Solving real-world multi-objective engineering optimization problems with an Election-Based Hyper-Heuristic*

Vinicius Renan de Carvalho[1 [0000−0002−4623−7244]] and Jaime Simão Sichman[1 [0000−0001−8924−9643]]

Intelligent Techniques Laboratory (LTI)
University of São Paulo (USP)
{vrcarvalho,jaime.sichman}usp.br

Abstract. Hyper-heuristics are high-level methodologies responsible for automatically discover how to combine elements from a low-level heuristic set in order to solve optimization problems. Agents, in turn, are autonomous component responsible for sensing an environment and performing some actions according to their perceptions. Thus, agent-based techniques seem suitable for the design of hyper-heuristics. In a previous work we proposed MOABHH [5], an agent-based hyper-heuristic framework for choosing the best multi-objective evolutionary algorithm (MOEA). Our approach performs a cooperative voting procedure, considering a set of quality indicator voters, to define which MOEA should generate more new solutions during execution time. However, MOABHH was just applied to solve benchmark problems, without being tested in real-world problems. Thus, this paper evaluates MOABHH in four real-world multi-objective engineering problems. For this purpose, an additional MOEA and new quality indicators better adapted to real-world problems were used. The obtained results show that our strategy aways find solutions at least equals to the ones generated by the best algorithm, and sometimes even overcomes these results.

Keywords: Hyper-heuristics · Multi-objective Evolutionary algorithms · Voting methods · Agent cooperation. · Crashworthiness · Car Side Impact · Machining · Water Resource Planning

1 Introduction

Multi-objective problems (MOPs) are ubiquitous in many real-world problems. In these problems, the solutions should optimize different and often conflicting criteria. Usually, classical exact optimization methods cannot be used to deal with MOPs and more sophisticated heuristic techniques are required. Multi-objective evolutionary algorithms (MOEAs) have been successfully applied to the solution of MOPs [2]. MOEAs are heuristic techniques that allow a flexible

* Supported by CNPq-Brazil.

representation of the solutions and do not impose continuity conditions on the functions to be optimized. Due to the general and abstract characteristics of MOEAs, researchers have proposed several algorithms to cope with MOPs [14]. No single MOEA, however, can outperform the others in all problems, and algorithms perform differently depending on the problem characteristics. Therefore, techniques able to choose the most suitable MOEA for a given problem have raised to help in this difficult task, thus, diminishing part of the user effort to test all these algorithms; this new field of research is called *Hyper-heuristics*.

An Hyper-heuristic is a high-level heuristic that can be used to reduce the difficulty of selecting the most suitable heuristic for a given problem. According to Burke et al. [4], hyper-heuristics are considered both as (i) selection methodologies that help to choose a low-level heuristic (as a MOEA) and (ii) heuristic generation methodologies that can generate new low-level heuristics from a given set of components. Most research in the field of Hyper-heuristic has been limited to treat single-objective optimization problems [16]. However, there are some interesting studies focused on multi-objective optimization [16] [25] [1].

In a preliminary research [5], we proposed a Multi-Objective Agent-Based Hyper-Heuristic (MOABHH), an agent-based multi-objective hyper-heuristic focused on selecting the most suitable multi-objective evolutionary algorithm during execution time. MOABHH used the concept of voting to define which algorithm should receive a bigger participation in the generation of solutions. As a voting procedure, we applied the Copeland voting method and employed a set of *voter agents* responsible for evaluating algorithms performance according to different *quality indicators*; these are usually used by the MOP community to compare the performance of MOEAs.

The MOP community also uses well-known benchmarks to perform comparative evaluations; these latter provide generic test suites, enabling researchers to compare their multi-objective numerical and combinatorial optimization problem results (regarding effectiveness and efficiency) with others, over a spectrum of algorithms instantiations [6]. So far, MOABHH employed the WFG benchmark to this end. However, the fact that an algorithm successfully "passes" all submitted test functions in a benchmark doesn't guarantee a continued effectiveness and efficiency when it is applied to real-world problems [6].

