VNU Journal of Science: Comp. Science & Com. Eng., Vol. 32, No. 1 (2016) 10–32

Hyper-volume Evolutionary Algorithm

Khoi Nguyen Le[1,*], Dario Landa-Silva[2]

[1]*VNU University of Engineering and Technology, Hanoi, Vietnam*
[2]*The University of Nottingham, Nottingham, United Kingdom*

Abstract

We propose a multi-objective evolutionary algorithm (MOEA), named the Hyper-volume Evolutionary Algorithm (HVEA). The algorithm is characterised by three components. First, individual fitness evaluation depends on the current Pareto front, specifically on the ratio of its dominated hyper-volume to the current Pareto front hyper-volume, hence giving an indication of how close the individual is to the current Pareto front. Second, a ranking strategy classifies individuals based on their fitness instead of Pareto dominance, individuals within the same rank are non guaranteed to be mutually non-dominated. Third, a crowding assignment mechanism that adapts according to the individual's neighbouring area, controlled by the *neighbouring area radius* parameter, and the archive of non-dominated solutions. We perform extensive experiments on the multiple 0/1 knapsack problem using different greedy repair methods to compare the performance of HVEA to other MOEAs including NSGA2, SEAMO2, SPEA2, IBEA and MOEA/D. This paper shows that by tuning the **neighbouring area radius** parameter, the performance of the proposed HVEA can be pushed towards better convergence, diversity or coverage and this could be beneficial to different types of problems.

Received 05 December 2015, revised 20 December 2015, accepted 31 December 2015

Keywords: Multi-objective Evolutionary Algorithm, Pareto Optimisation, Hyper-volume, Knapsack Problem.

1. Introduction

The development of heuristic and evolutionary techniques to solve real-world multi-objective optimisation problems is an an active research area. In Pareto-based multi-objective optimisation, a set of non-dominated solutions, also known as Pareto front, is sought so that the decision-maker can select the most appropriate one. Evolutionary xalgorithms and other population based heuristics seem well suited to deal with Pareto based multi-objective optimisation problems because they can evolve a population of solutions towards the Pareto-optimal front in a single run. A good multi-objective evolutionary algorithm (MOEA) should be able to obtain Pareto fronts that are both well-distributed and well-converged. An important issue in a MOEA is how to establish superiority between solutions within the population, i.e. how to assess solution fitness in a multi-objective sense. In this paper, we proposed the Hyper-Volume Evolutionary Algorithm (HVEA), a MOEA that employs the concepts of volume dominance proposed by Le et at [1] to assess solution fitness in multi-objective optimisation.

The paper is organised as follows. Section 2 describes the HVEA algorithm. In Section 3, most well-known and recent MOEAs in the literature together with the multiple 0/1 knapsack problem are discussed. In Section 4, the experimental results and discussion are presented. Then some conclusions and perspectives are discussed in Section 5.

[*] Corresponding author. Email: khoi.n.le@vnu.edu.vn

2. Hyper-volume Evolutionary Algorithm

We propose a new approach to multi-objective optimisation, the *Hyper-volume Evolutionary Algorithm* (HVEA). HVEA deploys techniques that are well established in the literature as well as presents new ones in order to find the Pareto front of the problem. As other population-based MOEAs, the proposed HVEA

- Uses a population of current solutions and an archive for storing the best solutions found so far,
- Assigns scalar fitness values to individuals and uses the Pareto dominance relationship,
- Employs a crowding strategy to maintain the diversity of the population and, if necessary, control the size of the archive during the search.

However, HVEA is distinguished by four features:

- The fitness of an individual is calculated using the hyper-volume of that individual and the hyper-volume of the current representative Pareto front.
- Individuals are ranked based on their fitness values, individuals having fractional difference in fitness values between them are classified into the same rank.
- A niching technique to preserve the diversity of the population based on distance between the individual and other solutions in its neighbouring area.
- Offspring that improve the current Pareto front (in other words offspring that dominate individuals in the current Pareto set) are guaranteed to be selected not only for the archive but also for the tournament selection to fill the mating pool.

Overall, HVEA works as follows:

Step 1: *Initialisation:* Start with an empty population $\boldsymbol{P}$ and create an initial archive $\overline{\boldsymbol{P}}$ (each of size N).

Step 2: *Fitness assignment:* Calculate fitness values of individuals in $\boldsymbol{P} \bigcup \overline{\boldsymbol{P}}$ (*cf. Section 2.1*).

Step 3: *Ranking assignment:* Assign ranking to individuals in $\boldsymbol{P} \bigcup \overline{\boldsymbol{P}}$ (*cf. Section 2.2*).

Step 4: *Environmental selection:* Repeatedly copy all individuals having the best ranking from $\boldsymbol{P}$ to $\overline{\boldsymbol{P}}$ and assign their crowding values until the size of $\overline{\boldsymbol{P}}$ exceeds N then reduce the size of $\overline{\boldsymbol{P}}$ by means of the truncation operator (*cf. Section 2.3* and *Section 2.4*).

Step 5: *Termination:* Stopping criteria are satisfied then present all non-dominated individuals in $\overline{\boldsymbol{P}}$ as solutions to the problem, and then the algorithm stops here.

Step 6: *Generating offspring:* Apply crossover and mutation operators on parents, which are selected with binary tournaments, to produce offspring (*cf. Section 2.5*). Go to Step 2.

2.1. Fitness Assignment

The fitness assignment procedure of HVEA deploys the concept of volume dominance proposed by Le et at [1]. Le et at. proposed a new relaxed Pareto dominance for multi-objective optimisation, named volume dominance. They assign *strength* values to individuals in the population using the ratio between the hyper-volume of an individual and the hyper-volume of the current representative Pareto front. The dominance relationship between two individuals is established by comparing *strength*s of the pair with respect to the threshold strength r_{Str}. The fitness assignment of HVEA is as follows. Let $V(\boldsymbol{x})$ be the hyper-volume of individual $\boldsymbol{x}$ w.r.t. the reference point $\vec{r^x}$; ***ParetoFront*** is the current Pareto front of the archived population $\overline{\boldsymbol{P}}$; the current representative Pareto front, $\boldsymbol{RP}(\boldsymbol{x})$, is the set of all individuals in ***ParetoFront*** that Pareto-dominate $\boldsymbol{x}$.

$$\boldsymbol{RP}(\boldsymbol{x}) = \left\{\boldsymbol{x}^* \mid \boldsymbol{x}^* \geq \boldsymbol{x} \bigwedge \boldsymbol{x}^* \in \boldsymbol{ParetoFront}\right\} \quad (1)$$

$$x_i^{ref} = \sup\left\{f_i(\boldsymbol{x}^*) \mid \boldsymbol{x}^* \in \boldsymbol{RP}(\boldsymbol{x})\right\} \quad (2)$$

$V^{ref}(\boldsymbol{x})$ is the hyper-volume of $\boldsymbol{x^{ref}}$, given by (2), w.r.t. the reference point $\vec{r^{\boldsymbol{x}}}$, given by (4), then the fitness value of individual $\boldsymbol{x}$ is as follows:

$$fitness(\boldsymbol{x}) = 1 - \frac{V(\boldsymbol{x})}{V^{ref}(\boldsymbol{x})} \quad (3)$$

As $\boldsymbol{RP}(\boldsymbol{x})$ is an empty set if $\boldsymbol{x}$ is a Pareto solution, (3) is relaxed by setting $fitness(\boldsymbol{x}) = 0$.

The estimation of the reference point $\vec{r^{\boldsymbol{x}}}$ and the hyper-volume $V(\boldsymbol{x})$, $V^{ref}(\boldsymbol{x})$ is identical to the one proposed by Le et al. [1]. The reference point $\vec{r^{\boldsymbol{x}}} = \left(r_1^{\boldsymbol{x}}, r_2^{\boldsymbol{x}}, \ldots, r_m^{\boldsymbol{x}}\right)$ is as follows:

$$r_i^{\boldsymbol{x}} = f_i(\boldsymbol{x}) - (r_i^{\sup} - r_i^{\inf}) \quad (4)$$

for $\forall i = 1, 2, \ldots, m$

$$r_i^{\inf} = \inf\left\{f_i(\boldsymbol{x}^*) \mid \boldsymbol{x}^* \in \boldsymbol{P} \bigcup \overline{\boldsymbol{P}}\right\} \quad (5)$$

$$r_i^{\sup} = \sup\left\{f_i(\boldsymbol{x}^*) \mid \boldsymbol{x}^* \in \boldsymbol{P} \bigcup \overline{\boldsymbol{P}}\right\} \quad (6)$$

and the hyper-volume $V(\boldsymbol{x})$, $V^{ref}(\boldsymbol{x})$:

$$V(\boldsymbol{x}) = \prod_{i=1}^{m} (f_i(\boldsymbol{x}) - r_i^{\boldsymbol{x}}) \quad (7)$$

$$V^{ref}(\boldsymbol{x}) = \prod_{i=1}^{m} (x_i^{ref} - r_i^{\boldsymbol{x}}) \quad (8)$$

Fig. 1 illustrates the estimation of the above terms that are required for the calculation of individual fitness in HVEA. Lower values of $fitness(\boldsymbol{x})$ are preferred. The procedure to determine the fitness values of individuals in the offspring population $\boldsymbol{P}$ and the archive $\overline{\boldsymbol{P}}$ is as follows:

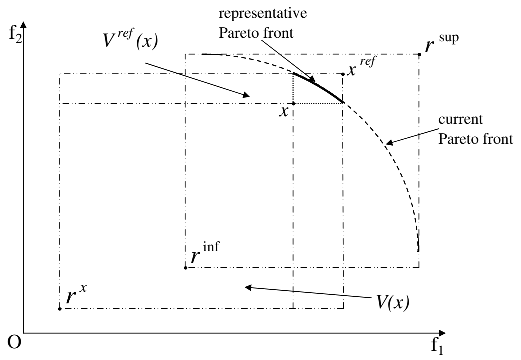


Fig. 1: The fitness assignment in HVEA.

Step 1: Use all individuals in $\boldsymbol{P}$ and $\overline{\boldsymbol{P}}$ to update:

$$\vec{r^{\inf}} = \left(r_1^{\inf}, r_2^{\inf}, \ldots, r_m^{\inf}\right) \quad (9)$$

$$\vec{r^{\sup}} = \left(r_1^{\sup}, r_2^{\sup}, \ldots, r_m^{\sup}\right) \quad (10)$$

where $r_i^{\inf}$ and $r_i^{\sup}$ are given by (5) and (6) respectively.

Step 2: Update ***ParetoFront*** using all newly generated offspring in $\boldsymbol{P}$. Assign fitness value of -1 to offspring that Pareto-dominate at least one individual in ***ParetoFront*** of the previous generation. Assign fitness value of 0 to the remaining individuals in ***ParetoFront*** of the current generation. Apply (3) to assign fitness value to all remaining individuals in $\boldsymbol{P}$ and $\overline{\boldsymbol{P}}$.

Assigning fitness value of -1 to offspring that Pareto-dominate at least one individual in ***ParetoFront*** enables HVEA to distinguish individuals that make a direct contribution in moving the Pareto front forward during the environmental selection and mating selection. This fitness assignment strategy emphasises the idea of preferring individuals near the Pareto front especially individuals that improve the Pareto front of the previous generation.

2.2. Ranking Assignment

As commonly found in the literature, the ranking assignment in evolutionary algorithms classifies individuals with similar (or identical) characteristic(s) into the same category(ies). The characteristic usually used is the Pareto non-dominance. In other words, Pareto non-dominated individuals are classified into one category (rank). Therefore it is guaranteed that individuals with the same rank are all mutually non-dominated. An example is NSGA2 [2] with its fast non-dominated sorting approach. HVEA takes a paradigm shift in ranking individuals. In HVEA, it is the case that an individual is Pareto dominated by another individual having the same rank. The main intention of this mechanism is to allow slightly worse quality individuals to be able to compete for survival and/or selection. HVEA computes ranking from the individual fitness value as follows:

$$rank(\boldsymbol{x}) = \left\lfloor fitness(\boldsymbol{x}) \times \frac{1}{\mu} \right\rfloor \tag{11}$$

where parameter μ indicates the 'size' of each ranking level whereas $\frac{1}{\mu}$ could be referred as the desired number of ranks (desired number of 'fronts'). The advantage of this ranking assignment mechanism is that only Pareto non-dominated individuals are assigned rank 0 with exception of individuals, which improve the Pareto front, are assigned rank -1. Other Pareto dominated individuals are assigned rank with some noise induction by applying (11).

