Hybrid Metaheuristics based on MOEA/D for 0/1 Multiobjective Knapsack Problems: A comparative study

Ahmed Kafafy*, Ahmed Bounekkar†, Stéphane Bonnevay‡
Laboratoire ERIC - Université Claude Bernard Lyon 1,
Ecole Polytechnique Universitaire, 15 Boulevard Latarjet,
69622 Villeurbanne cedex, France
*ahmedkafafy80@gmail.com, †bounekkar@univ-lyon1.fr, ‡stephane.bonnevay@univ-lyon1.fr

***Abstract*—Hybrid Metaheuristics aim to incorporate and combine different metaheuristics with each other to enhance the search capabilities. It can improve both of intensification and diversification toward the preferred solutions and concentrates the search efforts to investigate the promising regions in the search space. In this paper, a comparative study was developed to study the effect of the hybridization of different metaheuristics within MOEA/D framework. We study four proposals of hybridization, the first proposal is to combine adaptive discrete differential evolution operator with MOEA/D. The second one is to combine the path-Relinking operator with MOEA/D. the third and the fourth proposals combine both of them in MOEA/D. The comparative study uses a set of MOKSP instances commonly used in the literature to investigate the hybridization effects as well as a set of quality assessment indicators. The experimental results indicate that the proposals are highly competitive for most test instances and can be considered as viable alternatives.**

***Index Terms*—Multiobjective Optimization, Metaheuristics, Evolutionary Algorithm, Differential Evolution, Path-relinking, 0/1 MOKSP.**

I. Introduction

Many of real-world problems can be modeled as multiobjective combinatorial optimization problems (MOCOP), which are often characterized by their large size and the presence of multiple, conflicting objectives. In general, the basic task in multiobjective optimization is the identification of the set of Pareto optimal solutions or even a good approximation set to the Pareto front (PF). Despite the progress in solving MOCOP exactly, the large size often means that Metaheuristics (MH) are required for their solution in reasonable time.

Multi-objective Evolutionary Algorithms (MOEAs) are a very active research area. Solving MOOPs and their applications using evolutionary algorithms have been investigated by many authors [4], [7], [17], [18]. Pareto dominance based MOEAs such as SPEA [17], NSGAII [3] and SPEA2 [18] have been dominantly used in the recent studies. Based on many traditional mathematical programming methods for approximating the PF [12], the approximation of the PF can be decomposed into a number of single objective subproblems. Some of the MOEAs adopt this idea such as MOGLS [8], MOEA/D [15]. Many of the search algorithms attempt to obtain the best from a set of different metaheuristics that perform together, complement each other and augment their exploration capabilities. They are commonly called Hybrid MH. Diversification and intensification [1] are the two major issues when designing a global search method. A search algorithm must balance between sometimes-conflicting two goals. The design of the Hybrid MH can give the ability to control this balance [11].

This paper tends to study and analyze the effect of hybridization of adaptive discrete differential evolution operator and/or path-Relinking operator with the MOEA/D framework in handling MOCOP. The competitive results achieved by discrete DE in [16] motivated us to hybrid discrete DE within MOEA/D framework. Moreover, path-relinking could improve the search if it applied on high quality solutions [10]. This work is partially related to our previous work in [10] in which a new hybrid approach (HEMH) was developed. But here, we only concentrate on studying the effect of incorporating differential evolution and/or path-relinking in the MOEA/D. The main goals are to determine the benefits and limitations of those techniques by studying possible combinations and their effects on the search capabilities. The rest of the paper is organized as follows: section II presents some of the basic concepts and definitions. In section III, the MOEA/D framework was reviewed. In section IV, an overview on adaptive discrete differential evolution is highlighted. The path-Relinking strategy is discussed in section V. The proposed hybridization variants are presented in section VI. In addition, the experimental design and results are involved in sections VII and VIII respectively. Section IX presents some concluding remarks. Finally, the conclusions and some directions for further research are presented in section X.

II. Basic Concepts and Definitions

Without loss of generality, MOOP can be formulated as:

$$Maximize\, F(x) = (f_1(x), f_2(x), \cdots, f_m(x)) \qquad (1)$$
$$Subject\, to:\ x \in \Omega$$

Where $F(x)$ is the m-dimensional objective vector, $f_i(x)$ is the i^{th} objective to be maximized, $x = (x_1, \cdots, x_n)^T$ is the n-dimensional decision vector, Ω is the feasible decision space. In the case $x \in$ Z, the MOOP is called multiobjective

combinatorial optimization problem (MOCOP).

Definition 1: A solution x dominates y (noted as: $x \succcurlyeq y$) if: $f_i(x) \geq f_i(y) \forall i \in \{1, \cdots, m\}$ and $f_i(x) > f_i(y)$ for at least one i.

Definition 2: A solution x is called efficient (*Pareto-optimal*) if: $\nexists\, y \in \Omega : y \succcurlyeq x$

Definition 3: The Pareto optimal set (P^*) is the set of all efficient solutions:

$$P^* = \{x \in \Omega : \nexists\, y \in \Omega \, and \, F(y) \geq F(x)\}$$

Definition 4: The Pareto front (PF) is the image of the Pareto optimal set (P^*) in the objective space:

$$PF = \{F(x) = (f_1(x), \cdots, f_m(x)) : x \in P^*\}$$

Definition 5: Given a reference point r^* and a weight vector $\Lambda = [\lambda_1, \cdots, \lambda_m]$ such that $\lambda_i \geq 0, \forall i \in \{1, \ldots, m\}$, the *weighted sum* (F^{ws}) and the *weighted Tchebycheff* (F^{Tc}) scalarizing functions corresponding to (1) are defined by (2) and (3) respectively as:

$$Maximize\, F^{ws}(x, \Lambda) = \sum_{i=1}^{m} \lambda_i f_i(x) \tag{2}$$

$$F^{Tc}(x, r^*, \Lambda) = Max_{1 \leq i \leq m} \{\lambda_i (r_i^* - f_i(x))\} \tag{3}$$

Given a set of m knapsacks and a set of n items, the 0/1 multiobjective knapsack problem (MOKSP) can be formulated as:

$$Maximize\, f_i(x) = \sum_{j=1}^{n} c_{ij} x_j, \forall i \in \{1, \cdots, m\} \tag{4}$$

$$Subject\, to : \sum_{j=1}^{n} w_{ij} x_j \leq W_i, \forall i \in \{1, \cdots, m\} \tag{5}$$

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

Where, $c_{ij} \geq 0$ is the *profit* of the j^{th} item in the i^{th} knapsack, $w_{ij} \geq 0$ is the *weight* of the j^{th} item in the i^{th} knapsack, and W_i is the *capacity* of the i^{th} knapsack. When x_j=1, it means that the j^{th} item is selected and put in all knapsacks.