Hence, this work evaluates MOABHH in 4 (four) real-world multi-objective engineering problems. The remainder of the paper is organized as follows. In the next section, we provide an overview of multi-objective optimization and present the 4 real-world engineering optimization problems used in this work. In Section 3, multi-objective evolutionary algorithms are briefly introduced. MOABHH is presented in Section 4, and the experimental setup and results are presented in Section 5. Finally, Section 6 presents our conclusions and further work.

2 Multi-Objective Engineering Optimization Problems

A multi-objective optimization problem (MOP) is defined in terms of a search space of allowed n decision values $(x_1, ..., x_n)$, and an objective function vector

$(f_1, ..., f_m)$, with size m, mapping parameter vectors into objective space. In these optimization problems, we aim to find the set of optimal trade-off solutions known as the Pareto optimal set [3]. Next, we describe the four real-world MOPs used in this work. Three of these problems are are constrained, i.e., present additional constraints to the values that can be attributed to a variable.

2.1 Crashworthiness

The vehicle crashworthiness problem [15] is a three-objective non-constrained problem where the crash safety level of a vehicle is optimized. In this problem, a higher safety level means how well a vehicle can protect the occupants from the effects of a frontal accident.

In this problem, there are five decision variables that represent the thickness of reinforced members around the car front, bounded as $1mm \leq x_i \leq 3mm$; and three objective functions to evaluate: (i) the mass of the vehicle $f_1(x)$ (Equation 1), (ii) an integration of collision acceleration in the full frontal crash $f_2(x)$ (Equation 2), (iii) the toe-board intrusion in the 40% offset-frontal crash $f_3(x)$ (Equation 3).

$$\begin{aligned} f_1(x) = 1640.2823 + 2.3573285x_1 + 2.3220035x_2 \\ +4.5688768x_3 + 7.7213633x_4 + 4.4559504x_5 \end{aligned} \tag{1}$$

$$\begin{aligned} f_2(x) = 6.5856 + 1.15x_1 - 1.0427x_2 + 0.9738x_3 + 0.8364x_4 - 0.3695x_1x_4 \\ +0.0861x_1x_5 + 0.3628x_2x_4 - 0.1106x_1x_1 - 0.3437x_3x_3 + 0.1764x_4x_4 \end{aligned} \tag{2}$$

$$\begin{aligned} f_3(x) = -0.0551 + 0.0181x_1 + 0.1024x_2 + 0.0421x_3 \\ -0.0073x_1x_2 + 0.024x_2x_3 - 0.0118x_2x_4 - 0.0204x_3x_4 \\ -0.008x_3x_5 - 0.0241x_2x_2 + 0.0109x_4x_4 \end{aligned} \tag{3}$$

2.2 Car Side Impact

Car Side Impact Problem [12] is a three-objective constrained problem which involves the optimization of a vehicle side impact crashworthiness. This problem uses seven decision variables describing the thickness of B-Pillars, floor, cross-members, door beam, roof rail, etc. Three objective functions are considered: (i) the weight of car $f_1(x)$ (Equation 4), (ii) the pubic force experienced by a passenger $f_2(x)$ (Equation 5), and (iii) the average velocity of the V-Pillar responsible for withstanding the impact load $f_3(x)$ (Equation 6).

$$f_1(x) = 1.98 + 4.9x_1 + 6.67x_2 + 6.98x_3 + 4.01x_4 + 1.78x_5 + 0.00001x_6 + 2.73x_7 \tag{4}$$

$$f_2(x) = 4.72 - 0.5x_4 - 0.19x_2x_3 \tag{5}$$

$$f_3(x) = 0.5 * (10.58 - 0.674x_1x_2 - 0.67275x_2 + 16.45 - 0.489x_3x_7 - 0.843x_5x_6) \tag{6}$$

This problem has eight additional constraints. For the spacing purpose, we omit them. It can be seen in [12] or in the external attach [1].

