




A Penalty-Based MOEA/D for Multiobjective 0/1 Knapsack problems

Ahmed Hassan Alhindi

*Department of Computer Science,
College of Computer & Information Systems,
Umm Al-Qura University
Email: alhindi@uqu.edu.sa*

Access this article online	
Quick Response Code:	Website: www.uqu.edu.sa/jea E-mail: jea@uqu.edu.sa Table of Contents - Current issue: https://uq.sa/CFwcAm
	
© Umm Al-Qura University Journal for E & A, Vol.9 Issue No.2, pp.19-30 April 2019 Under Legal Deposit No. p- ISSN: 1658-4635 / e- ISSN: 1658-8150	

A Penalty-Based MOEA/D for Multiobjective 0/1 Knapsack problems

Ahmed Hassan Alhindi

Department of Computer Science,
College of Computer & Information Systems,
Umm Al-Qura University
Email: alhindi@uqu.edu.sa

Abstract

The multiobjective evolutionary algorithm based on decomposition (MOEA/D) is a generic multiobjective evolutionary optimization algorithm. It decomposes a multiobjective problem into a number of scalar optimization subproblems with a neighbourhood structure and optimizes them simultaneously to approximate the Pareto-optimal set. For effective performance of MOEA/D, scalar optimization subproblem has to be able to escape Pareto local optima. In this paper, a penalty method for different subproblems with penalizing scheme is proposed (PB-MOEA/D) to overcome this shortcoming. Our experimental results on the multiobjective 0/1 knapsack problem test instances show that subproblems with a penalty method and penalizing scheme yield superior performance over implementations without. This paper argues that a strategy for escaping Pareto local optimal solutions is necessary in multiobjective evolutionary algorithms for improving algorithms performance. It also explains why PB-MOEA/D performs better.

Keywords: Decomposition, evolutionary multiobjective optimization, self adaptation, penalty method, guided local search, local optima, genetic algorithm.

Introduction

A multiobjective optimization problem (MOP) can be stated as follow:

$$\begin{aligned} &\text{maximize} && F(x) = (f_1(x), \dots, f_m(x)) \\ &\text{subject to} && x \in \Omega \end{aligned} \tag{1}$$

where x is a potential solution, Ω is the discrete search space and $F(x)$ consists of m scalar objective functions $f_1(x), \dots, f_m(x)$. Very often, an improvement in one objective will cause a degradation of another. No single solution can optimize all the objectives at the same time. A decision maker has to balance these objectives in an optimal way. The concept of Pareto optimality is commonly used to best trade-off solutions.

Given two solutions $x, y \in \Omega$, x is said to dominate y if and only if $f_i(x) \leq f_i(y)$ for

every i and $f_i(x) < f_i(y)$ for at least one index $j \in \{1, \dots, m\}$. x^* is called Pareto optimal to (1) if no other solution can dominate x . The set of all the Pareto optimal solutions is called the Pareto set (PS), the set $\{F(x) | x \in PS\}$ is called the Pareto front (PF) [1].

The goal of multiobjective evolutionary algorithms (MOEAs) is to produce a number of solutions to approximate the PF in a single run [1-3]. Such approximation can be very useful for a decision maker to understand their problem and make their final decision. With Pareto dominance based MOEAs (e.g. [4]) and Hypervolume based MOEAs (e.g. [5]), Decomposition based MOEAs (e.g. MOEA/D [6]) have been widely accepted as a major approach in the area of multiobjective evolutionary computation.

Multiobjective evolutionary algorithm based on decomposition (MOEA/D) [6] is a recent MOEA. Using conventional aggregation approaches, MOEA/D decomposes the approximation of the PF into a number of single objective optimization subproblems. The objective of each subproblem is a weighted aggregation of all objectives in the MOP under consideration. Neighbourhood relations among these subproblems are defined based on the distances among their aggregation weight vectors. Each subproblem is optimized by using information mainly from its neighbouring subproblems. Each single objective subproblem plays a crucial role in MOEA/D.