2.3. Environmental Selection

Let $\mathcal{F}_i$ is the set of individuals in $\boldsymbol{P} \bigcup \overline{\boldsymbol{P}}$ with $rank(\boldsymbol{x}) = i$:

$$\mathcal{F}_i = \left\{\boldsymbol{x} \mid rank(\boldsymbol{x}) = i \wedge \boldsymbol{x} \in \boldsymbol{P} \bigcup \overline{\boldsymbol{P}}\right\} \tag{12}$$

One set at a time, all individuals from each set $\mathcal{F}_i$, with $i = -1, 0, 1, 2, \ldots$, are copied to the archive $\overline{\boldsymbol{P}}$ until the size of $\overline{\boldsymbol{P}}$ equals to or exceeds N. If the size of $\overline{\boldsymbol{P}}$ equals to N, the environmental selection is completed. Otherwise, individuals in the last front $\mathcal{F}_l$ copied to $\overline{\boldsymbol{P}}$, are removed from $\overline{\boldsymbol{P}}$ until reducing the size of $\overline{\boldsymbol{P}}$ to N. The last copied front $\mathcal{F}_l$ satisfies the following conditions:

$$\sum_{i=-1}^{l-1} |\mathcal{F}_i| < N \tag{13}$$

$$\sum_{i=-1}^{l-1} |\mathcal{F}_i| + |\mathcal{F}_l| > N \tag{14}$$

Individuals in $\mathcal{F}_l$ are sorted by their crowding values. The individual with the highest crowding value in $\mathcal{F}_l$ is removed from $\overline{\boldsymbol{P}}$ and $\mathcal{F}_l$. The crowding value of remaining individuals in $\overline{\boldsymbol{P}}$, not just $\mathcal{F}_l$, are updated accordingly. These processes are repeated until the size of $\overline{\boldsymbol{P}}$ equals to N.

2.4. Crowding Assignment

As NSGA2 and SPEA2, HVEA employs the distance-based approach to estimate the density of individuals around a particular individual. However HVEA's crowding assignment mechanism is different from NSGA2 and SPEA2. Both NSGA2 and SPEA2 only consider "one neighbour" in the neighbouring area in order to determine the density of a particular individual. NSGA2 combines the distance to the adjacent neighbour in each objective to estimate the crowding of a particular individual whereas SPEA2 considers the distance to the k-th nearest neighbour. HVEA takes into account all individuals in the *neighbouring area* to determine the crowding of a particular individual. The *neighbouring area* is defined by the current state of the combined population $\boldsymbol{P} \bigcup \overline{\boldsymbol{P}}$ and a *radius* ω. An individual $\boldsymbol{x}^*$ is in the *neighbouring area* of an individual $\boldsymbol{x}$ (in other word $\boldsymbol{x}^*$ and $\boldsymbol{x}$ are neighbours) if the following condition is satisfied:

$$\left|x_i - x_i^*\right| \leq \left(r_i^{\sup} - r_i^{\inf}\right) \times \omega \tag{15}$$

for $\forall i = 1, 2, \ldots, m$ and $r_i^{\inf}$ and $r_i^{\sup}$ are given in (5) and (6). Let $\boldsymbol{NB}(\boldsymbol{x})$ is the set of all neighbouring individuals of $\boldsymbol{x}$ that means $\boldsymbol{x}^* \in \boldsymbol{NB}(\boldsymbol{x})$ if and only if $\boldsymbol{x}^*$ and $\boldsymbol{x}$ are satisfied (15). Then, the crowding value of $\boldsymbol{x}$ is as follows:

$$crowding(\boldsymbol{x}) = \sum_{\boldsymbol{x}^* \in \boldsymbol{NB}(\boldsymbol{x})} (d(\boldsymbol{x}, \boldsymbol{x}^*) + 1)^{-1} \tag{16}$$

where $d(\boldsymbol{x}, \boldsymbol{x}^*)$ is the Euclidean distance between $\boldsymbol{x}$ and $\boldsymbol{x}^*$:

$$d(\boldsymbol{x}, \boldsymbol{x}^*) = \sqrt{\sum_{i=1}^{m} (x_i - x_i^*)^2} \tag{17}$$

and lower value of $crowding(\boldsymbol{x})$ is preferred.

Apart from considering all individuals in the *neighbouring area* to estimate the crowding of a particular individual, another main difference in crowding assignment mechanism between HVEA and NSGA2, SPEA2 is that NSGA2 only uses each front separately to estimate crowding value of individuals in that front, SPEA2 uses all individuals in both the offspring population $\boldsymbol{P}$ and the archive $\overline{\boldsymbol{P}}$ to estimate crowding value of individuals, but HVEA only uses individuals in fronts, which are already added to the archive $\overline{\boldsymbol{P}}$, to estimate the crowding value. Additionally, the crowding value of individual $\boldsymbol{x} \in \overline{\boldsymbol{P}}$ is repeatedly adjusted when a new front $\mathcal{F}_i$ is added to $\overline{\boldsymbol{P}}$ which is not the case in SPEA2. The pseudocode for the crowding assignment of front $\mathcal{F}_i$ $(-1 \le i \le l)$ is as follows:

```
Procedure CrowdingAssignment($\mathcal{F}_i$)
begin
  for each $x$ in $\mathcal{F}_i$
    for each $x^*$ in $\mathcal{F}_i$ ($x^* \neq x$)
      if $x^*$ and $x$ are neighbours
      then add $(d(x,x^*) + 1)^{-1}$ to $crowding(x)$ and $crowding(x^*)$
      endif
    endfor
    for each front $\mathcal{F}_j$ ($-1 \le j < i$)
      for each $x^*$ in $\mathcal{F}_j$
        if $x^*$ and $x$ are neighbours
        then add $(d(x,x^*) + 1)^{-1}$ to $crowding(x)$ and $crowding(x^*)$
        endif
      endfor
    endfor
  endfor
end
```

2.5. Generating Offspring

HVEA chooses parents to fill the mating pool by binary tournament selection. Individuals competed for the second parent are different from the first parent. The binary tournament selection prioritises the following order: *ranking*, *crowding*, *fitness* values of individuals. Crossover and mutation are applied on the mating pool to form the offspring population $\boldsymbol{P}$. Any duplicated offspring dies out of the combined population $\boldsymbol{P} \bigcup \overline{\boldsymbol{P}}$. This guarantees that there is no duplication in the combined population $\boldsymbol{P} \bigcup \overline{\boldsymbol{P}}$ before the environmental selection process. Consequently, there is no duplication in the archive $\overline{\boldsymbol{P}}$.

3. Comparative Case Study

3.1. Multi-objective Evolutionary Algorithms

The performance of HVEA is compared to NSGA2 [2], SEAMO2 [3], SPEA2 [4], IBEA [5] and MOEA/D [6].

3.1.1. Non-dominated Sorting Genetic Algorithm 2 (NSGA2)

Deb et at. [2] used a fixed population and a fixed archive of size N for their NSGA2. They proposed a fast non-dominated sorting approach in NGSA2 to classify individuals in a population into different non-domination levels or different fronts. NSGA2 employed a density estimation metric to preserve the population diversity. It required to sort each front according to each objective function value in ascending order of magnitude. The boundary individuals are assigned an infinite distance value. The density of individuals surrounding other intermediate individual in the front is the sum of the average normalised distance of two individuals on either side of this individuals along each of the objectives. The mating pool is filled by the binary tournament selection with the following priority, non-domination level then crowding distance. The offspring population is formed by applying crossover and mutation on the mating pool. The best fronts of the offspring population and the archive combined are selected for the next archive. If the size of the archive is greater than N, individuals in the last front is removed to reduce the size of the archive to N based on their crowding value. There is no treatment to prevent duplication of individuals whilst forming the archive. The duplication individuals could be spotted by the crowding value and removed during the truncation of the archive but it is not exhaustive.

3.1.2. Simple Evolutionary Algorithm for Multi-objective Optimization 2 (SEAMO2)

SEAMO2 [3] uses a steady-state population and a simple elitist replacement strategy. Each individual of the population act as the first parent once and a second parent is chosen at random.

Crossover and mutation is applied on the pair of parents to produce offspring. The offspring replaces one of the parents if either its objective vector improves on any *best-so-far* objective function or it dominates that parent. If neither the offspring dominates the parents nor the parents dominate the offspring, the offspring replaces a random individual in the population that the offspring dominates. SEAMO2 does not allow duplication in its population. Therefore, any duplicated offspring dies before the replacement process.

3.1.3. Strength Pareto Evolutionary Algorithm 2 (SPEA2)

Zitzler et at. [4] employed a fixed size archive to store non-dominated individuals in addition to a population for SPEA2. SPEA2 uses a fine-grained fitness assignment strategy which takes for each individual into account how many individuals it dominates and it is dominated by. Each individual in both the archive and the population is assigned a strength value $S(i)$, indicating the number of individuals it dominated. The raw fitness $R(i)$ of an individual is defined as the sum of the strength value of all individuals by which it is dominated.

$$S(i) = \left|\left\{j \mid j \in \boldsymbol{P} \bigcup \overline{\boldsymbol{P}} \wedge i \succ j\right\}\right| \qquad (18)$$

$$R(i) = \sum_{j \in \boldsymbol{P} \cup \overline{\boldsymbol{P}} \wedge j \succ i} S(j) \qquad (19)$$

SPEA2 uses an adaptation of the k-th nearest neighbour method to estimate the density of an individuals. The density of an individual is estimated as the inverse of the distance to the k-th nearest neighbour. Then, the density is added to the raw fitness Ri to yields the fitness of an individual i. Similarly to NSGA2, SPEA2 fills the mating pool using binary tournament selection on the archive based on the fitness of individuals participated in the tournament. Crossover and mutation is applied on the mating pool to create the offspring population. From the offspring and the archive population, all individuals having fitness less than one, i.e. all non-dominated individuals in $\boldsymbol{P} \bigcup \overline{\boldsymbol{P}}$, are copied to the next archive. Zitzler et at. pointed out that there are two situation: either the archive is too small or too large. If the archive is too small, the best dominated individuals, based on the fitness value, are copied to fill $\overline{\boldsymbol{P}}$. In the later situation, non-dominated individuals in the archive are iteratively removed until the archive's size is not exceeded. The removal of non-dominated individuals from the archive is carefully managed by using an archive truncation method that guarantees the preservation of boundary solutions. As in NSGA2, SPEA2 relies on the archive truncation to remove duplicated individuals but it does not guarantee that the archive contains no duplication.

3.1.4. Indicator Based Evolutionary Algorithm (IBEA)

Zitzler and Künzli [5] proposed a general framework indicator-based evolutionary algorithm (IBEA). IBEA could be referred as classic MOEAs but guided by a general preference information of the dominance relationship, a *binary quality indicator*. IBEA needs neither specifications of weights or targets (as in aggregation methods) nor the dominance relationship and distribution techniques (as in classic MOEAs). IBEA uses this *binary indicator* guiding the search to optimise the approximation set. The *binary quality indicator* in IBEA maps an ordered pair of individuals to a real number which therefore could be used for fitness calculation. Zitzler and Künzli [5] proposed two indicators, the additive ϵ-indicator I_{ϵ^+} and the hyper-volume-indicator I_{HD}. For an order pair of individuals $(\boldsymbol{x}^1, \boldsymbol{x}^2)$, $I_{\epsilon^+}(\boldsymbol{x}^1, \boldsymbol{x}^2)$ gives minimum distance for which $\boldsymbol{x}^1$ is translated in each in objective space to weakly dominate $\boldsymbol{x}^2$. $I_{HD}(\boldsymbol{x}^1, \boldsymbol{x}^2)$ measures the space volume that is dominated by $\boldsymbol{x}^2$ but not by $\boldsymbol{x}^1$ with respect to a predefined reference point. The fitness of an individual is estimated as the sum of an exponential function of the indicator values. A positive constant $\kappa = 0.05$ is applied to the indicator value to amplify the influence of dominating individuals over dominated one. It is suggested to use adaptive scaling for both indicator values and objective values to avoid the issue of estimating a good reference point. As aforementioned population based MOEAs (NSGA2 and SPEA2), IBEA uses a fixed size archive and an offspring population. Offspring is produced by recombination and mutation on a pair of parents selected from the archive. The archive and the offspring population is combined and truncated to generate the new archive for the next iteration.

3.1.5. Multi-objective Evolutionary Algorithm Based on Decomposition (MOEA/D)

MOEA/D, proposed by Zhang and Li [6], decomposes a multi-objective optimisation problem into a number of scalar optimisation sub-problems and optimises them simultaneously. Each sub-problem is optimised by utilising information from it neighbouring sub-problems. Non-dominated individuals found during the search are stored to an external archive. MOEA/D predefines a set of even spread weight vector $\left\{\lambda^1, \ldots, \lambda^N\right\}$, where N is the number of sub-problems, or the population size of MOEA/D. Each sub-problem ith corresponds to a weight vector λ^i. The neighbourhood of the ith sub-problem consists of all sub-problems with the weight vectors closest to λ^i which include the ith sub-problem itself. Then, the population of MOEA/D consists the best solution $\boldsymbol{x}^i$ found so far for each sub-problem ith. At each iteration, an offspring to ith sub-problem is produced by recombination and mutation on parents which are the current solutions to the neighbohood of the ith sub-problem. The offspring replaces current solutions of the ith sub-problem and its neighbouring sub-problems if the fitness value of the offspring is better than that of these current solutions. See [6] for more details of MOEA/D and the fitness calculation.

3.1.6. Adaptive Evolutionary Multi-objective Simulated Annealing (EMOSA) [7]

Li and Landa-Silva [7] proposed an improved version of MOEA/D by incorporating an advanced local search technique (simulated annealing) and adapting the weight vectors, so called EMOSA. EMOSA deploys the cooling technique in simulated annealing to adaptively modified weight vectors at the lowest temperature in order to diversify the population towards the unexplored part of the search space. EMOSA also uses simulated annealing technique to further improve each individual in the population. Finally, EMOSA use ϵ-dominance to update the external population in order to maintain the diversity of the external population. The performance of EMOSA is then compared against a set of multi-objective simulated annealing algorithms and a set of multi-objective memetic algorithms using multi-objective 0/1 knapsack problem and multi-objective travelling salesman problem. EMOSA performs better than all these algorithms and is capable of finding very good distribution of non-dominated solutions.