[1] https://github.com/vinixnan/PublicData/raw/master/Contrainsts.pdf

2.3 Machining

The machining problem [10] formulates machining recommendations under multiple criteria. This problem considered tests on aluminum cut with VC-3 carbide cutting tools as a basis to test the approach, which has significant automotive industry applications [10]. This problem has three decision variables and four objective functions. Speed (v), feed (j) and depth of cut (d) are attributes considered in the three decision variables definition: $x_1 = ln(v)$, $x_2 = ln(1000j)$ and $x_3 = ln(1000d)$. Four objectives are considered: (i) minimizing the surface roughness $f_1(x)$ (Equation 7); (ii) maximizing the surface integrity $f_2(x)$ (Equation 8), which refers to the amount of undamaged primary silicon at and immediately below the surface; (iii) maximizing the tool life $f_3(x)$ (Equation 9), which is generally defined as the machining time to reach a fixed amount of uniform flank wear; and (iv) maximizing the metal removal rate $f_4(x)$ (Equation 10), which is a measure of parts made per unit machining time.

$$f_1(x) = -7.49 + 0.44x_1 - 1.16x_2 + 0.61x_3 \tag{7}$$

$$f_2(x) = -4.31 + 0.92x_1 - 0.16x_2 + 0.43x_3; \tag{8}$$

$$f_3(x) = 21.90 - 1.94x_1 - 0.30x_2 - 1.04x_3 \tag{9}$$

$$f_4(x) = -11.331 + x_1 + x_2 + x_3 \tag{10}$$

This problem has three additional constraints. For the spacing purpose, we omit them. It can be seen in [10] or in the external attach [1].

2.4 Water Resource Planning

The Water Resource Planning [23] is a five-objective constrained problem which involves optimal planning for a storm drainage system in an urban area. The problem variables are the local detention storage capacity x_1, the maximum treatment rate x_2 and the maximum allowable overflow rate x_3. There are five objective functions to be minimized: (i) the drainage network cost $f_1(x)$ (Equation 11), (ii) the storage facility cost $f_2(x)$ (Equation 12), (iii) the treatment facility cost $f_3(x)$ (Equation 13), (iv) the expected flood damage cost $f_4(x)$ (Equation 14), and (v) the expected economic loss due to flood $f_5(x)$ (Equation 15).

$$f_1(x) = 106780.37 * (x_2 + x_3) + 61704.67 \tag{11}$$

$$f_2(x) = 3000x_1 \tag{12}$$

$$f_3(x) = \frac{305700 * 2289x_2}{(0.06 * 2289)^{0.65}} \tag{13}$$

$$f_4(x) = 250 * 2289 * \exp(-39.75x_2 + 9.9x_3 + 2.74) \tag{14}$$

$$f_5(x) = 25 * (\frac{1.39}{x_1x_2} + 4940x_3 - 80) \tag{15}$$

This problem has seven additional constraints. For the spacing purpose, we omit them. It can be seen in [23] or in the external attach [1].

3 Multi-Objective Evolutionary Algorithms

Multi-Objective Evolutionary Algorithms (MOEAs) are algorithms inspired by nature, in special by Darwin's survival of the fittest. Thus, solutions for an optimization problem are individuals that reproduce, suffer eventually some mutation, and compete for survival.

Some MOEAs employ the concept of Pareto Dominance [18] in order to find a non-dominated set. This is performed by selecting solutions which do not dominate one another, that is, no solution is better than another with respect to all objective functions.

Hence, different replacement strategies have been proposed by several algorithms in the literature. We present four of them that presently are the most popular ones. All of them consider a population of solutions P, a population of offspring solutions O, and a maximum population size N.

3.1 Indicator-Based Evolutionary Algorithm

(IBEA) [28] performs a replacement considering a specific quality indicator (explained in Subsection 3.6). Thus, this algorithm selects surviving solutions from $P \cup O$ by removing the ones who contribute less to increase the given quality indicator. The quality indicator which is usually adopted is Hypervolume.

3.2 Non-dominated Sorting Genetic Algorithm-II

(NSGA-II) [9] performs the replacement strategy considering Pareto Dominance and Crowding Distance selection. This latter measures how close a solution is to its neighbors. Large values allow better diversity in the population. Thus, NSGA-II selects surviving solutions from $P \cup O$ first taking non-dominated solutions to compose P' and while $|P'| < N$, it adds dominated solutions according to the Crowding Distance values.

