HEMH2: An Improved Hybrid Evolutionary Metaheuristics for 0/1 Multiobjective Knapsack Problems

Ahmed Kafafy, Ahmed Bounekkar, and Stéphane Bonnevay

Laboratoire ERIC - Université Claude Bernard Lyon 1,
Ecole Polytechnique Universitaire, 15 Boulevard Latarjet,
69622 Villeurbanne cedex, France
ahmedkafafy80@gmail.com,{bounekkar,stephane.bonnevay}@univ-lyon1.fr

Abstract. Hybrid evolutionary metaheuristics tend to enhance search capabilities, by improving intensification and diversification, through incorporating different cooperative metaheuristics. In this paper, an improved version of the Hybrid Evolutionary Metaheuristics (HEMH) [7] is presented. Unlike HEMH, HEMH2 uses simple inverse greedy algorithm to construct its initial population. Then, the search efforts are directed to improve these solutions by exploring the search space using binary differential evolution. After a certain number of evaluations, path relinking is applied on high quality solutions to investigate the non-visited regions in the search space. During evaluations, the dynamic-sized neighborhood structure is adopted to shrink/extend the mating/updating range. Furthermore, the Pareto adaptive epsilon concept is used to control the archiving process with preserving the extreme solutions. HEMH2 is verified against its predecessor HEMH and the MOEA/D [13], using a set of MOKSP instances from the literature. The experimental results indicate that the HEMH2 is highly competitive and can achieve better results.

Keywords: Hybrid Metaheuristics, Adaptive Binary Differential Evolution, Path Relinking, 0/1 Multiobjective Knapsack Problems.

1 Introduction

Multiobjective combinatorial optimization problems (MOCOP) are often characterized by their large size and the presence of multiple and conflicting objectives. The basic task in multiobjective optimization is to identify the set of Pareto optimal solutions or even a good approximation to Pareto front (PF). Despite the progress in solving MOCOP exactly, the large size often means that metaheuristics are required for their solution in a reasonable time.
Solving multiobjective optimization problems (MOOP) using evolutionary algorithms (MOEAs) has been investigated by many authors [3, 13, 15, 16]. Pareto dominance based MOEAs such as SPEA [15], NSGAII [3] and SPEA2 [16] have been dominantly used in the recent studies. Based on many traditional mathematical programming methods for approximating PF [10], the approximation of PF can be decomposed into a set of single objective subproblems. This idea is adopted in MOGLS [5] and MOEA/D [13]. 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* metaheuristics. Search algorithms must balance between sometimes-conflicting two goals, intensification and diversification [1]. The design of hybrid metaheuristics can give the ability to control this balance [9]. This paper tends to improve the hybrid evolutionary metaheuristics (HEMH) proposed in [7] through developing a new version called HEMH2 with two variants HEMH_{de} and HEMH_{pr}. The motivations of this work are to overcome the limitations from which the performance of HEMH suffers. In HEMH2, an adaptive binary differential evolution is used as a reproduction operator rather than classical crossover. Also, path relinking is applied on high quality solutions generated after a certain number of evaluations. Moreover, all improvement proposals and their effects on the search process will be discussed in details. The rest of the paper is organized as follows: section 2 presents some of the basic concepts and definitions. HEMH framework is reviewed in section 3. Section 4 explains the adaptive binary differential evolution. Path relinking strategy is discussed in section 5. The proposed HEMH2 and its variants are presented in section 6. Additionally, the experimental design and results are involved in sections 7 and 8 respectively. Finally, the conclusions and future works are involved in section 9.

2 Basic Concepts and Definitions

Without loss of generality, the MOOP can be formulated as:

$$\begin{gathered} Maximize\, F(x) = (f_1(x), f_2(x), \cdots, f_m(x)) \\ Subject\, to:\ x \in \Omega \end{gathered} \tag{1}$$

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 $x \in \mathbb{Z}$, the MOOP is called 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 said to ϵ-dominate a solution y for some $\epsilon > 0$ (noted as: $x \succcurlyeq_\epsilon y$) if and only if: $f_i(x) \geq (1+\epsilon) f_i(y)$, $\forall i \in \{1, \cdots, m\}$.