3.1.7. $\mathcal{S}$-Metric Selection EMOA (SMS-EMOA)

SMS-EMOA is a steady state population and selection scheme, proposed by Beume et al. [8]. SMS-EMOA's goal is to maximise the hyper-volume, $\mathcal{S}$-metric value, of the population by removing an individual which exclusively contributes the least hyper-volume. At each iteration, an offspring is produced by recombination and mutation. The offspring replace an individual in the population $\boldsymbol{P}_t$ if it leads to higher quality of the population with respect to the $\mathcal{S}$-metric. The resulting population $\boldsymbol{P}_{t+1}$, formed by combining the offspring and the previous population $\boldsymbol{P}_t$, is partition into different non-dominated sets, fronts, using the fast non-dominated sorting algorithm as describe in NSGA2 [2]. The first front $\mathcal{R}_1$ contains all non-dominated solutions of $\boldsymbol{P}_{t+1}$, the second front contains all individual that are non-dominated in the set $(\boldsymbol{P}_{t+1} \backslash \mathcal{R}_1)$, etc.. An individual r is discarded from the last front $\mathcal{R}_v$, the worst ranked front, for which that individual exclusively contributes the least hyper-volume to the last front $\mathcal{R}_v$.

$$r = \arg\min_{s \in \mathcal{R}_v} \left[\Delta_{\mathcal{S}}(s, \mathcal{R}_v)\right] \tag{20}$$

$$\Delta_{\mathcal{S}}(s, \mathcal{R}_v) = \mathcal{S}(\mathcal{R}_v) - \mathcal{S}(\mathcal{R}_v \backslash \{s\}) \tag{21}$$

Beume et al. also investigated several selection variants of SMS-EMOA and pointed out that if there is more than one front in P_{t+1} ($v > 1$), the individual with the most number of dominating points should be discarded. The equation (20) should be replaced by the following equation (22):

$$r = \arg\max_{s \in \mathcal{R}_v} \left[d\left(s, \boldsymbol{P}_{t+1}\right) \right] \tag{22}$$

$$d\left(s, \boldsymbol{P}_t\right) = \left|\{y \in \boldsymbol{P}_t | y \prec s\}\right| \tag{23}$$

The main motivation of this selection variant is to reduce the runtime complexity and to emphasise on sparsely filled regions of the solution space. It is expected that dominating individual will rise in rank to better fronts and fill those vacancies [8]. The experimental results on continuous problems including ZDT and DTLZ indicated that SMS-EMOA outperforms NSGA2 and SPEA2.

It is emphasised that HVEA is clearly different from SMS-EMOA. SMS-EMOA uses the exclusive hyper-volume contribution of individuals in the worst front to eliminate an individual during the steady-state selection scheme. HVEA use the hyper-volume of an individual to determine its fitness and its ranking. The next archive is selected from the archive population combining with the offspring population using only individual ranking and crowding.

3.1.8. Direction-based Multi-objective Evolutionary Algorithm (DMEA)

Bui et al. [9] proposed a population-based evolutionary algorithm that evolves a population along directions of improvement. They are two types of directions employed in DMEA i.e. (1) the convergence direction between an archived non-dominated solution and a dominated solution from the parental pool and (2) spread direction between two non-dominated solutions in the archive. These directions are used to perturb the parental population at each generation. A half of the next-generation parental pool is formed by non-dominated whose spread is aided by a niching criterion applied in the decision space (i.e. using spread direction), whereas the other half is filled by non-dominated dominated solutions from the combined population using convergence direction. The archive is updated by scanning a bundle of rays from the estimated ideal point into the part of objective space containing the current pareto front. For each ray scanned, the non-dominated solution in the combined population, which are closest to the ray, is selected for the next archive. The experimental result on continuous problems including ZDT and DTLZ obtained by DMEA is slightly better than those by NSGA2 and SPEA2.

3.1.9. Direction-based Multi-objective Evolutionary Algorithm II (DMEA-II)

The DMEA is further improved and so called DMEA-II [10]. The main differences between these two algorithms are: (1) an adaptive approach for choosing directional types in forming the parental pool, (2) a new concept for ray-based density for niching and (3) a new selection scheme based on this ray-based density. In DMEA, a half of the parental pool is formed by using the convergence direction and the other half is formed by using the spread direction. DMEA-II deployed an adaptive approach, in which this ratio is depended on the number of non-dominated solutions in the current population. The higher the number of non-dominated solutions in the current population is, the more use of the spread direction. In other words, if the current population contains all non-dominated solutions then the spread direction is used to generate the whole parental population. The second improvement is a new ray-based density approach by counting the number of ray that a solution is the closest one. Then a half of the archive is updated by non-dominated solutions with highest ray-based density, whereas the other half is formed by solutions closest to each ray. Continuous problems including ZDT, DTLZ and UF are used as benchmark problems for comparison. Experimental results show that DMEA-II outperforms DMEA, NSGA2 and SPEA2 but remains competitive to MOEA/D.

3.2. Experimental Design

3.2.1. Multi-objective Optimisation Problem

The performance of HVEA is assessed by comparing to NSGA2, SEAMO2, SPEA2, IBEA and MOEA/D on the multiple 0/1 knapsack problem proposed by Zitzler and Thiele [11]. The multiple 0/1 knapsack is defined as a set of n items and a set of m knapsacks, weight and profit associated with each item in a knapsack, and an upper bound for the knapsack capacity. The goal is to find a subset of items that maximise the profit in each knapsack and the knapsack weight does not exceed its capacity.

$p_{i,j}$ = profit of item j in knapsack i

$w_{i,j}$ = weight of item j in knapsack i

c_i = capacity of knapsack i

find a vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n$ such that:

$$\forall i \in \{1,2,\ldots,m\} : \sum_{j=1}^{n} w_{i,j} \times x_j \le c_i \qquad (24)$$

and maximise $f(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_1(\boldsymbol{x}), \ldots, f_m(\boldsymbol{x}))$, where

$$f_i(\boldsymbol{x}) = \sum_{j=1}^{n} p_{i,j} \times x_j \qquad (25)$$

and $x_j = 1$ if and only if item j is selected.

There are 9 instances for the multiple 0/1 knapsack problem [11]. The population size for each instance is as follows:

We perform 50 independent runs for statistical analysis, each run is of 2000 generations. It is noticed that MOEA/D predefines the set of even spread weight vector $\left\{\lambda^1, \ldots, \lambda^N\right\}$ where $N = C^{m-1}_{H+m-1}$ (H: controlling parameter). It is difficult to change N to the required population size. Therefore it is suggested to respect the population size set by MOEA/D but alter the number of generations for MOEA/D as follows

$$g = \left\lfloor \frac{S}{N} \times 2000 \right\rfloor \qquad (26)$$

Table 1: Parameter Setting for The Multiple 0/1 Knapsack Problem

Instance	**Population Size (S)**	**N in MOEA/D**
ks2_250	150	150
ks2_500	200	200
ks2_750	250	250
ks3_250	200	351
ks3_500	250	351
ks3_750	300	351
ks4_250	250	455
ks4_500	300	455
ks4_750	350	455

There are several representations, repair methods and offspring productions for the multiple 0/1 knapsack problem. Our preliminary experiment show that the performance of MOEAs on the multiple 0/1 knapsack problem is affected by those. Therefore we outline 4 different repair methods and corresponding representations.

Zitzler and Thiele [11] represented solutions to the multiple 0/1 knapsack problem as binary strings. Offspring is produced by applying one point crossover followed by bit-flip mutation on a pair of parents. The knapsack capacity constraints are confronted by a greedy repair method. This repair method repeatedly removed items from the encoded solutions until all capacity constraints are satisfied. Items are deleted based on the order determined by the maximum profit/weight ration per item (q_j); for item j the maximum profit/weight ratio (q_j) is given by the equation (27):

$$q_j = \max_{i=1}^{m} \left\{ \frac{p_{i,j}}{w_{i,j}} \right\} \qquad (27)$$

Items with lowest q_j are removed first until the capacity constraints are fulfilled. This mechanism diminishes the overall profit as little as possible [11].

Mumford [12] used a different set of representation and repair method for the multiple 0/1 knapsack problem. The solution to the knapsack problem in SEAMO2 is represented as a simple permutation of items to be packed. Offspring is produced from a pair of parents by applying cycle crossover followed by random mutation swapping two arbitrarily selected items within a single permutation list. The repair method starts from the beginning of the permutation list, packing item once at a time until the weight for any knapsack exceeds its capacity. Packing is discontinued and the final item is removed from all knapsacks.

Jaszkiewicz [13] used binary strings to represent solutions of the multiple knapsack problem. As Zitzler and Thiele [11], Jaszkiewicz used one point crossover followed by bit-flip mutation on a pair of parents to produce offspring. However Jaszkiewicz used the weighted linear scalarising approach to repair infeasible solutions rather than the maximum profit/weight ratio (q_j) [11]. Items are sorted according to the following ratio:

$$l_j = \frac{\sum_{i=1}^{n} \lambda_i \times p_{i,j}}{\sum_{i=1}^{n} w_{i,j}} \tag{28}$$

where $\vec{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_n)$ is the weight vector used in the current iteration. Later, Jaszkiewicz improved this greedy repair method which is subsequently deployed in MOEA/D. Jaszkiewicz's improved greedy repair method is as follows.

Let set $J = \left\{j | 1 \leq j \leq n \wedge x_j = 1\right\}$ is a subset of selected items and set $I = \left\{i | 1 \leq i \leq m \wedge \sum_{j=1}^{n} w_{i,j} \times x_j > c_i\right\}$ is a set of overfilled knapsacks. Repeatedly select item $k \in J$ to remove until none knapsack is overfilled, such that:

$$k = \arg\min_{j \in J} \frac{f(\boldsymbol{x}^{j-}) - f(\boldsymbol{x})}{\sum_{i \in I} w_{i,j}} \tag{29}$$

where $\boldsymbol{x}^{j-}$ is different from $\boldsymbol{x}$ only by item j, $x_i^{j-} = x_i$ for all $i \neq j$ and $x_j^{j-} = 0$, and $f(\boldsymbol{x})$ is the fitness value of $\boldsymbol{x}$. We investigated two approaches, weighted sum approach and Tchebycheff approach, determining the fitness value of $\boldsymbol{x}$ addressed in MOEA/D [6].

$$f^{ws}\left(\boldsymbol{x} \middle| \vec{\lambda}, \vec{z}\right) = \sum_{i=1}^{m} \lambda_i \times (z_i - f_i(\boldsymbol{x})) \tag{30}$$

$$f^{te}\left(\boldsymbol{x} \middle| \vec{\lambda}, \vec{z}\right) = \max_{1 \leq i \leq m} \{\lambda_i \times |z_i - f_i(\boldsymbol{x})|\} \tag{31}$$

where $\vec{z} = (z_1, z_2, \ldots, z_m)$ is the reference point with respect to the current population, $z_i = \max\{f_i(\boldsymbol{x}) | \boldsymbol{x} \in \boldsymbol{P}\}$.

As aforementioned, our initial investigation shows that there is difference in performance regarding the greedy repair methods and their corresponding representations and recombination methods for the multiple 0/1 knapsack problem. Therefore we assess the performance of HVEA, NSGA2, SEAMO2, SPEA2, IBEA and MOEA/D using different methods separately.

3.2.2. Performance Metrics

We use the hyper-volume measurement proposed by Zitzler and Thiele [11], the generational distance and the inverted generational distance measurements for evaluation.

Regarding the hyper-volume metric, Zitzler and Thiele suggested that the reference point to estimate the hyper-volume at the origin in the objective space. However, this suggestion gives extreme points in the objective space much higher exclusive contribution to the hyper-volume metric especially when the objective values of the solutions are high. Therefore, we propose that the reference point $\vec{r} = (r_1, r_2, \ldots, r_m)$ should be close to the solution set $\boldsymbol{S}$ obtained by all algorithm. The estimation of the reference point is as follows:

$$u_i = \max_{\boldsymbol{x} \in \boldsymbol{S}} \{f_i(\boldsymbol{x})\} \tag{32}$$

$$l_i = \min_{\boldsymbol{x} \in \boldsymbol{S}} \{f_i(\boldsymbol{x})\} \tag{33}$$

$$r_i = l_i - (u_i - l_i) \times 0.1 \tag{34}$$

The generational distance measures the distance from a non-dominated solutions set $\boldsymbol{P}_f$ to the true Pareto front $\boldsymbol{P}_t$:

$$gd(\boldsymbol{P}_f, \boldsymbol{P}_t) = \frac{\sqrt{\sum_{\boldsymbol{x} \in \boldsymbol{P}_f} (d(\boldsymbol{x}, \boldsymbol{P}_t))^2}}{|\boldsymbol{P}_f|} \tag{35}$$

where $d(\boldsymbol{x}, \boldsymbol{P}_t)$ is the minimum Euclidean distance between $\boldsymbol{x}$ and points in $\boldsymbol{P}_t$:

$$d(\boldsymbol{x}, \boldsymbol{P}_t) = \min_{\boldsymbol{x}^* \in \boldsymbol{P}_t} \left\{ \sqrt{\sum_{i=1}^{m} \left(f_i(\boldsymbol{x}) - f_i(\boldsymbol{x}^*)\right)^2} \right\} \quad (36)$$

The generational distance measurement indicates how close a non-dominated solutions set to the true Pareto front is. In other words, this metric indicates the convergence of a non-dominated solutions set to a particular part of the true Pareto front. The lower the value of the generational distance, the closer of a non-dominated solutions set to a particular part of the true Pareto front, the better performance of the algorithm.

