.75	334.12	509.03
30	100	0.25	32.92	132.79
		0.50	43.50	160.10
		0.75	22.45	141.25
30	250	0.25	164.45	323.35
		0.50	207.43	331.13
		0.75	110.25	334.17
30	500	0.25	395.86	557.04
		0.50	380.26	607.81
		0.75	387.51	612.07
Total Average Value			172.84	325.93

ED-PSO and PSO

Table 2. compares results of ED-PSO algorithm and PSO algorithm.As can be seen from Table 2., cases with number of items n less than 500 are solved, ED-PSO is slightly better than PSO. Since case 5-100 is of relatively small size, ED-PSO and PSO both can find all the optimal solutions; for cases 5-250 and 10-100, ED-PSO find more optimal solutions than PSO, superior to PSO; for other three cases, although it cannot be determined that what ED-PSO find is optimal solution, but it can be seen from the comparison of % gap, %gap of ED-PSO is less than %gap of PSO, means ED-PSO solution is closer to the optimal solution. For solving the three cases with number of items n is 500, ED-PSO is inferior to PSO, especially case 30-500, the gap between ED-PSO and PSO is relatively larger than other cases, for case 5-500 the average % gap of ED-PSO is slightly higher than PSO, but for case 10-500, average % gap of ED-PSO and PSO are the same.

In summary, ED-PSO is slightly superior to PSO in solving large-scale knapsack problem, but for ultra-large-scale knapsack problem, the performance of MKPGA is not as good as PSO. Though average %gap of ED-PSO is 0.002 larger than that of PSO, in some cases solving, ED-PSO can find better solutions than PSO, just slightly inferior to PSO in solving cases with 500 item numbers. It is can be said that the overall performance of ED-PSO is basically the same as that of PSO. n/k represents solutions which cannot be determined be the optimal solutions (the following the same).

Table2: Comparison Table of ED-PSO and PSO Calculation Results

Problem			**PSO**		**ED-PSO**	
Size	NO. of packages	α	Average %gap	NO. of Optimal Solutions	Average %gap	NO. of Optimal Solutions
5	100	0.25	0.979	10	0.990	10
		0.50	0.451	10	0.452	10
		0.75	0.358	10	0.319	10
	Average		0.596		0.587	
5	250	0.25	0.227	8	0.226	9
		0.50	0.126	5	0.113	6
		0.75	0.089	5	0.077	9
	Average		0.141		0.138	
10	100	0.25	1.562	10	1.563	10
		0.50	0.755	9	0.791	10
		0.75	0.482	10	0.482	10
	Average		0.956		0.946	
5	500	0.25	0.084	n/k	0.092	n/k
		0.50	0.042	n/k	0.043	n/k
		0.75	0.024	n/k	0.027	n/k

	Average		0.051		0.054	
10	250	0.25	0.501	n/k	0.491	n/k
		0.50	0.252	n/k	0.237	n/k
		0.75	0.153	n/k	0.149	n/k
	Average		0.301		0.293	
10	500	0.25	0.228	n/k	0.225	n/k
		0.50	0.105	n/k	0.111	n/k
		0.75	0.073	n/k	0.069	n/k
	Average		0.136		0.136	
30	100	0.25	2.915	n/k	2.914	n/k
		0.50	1.342	n/k	1.331	n/k
		0.75	0.829	n/k	0.829	n/k
	Average		1.696		1.692	
30	250	0.25	1.179	n/k	1.166	n/k
		0.50	0.521	n/k	0.518	n/k
		0.75	0.305	n/k	0.302	n/k
	Average		0.669		0.663	
30	500	0.25	0.613	n/k	0.689	n/k
		0.50	0.264	n/k	0.299	n/k
		0.75	0.166	n/k	0.174	n/k
	Average		0.348		0.388	
	Total Average		0.542		0.545	

Comparing ED-PSO with Other Heuristic Algorithms

Table 3. is a comparison of ED-PSO with other heuristic algorithms, where M&Q (Magazine and Oguz), V&Z (Volgenant and Zoon) [7,8,9].