Definition 3: A solution x is called efficient (*Pareto-optimal*) if: $\nexists\, y \in \Omega : y \succcurlyeq x$

Definition 4: The Pareto optimal set (P^*) is the set of all efficient solutions:

$$P^* = \{x \in \Omega \,|\, \nexists\, y \in \Omega\,,\ y \succcurlyeq x\}$$

Definition 5: The Pareto front (PF) is the image of P^* in the objective space:

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

Definition 6: Given a reference point r^* and a weight vector $\Lambda = [\lambda_1, \cdots, \lambda_m]$ such that $\lambda_i \geq 0, \forall i \in \{1, \cdots, m\}$, the *weighted sum* (F^{ws}) and the *weighted Tchebycheff* (F^{Tc}) scalarizing functions corresponding to (1) can be defined as:

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

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

Definition 7: Given a set of m knapsacks and a set of n items, the 0/1 Multi-objective Knapsack Problem (MOKSP) can be written as:

$$\begin{gathered} Max:\ f_i(x) = \sum_{j=1}^{n} c_{ij} x_j,\ \forall i \in \{1, \cdots, m\} \\ s.t.:\ \sum_{j=1}^{n} w_{ij} x_j \leq W_i,\ \forall i \in \{1, \cdots, m\} \\ x = (x_1, \cdots, x_n)^T \in \{0,1\}^n \end{gathered} \tag{4}$$

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 put in all knapsacks.
The MOKSP is NP-hard and can model a variety of applications.

3 HEMH, an overview

In HEMH, a combination of different cooperative metaheuristics is provided to handle 0/1 MOKSP. The MOEA/D framework [13] is adopted to carry out the combination. The *weighted sum* defined in (2) is considered to decompose the MOKSP in (4) into a set of N single objective subproblems, based on a set of N evenly distributed weight vectors $\{\Lambda^1, \cdots, \Lambda^N\}$. HEMH attempts to simultaneously optimize these subproblems. The HEMH framework consists of the following:

- A population P of N individuals, $P = \{x^1, \cdots, x^N\}$, where x^i represents the current solution of the i^{th} subproblem.
- A set of N evenly distributed weight vectors $\{\Lambda^1, \cdots, \Lambda^N\}$, correspond to the N subproblems. Each $\Lambda = [\lambda_1, \cdots, \lambda_m]$ has m components correspond to m-objectives, such that: $\sum_{i=1}^{m} \lambda_i = 1$, $\forall \lambda_i \in \{0/H, \cdots, H/H\}$, and $H \in \mathbb{Z}^+$.
- A neighborhood B_i for each subproblem $i \in \{1, \cdots, N\}$, which includes all subproblems with the T closest weight vectors $\{\Lambda^{i1}, \cdots, \Lambda^{iT}\}$ to Λ^i.
- An *archive* to collect all efficient solutions explored over the search process.

The HEMH consists of two basic phases, *initialization* and *main loop*. In the initialization phase, an initial population of high quality solutions is constructed by applying DMGRASP on each subproblem. Then, the search efforts are concentrated on the promising regions to explore new efficient solutions. In the main loop phase, for each subproblem i, the mating/updating range M_i is chosen to be either the neighborhood B_i or the whole population P. Then, two individuals are randomly selected from M_i for reproduction. Single point crossover and mutation or greedy randomize path relinking is applied on the selected individuals to generate a new offspring, which is used to update M_i and the archive. This process is repeated until a certain number of evaluations. We refer to [7] for more details. The limitations that affect the HEMH performance are briefed as:

- Despite using DMGRASP achieves high quality initial solutions, it consumes more time and evaluations especially with large populations. Thus, the second phase will not have enough chance to improve the search process.
- Collecting all efficient solutions causes waste in time and storage space especially in many objective cases. So, archiving process should be controlled.
- For each subproblem i, the mating/updating range M_i is either the fixed (static) size neighborhood B_i or the whole population P. This may cause less execution of path relinking, or consume more time.
- Reproduction is made only by single point crossover and mutation or greedy randomized path relinking.
- Path relinking adopts single bit flipping per move and also uses local search to improve the generated solution. This causes more time consumption.

From the above, this paper presents an improved version of HEMH called HEMH2 with two variants HEMH$_{de}$ and HEMH$_{pr}$, which have the ability to overcome those limitations and can achieve an enhanced performance.

4 Adaptive Binary Differential Evolution

Differential evolution (DE) is a simple and efficient evolutionary algorithm to solve optimization problems mainly in continuous search domains [2, 11]. DE's success relies on the differential mutation, that employs difference vectors built with pairs of candidate solutions in the search domain. Each difference vector is scaled and added to another candidate solution, producing the so-called mutant vector. Then, DE recombines the mutant vector with the parent solution to generate a new offspring. The offspring replaces the parent only if it has an equal or better fitness. DE has some control parameters as the mutation factor F, that used to scale the difference vectors, and the crossover rate CR. In this paper, an adaptive binary DE strategy is introduced to improve the exploration capabilities of HEMH instead of classical crossover and mutation. This strategy is described in Alg. 1. Given a population P of N individuals, where each individual represented by a n-component 0/1 vector. The main idea is to select at random three distinct individuals x^a, x^b and x^c 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 binary differential mutation on the selected individuals according to (5). First, the *difference vector* is calculated by applying logical XOR on the two parent differential individuals x^b and x^c. Then, v^i is determined by applying logical OR on the parent based individual x^a and the difference vector previously obtained. Finally, the new generated offspring u^i is produced by applying crossover according to (6).

