Hybrid Estimation of Distribution Algorithm for Multiobjective Knapsack Problem

Hui Li, Qingfu Zhang, Edward Tsang, and John A. Ford

Department of Computer Science, University of Essex
Wivenhoe Park, Colchester CO4 3SQ, United Kingdom
{hlil,qzhang,edward,fordj}@essex.ac.uk

Abstract. We propose a hybrid estimation of distribution algorithm (MOHEDA) for solving the multiobjective 0/1 knapsack problem (MOKP). Local search based on weighted sum method is proposed, and random repair method (RRM) is used to handle the constraints. Moreover, for the purpose of diversity preservation, a new and fast clustering method, called stochastic clustering method (SCM), is also introduced for mixture-based modelling. The experimental results indicate that MOHEDA outperforms several other state-of-the-art algorithms.

1 Introduction

Over the past twenty years, numerous multiobjective evolutionary algorithms (MOEAs) have been proposed for multiobjective optimization problems (MOPs) [2]. Compared with classical methods, MOEAs are more suitable for solving MOPs for the following reasons: (i) multiple solutions can be found in a single run of a MOEA; (ii) a good spread of the nondominated solutions can be reached; and (iii) a MOEA is less susceptible to the shape or continuity of the Pareto-optimal front. Due to the conflicting relationships between objectives, it is unlikely to find such a solution that optimizes all objectives simultaneously. In practice, it is a hard task to find all nondominated solutions when, as is often the case, the number of Pareto-optimal solutions is huge or even infinite. Therefore, when applying an evolutionary algorithm to MOPs, two major issues should be addressed:

- The resultant nondominated front should be as close to the Pareto-optimal front as possible (convergence);
- The nondominated solutions should be distributed as uniformly (in most cases) and widely along the Pareto-optimal front as possible (diversity).

The performance of MOEAs can be improved by local search. IMMOGLS [5] should be regarded as the first algorithm to hybridize MOEAs with local search. In this algorithm, all weights are generated randomly. Jaszkiewicz (1998) proposed multiobjective genetic local search (MOGLS)[1]. Weighted Tchebycheff scalarizing functions are used to evaluate the fitness value for each individual solution. Local search and selection are implemented on the basis of the current

scalarizing function. In MOGLS, all weights of the scalarizing function are also created in a random way. Without using the linear combination of objectives, Knowles and Corne (2000)[7] proposed a memetic Pareto archived evolutionary strategy (M-PAES), which combines the recombination operators with an evolutionary strategy (PAES). In M-PAES, the acceptance of a new solution generated by genetic search or local search depends not only on the Pareto-dominance relationship but also on the grid-type partition of the objective space.

Estimation of distribution algorithm (EDA) is a new paradigm in evolutionary computation. Recently, few multiobjective EDAs have been proposed and studied. Mixture-Based Iterated Density Estimation Evolutionary Algorithms (MIDEA) was proposed by Thierens and Bosman (2001)[3]. Multiobjective Bayesian Optimization Algorithms (mBOA) was introduced by Khan, Goldberg, and Pelikan (2002)[9]. Population-Based Ant Colony Optimization for MOP was developed by Guntsch1 and Middendorf (2003)[8]. This class of algorithms generalizes genetic algorithms by replacing the crossover and mutation operators with learning and sampling the probability distribution of the best individuals in the previous population.

However, the main goal of MOEAs is to improve the performance both in algorithmic convergence and in diversity preservation. To this end, we propose a hybrid estimation of distribution algorithm for solving the MOKP.

This paper is organized as follows. In section 2, MOPs and MOKP are briefly introduced. Local search is discussed in section 3. Section 4 contains stochastic clustering method. In the following section, MOHEDA is proposed. The experimental results are presented in section 6. In the final section, conclusions are given.