Arguably, different scalar optimization subproblems are responsible for different part of the PF, therefore, incorporating information collected at different search stages could improve the algorithm performance. When some subproblems trapped in a local optimal region, altering their objective functions by means of penalty (posting constraint) is required for helping these subproblems escape from the trapped region.

Penalty approach has proven to be very efficient and effective for changing the cost function (i.e. modification of the landscape structure) in an online manner to escape from the local optima [7-10]. Some advanced local search methods, such as guided local search (GLS) [7], adopt the penalty approach to escape local optima and obtained good solutions. In GLS, when the search get trapped in a local optimal solutions, the objective function is modified by means of penalties and thus the search can be guided out of the attraction region of this local solution. Using the GLS idea, this paper proposes to use a penalty method for subproblems in MOEA/D and dynamically modifying their cost function based on their previous performances. The resultant algorithm, called PB-MOEA/D, is compared with original MOEA/D proposed in [6] on combinatorial 0/1 Multiobjective Knapsack benchmark problems [11]. Our results indicate that penalty approach improves the performance of MOEA/D significantly.

Review on MOEA/D

There are several variants of MOEA/D. In this paper, we use the original MOEA/D [6] with the weighted sum approach. The MOEA/D requires N evenly spread weight

vectors $\lambda^1, \dots, \lambda^N$ to decompose (1). Each weight vector λ satisfies $\sum_{i=1}^m \lambda_i = 1$ and $\lambda_i \geq 0$ for all $i=1, \dots, m$. Then, the problem of approximation of the PF of (1) can be decomposed into N scalar optimization subproblems and the objective function of the i th maximization subproblem is:

$$\text{maximize } g^{ws}(x|\lambda) = \sum_{i=1}^m \lambda_i f_i(x) \quad (2)$$

During the search, MOEA/D maintains:

- 1- A population of N points $x^1, \dots, x^N \in \Omega$, where x^i is the current solution to the i th subproblem;
- 2- FV^1, \dots, FV^N , where FV^i is the Function value of solution x^i .
- 3- EP , external archive contains all the non-dominated solutions found so far.

At the beginning, MOEA/D initializes each subproblem i with solution x^i weight vector λ^i and the neighbourhood $B(i)$ of its λ^i for $i = 1, \dots, N$, which formed by the indexes of its T closest neighbours in B . The Euclidean distance is used to measure closeness between any (subproblems') weight vectors.

At each generation, MOEA/D does the following:

For each subproblem $i=1, \dots, N$ do

Step 1: Search for Solutions

- (a) Randomly select two indexes l, k from $B(i)$
- (b) Apply genetic operators on x^l, x^k to generate a new solution y , then, repair y if necessary, and compute $F(y)$

Step 2: Update Solutions

- (a) For each index $j \in B(i)$, if $g^{ws}(y|\lambda^j) \leq g^{ws}(x^j|\lambda^j)$, then set $x^j = y$ and $FV^j = F(y)$.
- (b) Remove from EP all the vectors dominated by $F(y)$. Add $F(y)$ to EP if not vectors in EP dominated $F(y)$.

A Penalty Method for MOEA/D

In combinatorial optimization problems, a solution often consists of defined properties that belong to the problem under consideration. Those properties are called features and are used to distinguish between solutions with different characteristics. Note that solutions features were originally introduced in GLS [7]. For clarity, we assume that the search space is $\Omega \in \{0,1\}^n$, although our approach can be generalized to other search spaces. We will give the idea and details of the proposed method for the 0/1 MOKP.

In MOEA/D, a new solution generated for a subproblem is very likely to be better,

but may be is close (in the same basin or plateaus) to its parent since the genetic operator does not utilize any global information gathered during the search. Penalty methods iteratively extract/collect/gather information from the previous search and posing constraints which modify the landscape and guide the search out of local optima and towards promising areas in the search space. For example, if a subproblem reaches a local optimum then an assumption can be made that the global optimum is unlikely to reside in the surrounding area. Constraints could then be introduced that exclude this area from being searched in future iterations. The idea behind the proposed penalty method is to utilize the global information and location information of the features gathered during the search to overcome the shortcoming of original MOEA/D.