$$v^i = x^a + (x^b \oplus x^c) \tag{5}$$

$$u^i_j = \begin{cases} v^i_j & \text{if } rnd(j) \leq CR, \text{ or } j \in e, \forall j = 1, \cdots, n. \\ x^i_j & \text{otherwise, } \forall j = 1, \cdots, n. \end{cases} \tag{6}$$

where $rnd(j) \in [0, 1]$ is a random number generated for the j^{th} component, n is the individual length, e is a random sequence selected from the range $\{1, \cdots, n\}$ to insure that at least one component of u^i is contributed by v^i and $CR \in [0, 1]$ denotes crossover rate. Here, CR is adapted periodically to avoid the premature convergence based on (7) proposed in [14].

$$CR = CR_0 \cdot e^{(-a.(G/G_{max}))} \tag{7}$$

where G and G_{max} are the current and the maximum evolutionary generations, CR_0 is the initial crossover rate. a is a constant.

Algorithm 1 :ABDEVOL$(x, x^a, x^b, x^c, CR_0, a)$

Inputs:
x: *Current solution*
x^a, x^b, x^c: *Parents individuals*
$CR_0 \in [0, 1]$: *Crossover rate*
a: *Plus constant*

Begin:
$CR \leftarrow CR_0 \cdot e^{(-a.(G/G_{max}))}$; ▷ *Adapt CR*
for all $j \in \{1, \cdots, n\}$ **do**: ▷ *For all items*
  $v_j = x^a_j + (x^b_j \oplus x^c_j)$; ▷ *Binary Diff. Mutation*
  $u_j \leftarrow \begin{cases} v_j & \text{if } rnd(j) \leq CR \vee j \in e, \\ x_j & \text{otherwise.} \end{cases}$
end for
return u;
End

Algorithm 2 :INVERSEGREEDY(x, Λ)

Inputs:
x: *Initial Solution*
$\Lambda = [\lambda_1, \cdots, \lambda_m]$: *Search direction*

Begin: $CL \leftarrow \varnothing$;
for all $j \in \{1, \cdots, n\}$ **do:** $x_j \leftarrow 1$; ▷ *Put all*
while $\exists j | j \notin CL \wedge Min^n_{j=1} \left(\frac{\sum_{i=1}^m \lambda_i c_{ij}}{\sum_{i=1}^m w_{ij}} \right)$ **do**:
  $CL \leftarrow$ APPEND(j);
end while
while x violates any constraint in (4) **do**:
  $j \leftarrow$EXTRACTTHEFIRST(CL);
  $x \leftarrow$REMOVEITEM(x, j);
end while
return x;
End

Algorithm 3 :PathRelinking(x^s, x^t, Λ)

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

Begin:
$x^* \leftarrow$GetTheBestOf(x^s, x^t, Λ);
if $x^* \neq x^s$ **then:** Swap(x^s, x^t);
$z^* \leftarrow F^{ws}(x^*, \Lambda); CL \& CL_{cmp} \leftarrow \varnothing$;
while $\exists j | j \notin CL \wedge x_j^s \neq x_j^t \wedge x_j^s = 0 \wedge Max_{j=1}^{n} \left(\sum_{i=1}^{m} \lambda_i c_{ij} / \sum_{i=1}^{m} w_{ij}\right)$ **do**:
  $CL \leftarrow$ Append(j);
end while
while $\exists j | j \notin CL_{cmp} \wedge x_j^s \neq x_j^t \wedge x_j^s = 1 \wedge Min_{j=1}^{n} \left(\sum_{i=1}^{m} \lambda_i c_{ij} / \sum_{i=1}^{m} w_{ij}\right)$ **do**:
  $CL_{cmp} \leftarrow$ Append(j);
end while
$x \leftarrow x^s$;
$\Delta(x, x^t) \leftarrow |\{j \in \{1, \cdots, n\} : x_j \neq x_j^t\}|$;
while $\Delta(x, x^t) \geq 2$ **do**: ▷ *Relinking loop*
  if $|CL| \neq 0 \wedge |CL_{cmp}| \neq 0$ **then**:
    $\ell^1 \leftarrow$ ExtractTheFirst(CL)
    $\ell^2 \leftarrow$ ExtractTheFirst(CL_{cmp})
  else if $|CL| > 1$ **then**:
    $\ell^1 \leftarrow$ ExtractTheFirst(CL)
    $\ell^2 \leftarrow$ ExtractTheFirst(CL)
  else:
    $\ell^1 \leftarrow$ ExtractTheFirst(CL_{cmp})
    $\ell^2 \leftarrow$ ExtractTheFirst(CL_{cmp})
  end if
  $x \leftarrow$ FilppBits(x, 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
  $\Delta(x, x^t) \leftarrow |\{j \in \{1, \cdots, n\} : x_j \neq x_j^t\}|$;
end while
return x^*;
End