2 Multiobjective Optimization

A general MOP consists of a set of d decision variables, a set of n objectives, and a set of m constraints. Both objectives and constraints are functions of the decision variables. The optimization goal is to

$$\begin{aligned} \text{maximize} \quad & f_i(\mathbf{x}) \qquad i = 1, \ldots, n \\ s.t. \quad & c_j(\mathbf{x}) \leq 0 \qquad j = 1, \ldots, m \end{aligned} \tag{1}$$

where $\mathbf{x} = (x_1, \ldots, x_d)$ is the decision vector in the decision space and $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_n(\mathbf{x}))$ is the objective vector in the objective space.

For any two decision vectors $\mathbf{x^{(1)}}$ and $\mathbf{x^{(2)}}$, $\mathbf{x^{(1)}}$ is said to *dominate* $\mathbf{x^{(2)}}$, denoted by $\mathbf{x^{(1)}} \succ \mathbf{x^{(2)}}$, if $f_k(\mathbf{x^{(1)}}) \geq f_k(\mathbf{x^{(2)}}), \forall k \in \{1, \ldots, n\}$ and there exists at least a $k' \in \{1, \cdots, n\}$ such that $f_{k'}(\mathbf{x^{(1)}}) > f_{k'}(\mathbf{x^{(2)}})$. A decisoin vector x^* is said to be *Pareto-optimal* if there is no any decision vector in the decision space which dominates x^*.

MOKP is a well-known NP-hard multiobjective combinatorial optimization problem, which has been studied by many researchers for testing the performance of MOEAs. It is a generalization of the 0-1 single objective knapsack problem.

Let p_{ij} be the profit of item j associated with knapsack i, ω_{ij} be the weight of item j associated with knapsack i, and c_i be the capacity of knapsack i, the goal is to find a binary vector $\mathbf{x} = (x_1, \ldots, x_d) \in \{0,1\}^d$ such that

$$\sum_{j=1}^{d} \omega_{ij} \cdot x_j \leq c_i, i = 1, 2, \ldots, m \tag{2}$$

and for which $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_n(\mathbf{x}))$ is maximized, where

$$f_i(\mathbf{x}) = \sum_{j=1}^{d} p_{ij} \cdot x_j, i = 1, \cdots, n. \tag{3}$$

3 Local Search

Local search is a simple iterative method for finding good approximate solutions. It should be pointed out that the acceptance function of a neighbour solution in MOPs are slightly different from that in single objective problems. For example, in M-PAES, a reference set is required to evaluate the Pareto-based fitness values of solutions. In this paper, the acceptance function $h(\mathbf{x})$ is defined as the linear combination of all objectives by:

$$h(\mathbf{x}) = \sum_{i=1}^{n} w_i \cdot f_i(\mathbf{x}) \tag{4}$$

where $w_i > 0, i = 1, \ldots, n$, is the weight for the i^{th} objective and $\sum_{i=1}^{n} w_i = 1$.

In IMMOGLS and MOGLS, the weights for the acceptance function are generated randomly. Here, the weights are determined by the start solution $\mathbf{x}$ in the following way. Two extra vectors in the objective space: $(f_{max}^{(1)}, \ldots, f_{max}^{(n)})$ and $(f_{min}^{(1)}, \ldots, f_{min}^{(n)})$, are required. In the context of evolutionary algorithm, $f_{max}^{(k)}$ and $f_{min}^{(k)}$ are the maximization and the minimization of the k^{th} objective in the population. Then, the raw weight w_i' is given by:

$$w_i' = f_i(\mathbf{x}) - f_{min}^{(i)}, i = 1, \ldots, n. \tag{5}$$

Take the order of magnitude into consideration, the unbiased coefficient $e_i, i = 1, \ldots, n$, is introduced as:

$$e_i = \frac{1}{f_{max}^{(i)} - f_{min}^{(i)}}. \tag{6}$$

Therefore, the weights for the acceptance function are:

$$w_i = \frac{w_i' \cdot e_i}{\sum_{j=1}^{n} w_j' \cdot e_j}, i = 1, \ldots, n. \tag{7}$$

The local search procedure for MOKP is described as follows:

Algorithm 1: Local Search

- (1) Select the starting solution $\mathbf{x}$, and generate a set of heuristic weights $w_i, i \in \{1, \ldots, n\}$ according to (7);
- (2) For each item $j \in I = \{n_1, \ldots, n_{|I|}\}$ with $x_j = 0$;
  - Set $x_j := 1$;
  - Repair the modified $\mathbf{x}$;
  - Record the set $\mathcal{I}_j$ of all removed items;
  - Compute $t_j := \sum_{i=1}^{n} w_i \cdot p_{ij} - \sum_{k \in \mathcal{I}_j} \sum_{i=1}^{n} w_i \cdot p_{ik}$;
  - Reset $\mathbf{x}$.
- (3) Sort the series $t_j, j \in \{n_1, \ldots, n_{|I|}\}$ and calculate $j_m := arg\max_{j \in I} t_j$;
- (4) Update $\mathbf{x}$ with $x_{j_m} := 1$ and set $x_k := 0$ for all $k \in \mathcal{I}_{j_m}$;
- (5) If the maximal number of local search steps is reached, stop; Otherwise, goto step 2.

Note that a portion of solutions generated by recombination operators or the move in local search might be infeasible. In this case, some repair strategies should be employed to handle the constraints. Greedy repair method (GRM) is a frequently-used approach. In this paper, we propose random repair method (RRM) for handling constraints and constructing the random neighbourhood. The items will be randomly removed from the knapsacks until the constraints are satisfied. In comparison with GRM, RRM is easy to implement and its computational time is less than that of GRM. We also note that the initial population with higher quality can be created by GRM.

4 Stochastic Clustering Method

As pointed out in [3], the diversity of the Pareto-optimal front can be maintained by the mixture of factorized probability distributions. The population is divided into a couple of clusters using the Euclidean leader algorithm. Each cluster is processed separately and a local probability model is built on its related cluster. In the context of population-based algorithms, each cluster consists of a number of good selected solutions which are currently optimal with respect to a certain weighted sum function. Then, different possible weighted sum functions can be simultaneously optimized in a single generation. In this paper, we propose a new and fast clustering method, called stochastic clustering method (SCM). The process of SCM is mathematically described in the remainder of this section.

Assume that $\prod_{i=1}^{n}[LB_i, UB_i]$ is the domain of a population P in the objective space, where $LB_i = \min_{\mathbf{x} \in P} f_i(\mathbf{x})$ and $UB_i = \max_{\mathbf{x} \in P} f_i(\mathbf{x})$. Suppose that we have a set of subdomains $D_j = \prod_{i=1}^{n}[LB_i^{(j)}, UB_i^{(j)}]$, $j = 1, \ldots, J$, where J is the number of subdomains. A quantitative metric for the objective with the maximal range in the subdomain $D_j, j = 1, \ldots, J$, is defined as:

$$r_j = \max_{i \in \{1, \ldots, n\}} \frac{UB_i^{(j)} - LB_i^{(j)}}{UB_i - LB_i}. \tag{8}$$

The index of the objective with the maximal range in the subdomain D_j is given by:

$$I_j = arg \max_{i \in \{1,\ldots,n\}} \frac{UB_i^{(j)} - LB_i^{(j)}}{UB_i - LB_i}. \tag{9}$$

Then, the K^{th} ($K = arg \max_{j \in \{1,\ldots,J\}} r_j$) subdomain with the maximal metric for the range is decomposed in the following way:

$$\begin{aligned}
LB_i^{(a)} &= LB_i^{(b)} = LB_i^{(K)} \text{ if } i \neq I_K, \\
UB_i^{(a)} &= UB_i^{(b)} = UB_i^{(K)} \text{ if } i \neq I_K, \\
LB_{I_K}^{(a)} &= LB_{I_K}^{(K)}, \\
UB_{I_K}^{(a)} &= LB_{I_K}^{(K)} + (\frac{1}{3} + \alpha)\Delta, \\
LB_{I_K}^{(b)} &= LB_{I_K}^{(K)} + (\frac{2}{3} - \alpha)\Delta, \\
UB_{I_K}^{(b)} &= UB_{I_K}^{(K)}
\end{aligned} \tag{10}$$

where $\Delta = UB_{I_K}^{(K)} - LB_{I_K}^{(K)}$ and α is a random number in $[0, 1]$. In this way, the subdomain D_K can be divided into two nonoverlapped subdomains $D_a = \prod_{i=1}^{n}[LB_i^{(a)}, UB_i^{(a)}]$ and $D_b = \prod_{i=1}^{n}[LB_i^{(b)}, UB_i^{(b)}]$. In SCM, there is no need to calculate the distances between individuals which are crucial for other clustering algorithms. Moreover, subdomains dynamically vary with generations.