In PB-MOEA/D, a set of features is defined to characterise solutions and allow us to constraint them. A feature ($feature_i$) can be any solution property and should has the following component;

- A cost function c_i , usually defined by the objective function.
- A penalty p_i , initially set to 0 and used to penalize occurrences of the feature in local optima. (i.e., indicate its attractiveness)
- An indicator function, $I_i(x)$, indicating whether the feature is present in a solution x or not as follows:

$$I_i(x) \begin{cases} 1, & \text{if } feature_i \text{ presented in } x \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

Since each subproblem has different search direction (i.e., weight vector), we use different feature set for each subproblem. The reason behind using different feature set is that each set of feature for each subproblem has different cost functions and different penalty values.

In PB-MOEA/D, the use of penalties are twofold: Firstly, altering the scalar objective function of subproblems. Secondly, bias genetic operators in order to regularise producing of spring.

Let the search space be $\Omega \in \{0,1\}^n$. In the following, we present the main components of PB-MOEA/D.

Augmented Cost Function

Following the idea of GLS, the PB-MOEA/D replaces the main cost function g^{ws} with an augmented cost function h to guide the search out of local optima. The idea behind it is to make a local optimum most costly than the surrounding search space, where the feature are not present. The scalar optimization problem in PB-MOEA/D

$$h(x) = g^{ws}(x|\lambda) + \delta \times \sum_{i=1} (p_i \times I_i(x)) \quad (4)$$

where x is a candidate solution, g^{ws} is the original scalarizing objective function of the subproblem, i ranges over all the features in FS, p_i is the penalty of feature i , I_i is the indicator function given in Eq.(3), and δ is the parameter that controls the degree of diversity of the search. The higher the value of δ the more divers the search will be. The δ parameter can be dynamically calculated as a function of a local optimum and the average number of features present by the following formula:

$$\delta = \alpha \times \frac{cost(x')}{M} \quad (5)$$

where $cost$ is the objective function of the problem, x' is the local optimum, M is the number of features presented in x' , and α is a tuning parameter $\in [0,1]$.

Guided Reproduction

Beside the augmented cost function, PB-MOEA/D adapts the reproduction cycle of the original MOEA/D [6] to use the information provided by the penalty operator of the features. The idea behind that is to influence the random selection of parents' features by their penalty.

For subproblem i , the guided reproduction of the child $y = (y_1, \dots, y_n) \in \{0,1\}^n$ will be as follow:

- 1- Randomly select two neighbouring subproblems l, k from its neighbourhood $B(i)$.
- 2- For each feature j in $\{1, 2, \dots, n\}$ do

- a. $Sum \leftarrow p_j^l + p_j^k$

- b. Randomly generate an integer $rand$ uniformly from $[0, sum-1]$. Then set

$$y_j = \begin{cases} x^l, & \text{if } rand < p_j^l \\ x^k, & \text{otherwise} \end{cases} \quad (6)$$

- 3- Compute $F(y)$

In the above steps, each feature i in the solution x^l competes against the corresponding feature in x^k for a place in the child. This competition is a weighted random selection, influenced by the penalty p_j^l and p_j^k of the respective features; thus the “lighter” feature will have a greater chance to propagate its information to the child.

Selective Penalty Modifications

During the evolution, when subproblem trapped at a local optimum x' the PB-MOEA/D penalizes (increment the penalty of the feature) its unfavourable/undesirable features. These are all the features present in x' which have maximum utility, $U(x, j)$, which defined as:

$$U(x', j) = \frac{c_j}{1 + p_j} \quad (7)$$

the idea is to penalize features, which have high costs first, such that they can be avoided in future search. In PB-MOEA/D, if a subproblem has not been updated for a specific number of iterations, we conclude that it has trapped in a local optimum.

Structure of PB-MOEA/D

The flow of PB-MOEA/D is presented below.