The inverted generational distance works on the opposite manner, measuring the diversity of a non-dominated solutions set along the whole true Pareto front or how close the true Pareto front to a non-dominated solutions set is. The lower the value of the inverted generational distance, the more diversity of a non-dominated solutions set, the better performance of the algorithm.

$$igd(\boldsymbol{P}_f, \boldsymbol{P}_t) = \frac{\sqrt{\sum_{\boldsymbol{x} \in \boldsymbol{P}_t} \left(d(\boldsymbol{x}, \boldsymbol{P}_f)\right)^2}}{|\boldsymbol{P}_t|} \quad (37)$$

The true Pareto front $\boldsymbol{P}_t$ is not available for every instance of the multiple 0/1 knapsack problem. Therefore we use the approximation of $\boldsymbol{P}_t$ estimated by Jaszkiewicz [13] as suggested in MOEA/D [6] to determine the generational distance and inverted generational distance measurements.

4. Experimental Results and Discussion

We set $\mu = 0.01$ to determine the rank of an individual based on its fitness as in equation (11). We experiment different *neighbouring area radius* values ω which is given in (15) for HVEA. The value of ω should lie in the range of 0 and 1 ($0 \leq \omega \leq 1$). We report results for $\omega = 0.01$ and $\omega = 1.0$ which abbreviate as hv1 and hv100 respectively in both sets of figures and tables and as HVEA$_{0.01}$ and HVEA$_{1.0}$ respectively in the text.

4.1. Comparison to MOEAs

The performance of HVEA is compared against NSGA2, SEAMO2, SPEA2, IBEA$_{\epsilon^+}$ and IBEA$_{HV}$ abbreviated as ns2, sea2, sp2, ib$_{\epsilon^+}$ (ibe) and ib$_{HV}$ (ibhv) respectively. The performance of these MOEAs is assessed using four greedy repair methods for the multiple 0/1 knapsack problem, included *binary encoding* approach proposed by Zitzler and Thiele [11], *permutation encoding* approach proposed by Mumford [12], *te binary encoding* (Tchebycheff) and *ws binary encoding* (weighted sum) different weighted linear scalarising approaches proposed by Jaszkiewicz [13].

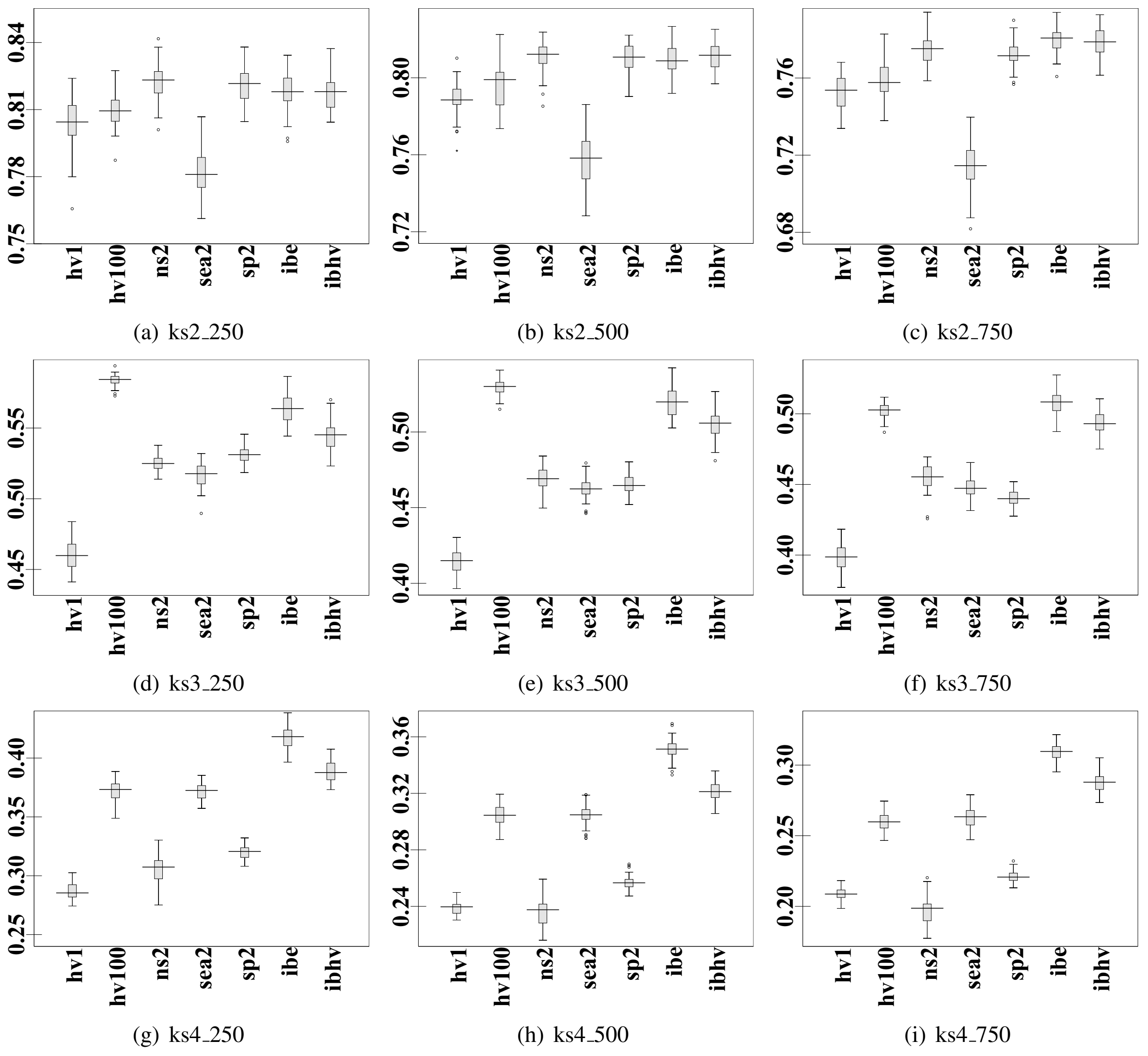
Table 2: Generational distance (permutation encoding)

Instance	hv1	hv100	ns2	sea2	sp2	ib$_{\epsilon+}$	ib$_{HV}$
ks2_250	3.20	3.24	1.99	4.57	**1.74**	3.31	3.45
ks2_500	10.10	10.37	6.00	14.47	**5.46**	10.65	12.02
ks2_750	17.86	17.48	12.73	26.54	**11.36**	21.86	23.71
ks3_250	6.96	14.59	24.31	8.04	10.83	6.15	**5.73**
ks3_500	16.68	34.11	54.61	18.24	18.04	13.91	**12.75**
ks3_750	26.57	54.02	73.69	29.58	23.35	22.61	**20.36**
ks4_250	11.43	38.73	44.97	12.60	19.93	12.27	**11.39**
ks4_500	22.37	66.97	95.02	27.28	27.74	22.97	**20.84**
ks4_750	34.44	99.30	141.16	42.77	34.04	35.25	**30.96**

Table 3: Inverted generational distance (permutation encoding)

Instance	hv1	hv100	ns2	sea2	sp2	ib$_{\epsilon+}$	ib$_{HV}$
ks2_250	10.13	9.06	**8.45**	11.49	8.67	8.48	8.48
ks2_500	10.96	9.98	10.07	13.93	9.86	9.50	**8.62**
ks2_750	48.07	46.02	42.92	64.14	44.40	38.07	**38.01**
ks3_250	21.11	**5.59**	10.37	18.35	12.41	13.25	15.74
ks3_500	48.73	**15.58**	25.60	43.54	39.14	33.71	37.69
ks3_750	69.65	**27.06**	39.73	61.16	61.03	48.20	54.06
ks4_250	19.33	**9.39**	13.57	14.98	14.53	12.30	15.03
ks4_500	43.24	**19.71**	30.15	33.84	39.64	29.83	35.81
ks4_750	68.36	**33.56**	49.35	54.58	65.93	49.84	57.93

Regarding the hyper-volume metric, Fig. 2,3,4,5 show that HVEA$_{1.0}$ outperforms or at least remains competitive to NSGA2, SEAMO2, SPEA2 on all 9 instances of the multiple 0/1 knapsack problem using all 4 different greedy repair methods. HVEA$_{1.0}$ is worse in only few cases out of 36 instance-repair method combinations which are mainly



(a) ks2_250 (b) ks2_500 (c) ks2_750

(d) ks3_250 (e) ks3_500 (f) ks3_750

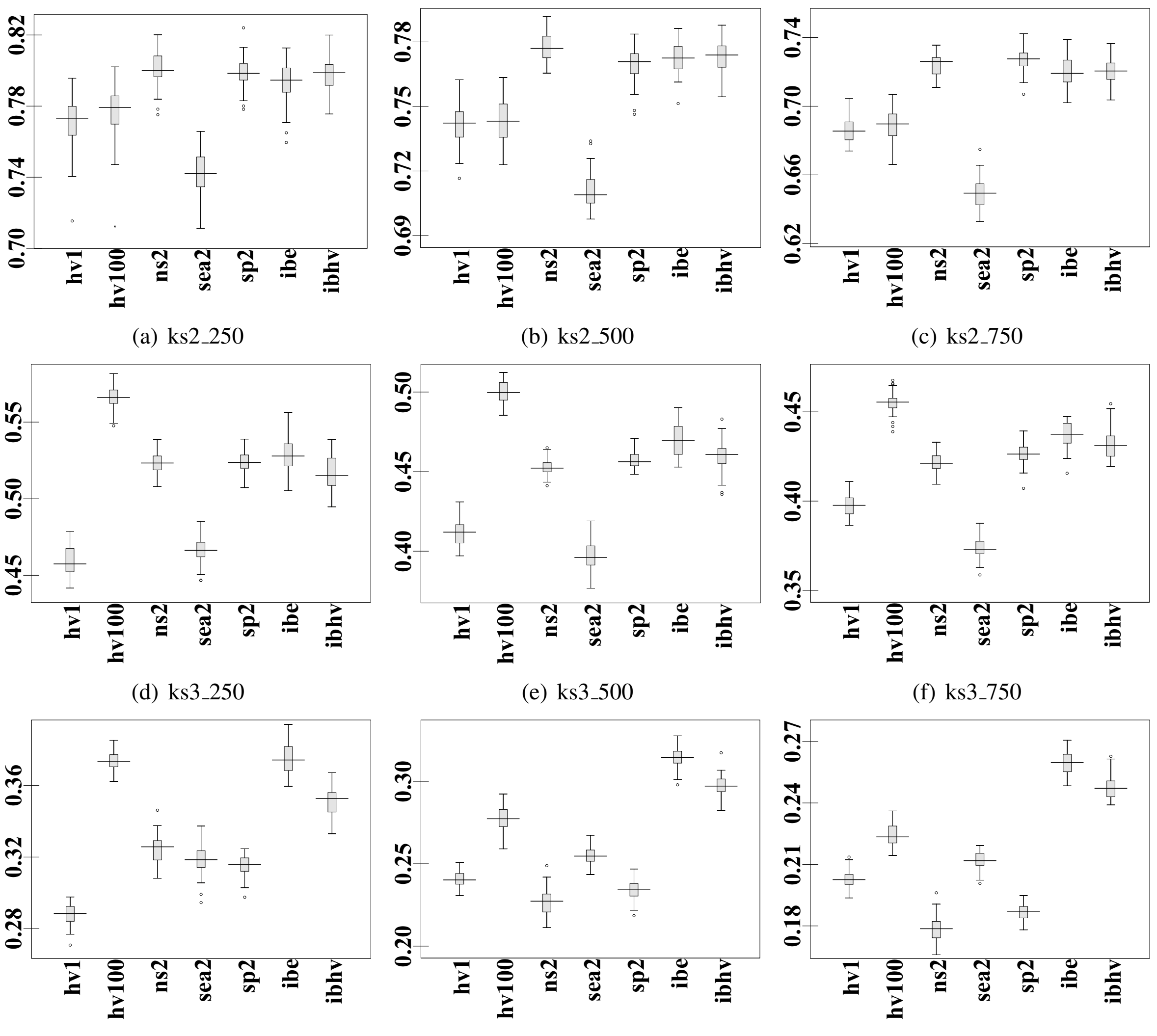
(g) ks4_250 (h) ks4_500 (i) ks4_750

Fig. 2: The percentage of hypervolume (permutation encoding).