For instance, we have two two-dimensional subdomains $D_1 = [0, 10] \times [0, 20]$ and $D_2 = [0, 20] \times [0, 30]$. Here, we assume that e_1 equals to e_2. Then, the metric values of D_1 and D_2 are $20 \cdot e_1$ and $30 \cdot e_2$ respectively. Therefore, we decompose D_2 into $[0, 20] \times [10 + \alpha 10, 30]$ and $[0, 20] \times [0, 10 + \alpha 10]$.

5 Hybrid Estimation of Distribution Algorithm

Estimation of Distribution Algorithms (EDAs) were first introduced by Mühlenbein and Paaß (1996) [6]. The detailed introduction of EDAs can be found in Larrañaga's book [10]. In EDAs, there are neither crossover nor mutation operators. Instead, a new population is sampled from a probability distribution, which is modelled from the selected best solutions. The dependencies among the variables involved can be explicitly and effectively captured and exploited. According to the types of dependencies among variables involved, EDAs can be classified into three categories: without dependencies (UMDA 1998, PBIL 1994, and cGA 1998), bivariate dependencies (MIMIC 1997, COMIT 1997, and BMDA 1999), and multivariate dependencies (EcGA 1999, FDA 1999, EBNA 2000, and BOA 2000).

Recently, hybrid estimation of distribution algorithms have been developed for solving global optimization [11]. To improve the performance of a MOEA both in convergence and in diversity, we propose a mixture-based hybrid estimation of distribution algorithm (MOHEDA). Univariate Marginal Distribution Algorithm (UMDA) is employed. In UMDA, the joint probability distribution of the selected

individuals at each generation, $p(\mathbf{x}, t)$, is factorized as a product of independent univariate marginal distributions $p(x_i, t), i = 1, \ldots, d$. This joint distribution can be written as:

$$p(\mathbf{x}, t) = p(\mathbf{x}|Pop^{Se}(t-1)) = \prod_{i=1}^{d} p(x_i, t) \tag{11}$$

where $Pop^{Se}(t-1)$ is the set of selected individuals from the previous population.

In MOHEDA, an elite population is maintained to store all nondominated solutions found so far. At the beginning of search, this population is empty. It is updated by the local optima obtained in the local search. The mating pool $P_m^{(t)}$ consists of the solutions selected from the elite population or current population. Using SCM, the domain of $P_m^{(t)}$ is divided into n_d subdomains $D_j, j = 1, \ldots, n_d$. The mixture-based probability models can be reformulated by:

$$p^{(k)}(\mathbf{x}, t) = p(\mathbf{x}|Pop^{Se}(k, t)) = \prod_{i=1}^{d} p^{(k)}(x_i, t) \tag{12}$$

where $Pop^{Se}(k, t)$ is the set of mating parents which belong to the k^{th} subdomain D_k. In the following, we summarize the framework of MOHEDA.

Algorithm 2: MOHEDA

- (1) *Initialization:* Generate an initial population $P^{(0)}$ with n_p individuals randomly and create an elite population $P_e^{(0)} := \emptyset$; Perform local search on $P^{(0)}$ and update $P_e^{(0)}$; Set t: =0.
- (2) *Selection:* Select n_p individuals from $P^{(t)} \cup P_e^{(t)}$ and place them in the mating pool $P_m^{(t)}$;
- (3) *Stochastic Clustering:* Decompose the domain of $P_m^{(t)}$ into n_d disjoint subdomains;
- (4) *Mixture-Based Modelling:* Estimate the distribution $p^{(k)}(\mathbf{x}, t)$ on each subdomain D_k, $k = 1, \ldots, n_d$;
- (5) *Sampling:* Sample the new population $P^{(t+1)}$ with n_p individuals according to the distributions $p^{(k)}(\mathbf{x}, t), k = 1, \ldots, n_d$;
- (6) *Local Search:* Select $\frac{n_p}{2}$ individuals from $P^{(t+1)}$, perform local search for each selected individual and update $P_e^{(t+1)}$ with the local optima;
- (7) *Termination:* If the termination criteria are satisfied, then stop. Otherwise, set t:=t+1 and go to step 2.