Step1: Initialization

Step 1.1 Compute the Euclidean distances between any two weight vectors and then find the T closest weight vectors to each weight vector. For each $i=1, \dots, N$, set $B(i) = i_1, \dots, i_T$ where $\lambda^{i_1}, \dots, \lambda^{i_T}$ are the T closest weight vectors to λ^i .

Step 1.2 Generate an initial population x^1, \dots, x^N by uniformly randomly sampling from the search space.

Step 1.3 Set $EP = \phi$.

Step 2: For each subproblem $i=1, \dots, N$ do

Step 2.1 Search for solutions

1. Randomly Select two indexes l, k from $B(i)$
2. Generate a new solution y from x^l and x^k following the steps in Sec 3.2, repair y if necessary and compute $F(y)$.

Step 2.2 Update solutions

1. For each index $j \in B(i)$, if $h(y|\lambda^j) \leq h(x^j|\lambda^j)$, then set $FV^j = F(y)$.
2. Remove from EP all vectors dominated by $F(y)$. Add $F(y)$ to EP if no vectors in EP dominated $F(y)$.

Step 2.3 Penalize Features

1. If the penalization condition is not met, go to Step 2.
2. For each feature j presents in the solution x^i , calculate its utility using Eq.7.
3. Find and penalize the largest utility feature.

Step 3: Termination: if a problem specific stopping condition is met, stop and output EP. Otherwise, go to Step 2.

To search for new solutions, Step 2 uses the augmented cost function h instead of the main cost function g . For generating a new solution, Step 2.1 calls the Guided Reproduction to generate a solution y and the update the neighbouring solutions and external population EP in step 2.2. When the penalization condition is met, Step 2.3 computes the utility for each feature presents in the optimal solution x^i of subproblem i . Then, the feature with the maximum utility is penalize.

4. Experimental Studies

4.1. The 0-1 Multiobjective Knapsack Problem (MOKP)

Given a set of n items and a set of m knapsacks, with

- $p_{ij} \geq 0$ the profit of item j in knapsack i ,
- $w_{ij} \geq 0$ the weight of item j in knapsack i ,
- $c_i =$ capacity of the knapsack i ,

The 0-1 multi-objective knapsack problem (MOKP) can be stated as follows:

$$\text{maximize} \quad f_i(x) = \sum_{j=0}^n p_{ij}x_j, \quad i = 1, \dots, m. \quad (7)$$

$$\text{subject to} \quad \sum_{j=0}^n w_{ij}x_j \leq c_i, \quad i = 1, \dots, m. \quad (8)$$

$$x = (x_1, \dots, x_n)^T \in \{0,1\}^n$$

$x^i = 1$ means that item i is selected and inserted into all the knapsacks. The 0-1 multi-objective knapsack problem is classified as an *NP*-hard problem and can be used to model different forms of applications in resource allocation [1]. When $m=1$, it is reduced to the 0-1 single knapsack problem.

4.2. Experimental Settings

- For generating new solutions, PB-MOEA/D used the guided reproduction operator described in (3.2), while MOEA/D used the conventional reproduction in [6].
- The value of N and H in bother algorithms for each test instance are listed in Table 1.
- Features and their costs in PB-MOEA/D: a possible feature to consider for MOKP could be items. The cost of a feature is the weight-to-profit ration over the all the objectives.

$$\text{i.e., } c_i = \frac{\sum_{j=1}^m \lambda_j \times \left(\frac{w_{ij}}{p_{ij}} \right)}{m}.$$

- Repair method for MOKP: we use the repair method of Jaszkiwicz, which is the same as in [6].
- Stopping condition: both algorithms stop after $N \times 500$ function evaluations.
- Other control parameters in PB-MOEA/D: $\alpha=0.01$ and $T=10$, which is the same for MOEA/D.

Table 1 Parameter settings of the MOEA/D for the nine test instances of the 0-1 knapsack problem. m is the number of knapsacks, and n is the number of items.

Instance		Parameter Setting for	
M	N	H	N
2	250	149	150
2	500	199	200
2	750	249	250
3	250	25	351
3	500	25	351
3	750	25	351
4	250	12	455
4	500	12	455
4	750	12	455

Performance metrics:

The following performance index is used in assessing the performance of the algorithms in out experimental studies.