Table 4: Running time in seconds (permutation)

Instance	hv1	hv100	ns2	sea2	sp2	ib$_{\epsilon+}$	ib$_{HV}$
ks2_250	7	6	6	**4**	48	50	50
ks2_500	14	14	14	**9**	97	96	91
ks2_750	27	28	26	**21**	165	160	144
ks3_250	14	18	11	**7**	103	103	158
ks3_500	28	38	23	**15**	179	175	242
ks3_750	46	66	39	**32**	272	264	368
ks4_250	24	52	18	**11**	184	183	354
ks4_500	44	114	36	**22**	294	286	545
ks4_750	68	204	56	**48**	415	407	771

in the lowest dimension space, the 2-knapsack problems. HVEA$_{1.0}$ only outperforms IBEA (both IBEA$_{\epsilon^+}$ and IBEA$_{HV}$) for the 3-knapsack problems. It looses its competitiveness to IBEA for 2,4-knapsack problems. IBEA$_{\epsilon^+}$ tends to be the most effective MOEA to provide the coverage (hyper-volume) for the Pareto sets. It is also noticed that HVEA$_{1.0}$ performs much better than HVEA$_{0.01}$. The reason is that HVEA$_{0.01}$ only looks at a tiny neighbouring area to determine the *crowding* of an individual whereas HVEA$_{1.0}$ examines the objective space as the whole picture, i.e. every individuals contributing towards the *crowding* of an individual. Results on the generational and inverted generational distance show more prominent evidence regarding this issue.



(a) ks2_250 (b) ks2_500 (c) ks2_750

(d) ks3_250 (e) ks3_500 (f) ks3_750

(g) ks4_250 (h) ks4_500 (i) ks4_750

Fig. 3: The percentage of hypervolume (binary encoding).

Table 5: Generational distance (binary encoding)

Instance	hv1	hv100	ns2	sea2	sp2	ib$_{\epsilon+}$	ib$_{HV}$
ks2_250	3.91	**3.78**	4.17	6.16	4.09	6.95	6.99
ks2_500	11.42	**11.03**	13.39	15.10	13.78	17.66	18.07
ks2_750	**26.86**	27.10	32.87	33.91	31.67	29.48	30.05
ks3_250	**8.45**	13.95	22.16	8.62	14.18	10.06	9.77
ks3_500	**20.83**	34.04	50.69	21.39	32.63	24.47	22.82
ks3_750	**35.71**	55.08	74.63	40.78	52.56	38.58	35.84
ks4_250	14.18	33.23	38.58	**13.91**	27.10	15.37	14.99
ks4_500	31.13	75.38	96.31	30.97	66.35	32.20	**28.85**
ks4_750	52.80	119.03	150.58	52.89	106.08	53.40	**44.81**

Table 2,5,8,11 show the generational distance from non-dominated sets to the true Pareto front. Table 3,6,9,12 show the inverted generational distance from the true Pareto front to non-dominated sets. Bold figures indicate the best results for each instance. Both sets of tables clearly indicate that HVEA$_{1.0}$ provides better diversity than HVEA$_{0.01}$ whereas HVEA$_{0.01}$ provides better convergence than HVEA$_{1.0}$ due to the use of *neighbouring area radius* ω. On the performance regarding the generational distance metric, HVEA$_{0.01}$ outperforms NSGA2 and SPEA2 but is competitive to SEAMO2 and IBEA. SEAMO2 is one of the best algorithms to provide convergence, although SEAMO2 only exploits Pareto dominance without fitness, ranking and crowding estimation. The reason behinds it is that SEAMO2 is a steady-state algorithm which allows offspring to compete for recombination immediately within the current generation. It is not the case for the other population-based MOEAs. However it leads SEAMO2 to be one of the worst MOEAs to provide diversity for non-dominated sets. The inverted generational distance metric show that $HVEA_{1.0}$ outperforms other MOEAs in most cases of the 36 instance-repair method combination. $HVEA_{1.0}$ is clearly the best algorithm to provide diversity in high dimensional knapsack, 3,4-knapsack problems. There is no strong evidence for which algorithm is the best in the 2-knapsack problem.

Table 6: Inverted generational distance (binary encoding)

Instance	hv1	hv100	ns2	sea2	sp2	$ib_{\epsilon+}$	ib_{HV}
ks2_250	14.55	13.88	**9.75**	16.81	10.51	10.67	9.89
ks2_500	17.97	17.91	**11.75**	20.95	12.85	12.19	11.79
ks2_750	81.69	81.09	54.68	98.40	**54.63**	56.50	55.30
ks3_250	20.33	**6.76**	10.20	20.46	11.16	14.33	16.36
ks3_500	46.55	**22.41**	27.22	49.62	31.86	36.09	39.43
ks3_750	64.04	**40.16**	43.82	72.18	48.56	53.63	56.87
ks4_250	18.37	**8.47**	12.53	16.64	12.67	13.45	15.90
ks4_500	39.55	**22.05**	30.54	37.02	30.33	29.45	34.11
ks4_750	62.33	**38.87**	51.83	60.39	51.45	49.51	55.66

Table 7: Running time in seconds (binary)

Instance	hv1	hv100	ns2	sea2	sp2	$ib_{\epsilon+}$	ib_{HV}
ks2_250	8	8	7	**6**	50	51	51
ks2_500	18	18	17	**14**	99	96	95
ks2_750	31	32	30	**26**	165	157	149
ks3_250	16	20	12	**9**	104	103	145
ks3_500	31	39	25	**21**	175	170	236
ks3_750	50	61	43	**36**	266	256	350
ks4_250	27	50	19	**14**	185	183	351
ks4_500	47	107	37	**28**	281	278	537
ks4_750	73	168	60	**48**	396	392	746

The running time (in seconds) is summarised in table 4,7,10,**??**. SPEA2 and IBEA are the slowest algorithms due to the complexity in computation of fitness values (SPEA2) and indicator values (IBEA). SEAMO2, NSGA2, HVEA are much faster algorithms.

Taken all the above criteria into account, it is concluded that HVEA outperforms NSGA2, SEAMO2, and SPEA2 but remains competitive to IBEA with much faster computational time than IBEA.

Table 8: Generational distance (te binary encoding)

Instance	hv1	hv100	ns2	sea2	sp2	$ib_{\epsilon+}$	ib_{HV}
ks2_250	**4.42**	4.46	5.18	6.94	5.41	6.84	6.93
ks2_500	**16.33**	16.60	17.68	23.92	17.62	18.35	19.01
ks2_750	44.67	43.98	46.88	64.90	44.96	**41.62**	43.42
ks3_250	13.99	18.92	28.85	**11.79**	21.06	13.64	13.50
ks3_500	36.35	44.72	70.27	33.93	50.40	31.61	**29.84**
ks3_750	64.61	78.85	113.39	67.64	85.45	59.47	**56.29**
ks4_250	22.82	40.94	47.44	**16.53**	37.85	17.53	17.29
ks4_500	51.52	89.40	114.67	40.71	89.75	43.05	**37.09**
ks4_750	90.24	144.94	183.81	79.45	148.89	78.48	**67.98**

Table 9: Inverted generational distance (te binary encoding)

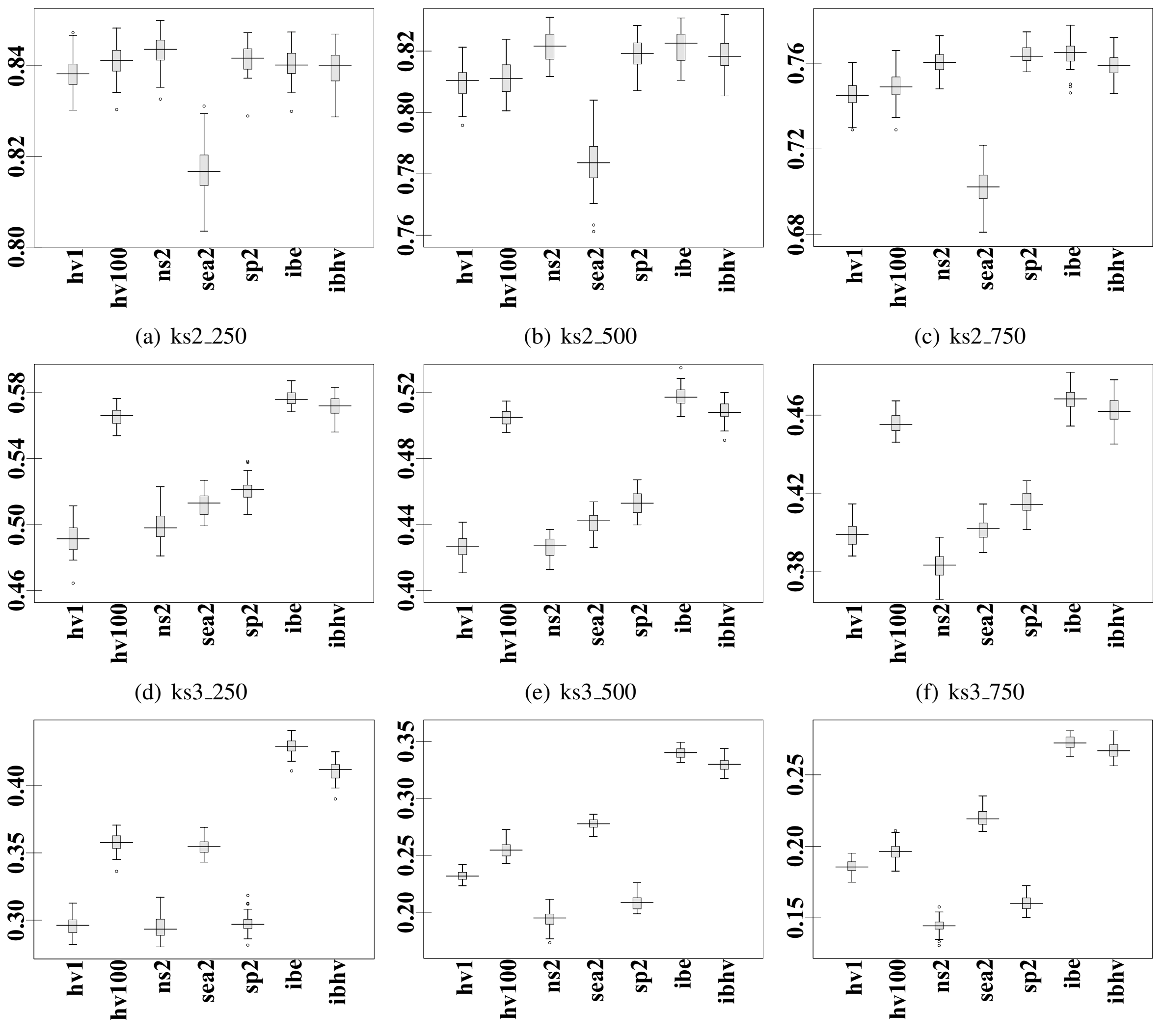
Instance	hv1	hv100	ns2	sea2	sp2	$ib_{\epsilon+}$	ib_{HV}
ks2_250	3.06	2.93	**2.62**	4.52	2.87	2.84	2.89
ks2_500	6.88	6.80	5.79	8.72	6.03	**5.52**	5.72
ks2_750	44.16	42.82	41.38	56.88	39.69	**38.89**	40.27
ks3_250	14.83	**7.03**	12.19	17.76	9.19	9.10	11.10
ks3_500	37.33	**19.40**	30.71	40.87	26.50	23.96	27.44
ks3_750	52.74	**36.67**	52.38	59.90	42.18	37.78	41.14
ks4_250	15.02	10.27	14.10	14.49	11.77	**8.90**	11.58
ks4_500	35.20	25.91	35.61	32.69	30.65	**22.04**	26.31
ks4_750	56.33	45.72	60.56	53.05	52.58	**37.70**	43.18

Table 10: Running time in seconds (te binary)

Instance	hv1	hv100	ns2	sea2	sp2	$ib_{\epsilon+}$	ib_{HV}
ks2_250	13	13	12	**10**	54	55	60
ks2_500	31	31	30	**26**	112	109	117
ks2_750	58	58	56	**51**	192	182	198
ks3_250	22	28	18	**16**	109	110	150
ks3_500	51	61	46	**40**	195	191	256
ks3_750	93	109	86	**76**	310	301	392
ks4_250	36	66	30	**24**	195	194	367
ks4_500	75	143	66	**57**	307	308	565
ks4_750	134	256	129	**108**	462	461	810

4.2. Comparison to MOEA/D

We compare HVEA against MOEA/D separately from other MOEAs due to the setting and approach employed by MOEA/D. As discussed in Section 3.2.1, it is problematic to alter the population size N in MOEA/D to that population size S suggested for the multiple 0/1 knapsack problem. Furthermore, MOEA/D stores all non-dominated solutions found so far in an external archive which is