The MOKSP is NP-hard and can model a variety of applications. It was formulated and solved by Zitzler and Thiele [16]. Since then, it has become a standard benchmark that has been solved by many other researchers [3], [8], [15].

III. MOEA/D Framework

The MOEA/D [15] is a recently developed MOEA in which the decomposition idea is applied instead of dominance relation. The MOEA/D framework can be explained as a cellular MOEA [6] with a neighborhood structure in the m-dimensional weight space. A single cell with a single individual is located at the same place as each weight vector in the m-dimensional weight space. That is, each cell has its own weight vector, which is used in the scalarizing function for evaluating the individual in that cell. Neighbors of a cell are defined by the Euclidean distance between cells in the weight space. The efficient solutions obtained over the search process are maintained in an external archive. To generate an offspring for a cell, two parents are randomly selected from its neighbors to apply reproduction. The offspring is compared with the individual in the current cell using the scalarizing function. If the offspring is better, the current individual is replaced with the offspring. The offspring is also compared with each neighbor using the scalarizing function with the weight vector of that neighbor. All neighbors, which are inferior to the offspring, are replaced with the offspring. MOEA/D requires the following components:

- A population of N individuals $\{x^1, \cdots, x^N\}$, where x^i is the current solution of the i^{th} subproblem.
- A set of N weight vectors $\{\Lambda^1, \cdots, \Lambda^N\}$, correspond to N single objective subproblems. Each weight vector $\Lambda = [\lambda_1, \cdots, \lambda_m]$ has m component correspond to the m-objective problem, such that: $\sum_{i=1}^{m} \lambda_i = 1, \forall \lambda_i \in \{0/H, 1/H, \cdots, H/H\}$, $\forall i \in \{1, \cdots, m\}$, $H \in Z^+$.

For each $i \in \{1, \cdots, N\}$, a neighborhood $B(i)$ of the i^{th} subproblem includes all the subproblems with the T closest weight vectors $\{\Lambda^{i1}, \cdots, \Lambda^{iT}\}$ to Λ^i in terms of Euclidean distance.

IV. Adaptive Discrete Differential Evolution

Differential Evolution (DE) is a simple and efficient evolutionary algorithm to solve optimization problems mainly in continuous search domains [2], [13]. It can memorize the best individuals and share the evolutionary information. It also utilizes the competition and the cooperation of individuals in population to guide the search process. DE has shown to be not only very effective as a global optimizer in many application domains, but also very robust in the sense of being able to produce a small variance of the outcomes for a number of independent runs [13].

The success of the DE algorithm relies on the differential mutation operator. It employs difference vectors built with pairs of candidate solutions in the search domain. The difference vector are scaled and added to a third point, producing the so-called mutant vector. In this paper, we propose to use the differential mutation operator as an additional operator within the MOEA/D framework. We choose the adaptive discrete differential evolution strategy proposed in [16] to study its effect on the MOEA/D exploration capabilities in the discrete domains. This strategy is described in Alg. 1. Assume P is a population of N individuals. The main idea is to select at random three distinct individuals x_{r1}, x_{r2}, x_{r3} from P for each target individual $x_i \in P, \forall i \in \{1, \cdots, N\}$. The mutant individual v_i is produced by applying the differential uniform mutation on which called the parent base individual x_{r1} with the rate p_m. p_m is calculated based on the parent differential individuals (x_{r2}, x_{r3}) as follows:

$$p_m = F \cdot (H_{Dist}(x_{r2}, x_{r3}) / n) \tag{6}$$

where H_{Dist} is the Hamming distance, n is the individual length and F denotes the scaling factor. Then, crossover is used to produce the new individual u_i as follows:

$$u_i^j = \begin{cases} v_i^j & \text{if } rnd(j) \leq CR, \text{ or } j = e, \forall j = 1, \cdots, n. \\ x_i^j & \text{otherwise, } \forall j = 1, \cdots, n. \end{cases} \tag{7}$$

where $rnd(j) \in [0, 1]$ is the j^{th} random number generated by random number generator, e is a component of a random sequence S selected from $\{1, \cdots, n\}$ to insure that at least

one component of u_i is contributed by v_i and $CR \in [0,1]$ denotes the crossover factor. The mutation scaling factor F and the crossover factor CR are adapted periodically to avoid premature convergence as follows:

$$F = F_0 \cdot e^{(-a_1 \cdot (G/G_{max}))} \quad (8)$$

$$CR = CR_0 \cdot e^{(-a_2 \cdot (G/G_{max}))} \quad (9)$$

where G, G_{max} are the current and the maximum evolutionary generation respectively, F_0, CR_0 are the initial values of the scaling factor and the crossover operator respectively. a_1 and a_2 are plus constants. Finally, the new generated individual u_i is returned.

Algorithm 1 :DIFFEVOLUTION($x, y_1, y_2, y_3, F_0, CR_0, a_1, a_2$)

Inputs:
x & y_1, y_2, y_3: *current & 3 parent individuals*
$F_0, CR_0 \in [0,1]$: *scaling factor and crossover rate*

Begin:
$F \leftarrow F_0 \cdot e^{(-a_1 \cdot (G/G_{max}))}$;
$CR \leftarrow CR_0 \cdot e^{(-a_2 \cdot (G/G_{max}))}$;
$p_m \leftarrow F \cdot (H_{Dist}(y_2, y_3)/n)$;
$v \leftarrow$ MUTATION(y_1,p_m); ▷ *Mutation*
for all $j \in \{1, \cdots, n\}$ **do** ▷ *Crossover*
  if $(rnd(j) \leq CR \vee j = e)$ **then**
    $u^j \leftarrow v^j$;
  else
    $u^j \leftarrow x^j$;
  end if
end for
return u;

V. PATH-RELINKING

Path-relinking was suggested to integrate intensification and diversification strategies in the context of tabu search and scattered search [5]. It generates new solutions by exploring trajectories that connect high quality solutions. Starting from the starting solution x^s, path-relinking generates a path in the neighborhood space that leads toward the guiding solution x^t. It selects moves that introduce attributes contained in x^t, and incorporating them in an intermediate solution initially originated in x^s. It is observed that better solutions are found when the relinking procedure starts from the best of x^s and x^t. Because starting from the best one gives the algorithm a better chance to investigate in more detail the neighborhood of the most promising solutions [14]. In this paper, path-relinking will be used as an intensification strategy in the MOEA/D framework, integrated with reproduction by crossover and mutation. It will be invoked in the higher generations to guarantee applying the relinking process on high quality solutions to improve the performance and enhance the efficiency.