3.3 Strength Pareto Evolutionary Algorithm

(SPEA2) [29] performs the replacement strategy considering Pareto Dominance and the use of Strength values. Strength values are computed considering, for each solution, the number of solutions that it dominates and those that it is dominated by. Thus, SPEA2 selects surviving solutions from $P \cup O$ first taking non-dominated solutions to compose P' and while $|P'| < N$, it adds dominated solutions according to Strength values.

3.4 Generalized Differential Evolution 3

Differently from previously presented algorithms, (GDE3) [13] doesn't employ a crossover operator to generate new solutions. This algorithm instead uses the differential evolution operator [21]. Differently from the crossover operator, DE

generates new solutions by combining more than three different solutions. In most of the cases, a DE is just applied to continuous optimization problems, because DE generates offspring parameters by calculating weighted difference among solutions parameters. In order to define which solution should survive, GDE3 employs the same surviving selection method used by NSGA-II.

3.5 Remarks

In our previous work [5], we used NSGA-II, SPEA2 and IBEA as MOEAs. In the current work, we added GDE3 to this set, because this algorithm uses a different way to combine solutions instead of applying crossover. This helps MOABHH because of its different way to explore the search space.

3.6 Quality Indicators

There are some quality indicators to assess a MOEA performance. In this session, some of them are presented:

3.7 AE

The Algorithm Effort quality indicator can be defined as the ratio of the total number of function evaluations N_{eval} over a fixed period of simulation time T_{run} [22]. This indicator is interesting to evaluate how fast a MOEA can generate solutions.

$$AE = \frac{T_{run}}{N_{eval}} \tag{16}$$

There are other quality indicators where the returning set of solutions (S) are employed in order to evaluate a MOEA quality. That is the case of the following quality indicators:

3.8 RNI

The ratio of non-dominated solutions [22] evaluates the percent of non-dominated solutions $ND(S)$ in the population S, as shown in Equation 17. Higher RNI values are better than lower ones.

$$RNI(S) = \frac{|ND(S)|}{|S|} \tag{17}$$

3.9 Hypervolume

The hypervolume [30] of a non-dominated solution set S is the size of the part of the objective space that is dominated collectively by the solutions in S [26]. Thus, the hypervolume indicator computes the area (or volume when more than two objectives are employed) in the search space [30]. Equation 18 presents how to calculate this indicator, where v_i is the volume. Higher hypervolumes are preferred to lower ones.

$$HV(S) = volume(\cup_{i=1}^{|S|} v_i) \tag{18}$$

3.10 HR

The Hyper-area Ratio (HR) [24] employs the hypervolume of a solution set A divided by the hypervolume value of a Reference Front B. Higher values are preferred to lower ones.

$$HV(S,P) = \frac{HV(A)}{HV(B)} \tag{19}$$

3.11 ER

Pareto Dominance Indicator (ER) [11] considers the solutions intersection between two given sets A and B, which can be provided by different algorithms or used to compare a solution set S with a Pareto Front P. Equation 20 presents ER, where the size of the intersection is compared with the size of B. Higher values are preferred to lower ones.

$$ER(A,B) = \frac{|A \cap B|}{|B|} \tag{20}$$

3.12 UD

The Uniform distribution of non-dominated population evaluates how distributed are the solutions along the search space. The distribution should be as uniform as possible to achieve consistent gaps among neighboring individuals in the population [22]. This quality indicator is calculated according to Equation 21.

$$UD(S) = \frac{1}{1 + S_{nc}} \tag{21}$$

where S_{nc} is the standard deviation of niche count of the overall set of $NonDominated(S)$ (Equation 22).

$$S_{nc}(ND) = \sqrt{\frac{\sum_{i}^{|ND|}(nc(nd_i) - \overline{nc}(ND))^2}{|ND| - 1}} \tag{22}$$

where $|ND|$ is the size of the non-dominated set ND of the population S; $nc(nd_i)$ is the niche count of i^{th} a solution;

$$nc(nd_i) = \sum_{j,j \neq i}^{|ND|} f(i,j), f(i,j) = \left\{ \begin{array}{l} 1 \text{ if } dist(i,j) < \sigma_{share} \\ 0 \text{ otherwise} \end{array} \right\} \tag{23}$$

$\overline{nc}(ND)$ is the mean value of $nc(nd_i)$ and dist(i, j) is the distance between individual i and j in the objective domain.