(a) ks2_250 (b) ks2_500 (c) ks2_750

(d) ks3_250 (e) ks3_500 (f) ks3_750

(g) ks4_250 (h) ks4_500 (i) ks4_750

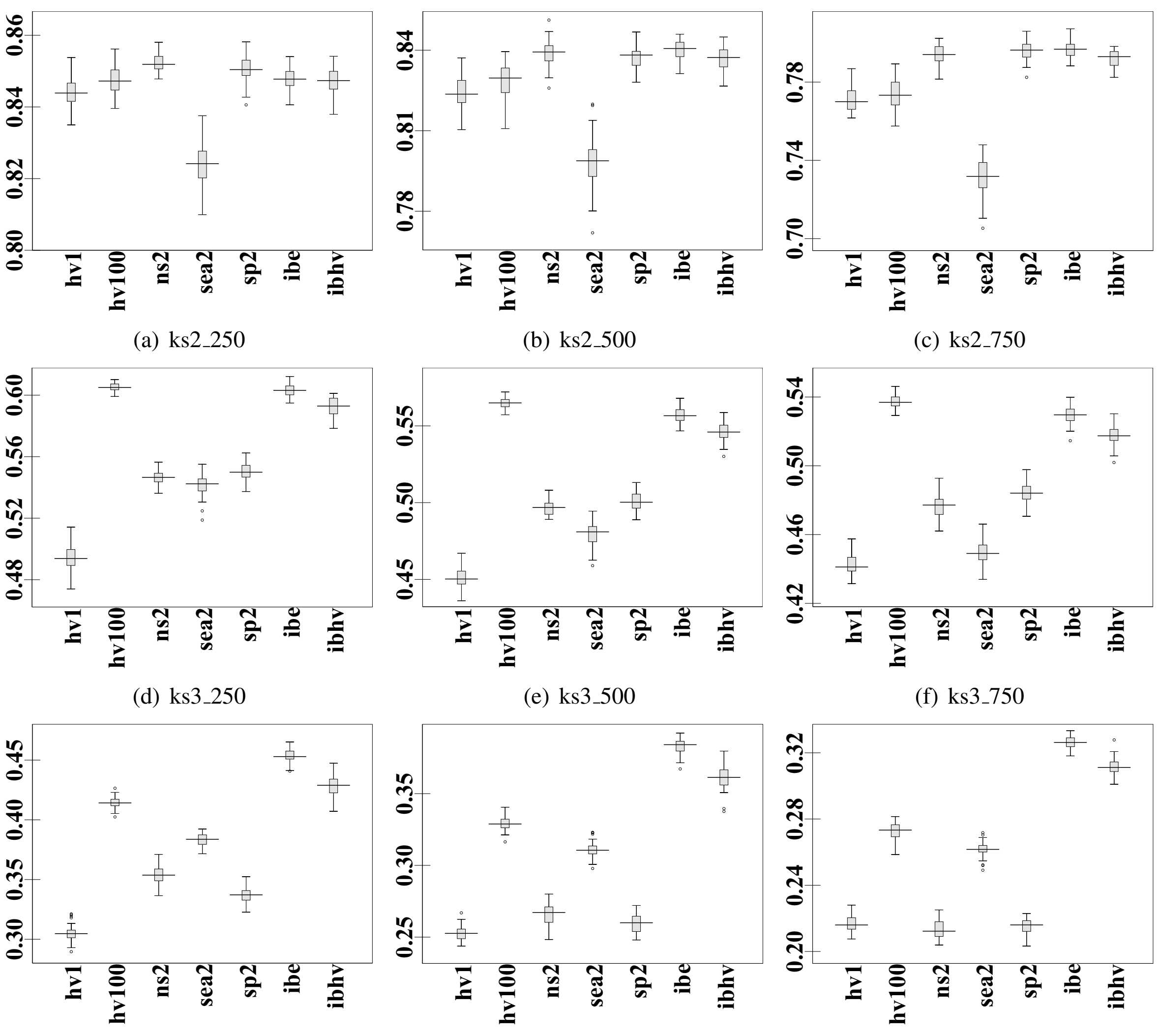
Fig. 4: The percentage of hypervolume (te binary encoding).

then reported as the final result. Therefore, it is suggested to adapt HVEA to the setting similar to MOEA/D included the population size and an external archive for non-dominated solutions found so far. Experimental results, including figures and tables, are presented in a similar format as in Section 4.1. MOEA/D employed Tchebycheff and weight sum approach for the fitness calculation are abbreviated as “mo/t” and “mo/w”. The extracted population, the truncated external population, is also reported so that the size of the extracted population does not exceed the suggested population size S. NSGA2’s truncation approach is used here. Fig. 6,7,8,9 present extracted populations using prefix “e”.

Table 11: Generational distance (ws binary encoding)

Instance	hv1	hv100	ns2	sea2	sp2	ib$_{\epsilon+}$	ib$_{HV}$
ks2_250	2.02	**1.96**	2.24	3.89	2.07	2.91	2.92
ks2_500	6.13	**6.07**	6.70	10.80	6.74	8.07	8.59
ks2_750	16.53	**16.38**	18.42	27.96	17.26	17.40	17.62
ks3_250	9.52	12.17	20.82	**7.26**	14.23	7.87	7.46
ks3_500	20.87	27.25	47.72	16.50	31.60	16.88	**15.90**
ks3_750	30.72	42.91	67.05	25.88	45.73	27.35	**24.99**
ks4_250	17.06	29.66	35.52	**12.52**	27.13	13.59	13.35
ks4_500	33.58	60.66	82.87	25.34	59.50	27.40	**23.68**
ks4_750	54.58	97.91	131.27	42.68	96.22	44.91	**38.29**

MOEA/D outperforms HVEA in most cases out of 36 instance-repair method combinations



(a) ks2_250 (b) ks2_500 (c) ks2_750
(d) ks3_250 (e) ks3_500 (f) ks3_750
(g) ks4_250 (h) ks4_500 (i) ks4_750

Fig. 5: The percentage of hypervolume (ws binary encoding).

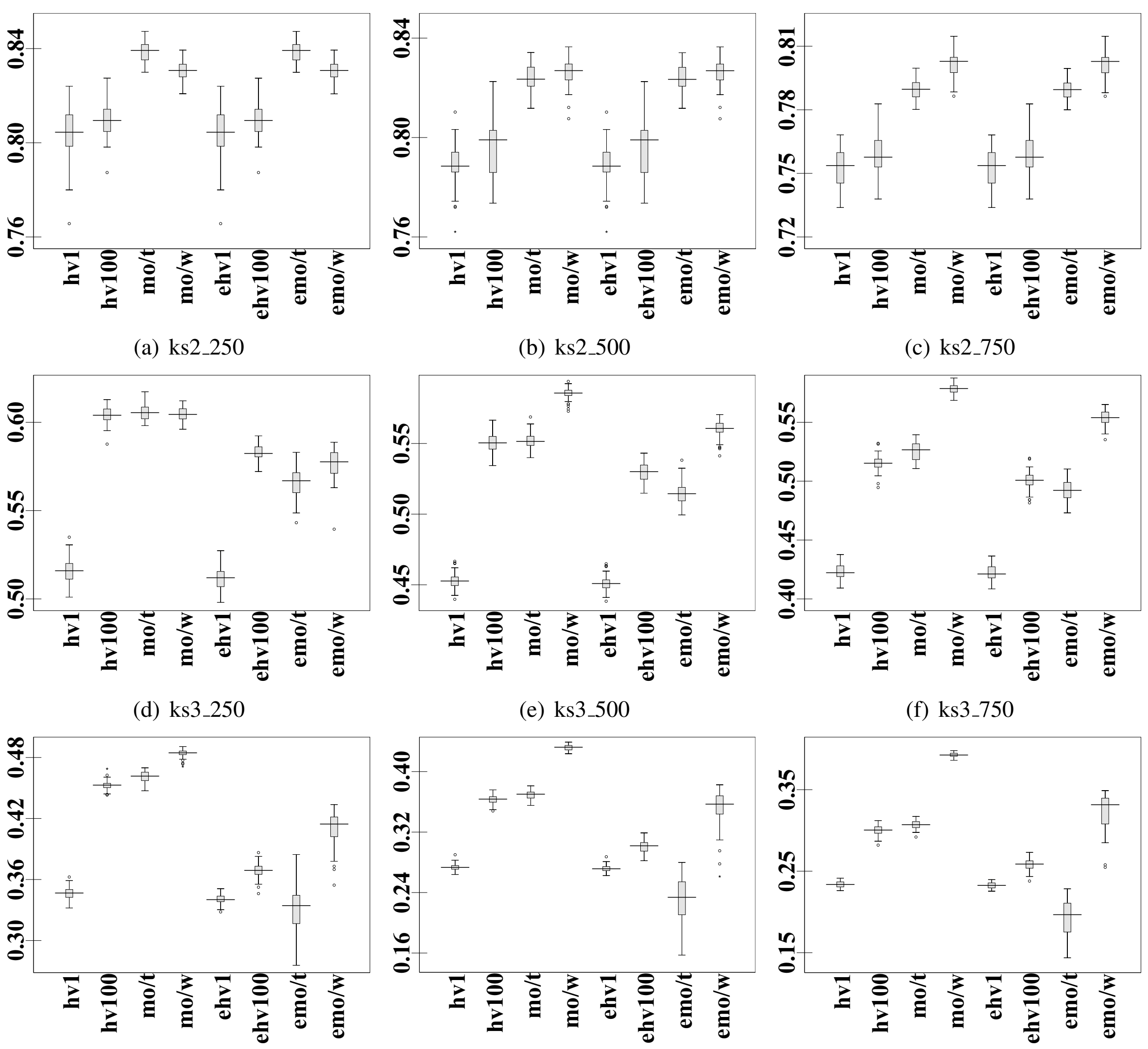
on the hyper-volume metric and the inverted generational distance metric. The convergence of MOEA/D is only better than that of HVEA in the linear scalarising greedy repair method. However the running time of MOEA/D is significantly higher than that of HVEA. In the other 2 repair methods, HVEA$_{0.01}$ outperforms MOEA/D regarding the generational distance metric. Under a restriction of the size of the population, where the truncation is applied to the external population, HVEA is able to compete to MOEA/D. However HVEA is much faster than MOEA/D. The running time (in seconds) is reported in Table 18, 15, 21, 24.

Table 12: Inverted generational distance (ws binary encoding)

Instance	hv1	hv100	ns2	sea2	sp2	ib$_{\epsilon+}$	ib$_{HV}$
ks2_250	3.80	3.40	**2.70**	4.90	3.05	3.14	3.34
ks2_500	7.02	6.62	4.97	9.05	5.27	**4.53**	4.78
ks2_750	41.14	40.31	28.86	52.55	27.86	**26.11**	27.82
ks3_250	16.83	**4.78**	9.71	17.31	8.93	9.06	11.05
ks3_500	40.25	**13.21**	22.43	40.48	24.74	23.92	27.69
ks3_750	56.43	**23.72**	35.38	59.39	37.02	36.85	41.70
ks4_250	16.08	**7.75**	11.47	14.33	11.35	9.17	11.85
ks4_500	37.35	**18.10**	27.27	32.41	28.35	22.44	27.55
ks4_750	58.56	**31.92**	46.15	52.21	47.00	38.13	45.26

4.3. Further Discussion

HVEA employs a parameter μ in equation (11) to define the size of each front. In this paper,



(a) ks2_250 (b) ks2_500 (c) ks2_750

(d) ks3_250 (e) ks3_500 (f) ks3_750

(g) ks4_250 (h) ks4_500 (i) ks4_750

Fig. 6: The percentage of hypervolume (permutation encoding).

μ is set to 0.01, which is not only bounded to the multiple 0/1 knapsack problem, could be used for other multi-objective optimisation problems. It is suggested that this parameter μ is fixed to 0.01 rather than tunable. The more important parameter in HVEA is ω, the *neighbouring area radius*, which is in the range $[0, 1]$. Section 4.1 shows a significant difference in performance between $\text{HVEA}_{0.01}$ and $\text{HVEA}_{1.0}$. $\text{HVEA}_{1.0}$ gives a much better performance regarding the hyper-volume metric and the diversity of the non-dominated set. However $\text{HVEA}_{0.01}$ is better in convergence than $\text{HVEA}_{1.0}$. One could tune ω to obtain desirable requirement. For example, ω could be set to 1.0 for theoretical problems such as the multiple 0/1 knapsack problem to obtain non-dominated set with better diversity (generational distance metric) and better coverage (hyper-volume metric). However, for real-world applications, where running time is expensive, $\omega = 0.01$ could be deployed. Furthermore, in real-world applications, extreme solutions are likely less of interest whereas better convergence, which could strike a good balance amongst objectives, is more important. We experiment different ω values in the range $[0, 1]$

but the performance of HVEA degrades quite significantly. Therefore, it is suggested to use $\omega = 1.0$ for better diversity and better coverage and $0.01 \leq \omega \leq 0.05$ for better convergence and faster computational time.