Inverted Generational Distance (IGD-metric)[14]

The IGD from the Pareto-optimal front PF^* to the non-dominated solutions set P found so far is defined as:

instance		MOEA/D		PB-MOEA/D		h
m	n	mean	stdev	mean	stdev	
2	250	46.85	4.6	25.94	1.74	1
	500	118.55	10.5	39.40	2.58	1
	750	322.15	22.9	86.48	5.84	1
3	250	125.20	8.37	57.57	2.41	1
	500	335.63	19.51	116.94	6.58	1
	750	603.53	27.50	181.40	10.62	1
4	250	200.50	4.64	114.5	2.6	1
	500	521.14	16.47	240.6	5.5	1
	750	967.74	30.75	381.1	7.6	1

Table 2: The IGD statistics of the final approximation obtained by the two methods.

$$IGD(\mathcal{PF}^*, \mathcal{P}) = \frac{\sum_{x \in \mathcal{P}} d(x, \mathcal{P})}{|\mathcal{PF}^*|}, \quad (9) \quad (9)$$

where $d(x, P)$ is the minimum Euclidean distance between x and each element in P . The smaller the value of IGD, the closer the solutions are to the Pareto-optimal front; for example, $IGD=0$ indicates that all the solutions generated are in the Pareto-optimal front. Likewise, the higher the value of IGD obtained, the farther the set of non-dominated solutions is from the Pareto-optimal front.

Results

PB-MOEA/D is tested on the 0/1 multiobjective knapsack problem. The inverted generational distance (IGD) performance measure is used. The IGD measures both the convergence as well as the diversity among the solutions. Both MOEA/D and PB-MOEA/D are implemented in Java.

We conducted a parameter sensitivity investigation of T for PB-MOEA/D using six different values (10, 20, 30, 40, 50, and 60). By observing the mean of IGD values over 30 runs, we can conclude that the PB-MOEA/D is not very sensitive to the setting of T to most of the instances. We should point out that a large value of T could increase the computation overhead in allocation of individual solutions to subproblems.