The proposed path-relinking procedure receives the inputs listed in Alg. 2. Firstly, the best of x^s and x^t is chosen to start with. Then, the best fitness z^* and the best solution x^* are initialized. The candidate lists CL and CL_{cmp} are constructed. Every unmatched j between x^s and x^t with $x^s_j = 0$ is inserted into CL in descending order according to the ratio in (10). whereas, every unmatched j between x^s and x^t with $x^s_j = 1$ is inserted into CL_{cmp} in increasing order according to (10).

$$\sum_{i=1}^{m} \lambda_i \, c_{ij} / \sum_{i=1}^{m} w_{ij} \quad (10)$$

The procedure builds the path that connects x^s with x^t gradually by creating intermediate points through execution of the relinking loop. Initially, the intermediate solution x is set to x^s. Then, the number of unmatched items between x and x^t ($\Delta(x, x^t)$) is calculated. The next move is carried out by selecting two of unmatched ℓ^1, ℓ^2 to be matched. If both CL and CL_{cmp} are not empty, then the first elements of CL and CL_{cmp} are extracted to be ℓ^1 and ℓ^2 respectively. Else if one of them is empty, then, the first and second element of the non empty one will be extracted to be ℓ^1 and ℓ^2 respectively. The new intermediate x is obtained by flipping the two items (x_{ℓ^1}, x_{ℓ^2}) corresponding to the selected indexes ℓ^1 and ℓ^2 in the current intermediate x. If x is infeasible, the Greedy-Repair is invoked to get the feasible solution y. Then, z^* and x^* are updated by y. This process is repeated until there is only one unmatched item between the current intermediate x and the guiding x^t. Finally x^* is returned.

Algorithm 2 :PATHRELINKING(x^s, x^t, Λ)

Inputs:
x^s, x^t: *Starting & Guiding solutions*
$\Lambda = [\lambda_1, \cdots, \lambda_m]$: *weight vector of the current subproblem*

Begin:
$x^* \leftarrow$GETBESTOF(x^s, x^t);
$z^* \leftarrow F^{ws}(x^*, \Lambda); CL, CL_{cmp} \leftarrow \phi$;
while $(\exists j : x^s_j \neq x^t_j \wedge x^s_j = 0 \wedge Max_{j \in \{1,\cdots,n\}} \frac{\sum_{i=1}^{m} \lambda_i c_{ij}}{\sum_{i=1}^{m} w_{ij}})$ **do**
  $CL \leftarrow CL \cup \{j\}$
end while
while $(\exists j : x^s_j \neq x^t_j \wedge x^s_j = 1 \wedge Min_{j \in \{1,\cdots,n\}} \frac{\sum_{i=1}^{m} \lambda_i c_{ij}}{\sum_{i=1}^{m} w_{ij}})$ **do**
  $CL_{cmp} \leftarrow CL_{cmp} \cup \{j\}$
end while
$x \leftarrow x^s; \Delta(x, x^t) \leftarrow \left\{ j : j \in \{1, \cdots, n\} : x_j \neq x^t_j \right\}$
while $(|\Delta(x, x^t)| \geq 2)$ **do** ▷ *Relinking loop*
  if CL and CL_{cmp} are not empty **then**
    $\ell^1 \leftarrow$ EXTRACTFIRSTELEM(CL) ▷ *Extract the 1st element*
    $\ell^2 \leftarrow$ EXTRACTFIRSTELEM(CL_{cmp})
  else if $|CL| > 1$ **then**
    $\ell^1 \leftarrow$ EXTRACTFIRSTELEM(CL)
    $\ell^2 \leftarrow$ EXTRACTFIRSTELEM(CL)
  else
    $\ell^1 \leftarrow$ EXTRACTFIRSTELEM(CL_{cmp})
    $\ell^2 \leftarrow$ EXTRACTFIRSTELEM(CL_{cmp})
  end if
  $x \leftarrow$ FILPPBIT(ℓ^1, ℓ^2);$y \leftarrow$REPAIR(x, Λ)
  if $(F^{ws}(y, \Lambda) > z^*)$ **then**
    $x^* \leftarrow y; z^* \leftarrow F^{ws}(y, \Lambda)$;
  end if
  $\Delta(x, x^t) \leftarrow \left\{ j : j \in \{1, \cdots, n\} : x_j \neq x^t_j \right\}$
end while
return x^*;

VI. PROPOSED HYBRIDIZATION VARIANTS

In this work, we study the effect of using both of adaptive differential evolution operator proposed in [16] and/or proposed path-relinking as a reproduction operator instead of standard reproduction (crossover and mutation) in MOEA/D framework. So we have four algorithm variants, the first variant is called MOEAD$_{de}$ in which the adaptive discrete differential evolution completely replaces crossover and mutation operators. The second variant is called MOEAD$_{pr}$ in which the proposed path-relinking operator is applied with the crossover

and standard mutation after a certain number of evaluations to guarantee the existence of high quality solutions. Pseudo codes in Alg. 3 and Alg. 4 describe MOEAD$_{pr}$ and MOEAD$_{de}$ respectively. In the third and the fourth variants, both of differential evolution and path-relinking replaces crossover and mutation, they are called MOEAD$_{dp1}$ and MOEAD$_{dp2}$ respectively. Fig.1 depicts these different variants.