In MOHEDA, both repair methods are taken into account. In the initialization, we use GRM to generate high quality solutions. Besides, new solutions produced by sampling and local search are repaired by RRM. Note that the size of the elite population will probably be increased since more and more new possible nondominated solutions may be found during the evolution. To maintain the limited size n_e of the elite population, we remove some individuals randomly until the number of elite individuals is less than n_e.

6 Experiments

In this section, the performance metrics of MOEAs are introduced. Some experiments are devised to test the performance of MOHEDA both in convergence and in diversity. The numerical results are briefly discussed.

6.1 Performance Metrics

How to assess the performance of MOEAs is very difficult. Many performance metrics have been proposed and studied for convergence and diversity. In this section, three metrics are briefly discussed.

The coverage metric proposed by Zitzler is used to assess the performance of various MOEAs. The function $\mathcal{C}(A, B)$ maps the ordered algorithm pair (A, B) into a real number in $[0, 1]$. The value $\mathcal{C}(A, B)$ means the percentage of the solutions in A covered by the solutions in B.

In order to evaluate the convergence of MOGLS and MOHEDA, the average values (*front error*) of the minimal, mean and maximal distances between nondominated front and the Pareto-optimal front in 20 runs are calculated. The lower value of the distances, the closer to the Pareto-optimal front the resultant nondominated front is.

Consider not only the standard deviation of distances between nondominated solutions but also the maximal spread, we propose a new metric to evaluate the spread of the set A of nondominated solutions. This metric can be formulated as:

$$D = \frac{\sum_{k=1}^{n}(f_{max}^{(k)} - f_{min}^{(k)})}{\sqrt{\frac{1}{|A|}\sum_{i=1}^{|A|}(d_i - \bar{d})^2}} \tag{13}$$

where d_i is the Euclidean distance between the i^{th} solution and its closest neighbour and $\bar{d}$ is the mean distance. The greater value of the diversity metric, the better the diversity of nondominated solutions is.

6.2 Experimental Settings

The proposed MOHEDA has been compared with MOGLS. Nine test instances including 2, 3 and 4 objectives, associated with 250, 500 and 750 items respectively, are considered. The profits, weights and capacities in our experiments are identical to the data reported in [1][3], which can be downloaded from the link http://www.tik.ee.ethz.ch/∼zitzler/testdata.html/.

In Jaszkiewicz's paper [1], MOGLS has been demonstrated to be the best algorithm for MOKP. MOGLS outperforms several other MOEAs, such as NSGA, SPEA and M-PAES. Thus, we only compared our algorithm with MOGLS. MOHEDA was implemented in C++. All experiments were performed on identical PCs (AMD Athlon 2400MHZ) running Linux.

Three parameters: the size of the population n_p, the number of subdomains n_d and the limited size of the elite population n_e, are given in TABLE I. To be

consistent with the parameter settings of other algorithms , the size of population and the maximal number of objective function evaluations are determined in the same way as SPEA [4] and MOGLS [1]. The search was terminated after completing $n_p \times 500$ objective function evaluations. We performed MOHEDA 20 runs on each of 9 instances.