Algorithm 4 :HEMH2($N, T, t, \epsilon, \gamma, CR_0, a$)

Inputs:
N: *Population size or no. of subproblems*
T,: *Min. neighborhood size*
t: *Max. replaced solutions*
ϵ: *Min. hamming distance*
γ: *Controls Path-relinking execution*
$CR_0 \in [0, 1]$, a: *Crossover rate, Plus constant*

Begin:
$W_v \leftarrow \{\Lambda^1, \cdots, \Lambda^N\}$; ▷ *Set of N weight vectors*
for $i \leftarrow 1$ to N **do**: ▷ *Construct Neighborhoods*
  $B_i \leftarrow [i1, \cdots, iN]$; ▷ *where* $\Lambda^{i1}, \cdots, \Lambda^{iN}$ *are*
end for ▷ *increasingly sorted by ED to* Λ^i
$Arch \leftarrow \varnothing$; ▷ *Empty archive*
$Evl \leftarrow 0$;
for $i \leftarrow 1$ to N **do**: ▷ *Initialization phase*
  $x^i \leftarrow$InverseGreedy(x^i, Λ^i);
  $P \leftarrow$AddSubProblem(x^i, Λ^i);
  $Extremes \leftarrow$Update(x^i);Update(Evl);
end for
while $(Evl < Mevls)$ **do**: ▷ *Main Loop*
  for $i \leftarrow 1$ to N **do**: ▷ *for each subproblem i*
    $x^a, x^b, x^c \leftarrow$ Selection(B_i, i);
      ▷ *Where:* $x^i \neq x^a \neq x^b \neq x^c$
    $x^j, x^k \leftarrow$ RandSelection(x^a, x^b, x^c);
    $D \leftarrow \Delta(x^j, x^k)$; ▷ *Hamming distance*
    $E \leftarrow \gamma \times Mevls$; ▷ *min. eval for PR*
    if $(D \geq \epsilon \wedge Eval \geq E)$ **then**:
      $y \leftarrow$PathRelinking(x^j, x^k, Λ^i);
    else:
      $u \leftarrow$ABDEvol$(x^i, x^a, x^b, x^c, CR_0, a)$;
      $y \leftarrow$Repair(u, Λ^i)
    end if
    $P \leftarrow$UpdateSolutions(y, t, B_i);
    $Arch \leftarrow$ UpdateArchive$^{pa\epsilon}(y)$;
    $Extremes \leftarrow$Update(y);Update(Evl);
  end for
end while
$Arch \leftarrow$ AddExtremes$(Extremes)$;
return $Arch$;
End

5 Path Relinking

Path relinking generates new solutions by exploring trajectories that connect high quality solutions. Starting from the starting solution x^s, path relinking builds a path in the neighborhood space that leads toward the guiding solution x^t. The relinking procedure has a better chance to investigate in more details the neighborhood of the most promising solutions if relinking starts from the best of x^s and x^t [12]. In this paper, a path relinking with two bits per move [8] is used as an intensification strategy, integrated with the adaptive binary DE. It will be invoked on the higher generations to guarantee applying the relinking process on high quality solutions. Alg. 3 describes the proposed procedure. 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. The procedure builds the path that connects x^s with x^t gradually by creating intermediate points through flipping two bits/move. Initially, the intermediate x is set to x^s. Then, the Hamming distance $\Delta(x, x^t)$ is calculated. The next move is carried out by flipping 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 the second elements 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} and x_{ℓ^2}. If x is infeasible, then x is repaired to get y that updates both x^* and z^*. This process is repeated until there is only two unmatched items between the current x and the guiding x^t.

6 The Proposed HEMH2

Motivated by the results achieved in [8], some proposals are adopted to improve HEMH performance and to overcome the limitations discussed. The main differences between HEMH2 and its predecessor are briefly presented as follows:

- Initial population is created using the simple inverse greedy algorithm (Alg.2) for each search direction rather than DMGRASP. The quality of the obtained initial solutions will be affected, but this will give a better chance to the second phase to improve and enhance the search process.
- Instead of collecting all efficient solutions, the Pareto-adaptive epsilon dominance ($pa\epsilon$-dominance) [4] is adopted to control the quality and the quantity of the efficient solutions collected in the archive.
- Dynamic neighborhood size that permit to shrink/extend the neighborhood for each subproblem is considered. Consequently, the parent solutions of a subproblem are always selected from its neighborhood. This can overcome the limitations of the binary differential mutation.
- The adaptive binary DE is used as a reproduction operator instead of crossover and mutation beside the path relinking.
- Path relinking is applied only after a certain number of evaluations as a post optimization strategy. This action guarantees the existence of high quality solutions. Moreover, path relinking flips two bits at each relinking step.
- In HEMH2, local search is avoided either after path relinking or after inverse greedy construction as proposed in HEMH.