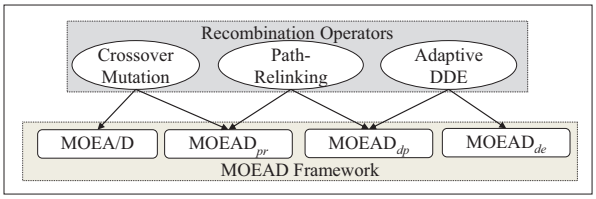


Fig. 1. Hybridization variants

In Both MOEAD$_{de}$ and MOEAD$_{pr}$, the set of uniform weight vectors Λ is calculated, followed by construction of the neighborhood structure. The initial population is also generated (lines 2-5). Then the main loop is executed until achieving the maximum evaluations (line 6). To generate a new offspring for each subproblem i, the mating/updating range (M) is determined to be either the neighborhood of the i^{th} subproblem (*Local*), or the whole population (*Global*) according to a certain probability (σ). This can give a better chance for selecting distinct parents, which encourages the path-relinking to be invoked in MOEAD$_{pr}$, or allows the differential evolution to operate on distinct individuals in MOEAD$_{de}$. Then, parent selection is performed. In case of MOEAD$_{pr}$ (Alg. 3), two parents x^j and x^k are randomly selected from M. Then, the path-relinking operator is used only if the hamming distance between the two selected parent is greater than a certain value ϵ and the number of evaluations $Eval$ exceeds a certain ratio (γ) of the maximum evaluations allowed to guarantee applying path-relinking on high quality solutions. Else, the standard reproduction operator is applied to generate the new offspring. In case of MOEAD$_{de}$ variant (Alg. 4), three distinct parent individuals are randomly selected to apply adaptive discrete differential evolution on them. The new generated offspring is evaluated, and used to update the reference point z and also updating the population according to the parameter t, which is used to limit the number of replaced solutions. Finally, the efficient solution set in the final population is returned as an output. In both MOEAD$_{dp1}$ and MOEAD$_{dp2}$, some modifications are applied on MOEAD$_{de}$ (Alg. 4) to involve path-relinking after certain number of evaluations that carried out to assure the existence of high quality solutions. These modifications can be briefed as follows: when the number of evaluations $Eval$ exceeds a certain value ($\gamma \times MaxEvals$) previously determined to involve path-relinking, we have three selected parents x^a, x^b and x^c in the selection step (Alg. 4:line 9). If we randomly choose two of them which have hamming distance greater than a certain value (ϵ) to apply path-relinking on instead of differential evolution, we will get the MOEAD$_{dp1}$ variant. On the other hand, assuming the hamming distance conditions[1] are satisfied, if we apply path-relinking on the three selected parents(x^a, x^b and x^c) in the following manner: randomly choosing two individuals (x^a,x^c) to apply path-relinking producing a new individual y, then applying path-relinking on y and x^b, we will get MOEAD$_{dp2}$ variant. The pseudo code of both MOEAD$_{dp1}$ and MOEAD$_{dp2}$ can be obtained by replacing lines 10-11 in Alg. 4 by the pieces of code shown in Alg. 5 and 6 respectively.

Algorithm 3 :MOEAD$_{pr}(N, T, t, \delta, \epsilon, \gamma)$

Inputs:
N, T, t: *Population size, Neighborhood size & No. of replaced solutions*
$\delta \in [0,1]$: *Prob. of selecting parents from neighborhood*
ϵ, γ: *Min. Hamming distance, Min. evaluations allowed for path-relink*

1: **Begin:**
2: $\Lambda \leftarrow$INITIALIZEWEIGHTVECTORS( );
3: $B \leftarrow$INITIALIZENEIGHBORHOOD( );
4: $P \leftarrow$INITIALIZEPOPULATION( );
5: $z \leftarrow$INITIALIZEREFPOINT( ); $Eval \leftarrow 0$;
6: **while** $(Eval < MaxEvals)$ **do** ▷ *main loop*
7: **for all** $(i \in \{1, 2, \cdots, N\})$ **do**
8: $M \leftarrow \begin{cases} B(i) & \text{if}(rnd \in [0,1] < \delta) \\ P & \text{otherwise} \end{cases}$
9: $x^j, x^k \leftarrow$ SELECTION(M, i) ▷ *select 2 elements*
10: **if** $(H_{Dist}(x^j, x^k) \geq \epsilon \wedge Eval \geq \gamma \times MaxEvals)$ **then**
11: $y \leftarrow$PATHRELINKING(x^j, x^k, Λ^i);
12: **else**
13: $u \leftarrow$REPRODUCTION(x^j, x^k);
14: $y \leftarrow$REPAIR(u, Λ^i)
15: **end if**
16: EVALUATEFITNESS(y);
17: $z \leftarrow$UPDATEREFPOINT(y);
18: $M \leftarrow$UPDATESOLUTIONS(y, t);
19: $Eval \leftarrow$UPDATE( );
20: **end for**
21: **end while**
22: **return** P;

Algorithm 4 :MOEAD$_{de}(N, T, t, \delta, F_0, CR_0, a_1, a_2)$

Inputs:
N, T, t: *Population size, Neighborhood size & No. of replaced solutions*
$\delta \in [0,1]$: *Prob. of selecting parents from neighborhood*
$F_0, CR_0 \in [0,1]$: *Scaling factor and Crossover rate*

1: **Begin:**
2: $\Lambda \leftarrow$INITIALIZEWEIGHTVECTORS( );
3: $B \leftarrow$INITIALIZENEIGHBORHOOD( );
4: $P \leftarrow$INITIALIZEPOPULATION( );
5: $z \leftarrow$INITIALIZEREFPOINT( ); $Eval \leftarrow 0$;
6: **while** $(Eval < MaxEvals)$ **do** ▷ *main loop*
7: **for all** $(i \in \{1, 2, \cdots, N\})$ **do**
8: $M \leftarrow \begin{cases} B(i) & \text{if}(rnd \in [0,1] < \delta) \\ P & \text{otherwise} \end{cases}$
9: $x^a, x^b, x^c \leftarrow$ SELECTION(M, i): $x^i \neq x^a \neq x^b \neq x^c$
10: $u \leftarrow$DIFFEVOLUTION$(x_i, x^a, x^b, x^c, F_0, CR_0, a_1, a_2)$;
11: $y \leftarrow$REPAIR(u, Λ^i);
12: EVALUATEFITNESS(y);
13: $z \leftarrow$UPDATEREFPOINT(y);
14: $M \leftarrow$UPDATESOLUTIONS(y, t);
15: $Eval \leftarrow$UPDATE( );
16: **end for**
17: **end while**
18: **return** P;

[1]it means that x^j and x^k can be chosen for path-relinking only if: $H_{Dist}(x^j, x^k) \geq \epsilon$

Algorithm 5 :MOEAD$_{dp1}(N, T, t, \delta, \epsilon, \gamma, F_0, CR_0, a_1, a_2)$

Replace lines (10-11) in Alg.4 by the following:
1: $x^j, x^k \leftarrow$ RANDOMSELECTION(x^a, x^b, x^c);
2: **if** $(H_{Dist}(x^j, x^k) \geq \epsilon \wedge Eval \geq \gamma \times MaxEvals)$ **then**
3: $y \leftarrow$PATHRELINKING(x^j, x^k, Λ^i);
4: **else**
5: $u \leftarrow$DIFFEVOLUTION$(x_i, x^a, x^b, x^c, F_0, CR_0, a_1, a_2)$;
6: $y \leftarrow$REPAIR(u, Λ^i);
7: **end if**

Algorithm 6 :MOEAD$_{dp2}(N, T, t, \delta, \epsilon, \gamma, F_0, CR_0, a_1, a_2)$

Replace lines(10-11) in Alg.4 by the following:
1: $x^a, x^c \leftarrow$ RANDOMSELECTION(x^a, x^b, x^c);
2: **if** $(H_{Dist}(x^a, x^c) \geq \epsilon \wedge Eval \geq \gamma \times MaxEvals)$ **then**
3: $y \leftarrow$PATHRELINKING(x^j, x^k, Λ^i);
4: **if** $(H_{Dist}(x^b, y) \geq \epsilon)$ **then**
5: $y \leftarrow$PATHRELINKING(y, x^b, Λ^i);
6: **end if**
7: **else**
8: $u \leftarrow$DIFFEVOLUTION$(x_i, x^a, x^b, x^c, F_0, CR_0, a_1, a_2)$;
9: $y \leftarrow$REPAIR(u, Λ^i);
10: **end if**

VII. EXPERIMENTAL DESIGN

In this paper, all experiments have been performed on DELL PC with Intel Core i5-2400 CPU, 3.10 GHz and 4.0 GB of RAM. The comparative study for different algorithm variant was carried out on the set of test instances listed below in Table I, which are commonly used in literature. SPEA2 [18] algorithm is also used in this study.