Table 13: Generational distance (permutation encoding)

	External Population				Extracted Population			
Instance	**hv1**	**hv100**	**mo/t**	**mo/w**	**hv1**	**hv100**	**mo/t**	**mo/w**
ks2_250	**3.16**	3.21	5.98	7.00	**3.19**	3.23	6.18	7.18
ks2_500	**10.00**	10.17	17.57	16.05	**10.08**	10.33	18.64	16.84
ks2_750	17.49	**17.06**	29.28	24.43	17.78	**17.40**	33.04	26.86
ks3_250	**2.08**	2.84	4.64	3.86	**7.99**	9.90	14.77	13.18
ks3_500	**4.97**	6.81	10.31	6.84	**16.72**	27.44	37.67	26.73
ks3_750	**9.29**	12.21	17.22	9.99	**26.94**	48.53	59.89	38.67
ks4_250	**2.24**	3.75	3.80	2.92	**14.11**	30.34	20.63	17.17
ks4_500	**4.23**	6.79	8.21	5.11	**24.22**	57.09	39.98	34.94
ks4_750	**6.79**	10.63	13.32	7.47	**34.17**	86.46	61.01	51.07

Table 14: Inverted generational distance (permutation encoding)

	External Population				Extracted Population			
Instance	**hv1**	**hv100**	**mo/t**	**mo/w**	**hv1**	**hv100**	**mo/t**	**mo/w**
ks2_250	10.13	9.06	**3.07**	4.26	10.13	9.06	**3.07**	4.26
ks2_500	10.96	9.98	6.60	**6.55**	10.96	9.98	6.60	**6.55**
ks2_750	48.07	46.02	37.36	**31.90**	48.07	46.02	37.37	**31.90**
ks3_250	15.86	5.22	**5.20**	6.26	15.95	**6.52**	7.83	8.37
ks3_500	42.80	**15.02**	16.43	15.10	42.85	**16.99**	19.83	18.18
ks3_750	63.84	25.89	30.90	**23.10**	63.89	27.77	34.74	**27.54**
ks4_250	14.97	**5.32**	5.76	5.91	15.08	11.06	12.70	**9.77**
ks4_500	38.54	14.94	16.82	**13.49**	38.60	22.45	31.17	**20.77**
ks4_750	63.77	29.00	31.73	**22.26**	63.87	36.84	49.64	**31.97**

We further examine the performance of HVEA against other MOEAs using a fixed running time, although this approach is not widely adopted due to its low reliability. The running time (in seconds), reported in Table 25, is deduced from the minimum running time of all MOEAs on each knapsack instances over 2000 generations. The running time for the linear scalarising greedy repair method proposed by Jaszkiewicz [13] is twice as much as the ones proposed by Mumford [12] and Zitzler and Thiele [11].

Under the time restriction, HVEA outperforms NSGA2, SPEA2, IBEA$_{\epsilon^+}$ and IBEA$_{HV}$. HVEA only outperforms SEAMO2 on 2,3-knapsack problems. SEAMO2 is slightly better than HVEA on 4-knapsack problems. It is suggested that due to the steady-state approach and the simple Pareto dominance, SEAMO2 is able to perform more evaluations than any other MOEAs which could lead to a better performance. HVEA is slightly worse than but competitive to MOEA/D.

Table 15: Running time in seconds (permutation)

Instance	hv1	hv100	mo/t	mo/w
ks2_250	4	5	**3**	4
ks2_500	**8**	9	**8**	9
ks2_750	**15**	16	17	19
ks3_250	**16**	26	40	38
ks3_500	**29**	84	118	133
ks3_750	**46**	177	172	181
ks4_250	**47**	156	357	315
ks4_500	**112**	602	1025	972
ks4_750	**163**	870	1784	1807

Table 16: Generational distance (binary encoding)

	External Population				Extracted Population			
Instance	**hv1**	**hv100**	**mo/t**	**mo/w**	**hv1**	**hv100**	**mo/t**	**mo/w**
ks2_250	3.91	**3.78**	4.47	7.28	3.91	**3.78**	4.54	7.29
ks2_500	11.42	**11.03**	15.53	17.39	11.42	**11.03**	15.53	17.39
ks2_750	**26.86**	27.10	34.07	34.88	**26.86**	27.10	34.07	34.88
ks3_250	**3.26**	4.08	4.77	4.57	**10.18**	12.28	14.33	14.01
ks3_500	**10.67**	10.69	12.18	10.15	**24.41**	31.16	37.44	32.18
ks3_750	22.54	**22.11**	23.62	19.32	**38.93**	53.36	64.60	52.75
ks4_250	**3.26**	4.46	4.08	3.63	**16.27**	26.40	21.17	19.24
ks4_500	**9.37**	11.80	10.06	8.09	**35.47**	69.12	49.77	44.30
ks4_750	18.56	20.13	**16.93**	12.85	**57.80**	114.62	80.97	71.23

Table 17: Inverted generational distance (binary encoding)

	External Population				Extracted Population			
Instance	**hv1**	**hv100**	**mo/t**	**mo/w**	**hv1**	**hv100**	**mo/t**	**mo/w**
ks2_250	14.55	13.88	**2.78**	5.35	14.55	13.88	**2.79**	5.35
ks2_500	17.97	17.91	**5.36**	7.37	17.97	17.91	**5.36**	7.37
ks2_750	81.69	81.09	**33.08**	35.57	81.69	81.09	**33.08**	35.57
ks3_250	16.31	8.36	**5.22**	8.37	16.36	8.93	**7.41**	9.67
ks3_500	41.47	25.90	**17.05**	24.17	41.50	26.38	**20.07**	25.59
ks3_750	61.05	42.30	**30.87**	34.95	61.08	42.67	**33.83**	36.71
ks4_250	14.87	6.95	**6.10**	8.84	14.97	**10.12**	11.59	10.91
ks4_500	34.46	19.55	**17.22**	19.04	34.53	24.15	24.86	**22.39**
ks4_750	56.85	36.89	**32.02**	32.62	56.92	41.66	43.02	**36.28**

Table 18: Running time in seconds (binary)

Instance	hv1	hv100	mo/t	mo/w
ks2_250	8	8	**5**	**5**
ks2_500	18	19	**13**	**13**
ks2_750	32	33	**25**	**25**
ks3_250	25	35	**18**	**18**
ks3_500	**39**	56	**39**	40
ks3_750	**56**	74	59	57
ks4_250	**55**	139	113	87
ks4_500	**82**	285	359	342
ks4_750	**110**	427	536	538

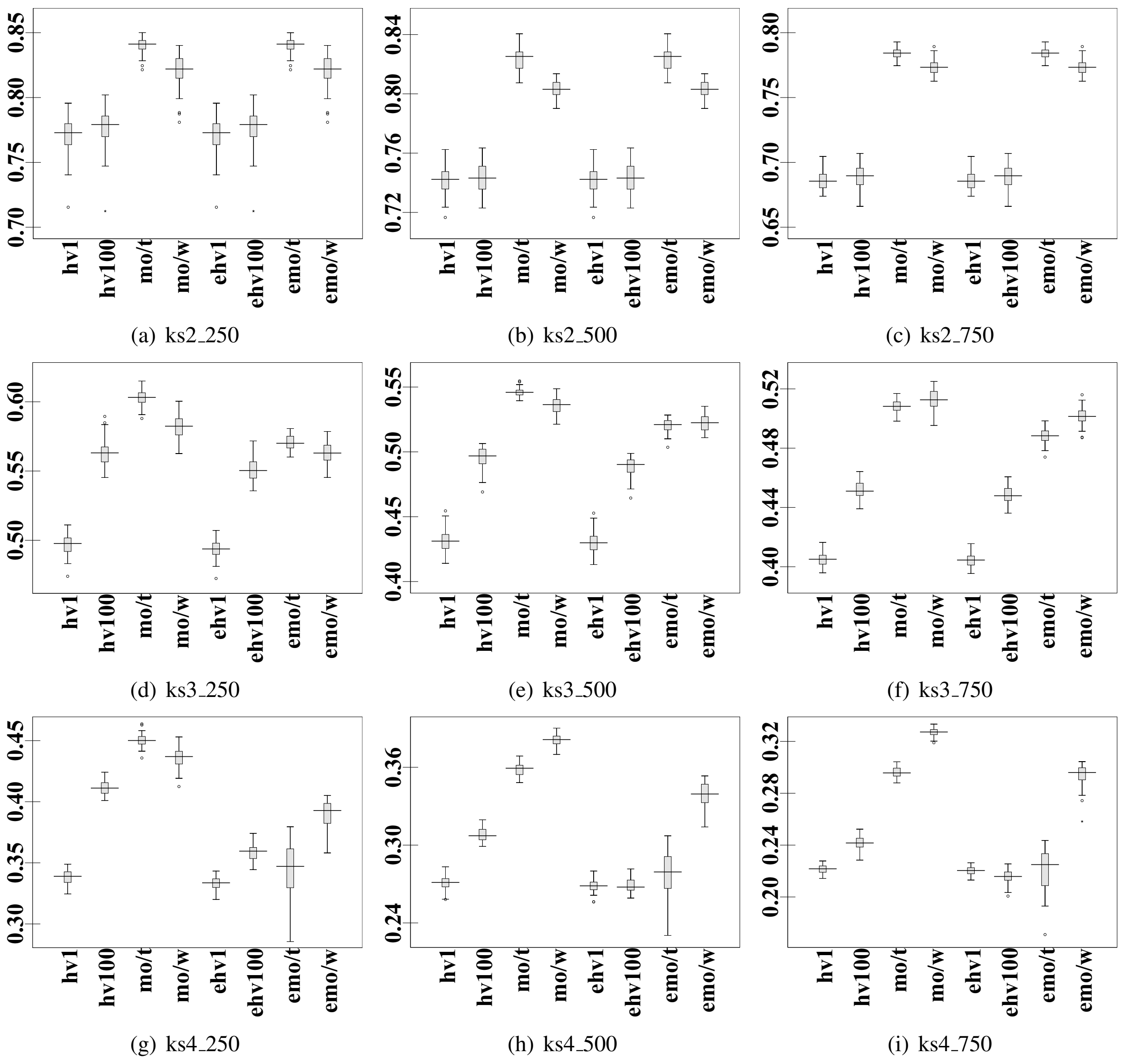


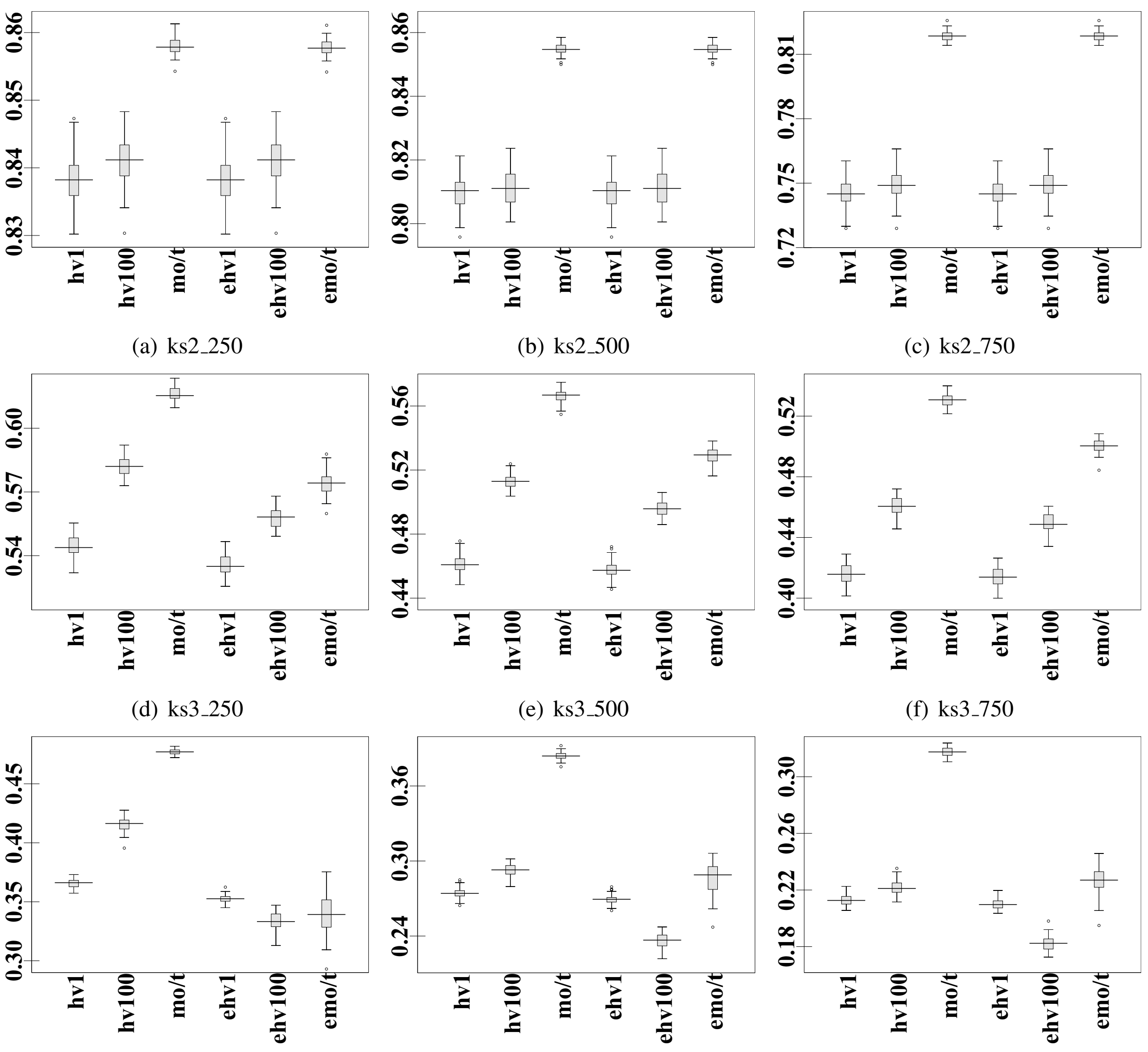
Fig. 7: The percentage of hypervolume (binary encoding).