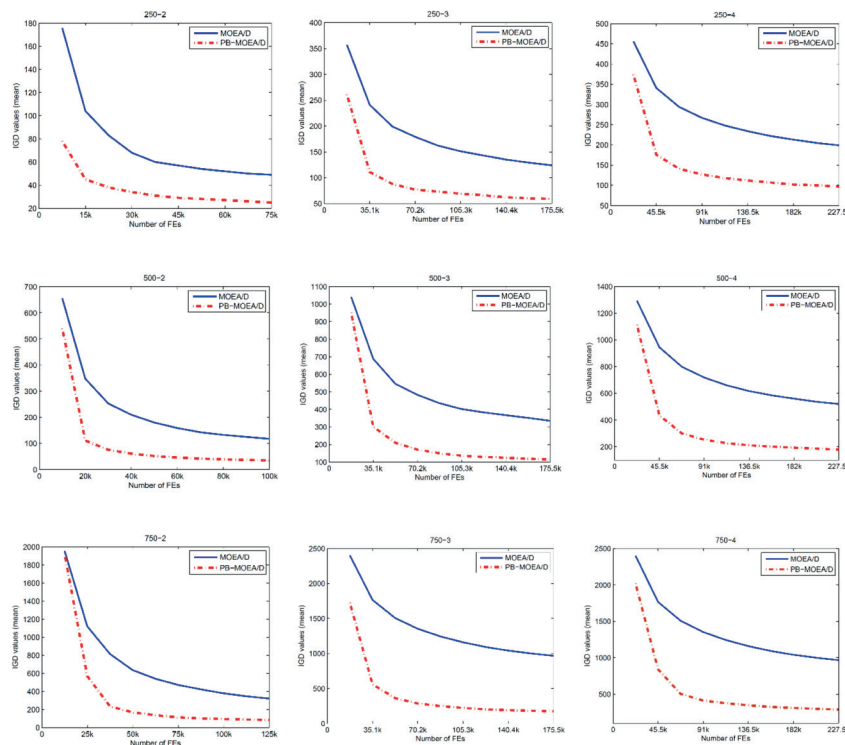
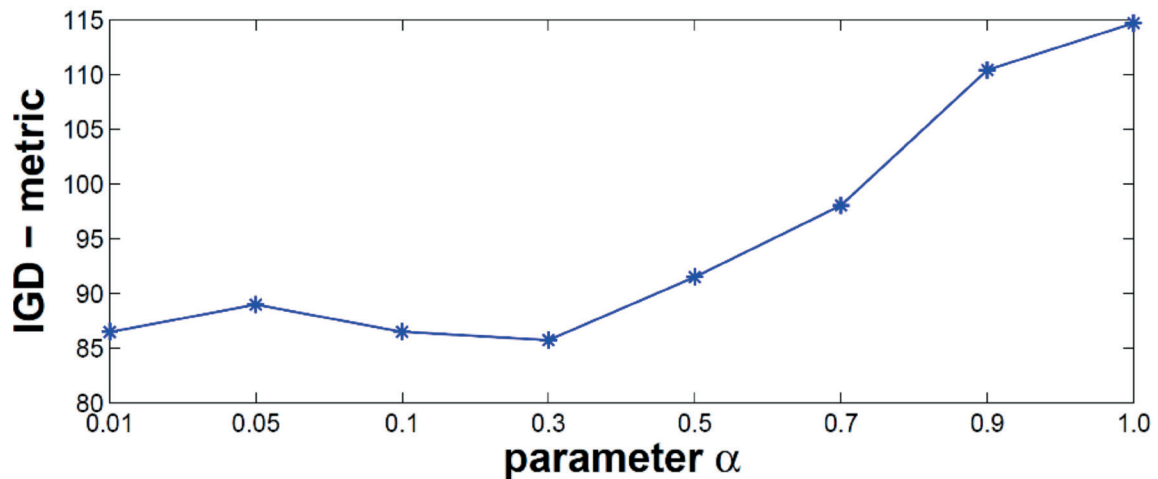


Figure 1: Convergence graphs in terms of mean of IGD obtained by MOEA/D and PB-MOEA/D for all the MOKP instances.

The comparison, in terms of the IGD metric, between the original MOEA/D and the proposed PB-MOEA/D are presented in Figure 1 and Table 2. The t -test at the 5% significance level has conducted to compare the final IGD values obtained by the two methods. Note that $h=1$ in Table 1 means that the difference is significance and $h=0$ implies that the t -test cannot detect a significance difference. It is clear from Table 2 that the quality of the final solutions obtained by PB-MOEA/D is significantly better than the original MOEA/D on all the instances. Figure 1 also indicates that the proposed PB-MOEA/D converges faster than the original MOEA/D on all the instances.



α is a major control parameter in PB-MOEA/D because it provides a means to control the influence of the information on the search process. To study the sensitivity of the performance to α in PB-MOEA/D, we have tested different settings of α in the implementation of PB-MOEA/D for knapsack instance 750-2. All the parameter settings are the same as in section 4.2 except the settings of α . As clearly shown in Figure 2, PB-MOEA/D performs very well with α from 0.01 to 0.3 on knapsack instance 750-2. Figure 2 also reveals that PB-MOEA/D does not work well on the same instance when $\alpha > 0.3$. This is because large α will increase the influence of information which led to solely remove the penalised features from the solution and the information (which may be wrong) will fully determine the search.

Conclusion

In this paper, a penalty approach with a penalizing scheme was integrated with MOEA/D. We have shown that the penalization scheme is necessary for improving the algorithm performance since collecting information were needed during the different search stages. Our experimental results on the multi-objective 0/1 knapsack problem test instances indicated that the proposed PB-MOEA/D outperforms the original MOEA/D.

Acknowledgements

The author would like to thank the Associate Editor and anonymous reviewers for their valuable comments, which greatly improved the quality of this paper.