TABLE I
Parameter Settings

Items	250			500			750		
Parameters	n_p	n_d	n_e	n_p	n_d	n_e	n_p	n_d	n_e
2 Knapsacks	150	15	500	200	20	500	250	25	500
3 Knapsacks	200	20	1000	250	25	1500	300	30	2000
4 Knapsacks	250	25	2500	300	30	3000	350	35	3500

6.3 Experimental Results

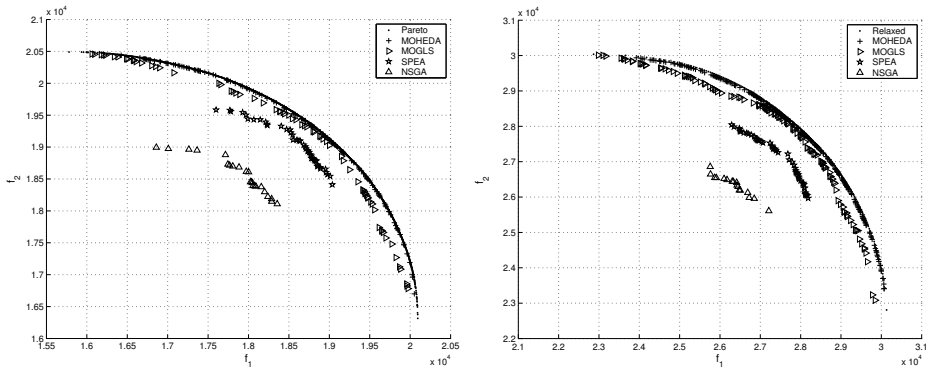

Fig. 1. Nondominated solutions of the instances with 2 knapsacks: 500 items (left), 750 items (right).

The mean coverage metrics of MOHEDA and MOGLS for nine test instances among 20 runs are given in TABLE II. In the case of 2 knapsacks, there are nearly 98% of the solutions of MOGLS covered by the solutions of MOHEDA. It can be observed from Fig.1 that the nondominated solutions of MOHEDA are well-distributed along the true Pareto-optimal front. For the instances with 3 knapsacks, MOHEDA still performs better than MOGLS, and about 65% of nondominated solutions of MOGLS are dominated by the solutions of MOHEDA. MOHEDA is slightly superior to MOGLS for the instances with 4 knapsacks. Nevertheless, the coverage metric $\mathcal{C}$(MOGLS,MOHEDA) decreases as the number of knapsacks increases. This might be caused by the immaturity of the population. If we increase the number of evolutionary generations, a better coverage metric could be obtained.

TABLE II
Coverage Metrics for 9 Test Instances

Mean coverage	$\mathcal{C}$(MOHEDA,MOGLS)			$\mathcal{C}$(MOGLS,MOHEDA)		
Items	250	500	750	250	500	750
2 knapsacks	0.01%	0.00%	0.02%	99.11%	98.61%	97.92%
3 knapsacks	6.05%	2.62%	4.09%	67.20%	67.14%	61.53%
4 knapsacks	3.61%	5.16%	4.92%	30.90%	11.60%	9.13%

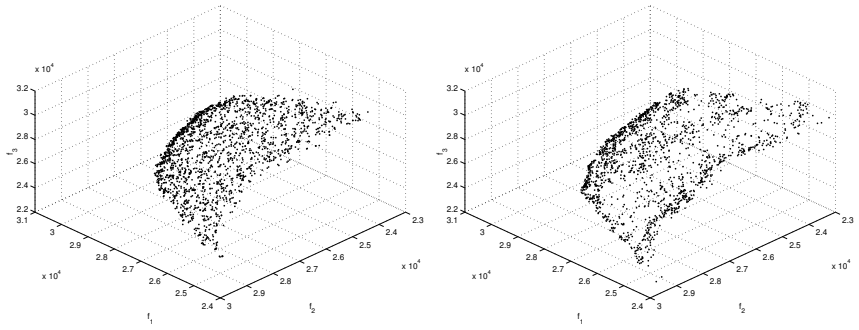

Fig. 2. Nondominated solutions of the instance with 3 knapsacks and 750 items: MOHEDA(left), MOGLS(right).