Now, the reasons behind the above proposals are explained. Firstly, there is no doubt that generating the initial population using DMGRASP can achieve better quality solutions, but it forces us to use small populations. In some cases, local search highly consumes more time and evaluations to investigate a small specified region in the search space. Consequently, the *main loop* phase has a small chance to improve the search process. To overcome this limitation, the inverse greedy construction is proposed. From the empirical results, the inverse greedy obtains solutions as close as possible to the boundary regions than simple greedy construction. Secondly, using $pa\epsilon$-dominance will control the size of the archive, especially in many objective cases. Consequently, saving more resources of time and storage space with persevering the quality of the collected solutions. Thirdly, the poor performance of the binary differential mutation occurs when treating differential individuals with large Hamming distance. Selecting parents from the whole population can encourage this scenario. In HEMH2, parents of each subproblem are always selected from its neighborhood which has a dynamic size, this guarantees obtaining individuals with suitable Hamming

distances. Fourthly, the adaptive binary DE empirically has the ability to explore the search space better than classical crossover and mutation. Thus, the performance of HEMH will be improved by adopting adaptive binary DE for reproduction rather than crossover. Finally, the proposed path relinking applies two bits flipping/move, that minimizes the whole relinking time. Also, avoiding local search saves both time and evaluations.
In Alg. 4, the HEMH2 procedure is introduced. Firstly, a set of N evenly distributed weight vectors is created. Then, the neighborhood structure is constructed for each subproblem i by assigning all subproblems sorted increasingly by the Euclidean distance between their weight vectors and the current weight vector Λ_i. After that, an initial population P is created by applying the inverse greedy (Alg.2) for each search direction. Now, the main loop is executed until achieving the maximum evaluations $Mevls$ (line 13). For each subproblem i, SELECTION routine is invoked to determine the *current size* of the neighborhood B_i such that: $|B_i| = T + r$, where T and r represent the number of different and repeated solutions in B_i respectively. This means, the SELECTION routine extends B_i size to guarantee the existence of at least T different solutions and randomly selects three of them x^a, x^b and x^c for reproduction. Two of the three selected parents x^j, x^k are chosen randomly. Then, path relinking is used only if the Hamming distance $\Delta(x^j, x^k)$ is greater than a certain value ϵ and the number of evaluations $Eval$ exceeds a certain ratio γ of the maximum evaluations $Mevls$ allowed to guarantee applying path relinking on high quality solutions. Else, the adaptive binary DE is applied to generate a new offspring y. The new generated offspring y is evaluated and used to update the neighborhood (B_i) according to the parameter t, which controls the number of replaced solutions. The Archive is also updated by y according to $pa\epsilon$-dominance [4]. Finally, the Extreme solutions are added to the archive which is returned as an output.
In order to study the effects of both adaptive binary DE and path relinking operators distinctly, two additional algorithms variants called HEMH$_{de}$ and HEMH$_{pr}$ are considered. Both of them have the same procedure as HEMH2 explained in Alg. 4 except that HEMH$_{de}$ only adopts adaptive binary DE for reproduction. Whereas, HEMH$_{pr}$ replaces the adaptive binary DE in HEMH2 procedure by crossover and mutation.

7 Experimental Design

In this paper, both MOEA/D and HEMH are involved to verify our proposals. The comparative study for different algorithms is carried out on a set of test instances from the literature [15] listed in Table 1. All experiments are performed on a PC with Intel Core i5-2400 CPU, 3.10 GHz and 4.0 GB of RAM.

7.1 Parameter settings

Here, the different parameters used for each algorithm are discussed. For MOEA/D and our proposals HEMH$_{de}$, HEMH$_{pr}$ and HEMH2, the parameter H which controls the population size N by the relation ($N = C_{m-1}^{H+m-1}$) is determined for each instance in Table 1 according to the complexity. The initial population used

in MOEA/D 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. For HEMH, the parameter $\acute{H}$ that controls the population size $\acute{N}$ is also considered. HEMH uses the parameters α=0.1 and β=0.5. The maximum number of evaluations ($Mevls$) is used as a stopping criterion for each algorithm. For fair comparison, the same archiving strategy based on $pa\epsilon$-dominance [4] are applied to each algorithm to get the final approximation set. The $pa\epsilon$-dominance uses the archive size (A_s) listed in Table 1. HEMH applies the path relinking proposed here. Single-point crossover and standard mutation are considered. Mutation was performed for each item independently with probability ($1/n$). The other control parameters are listed in Table 2. Finally, the statistical analysis is applied on 30 independent runs for each instance.