Table 3. Comparison Table of ED-PSO with Other Heuristic Algorithm Calculation Results

DIMENSION	BAGNUMBER	α	M&O	V&Z	ED-
5	100	0.25	13.69	10.30	0.989
		0.50	6.71	6.90	0.451
		0.75	5.11	5.68	0.318
	Average		8.50	7.63	0.586
5	250	0.25	6.64	5.85	0.225
		0.50	5.22	4.40	0.113
		0.75	3.56	3.59	0.077
	Average		5.14	4.61	0.138
5	500	0.25	4.39	4.11	0.092
		0.50	2.96	2.53	0.043
		0.75	2.31	2.41	0.027
	Average		3.40	3.02	0.054
10	100	0.25	15.88	15.55	1.562
		0.50	10.41	10.72	0.791
		0.75	6.07	5.67	0.482
	Average		10.79	10.65	0.945
10	250	0.25	11.73	10.53	0.491
		0.50	6.83	5.92	0.237
		0.75	4.42	3.77	0.149
	Average		7.66	6.74	0.292

From Table 3., ED-PSO algorithm is more efficient than other heuristic algorithms in solving the Multidimensional Knapsack Problem.

Conclusion

The ED-PSO algorithm used in experiment effectively combine Estimation of Distribution Algorithm and Particle Swarm Optimization to form a hybrid discrete particle swarm algorithm (ED-PSO). This algorithm can effectively solve combinatorial optimization problems. ED-PSO algorithm can combine global statistics and global optimal solution well to effectively solve the NP-hard problem. Use ED-PSO algorithm to solve the Multidimensional Knapsack Problem, in order to test the ability to solve problems of ED-PSO algorithm. The test results show that the ability to solve problems of ED-PSO algorithm is superior to that of traditional EDA algorithms, PSO algorithm, as well as heuristic algorithms. Meanwhile, ED-PSO algorithm using fewer controlling parameters makes the algorithm more easily to be achieved and operate more stably. The ED-PSO algorithm has room for improvement, such as: use efficient and ordered probability model to build distributed solution space, as well as use this algorithm to solve more difficult combinatorial optimization problems.

Acknowledgements

This work was supported by Scientific Research Program of the Higher Education Institution of XinJiang(No. XJEDU2010S48).

REFERENCES

[1] Kennedy, J., Eberhart, RC: Particle Swarm Optimization. Proceedings of IEEE International Conference on Neural Networks. NJ: Piscataway, (1995) 1942-1948Goguen JA, Parameterized Programming.*IEEE Transactions on Software Engineering*,10(5):528-543,1984.

[2] Zhang Yan etc., Overview of Particle Swarm Optimization and Its Improved Form, *Computer Engineering and Application*, 2005,41(2):62-63.

[3] Zhou Shude etc., Overview of Estimation of Distribution Algorithms, *AAS*, 2007,33(2):113-124.

[4] Fernanda Hembecker, Particle Swarm Optimization for the Multidimensional Knapsack Problem, *Federal University of Technology Parana (UTFPR)*, 2001

[5] Pelikan, M., Goldberg, DE, Lobo, F.: A Survey Of Optimization by Building and Using Probabilistic Models.*IlliGAL Technical Report 99018*,1999

[6] Cai Zixing, Artificial Intelligence and Its Applications: Graduate Book (third edition), *Tsinghua University Press*, 2005

[7] Ma Shaoping, Artificial Iintelligence, *Tsinghua University Press*, 2006

[8] PCChu and JEBeasley, A Genetic Algorithm for the Multidimensional Knapsack Problem, *Journal of Heuristics*, 1999,4:63-86

[9] Krzysztof Socha, Marco Dorigo. Ant colony optimization for continuous domains. *European Journal of Operational Research* (S0377-2217). 2008, 185(3): 1155-1173.

Authors

LIU Wen, since 2011, he is a PhD student in the School of Computer Science and Technology Dalian University of Technology,mainly researching in the field of Computer Application Technology.

Address: Dept.of Electrical Engineering,Institute of XinJinag Mechano-Electrical Vocational and Technical,Tianjin Road, No. 176, Urumqi,China 830011.E-mail:627952@qq.com.

TEL:86 13899918616.