TABLE III
Convergence and Diversity Metrics

Performance Metrics		Convergence Metric (×100)			Diversity Metric (×100)		
Algorithms	Instances	Minimum	Mean	Maximum	Minimum	Mean	Maximum
MOGLS	(2,250)	0.1571	0.6115	1.1477	1.5270	2.7060	4.0700
	(2,500)	0.3031	1.4220	2.6603	2.6020	4.3010	5.4600
	(2,750)	1.0110	2.4612	4.3301	4.3920	5.5090	7.2640
MOHEDA	(2,250)	0	0.0514	0.2274	4.5620	7.0230	9.1780
	(2,500)	0.0216	0.1745	0.4071	6.4270	10.2030	12.5360
	(2,750)	0.1892	0.5420	1.6539	5.3970	11.0930	14.3690

We visualize the nondominated fronts in the case of two and three objectives. It can be intuitively observed from Fig.1 and Fig.2 that the nondominated fronts of MOHEDA are superior to those of MOGLS in both convergence and diversity. Moreover, the mean quantitative metrics of convergence and diversity for the instances with 2 knapsacks in 20 runs are given in TABLE III. Here, we only give the results in the cases of 2 knapsacks in which the true or relaxed Pareto fronts are known to us. For all nine test instances, the values of the diversity metric of MOHEDA are greater than those of MOGLS. A better spread of

nondominated front is achieved by MOHEDA. Obviously, from the convergence metrics in TABLE III, the nondominated fronts of MOHEDA are closer to the Pareto-optimal front than those of MOGLS.

7 Conclusions

MOHEDA has been compared with MOGLS on MOKP. The experimental results show that MOHEDA outperforms MOGLS both in convergence and in diversity. The following techniques are effectively used in MOHEDA: (1) random repair method is used; (2) random neighbourhood structure is constructed in local search; (3) stochastic clustering method is employed; (4) a mixture-based UMDA is introduced. The distinct feature of MOHEDA is that multiple possible weighted sum functions can be optimized in a single generation. Moreover, the further work may focus on the performance analysis of MOHEDA.

References

1. A. Jaszkiewicz. On the Performance of Multiple-Objective Genetic Local Search on the 0/1 Knapsack Problem-A comparative Experiment, IEEE Transactions on Evolutionary Computation, Vol.6, No.4, August 2002.
2. Coello Coello, Carlos A. An Updated Survey of GA-Based Multiobjective Optimization Techniques. Technical Report Lania-RD-98-08, Laboratorio Nacional de Informática Avanzada (LANIA), Xalapa, Veracruz, M éxico, december 1998.
3. D. Thierens and P. A.N. Bosman. Multi-Objective Mixture-based Iterated Density Estimation Evolutionary Algorithms, Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001) (2001).
4. E. Zitzler and L. Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. IEEE Transactions on Evolutionary Computation, 3(4), pages 257-271, November 1999.
5. H. Ishibuchi and T. Murata. Multi-objective Genetic Local Search Algorithm, in Proceedings of 1996 IEEE International Conference on Evolutionary computation(ICEC'96), iscataway, NJ, May 20-22 1996, IEEE, pp. 119-124.
6. H. Mühlenbein and G. Paaß. From Recombination of Genes to the Estimation of Distributions I. Binary parameters. In Lecture Notes in Computer Science 1411: Parallel Problem Solving from Nature-PPSN IV,178-187, 1996.
7. J. Knowles and D. Corne. M-PAES: A Memetic Algorithm for Multiobjective Optimization, in Proceeding of 2000 Congress on Evolutionary Computation. Piscataway, NJ:IEEE Press, July 2000, vol. 1,pp. 325-332.
8. M. Guntsch and M. Middendorf. Solving Multi-criteria Optimization Problems with Population-Based ACO, EMO 2003, LNCS 2632, pp. 464-478.
9. Laumanns Marco, Ocenásek Jirí. Bayesian Optimization Algorithms for Multi-Objective Optimization, In: Parallel Problem Solving from Nature - PPSN VII, Granada, ES, Springer, 2002, p.298-307, ISBN 3-540-444139-5.
10. P., Larrañaga, and J.A., Lozano. Estimation of Distribution Algorithms: A New Tools for Evolutionary Computation. Kluwer Academic Publishers, 2001.
11. Q. Zhang, J. Sun, E. P. K. Tsang, J. A. Ford. Hybrid Estimation of Distribution Algorithm for Global Optimisation, accepted in Engineering computations, 2003.