TABLE I
SET OF KNAPSACK TEST INSTANCES

Instance	m	n	SPEA2(N)	$N(H)$	$MaxEvals$
KS252	2	250	150	150(149)	75000
KS502	2	500	200	200(199)	100000
KS752	2	750	250	250(249)	125000
KS253	3	250	200	300(23)	100000
KS503	3	500	250	300(23)	125000
KS753	3	750	300	300(23)	150000
KS254	4	250	250	364(11)	125000
KS504	4	500	300	364(11)	150000
KS754	4	750	350	364(11)	175000

A. Parameter settings

Here, the different parameters used for each algorithm are discussed. The population size (N) used in SPEA2 is shown in Table I. For MOEA/D and its variants MOEAD$_{de}$, MOEAD$_{pr}$, MOEAD$_{dp1}$ and MOEAD$_{dp2}$, the parameter H which controls both the number of weight vectors and the population size (N) is determined for each instance in Table I according to the complexity. The initial population used is randomly generated such that each member $x = (x_1, \cdots, x_n)^T \in \{0, 1\}^T$, where $x_i = 1$ with probability equal to 0.5. The maximum number of evaluations ($MaxEvals$) is used as a stopping criterion for each algorithm. For each algorithm, all efficient solutions rest in the final iteration is used as the final approximation set. In these experiments, single-point crossover and standard mutation were considered. Mutation was performed for each item independently with probability $(1/n)$. SPEA2 uses single point crossover with probability=1 and tournament selection with tournament size=2. The other control parameters are listed in Table II. Finally, the statistical analysis is applied on 30 independent runs for each test instance.

TABLE II
SET OF COMMON PARAMETER USED

Parameters	MOEA/D			
	-	-*de*	-*pr*	-*dp*
Neighborhood size: T	10	10	10	10
Max.no. of replaced solutions: t	2	2	2	2
Parents slection: δ	-	0.9	0.9	0.9
Ratio to apply Path relink: γ	-	-	0.7	0.7
Minimal Hamming Distance: ϵ	-	-	10	10
Initial crossover Prob.: CR_0	-	0.4	-	0.4
nitial scaling factor: F_0	-	0.4	-	0.4
Plus constants: a_1, a_2	-	2, 2	-	2, 2

B. Assessment Metrics

Let $A \subset \Re^m$ and $B \subset \Re^m$ be two approximations to the Pareto front (PF), $P^* \subset \Re^m$ be a set of uniformly distributed points along the PF(*Reference Set*) and $r^* \in \Re^m$ be a reference point. The following metrics can be expressed as follows:

1) The Set Coverage (I_C) [17]: This indicator is used to compare two approximation sets. The function I_C maps the ordered pair (A, B) to the interval $[0, 1]$ as:

$$I_C(A, B) = |u|u \in B, \exists v|v \in A : v \succcurlyeq u|/(|B|) \quad (11)$$

where $I_C(A, B)$ represents the percentage of the solutions in B that are dominated by at least one solution from A. $I_C(B, A)$ is not necessarily equal to 1-$I_C(A, B)$. If $I_C(A, B)$ is large and $I_C(B, A)$ is small, then A is better than B in a sense.

2) The Hypervolume (I_H) [17]: The hypervolume for a set A is defined as:

$$I_H(A) = \mathcal{L}(\cup_{u \in A} \{y|u \succcurlyeq y \succcurlyeq r^*\}) \quad (12)$$

where $\mathcal{L}$ is the Lebesgue measure of a set. This indicator describes the size of the objective space that is dominated by points of A and dominates r^*. Here, we use the referenced indicator such that: $I_{RH}(A) = I_H(P^*) - I_H(A)$ and r^* is chosen as the origin.

3) Generational/Inverted Generational Distance: Generational Distance (I_{GD}) and Inverted Generational Distance (I_{IGD}) of a set A are defined as:

$$I_{GD}(A, P^*) = \frac{1}{|A|} \sum_{u \in A} \{min_{v \in P^*} d(u, v)\} \quad (13)$$

$$I_{IGD}(A, P^*) = \frac{1}{|P^*|} \sum_{u \in P^*} \{min_{v \in A} d(u, v)\} \quad (14)$$

where $d(u, v)$ is the Euclidean distance between u,v in $\Re^m$. The $I_{GD}(A, P^*)$ measures the average distance from A to the nearest solution in P^* that reflects the closeness of A to P^*. In contrast, the $I_{IGD}(A, P^*)$ measures the average distance from P^* to the nearest solution in A that reflects the spread of A to a certain degree. The lower value of both $I_{GD}(A, P^*)$ and $I_{IGD}(A, P^*)$ means the better quality of A in terms of convergence and diversity respectively.

4) R-indicator (I_{R_3}) [9]: The I_{R_3} metric uses a set of utility functions u, which can be any scalar function. We use both a weighted sum and weighted Tchebycheff function with sufficiently large set of evenly distributed normalized weight vectors Λ. I_{R_3} can be evaluated as follows:

$$I_{R_3}(A, P^*) = \frac{\sum_{\lambda \in \Lambda} \left[u^*(\lambda, P^*) - u^*(\lambda, A)\right] / u^*(\lambda, P^*)}{|\Lambda|} \quad (15)$$

where $u^*(\lambda, A)$=$max_{z \in A} u(\lambda, z)$, $u(\lambda, z) = -(max_{1 \leq j \leq m} \lambda_j |z_j^* - z_j| + \rho \sum_{j=1}^{m} |z_j^* - z_j|)$, ρ is a small positive integer, and for each weight vector $\lambda \in \Lambda$, $\lambda = [\lambda_1, \cdots, \lambda_m]$ such that $\lambda_i \in [0, 1]$ and $\sum_{i=1}^{m} \lambda_i = 1$.