References

- K. Miettinen, Nonlinear multiobjective optimization. Springer, 1999, vol. 12.
- K. Deb et al., Multi-objective optimization using evolutionary algorithms. John Wiley & Sons Chichester, 2001, vol. 2012.
- C. Coello, G. Lamont, and D. Van Veldhuizen, Evolutionary algorithms for solving multiobjective problems. Springer-Verlag New York Inc, 2007, vol. 5.
- K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," Evolutionary Computation, IEEE Transactions on, vol. 6, no. 2, pp. 182–197, 2002.
- E. Zitzler and S. Kunzli, "Indicator-based selection in multiobjective search," in "Parallel Problem Solving from Nature-PPSN VIII. Springer, 2004, pp. 832–842.
- Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," Evolutionary Computation, IEEE Transactions on, vol. 11, no. 6, pp. 712–731, 2007.
- C. Voudouris, "Guided local search for combinatorial optimisation problems." Ph.D. dissertation, University of Essex, 1997.
- L. T. Leng, "Guided genetic algorithm," Ph.D. dissertation, Doctoral Dissertation. University of Essex, 1999.
- N. Tairan and Q. Zhang, "Population-based guided local search: Some preliminary experimental results," in Evolutionary Computation (CEC), 2010 IEEE Congress on. IEEE, 2010, pp. 1–5.
- A. Alsheddy and E. E. Tsang, "Guided pareto local search based frameworks for biobjective optimization," in Evolutionary Computation (CEC), 2010 IEEE Congress on. IEEE, 2010, pp. 1–8.
- E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach," Evolutionary Computation, IEEE Transactions on, vol. 3, no. 4, pp. 257–271, 1999.
- H. Ishibuchi, Y. Tanigaki, N. Akedo, and Y. Nojima, "How to strike a balance between local search and global search in multiobjective memetic algorithms for multiobjective 0/1 knapsack problems," in Evolutionary Computation (CEC), 2013 IEEE Congress on. IEEE, 2013, pp. 1643–1650.
- Y.-y. Tan, Y.-c. Jiao, H. Li, and X.-k. Wang, "Moea/d+ uniform design: A new version of moea/d for optimization problems with many objectives," Computers & Operations Research, vol. 40, no. 6, pp. 1648–1660, 2013

Received: 04/06/2018

Accepted: 31/01/2019

دمج دالة الغرامة في الخوارزميات التطورية متعددة الأهداف المعتمدة على التقسيم

أحمد حسن الهندي

أستاذ مساعد بقسم علوم الحاسب الآلي - كلية الحاسب الآلي ونظم المعلومات - جامعة أم القرى

Email: alhindi@uqu.edu.sa

الملخص:

الخوارزميات التطورية تعتمد في طريقة عملها على مبدأ التزاوج المعروف في علم الأحياء. هذا النوع من الخوارزميات غالباً ما يقوم بأداء جيد في حل المسائل أو المشاكل البحثية التي تهدف لإيجاد أفضل حل من خلال المفاضلة بين الخيارات المتاحة للمشكلة. سهولة وصعوبة المفاضلة بين البدائل و البحث عنها تعتمد على طبيعة المشكلة المراد حلها. خوارزميات البحث التطورية تواجه صعوبة في البحث عن الحلول حيث أنها قد تعلق في النقاط الحرجة والتي تعرف في الرياضيات بالمناطق الصغرى أو العظمى. من أجل التغلب على هذه المشكلة هذا البحث يهدف إلى تحسين عمل إحدى الخوارزميات التطورية والتي تعرف باسم الخوارزميات التطورية متعددة الأهداف و المعتمدة على التقسيم، وذلك عن طريق استخدام إحدى الطرق الرياضية والتي تعرف بدالة الغرامة. ولدراسة جدوى هذا التحسين فقد تم اختبار هذه الخوارزمية على إحدى المشاكل المعروفة في هذا المجال وتعرف باسم مشكلة الحقيبة. نتائج التجارب العملية أظهرت تحسن ملحوظ في أداء الخوارزمية.