Table 1: Set of knapsack instances

Inst.	m	n	$N(H)$	$\acute{N}(\acute{H})$	A_s	$Mevls$
KS252	2	250	150(149)	75(74)	150	75000
KS502	2	500	200(199)	100(99)	200	100000
KS752	2	750	250(249)	125(124)	250	125000
KS253	3	250	300(23)	153(16)	200	100000
KS503	3	500	300(23)	153(16)	250	125000
KS753	3	750	300(23)	153(16)	300	150000
KS254	4	250	364(11)	165(8)	250	125000
KS504	4	500	364(11)	165(8)	300	150000
KS754	4	750	364(11)	165(8)	350	175000

Table 2: Set of common parameters

Parameters	HEMH			
	-	**-*de***	**-*pr***	**-2**
Neighborhood size: T	10	10	10	10
Max. replaced sols: t	2	2	2	2
Parents selection: δ	0.9	-	-	-
Min. support: σ	{1}	-	-	-
Ratio controls PR: γ	-	-	0.8	0.8
Min. Ham. Distance: ϵ	10	-	10	10
Crossover rate: CR_0	-	0.4	-	0.4
Plus constant: a	-	2	-	2

7.2 Assessment Metrics

Let $A, B \subset \Re^m$ be two approximations to PF, $P^*, r^* \subset \Re^m$ be a reference set and a reference point respectively. The following metrics can be expressed as:

1. **The Set Coverage** (I_C) [15] is used to compare two approximation sets. The function I_C maps the ordered pair (A, B) to the interval $[0, 1]$ as:
$$I_C(A, B) = |\{u|u \in B, \exists v|v \in A : v \succcurlyeq u\}| / |B| \tag{8}$$
where $I_C(A, B)$ is 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) [15] for a set A is defined as:
$$I_H(A) = \mathcal{L}(\cup_{u \in A} \{y|u \succcurlyeq y \succcurlyeq r^*\}) \tag{9}$$
where $\mathcal{L}$ is the Lebesgue measure of a set. $I_H(A)$ describes the size of the objective space that is dominated by A and dominates r^*. We use the referenced indicator such that: $I_{RH}(A) = I_H(P^*) - I_H(A)$ and r^* is the origin.
3. **The Generational** (I_{GD}) and Inverted Generational Distance (I_{IGD}) of a set A are defined as:
$$\begin{aligned} I_{GD}(A, P^*) &= \tfrac{1}{|A|} \textstyle\sum_{u \in A} \{min_{v \in P^*} d(u, v)\} \\ I_{IGD}(A, P^*) &= \tfrac{1}{|P^*|} \textstyle\sum_{u \in P^*} \{min_{v \in A} d(u, v)\} \end{aligned} \tag{10}$$
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.

4. **R-indicator** (I_{R_3}) [6] uses a set of utility functions u, which can be any scalar function. both of weighted sum and weighted Tchebycheff functions with a sufficiently large set of evenly distributed normalized weight vectors (Λ) are used. I_{R_3} can be evaluated as follows:

$$I_{R_3}(A, P^*) = \frac{\sum_{\lambda \in \Lambda} [u^*(\lambda, P^*) - u^*(\lambda, A)] / u^*(\lambda, P^*)}{|\Lambda|} \quad (11)$$

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 $\sum_{i=1}^{m} \lambda_i = 1$.

Here, the reference set P^* for each instance is formed by gathering all efficient solutions found by all algorithms in all runs. Also, all approximation sets are normalized in the range [1,2].

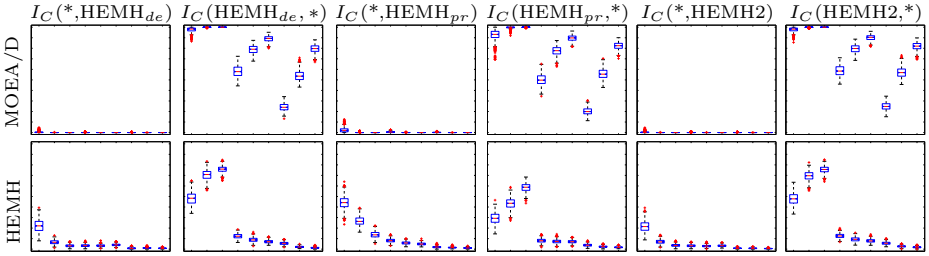

Fig. 1: Results of I_C indicator

8 Experimental Results

Here, the different simulation results are shown in details. Firstly, Fig.1 depicts the results of I_C metric. It contains a chart (with scale 0 at the bottom and 1 at the top) for the ordered pairs depicted. 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 1 (from top to down) respectively. It is clear from the results in Fig.1 that all proposals HEMH2, HEMH$_{de}$ and HEMH$_{pr}$ outperform the original MOEA/D in all test instances. Whereas, Both HEMH2 and HEMH$_{de}$ outperform HEMH for all test instances. Also, HEMH$_{pr}$ slightly performs better than HEMH in most test instances.