In these experiments, the reference set P^* is alternatively formed for each problem instance by gathering all nondominated solutions found by all of the compared algorithms in all runs. Also, all approximation sets are normalized in the range [1,2].

VIII. Experimental Results

Here, the different simulation results are shown in details. Firstly, Fig.2 depicts the results of I_C metric. It contains a chart (with scale 0 at the bottom and 1 at the top) for each ordered pair of the compared algorithms. Each chart consists of nine box plots representing the distribution of I_C values. Each box plot (from left to right) represents an instance in Table I (from top to down), respectively. A chart located in the raw of algorithm A and the column of algorithm B presents the values of coverage of the approximations generated by algorithm B by approximations generated by algorithm A. It is clear from the results in Fig.2 that all four hybrid variants outperform the original MOEA/D in most test instances. It is also clear that MOEAD$_{pr}$ has the best performance for all bi-objective test instances.

The results of I_{RH} listed in Table III contain the average of I_{RH} values achieved over 30 independent runs for each test instance for each algorithm. Fig.3(a) visualizes the average values. It is clear that all hybrid variants outperform the original MOEA/D for all 3 and 4 objective test instances especially Path-relinking based variants (MOEAD$_{pr}$, MOEAD$_{dp1}$ and MOEAD$_{dp2}$), since they have the minimum average values. We find also, MOEAD$_{pr}$ has the best performance with respect to most instances, while the differential evolution based variants (MOEAD$_{de}$, MOEAD$_{dp1}$ and MOEAD$_{dp2}$) have poor performance with respect to bi-objective instance compared with the original MOEA/D.

TABLE III
Results of Referenced Hypervolume (I_{RH})

Inst.	Algorithm					
	SPEA2	MOEAD	MOEAD$_{de}$	MOEAD$_{pr}$	MOEAD$_{dp1}$	MOEAD$_{dp2}$
KS252	4.45E-01	4.07E-02	1.48E-01	**2.96E-02**	1.41E-01	1.43E-01
KS502	6.51E-01	4.76E-02	1.72E-01	**3.06E-02**	1.68E-01	1.70E-01
KS752	7.32E-01	3.97E-02	1.19E-01	**2.86E-02**	1.13E-01	1.11E-01
KS253	1.58E+00	2.09E-01	1.80E-01	**1.40E-01**	1.64E-01	1.67E-01
KS503	2.11E+00	2.50E-01	2.12E-01	**1.40E-01**	1.86E-01	1.94E-01
KS753	2.41E+00	2.85E-01	2.80E-01	**1.21E-01**	2.62E-01	2.53E-01
KS254	4.00E+00	8.34E-01	6.49E-01	6.11E-01	**6.10E-01**	**6.10E-01**
KS504	5.05E+00	1.06E+00	6.55E-01	**4.92E-01**	5.66E-01	5.66E-01
KS754	5.76E+00	1.29E+00	7.50E-01	**5.07E-01**	6.09E-01	5.79E-01

In Table IV, the average values of the generational distance I_{GD} are listed. Additionally, Fig. 3(b) visualizes the average values. According to I_{GD} measure, it is clear that the proposed hybrid variants outperform the original MOEA/D for most instances since they have the minimum average values. Also, MOEAD$_{pr}$ outperforms with respect to bi-objective test instances, while both MOEAD$_{dp1}$ and MOEAD$_{dp2}$ have the superiority in the rest. In contrast, we find the differential evolution based variants especially MOEAD$_{de}$ achieve bad results than the original MOEA/D in bi-objective instances.

TABLE IV
Results of Generational Distance (I_{GD})

Inst.	Algorithm					
	SPEA2	MOEAD	MOEAD$_{de}$	MOEAD$_{pr}$	MOEAD$_{dp1}$	MOEAD$_{dp2}$
KS252	2.79E-03	1.39E-03	2.25E-03	**1.12E-03**	2.22E-03	2.33E-03
KS502	3.98E-03	1.53E-03	2.34E-03	**1.10E-03**	2.18E-03	2.36E-03
KS752	4.62E-03	1.28E-03	1.38E-03	**1.23E-03**	1.27E-03	1.24E-03
KS253	3.71E-03	1.88E-03	7.39E-04	9.77E-04	6.55E-04	**6.30E-04**
KS503	4.06E-03	2.13E-03	7.12E-04	1.24E-03	**5.93E-04**	6.08E-04
KS753	4.18E-03	2.10E-03	8.06E-04	1.05E-03	6.55E-04	**6.37E-04**
KS254	5.17E-03	2.37E-03	9.84E-04	1.33E-03	8.76E-04	**8.61E-04**
KS504	6.34E-03	3.18E-03	9.58E-04	1.02E-03	**5.98E-04**	6.17E-04
KS754	6.75E-03	3.69E-03	9.73E-04	1.39E-03	6.08E-04	**5.51E-04**

The experimental results of the inverted generational distance (I_{IGD}) are listed in Table V. Fig. 3(c) depicts these results. These results are identical with I_{GD}-metric, where MOEAD$_{pr}$ performs better in bi-objective and both MOEAD$_{dp1}$ and MOEAD$_{dp2}$ perform better with respect to many-objective.

The R-indicator I_{R_3} illustrated in Table VI and depicted by Fig. 3(d) confirms the results of the previous metrics. It indicates that MOEAD$_{pr}$ variant outperforms with respect to bi-objective test instances. Where, MOEAD$_{dp1}$ and MOEAD$_{dp2}$ have the best performance with respect to the other instances.

TABLE V
Results of Inverted Generational Distance(I_{IGD})

Inst.	Algorithm					
	SPEA2	MOEAD	MOEAD$_{de}$	MOEAD$_{pr}$	MOEAD$_{dp1}$	MOEAD$_{dp2}$
KS252	1.09E-02	1.00E-03	2.31E-03	**8.37E-04**	2.40E-03	2.25E-03
KS502	1.16E-02	9.72E-04	1.89E-03	**7.05E-04**	1.88E-03	1.89E-03
KS752	1.27E-02	7.96E-04	1.29E-03	**7.04E-04**	1.20E-03	1.16E-03
KS253	2.24E-03	6.11E-04	4.48E-04	4.99E-04	4.38E-04	**4.35E-04**
KS503	2.83E-03	6.12E-04	4.11E-04	4.82E-04	**3.99E-04**	4.01E-04
KS753	3.15E-03	5.93E-04	3.89E-04	4.27E-04	3.79E-04	**3.70E-04**
KS254	1.45E-03	6.80E-04	5.54E-04	6.07E-04	5.53E-04	**5.49E-04**
KS504	1.69E-03	6.51E-04	4.56E-04	4.82E-04	4.41E-04	**4.37E-04**
KS754	1.91E-03	6.88E-04	4.40E-04	4.81E-04	4.14E-04	**4.07E-04**