Table 19: Generational distance (te binary encoding)

	External Population			Extracted Population		
Instance	**hv1**	**hv100**	**mo/t**	**hv1**	**hv100**	**mo/t**
ks2_250	4.40	4.43	**2.83**	4.41	4.45	**3.05**
ks2_500	16.33	16.60	**9.88**	16.33	16.60	**10.03**
ks2_750	44.67	43.98	**23.54**	44.67	43.98	**23.63**
ks3_250	5.39	6.09	**4.54**	**14.62**	15.61	14.80
ks3_500	16.66	15.73	**10.49**	37.71	42.51	**37.10**
ks3_750	35.66	30.38	**19.88**	66.81	77.08	**65.21**
ks4_250	4.91	5.75	**3.61**	22.65	34.78	**20.41**
ks4_500	14.45	14.45	**8.09**	52.51	83.35	**46.14**
ks4_750	30.20	24.54	**13.47**	93.91	141.43	**65.90**

Table 20: Inverted generational distance (te binary encoding)

	External Population			Extracted Population		
Instance	**hv1**	**hv100**	**mo/t**	**hv1**	**hv100**	**mo/t**
ks2_250	3.06	2.93	**1.52**	3.06	2.93	**1.53**
ks2_500	6.88	6.80	**3.69**	6.88	6.80	**3.69**
ks2_750	44.16	42.82	**26.22**	44.16	42.82	**26.22**
ks3_250	10.22	6.18	**5.03**	10.66	7.68	**7.45**
ks3_500	30.69	19.09	**15.73**	30.86	20.57	**18.94**
ks3_750	47.42	35.89	**31.87**	47.63	37.57	**35.11**
ks4_250	10.89	7.15	**5.52**	**11.17**	12.61	12.42
ks4_500	29.43	22.33	**16.88**	29.56	29.20	**24.87**
ks4_750	49.19	42.32	**32.32**	49.37	50.01	**42.92**



(a) ks2_250 (b) ks2_500 (c) ks2_750

(d) ks3_250 (e) ks3_500 (f) ks3_750

(g) ks4_250 (h) ks4_500 (i) ks4_750

Fig. 8: The percentage of hypervolume (te binary encoding).

Table 21: Running time in seconds (te binary)

Instance	hv1	hv100	mo/t
ks2_250	14	13	**9**
ks2_500	33	32	**25**
ks2_750	62	59	**51**
ks3_250	26	42	**24**
ks3_500	**58**	81	60
ks3_750	**102**	126	105
ks4_250	**49**	164	98
ks4_500	**92**	331	424
ks4_750	**157**	527	769

Table 22: Generational distance
(ws binary encoding)

	External Population			Extracted Population		
Instance	hv1	hv100	mo/w	hv1	hv100	mo/w
ks2_250	1.91	1.85	**1.81**	2.01	**1.95**	1.96
ks2_500	6.13	6.07	**3.74**	6.13	6.07	**4.23**
ks2_750	16.53	16.38	**8.10**	16.53	16.38	**9.08**
ks3_250	2.61	3.24	**1.84**	**9.82**	10.42	10.28
ks3_500	7.64	8.11	**3.15**	21.95	25.21	**18.53**
ks3_750	14.23	14.80	**5.15**	33.44	41.42	**27.36**
ks4_250	3.02	3.66	**1.56**	17.80	24.37	**16.10**
ks4_500	7.42	9.11	**2.72**	35.33	54.99	**30.81**
ks4_750	14.49	16.17	**4.34**	58.33	93.31	**44.10**

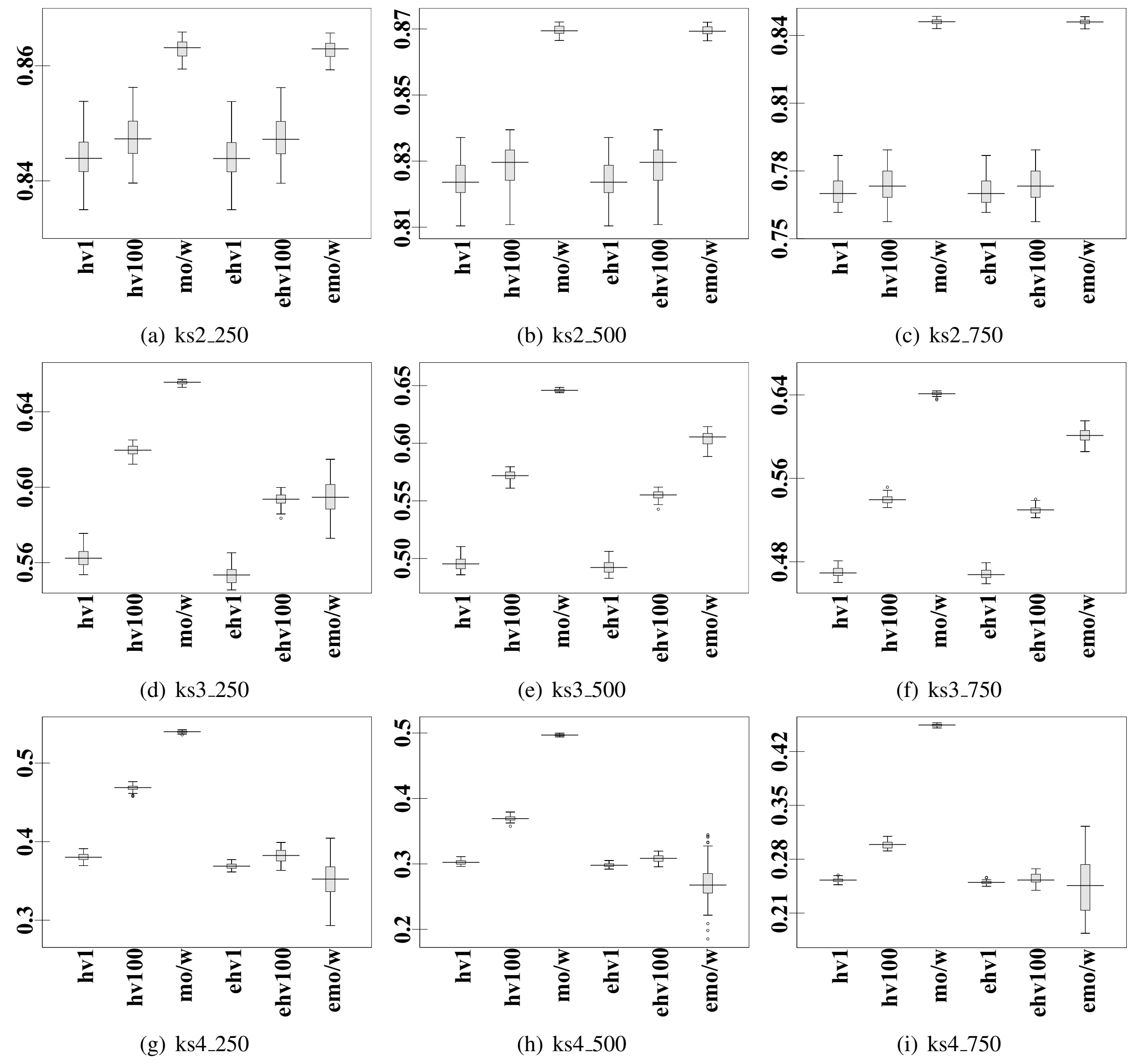


Fig. 9: The percentage of hypervolume (ws binary encoding).

Table 23: Inverted generational distance (ws binary encoding)

	External Population			Extracted Population		
Instance	**hv1**	**hv100**	**mo/w**	**hv1**	**hv100**	**mo/w**
ks2_250	3.80	3.39	**1.20**	3.80	3.40	**1.21**
ks2_500	7.02	6.62	**1.66**	7.02	6.62	**1.66**
ks2_750	41.14	40.31	**10.29**	41.14	40.31	**10.31**
ks3_250	10.80	4.31	**2.85**	11.13	**6.13**	8.32
ks3_500	31.98	14.04	**6.71**	32.08	15.44	**13.18**
ks3_750	48.87	24.76	**11.37**	48.95	25.93	**21.12**
ks4_250	11.45	4.99	**2.97**	11.66	**10.52**	13.93
ks4_500	30.56	14.94	**6.90**	30.66	**21.67**	31.76
ks4_750	50.08	29.11	**12.00**	50.20	**36.67**	47.92

Table 24: Running time in seconds (ws binary)

Instance	hv1	hv100	mo/w
ks2_250	9	13	**8**
ks2_500	**21**	30	22
ks2_750	**39**	55	43
ks3_250	**21**	44	33
ks3_500	**41**	82	107
ks3_750	**66**	124	137
ks4_250	**47**	180	339
ks4_500	**75**	326	885
ks4_750	**111**	500	1377

Table 25: Running Time (in seconds) for The Multiple 0/1 Knapsack Problem

Instance	**Mumford [12] & Zitzler [11]**	**Jaszkiewicz [13]**
ks2_250	5	10
ks2_500	15	30
ks2_750	30	60
ks3_250	15	30
ks3_500	30	60
ks3_750	45	90
ks4_250	30	60
ks4_500	45	90
ks4_750	60	120

5. Conclusions

In this paper we proposed a multi-objective evolutionary algorithm called Hyper-volume Evolutionary Algorithm (HVEA) which incorporates distinctive mechanisms for fitness assignment, ranking and crowding. The algorithm uses hyper-volume, Pareto dominance and distance between neighbouring solutions in order to drive improvement of the current Pareto front. Experiments conducted in this paper using the multiple 0/1 knapsack problem show that HVEA outperforms or remains very competitive against other MOEAs including NSGS2, SEAMO2, SPEA2, IBEA$_{\epsilon^+}$, IBEA$_{HV}$ and the high-performing MOEA/D. HVEA proposes a new individual fitness assignment strategy using the hyper-volume of an individual without the requirement of determining the reference point for the hyper-volume calculation. HVEA assesses the crowding of an individual by considering all individuals in its neighbourhood. By tuning the only parameter ω, the *neighbouring area radius*, the performance of the algorithm can be driven to attain a non-dominated set with specific characteristics. That is, the HVEA can be tuned to perform better on convergence, diversity or coverage and this could be useful in different applications when solving multi-objective optimisation problems.

The paper also extensively studied the multiple 0/1 knapsack problem using different greedy repair methods. In order to assess the performance of MOEAs fairly, the same greedy repair method should be used. Furthermore, we suggest that the greedy repair method proposed by Mumford [12] should be deployed while assessing the performance of MOEAs on the multiple 0/1 knapsack problems in order to minimise the effect of the greedy repair method on the overall performance.

For future work, it would be interesting to investigate the incorporation of the steady-state selection, deployed by SEAMO2 and MOEA/D, into fitness and/or ranking assignment population-based MOEAs. The reason behinds this it that the steady-state selection allows strong offspring to immediately participate in the recombination of the current generation which could lead to a better convergence whereas the fitness and/or ranking assignment population-based MOEAs consider the whole population during the environmental selection which could lead to a better diversity.

References

[1] K. Le, D. Landa-Silva, H. Li, An improved version of volume dominance for multi-objective optimisation, in: Proceedings of the 2009 International Conference on Evolutionary Multi-criterion Optimization (EMO 2009), Vol. 5467 of Lecture Notes in Computer Science, Springer, 2009, pp. 231–245.

[2] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: Nsga-ii, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182–197.

[3] C. Mumford, Simple population replacement strategies for a steady-state multi-objective evolutionary algorithm, in: Genetic and Evolutionary Computation GECCO 2004, Vol. 3102 of Lecture Notes in Computer Science, Springer, 2004, pp. 1389–1400.

[4] E. Zitzler, M. Laumanns, L. Thiele, Spea2: Improving the strength pareto evolutionary algorithm for multiobjective optimization, in: Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001), International Center for Numerical Methods in Engineering (CIMNE), 2002, pp. 95–100.

[5] E. Zitzler, S. Künzli, Indicator-based selection in multiobjective search, in: Parallel Problem Solving from Nature (PPSN VIII), Springer-Verlag, 2004, pp. 832–842.

[6] Q. Zhang, H. Li, Moea/d: A multiobjective evolutionary algorithm based on decomposition, IEEE Transactions on Evolutionary Computation 11 (6) (2007) 712–731.

[7] H. Li, D. Landa-Silva, An adaptive evolutionary multi-objective approach based on simulated annealing, MIT Evolutionary Algorithm 19 (4) (2011) 561–595.

[8] N. Beume, B. Naujoks, M. Emmerich, Sms-emoa: Multiobjective selection based on dominated hypervolume, European Journal of Operational Research 181 (3) (2007) 1653–1669.

[9] L. T. Bui, H. A. Abbass, M. Barlow, S. Wesolkowski, A. Bender, J. Liu, Dmea: A direction-based multiobjective evolutionary algorithm, Memetic Computing 3 (4) (2011) 271–285.

[10] L. Nguyen, L. T. Bui, H. A. Abbass, Dmea-ii: The direction-based multiobjective evolutionary algorithm-ii, Soft Computing (2013) 1–16.

[11] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach, IEEE Transactions on Evolutionary Computation 3 (4) (1999) 257–271.

[12] C. L. Valenzuela, A simple evolutionary algorithm for multi-objective optimization (seamo), in: Proceedings of the 2002 Congress on Evolutionary Computation - CEC 2002, 2002, pp. 717–722.

[13] A. Jaszkiewicz, On the performance of multiple-objective genetic local search on the0/1 knapsack problem - a comparative experiment, IEEE Transactions on Evolutionary Computation 6 (4) (2002) 402–412.