In Tables 3, 4, 5 and 6, the average values of the indicators I_{RH}, I_{GD}, I_{IGD} and I_{R_3} are listed respectively. Additionally, Figures 2, 3, 4 and 5 visualize theses values in the same order. Each of those tables contains the average values achieved over 30 independent runs for each test instance for each algorithm. Based on those results, it is clear that the proposals HEMH2, HEMH$_{de}$ and HEMH$_{pr}$ outperform the original MOEA/D and HEMH. Sience they have the minimum average values. Moreover, the proposed HEMH2 and HEMH$_{de}$ has the superiority for all test instances, followed by HEMH$_{pr}$, which confirm the results of I_C metric.

In some cases, it is observed that HEMH$_{de}$ achieves better results than HEMH2 and HEMH$_{pr}$, depending on adaptive binary DE, which reflects that the adaptive

binary DE capabilities in exploring the search space is more effective than path relinking and classical crossover and mutation.

Table 3: Average Referenced Hypervolume (I_{RH})

Inst.	Algorithms				
	MOEA/D	HEMH	HEMH$_{de}$	HEMH$_{pr}$	HEMH2
KP252	4.66E-02	1.02E-02	**6.07E-03**	9.33E-03	6.08E-03
KP502	5.67E-02	1.55E-02	5.57E-03	1.16E-02	**5.56E-03**
KP752	4.73E-02	1.64E-02	**3.84E-03**	7.34E-03	3.94E-03
KP253	2.24E-01	1.21E-01	8.85E-02	1.09E-01	**8.77E-02**
KP503	2.76E-01	1.16E-01	**7.39E-02**	9.01E-02	**7.39E-02**
KP753	2.89E-01	8.85E-02	6.48E-02	7.63E-02	**6.39E-02**
KP254	8.56E-01	5.55E-01	**4.26E-01**	4.90E-01	4.27E-01
KP504	1.07E+00	4.53E-01	3.96E-01	4.10E-01	**3.84E-01**
KP754	1.23E+00	3.93E-01	3.54E-01	3.64E-01	**3.41E-01**

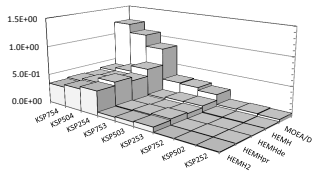

Fig. 2: Average Results

Table 4: Average Generational Distance (I_{GD})

Inst.	Algorithms				
	MOEA/D	HEMH	HEMH$_{de}$	HEMH$_{pr}$	HEMH2
KP252	1.50E-03	5.17E-04	3.40E-04	5.74E-04	**3.37E-04**
KP502	1.53E-03	4.41E-04	**1.44E-04**	3.12E-04	1.54E-04
KP752	1.33E-03	4.77E-04	**7.52E-05**	1.91E-04	7.81E-05
KP253	1.98E-03	6.83E-04	3.92E-04	6.82E-04	**3.88E-04**
KP503	2.25E-03	4.95E-04	2.76E-04	4.25E-04	**2.70E-04**
KP753	2.16E-03	3.63E-04	2.17E-04	2.77E-04	**2.07E-04**
KP254	2.52E-03	1.14E-03	**6.07E-04**	9.13E-04	6.14E-04
KP504	3.45E-03	6.63E-04	4.08E-04	5.14E-04	**3.62E-04**
KP754	3.79E-03	4.91E-04	2.88E-04	3.56E-04	**2.53E-04**

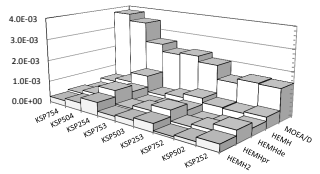

Fig. 3: Average Results

Table 5: Average Inv. Generational Dist. (I_{IGD})