IX. Concluding Remarks

Regarding the above results on MOKSP test instances, we can deduce the following remarks:

- MOEAD$_{pr}$ variant achieves better results than the original MOEA/D for all test instances in all used metrics.
- MOEAD$_{de}$ variant outperforms the original MOEA/D for test instance with 3 or 4 objective. Conversely, it

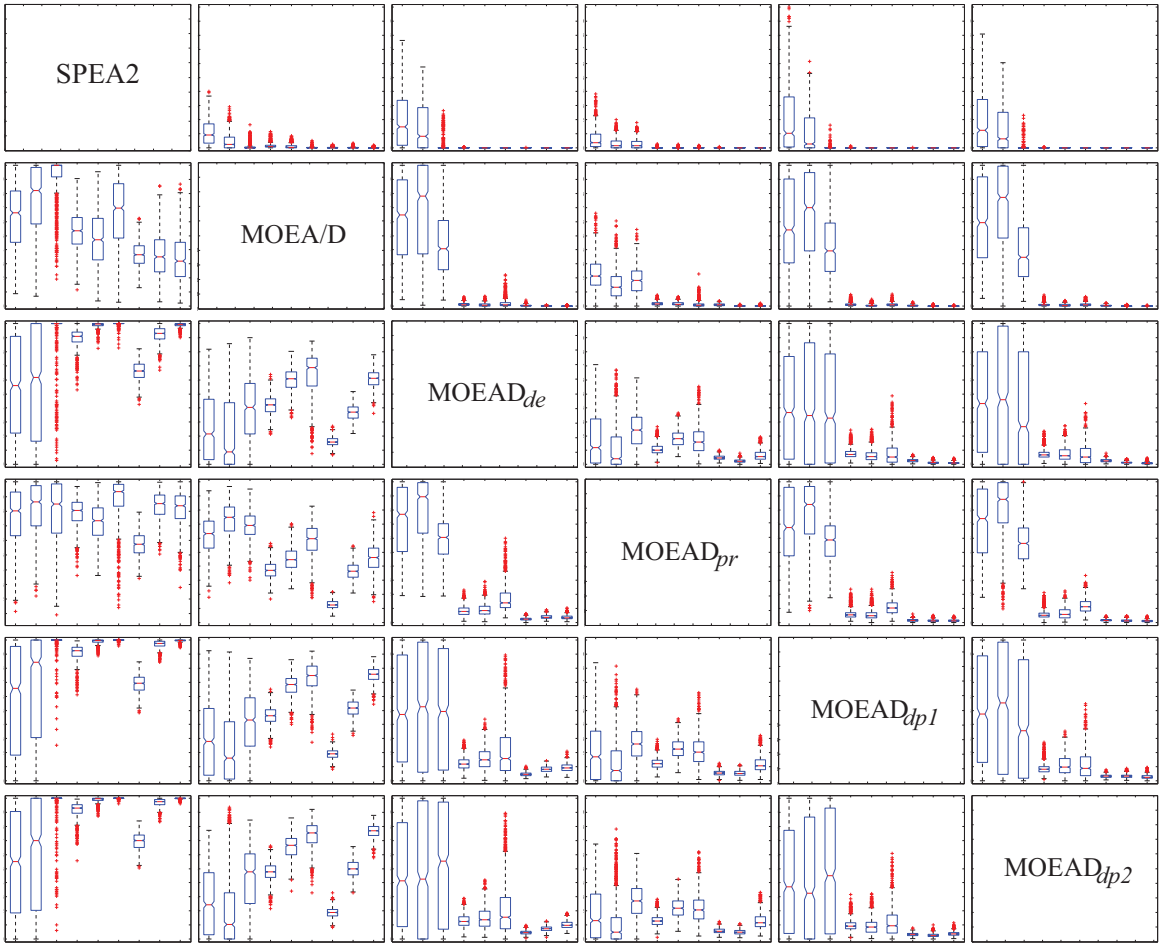


Fig. 2. Results of I_C indicator

TABLE VI
RESULTS OF R3 INDICATOR (I_{R_3})

Inst.	Algorithm					
	SPEA2	MOEAD	MOEAD$_{de}$	MOEAD$_{pr}$	MOEAD$_{dp1}$	MOEAD$_{dp2}$
KS252	4.28E-02	5.27E-03	1.08E-02	**3.85E-03**	1.09E-02	1.12E-02
KS502	7.98E-02	6.66E-03	1.40E-02	**4.64E-03**	1.33E-02	1.40E-02
KS752	9.61E-02	6.46E-03	8.11E-03	**5.64E-03**	7.23E-03	7.16E-03
KS253	6.07E-02	1.09E-02	6.29E-03	6.65E-03	5.82E-03	**5.79E-03**
KS503	9.74E-02	1.32E-02	7.08E-03	7.26E-03	**5.91E-03**	6.27E-03
KS753	1.21E-01	1.45E-02	8.34E-03	**6.52E-03**	7.25E-03	7.06E-03
KS254	7.24E-02	1.55E-02	1.03E-02	1.11E-02	9.66E-03	**9.63E-03**
KS504	1.06E-01	2.05E-02	1.08E-02	9.09E-03	**8.93E-03**	9.00E-03
KS754	1.38E-01	2.54E-02	1.21E-02	1.02E-02	9.17E-03	**8.81E-03**

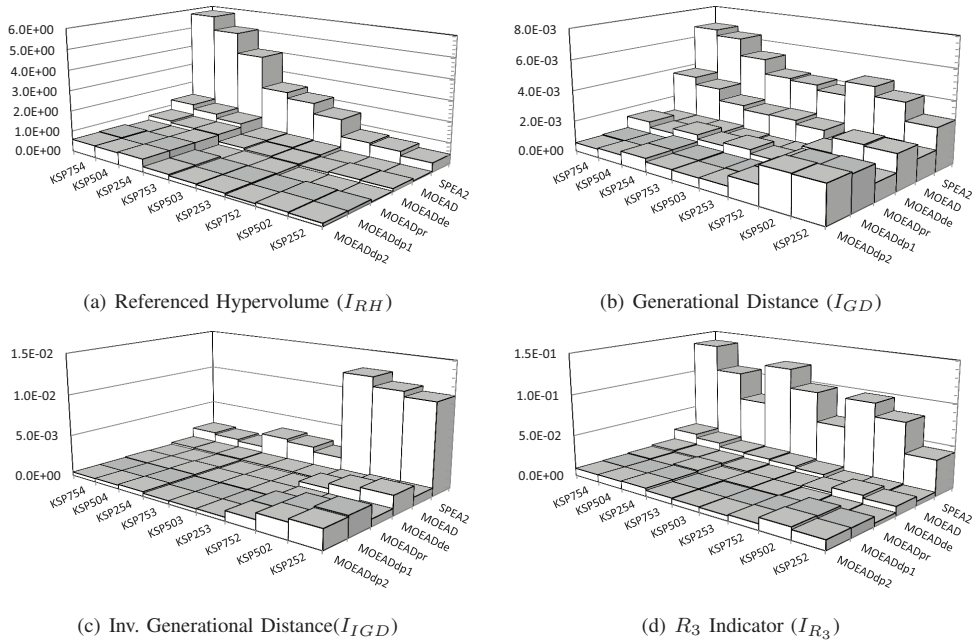
deteriorates the MOEA/D performance concerning bi-objective instances.