Inst.	Algorithms				
	MOEA/D	HEMH	HEMH$_{de}$	HEMH$_{pr}$	HEMH2
KP252	9.32E-04	4.83E-04	**3.49E-04**	4.51E-04	3.50E-04
KP502	7.31E-04	2.72E-04	**1.31E-04**	2.10E-04	1.32E-04
KP752	5.41E-04	2.43E-04	**7.89E-05**	1.14E-04	7.95E-05
KP253	6.31E-04	4.48E-04	**3.83E-04**	4.41E-04	3.84E-04
KP503	5.28E-04	3.33E-04	**2.58E-04**	2.99E-04	2.60E-04
KP753	4.55E-04	2.46E-04	**1.93E-04**	2.24E-04	1.95E-04
KP254	6.75E-04	5.69E-04	**4.88E-04**	5.33E-04	4.91E-04
KP504	5.95E-04	4.04E-04	**3.46E-04**	3.67E-04	3.46E-04
KP754	5.46E-04	3.28E-04	2.71E-04	2.85E-04	**2.70E-04**

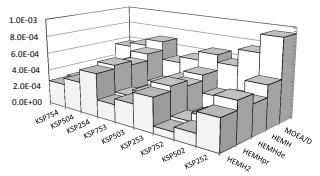

Fig. 4: Average Results

Table 6: Average R_3 indicator (I_{R_3})

Inst.	Algorithms				
	MOEA/D	HEMH	HEMH$_{de}$	HEMH$_{pr}$	HEMH2
KP252	6.05E-03	1.88E-03	**1.29E-03**	2.32E-03	1.30E-03
KP502	7.30E-03	2.06E-03	**6.89E-04**	1.62E-03	7.12E-04
KP752	6.88E-03	2.47E-03	**5.46E-04**	1.21E-03	5.54E-04
KP253	1.11E-02	5.62E-03	4.49E-03	5.24E-03	**4.48E-03**
KP503	1.34E-02	5.16E-03	3.83E-03	4.60E-03	**3.78E-03**
KP753	1.42E-02	4.25E-03	3.38E-03	3.92E-03	**3.32E-03**
KP254	1.61E-02	9.70E-03	7.86E-03	8.84E-03	**7.84E-03**
KP504	2.02E-02	7.91E-03	7.18E-03	7.40E-03	**7.06E-03**
KP754	2.39E-02	7.02E-03	6.02E-03	6.26E-03	**5.82E-03**

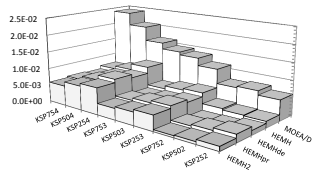

Fig. 5: Average Results

9 Conclusion

In this paper, an improved hybrid evolutionary metaheuristics HEMH2 and two other variants called HEMH$_{de}$ and HEMH$_{pr}$ were proposed to enhance HEMH

performance. The HEMH2 adopts the inverse greedy procedure in its initialization phase. Both adaptive binary DE and path relinking operators are used. The HEMH$_{de}$ only uses adaptive binary DE. Whereas, HEMH$_{pr}$ uses crossover and mutation beside path relinking. The proposals were compared with the original MOEA/D and HEMH using a set of MOKSP instances from the literature. A set of quality indicators was also used to assess the performance. The experimental results indicate the superiority of all proposals over the original MOEA/D and HEMH based on the assessment indicators used in this study. According to the results, we can deduce that the adaptive binary DE included in both HEMH2 and HEMH$_{de}$ has better exploration capabilities which overcome the local search capabilities contained in the original HEMH. Therefore both of HEMH2 and HEMH$_{de}$ outperform HEMH. In some cases, HEMH$_{de}$ can achieve highly competitive results compared with HEMH2 based on the adaptive binary DE which can achieve better performance than path relinking. In the future work, the tuning parameters of HEMH2 and its variants will be investigated. Moreover, other metaheuristics will be studied to improve the performance of HEMH2 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. Hernández-Díaz, A. G., Santana-Quintero, L. V. , Coello, C. A., Luque, J. M., Pareto-adaptive epsilon-dominance. Evolutionary Computation, vol. 15, no. 4, pp. 493-517, (2007).
5. 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).
6. Knowles, J., Corne, D.; On metrics for comparing nondominated sets. In IEEE International conference in E-Commerce technology, vol. 1 pp. 711-716(2002).
7. Kafafy A., Bounekkar A., Bonnevay S.:A hybrid evolutionary metaheuristics (HEMH) applied on 0/1 multiobjective knapsack problems.GECCO-2011: 497-504.
8. Kafafy A., Bounekkar A., Bonnevay S.:Hybrid metaheuristics based on MOEA/D for 0/1 multiobjective knapsack problems: A comparative study, IEEE World Congress on Computational Intelligence (WCCI2012), Brisbane, June 10-15, pp.3616-3623.
9. 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).
10. Miettinen, K. Nonlinear Multiobjective Optimization. Boston, MAA: Kluwer, (1999).
11. 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).
12. 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).
13. 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).
14. 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).
15. 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).
16. 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).