- The performance of both MOEAD$_{dp1}$ and MOEAD$_{dp2}$ is highly affected by MOEAD$_{de}$ performance. Since they depend on differential evolution strategy more than path-relinking.

In general, path-relinking operator has the ability to improve the performance of the MOEA/D for all instances especially with bi-objective instances. Whereas differential evolution improves the MOEA/D performance in 3 and 4 objectives instances. Consequently, the performance of their hybrid variants MOEAD$_{dp1}$ and MOEAD$_{dp2}$ is enhanced.

X. CONCLUSION

In this paper, four different hybridization variants within MOEA/D framework were presented. The first one is called MOEAD$_{de}$ which involves the adaptive discrete differential evolution as a recombination operator within MOEA/D Framework. The second is called MOEAD$_{pr}$, which uses the path-relinking operator with the standard reproduction operators. In the third and fourth variants both of differential evolution and path-relinking are used. The four proposals were compared with the original MOEA/D and SPEA2 using a set of MOKSP instances commonly used in the literature. A set of quality assessment indicators was also used to assess the performance. The experimental results indicate the superiority of all proposed hybrid variants over the original MOEA/D and SPEA2 for most test instances. In bi-objective test instances, we found that MOEAD$_{pr}$ has the superiority, while MOEAD$_{de}$ has poor performance. On the other hand, in case of instances with 3 or 4 objectives, the performance of the differential evolution is improved. Consequently, all proposed variants achieve better performance. They have an average performance highly



(a) Referenced Hypervolume (I_{RH})

(b) Generational Distance (I_{GD})

(c) Inv. Generational Distance(I_{IGD})

(d) R_3 Indicator (I_{R_3})

Fig. 3. Average Results for all used indicators

competitive with respect to the original MOEA/D and SPEA2 based on the assessment indicators used in this study. The general conclusion we have is: for bi-objective MOKSP test instances, path-relinking operator has the first rank followed by the standard crossover and mutation then differential evolution. But in MOKSP test instances with 3 or 4 objectives, differential evolution and path-relinking perform better than standard crossover and mutation. In the future work, these results will be exploited to improve our previous work in [10]. We will also study how to improve the performance of differential evolution on discrete domains. Moreover, we can use other metaheuristics to improve the performance of MOEA/D and to handle other types of combinatorial optimization problems.

REFERENCES

[1] Blum, C., Roli, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison, ACM Computing Surveys (CSUR), vol. 35, no.3, pp. 268-308. (2003).

[2] Chakraborty, U. K., Ed., Advances in Differential Evolution, ser. Studies in Computational Intelligence, vol. 143, Berlin: Springer-Verlag,(2008).

[3] Deb, K., Pratab, A., Agrawal S. and Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGAII. IEEE Trans. on Evolutionary Computation, vol.6, n. 2, 182-197, (2002).

[4] Farina, M., Deb, K. and Amato, P. Dynamic multiobjective optimization problems: test cases, approximations, and applications. IEEE Transactions on Evolutionary Computation, vol. 8, no. 5, pp. 425-442, (2004)

[5] Glover, F. and Laguna, M. Fundamentals of scatter search and path relinking. Control and Cybernetics, vol. 29, no. 3, pp. 653-684, (1999).

[6] Ishibuchi, H., Sakane, Y., Tsukamoto, N. and Nojima, Y. Effects of using two neighborhood structures on the performance of cellular evolutionary algorithms for many-objective optimization, In Proc. of IEEE Congress on Evolutionary Computation, 2508-2515, (2009).

[7] Jaszkiewicz, A. Do multiple objective metaheuristics deliver on their promises? A computational experiment on the set covering problem, IEEE Trans. on Evolutionary Computation, vol. 7, no. 2, pp. 133-143, (2003).

[8] Jaszkiewicz, A. 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, pp. 402-412, (2002).

[9] Knowles, J., Corne, D.; On metrics for comparing nondominated sets. In IEEE International conference in E-Commerce technology, vol. 1 pp. 711-716(2002).

[10] Kafafy A., Bounekkar A., Bonnevay S.: A hybrid evolutionary metaheuristics (HEMH) applied on 0/1 multiobjective knapsack problems. GECCO-2011: 497-504.

[11] Lozanoa, M. and Garca-Martnez, C. Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: Overview and progress report, Computers and Operations Research, vol. 37, no. 3, pp. 481-497, (2010).

[12] Miettinen, K. Nonlinear Multiobjective Optimization. Boston, MAA: Kluwer, (1999).

[13] Price, K. V. and Storn, R. M., and Lampinen, J. A., Differential Evolution: A Practical Approach to Global Optimization, 1st ed., ser. Natural Computing Series. Springer, (2005).

[14] Ribeiro, C., Uchoa, E. and Werneck, R. F. A hybrid GRASP with perturbations for the Steiner problem in graphs. INFORMS Journal on Computing, vol. 14, n. 3, 228-246 (2002).

[15] Zhang, Q. and Li, H. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. IEEE Trans. on Evolutionary Computation, vol 11, no. 6, pp 712-731 (2007).

[16] Zhang, M., Zhao, S. and Wang, X., Multi-objective evolutionary algorithm based on adaptive discrete Differential Evolution. IEEE Congress on Evolutionary Computation (CEC 2009), pp. 614-621, (2009).

[17] Zitzler, E. and Thiele, L. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto evolutionary algorithm. IEEE Transaction on Evolutionary Computation, vol. 3, pp. 257-271 (1999).

[18] Zitzler, E., Laumanns, M. and Thiele, L. SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In Proceeding. of Evolutionary Methods for Design, Optimization and Control with Application to Industrial Problems (EUROGEN 2001). 95-100, Athena, Greece, (2001).