3.13 Remarks

In our previous work [5], we used Hypervolume, RNI, GD [20], IGD [31], and Spread [20] as quality indicators. However, as some of them need previous problem knowledge, such as Spread, IGD and GD, they were not employed in the

current work.[2]. Moreover, in the current work we added UD, HR and ER to this set, because they don't need a Pareto Front specification. Thus, we can evaluate subpopulations of solutions by considering the whole population. We also employed AE, which help us to evaluate the computational efficiency time along the execution.

4 MOABHH framework

MOABHH (Multi-Objective Agent-Based Hyper-Heuristic) [5] is an agent-based hyper-heuristic framework focused selecting during execution time the most suitable MOEA by employing voting techniques. The main idea is to consider MOEAs as candidates and quality indicators as voters in an election. Then, with the election outcome, a hyper-heuristic agent can assign more or fewer resources for a given MOEA. This framework can be instantiated using different quality indicators and MOEAs. Figure 1 shows MOABHH interaction, where three kinds of agents (MOEAs, Voters and HH) share four kinds of *artifacts*. In this figure, Agents are represented by circles, Artifacts by parallelograms, instances by rectangles. Solid arrows mean writing permission and dotted lines mean reading permission.

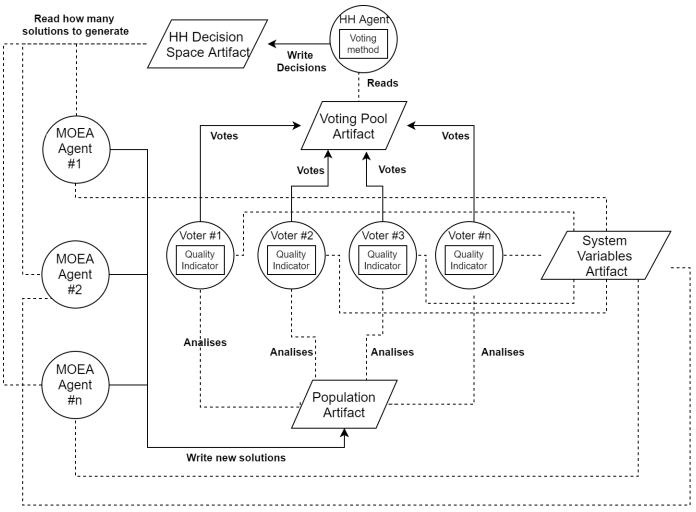

Fig. 1. MOABHH interaction, adapted from [5]

[2] These indicators are not described here.

4.1 Artifacts

Artifacts are non-autonomous, function-oriented, stateful entities, designed by Multi-Agent Systems programmers. They are controllable and observable, and that are used to model the tools and resources used by agents [19]. MOABHH employs four different artifacts:

- *Population artifact* keeps the main current population of solutions. This artifact is used by *Indicator voter agents*. When MOABHH starts, the Problem Manager agent randomly generates the first population, and then assigns it to this artifact;
- *Voting pool artifact* keeps all the voting preferences necessary to perform an election. This artifact is used by *voter agents* and the HH Agent, who defines which MOEA wins the election according to its voting rules;
- *HH decision space artifact* keeps all HH agent decisions. All decisions can be read by MOEA agents. These decisions define how many solutions each MOEA agent has to generate in the next cycle;
- *System variables artifact* keeps the current values of the optimization problem parameters: (i) reference points for quality indicators and MOABHH variables such population size, the number of generations in the training phase (δ) and the number of generations that each MOEA can execute (γ). This artifact is readable by all agents, and may be written by the Problem Manager Agent.

4.2 Agents

An agent is a computer system that is situated in some environment, and is capable of autonomous action in this environment in order to meet its objectives [27]. MOABHH uses four different types of agents:

- The *Problem Manager agent* is responsible for receiving all MOABHH parameters and set them in the *System Variables* artifact;
- The *MOEA agent* contains a particular evolutionary algorithm instance. This agent is responsible for generating a given number of solutions in a generation, where the number is defined by HH agent. After generating new solutions, this agent adds the generated solutions to *Population Artifact*. All generated solutions are associated with their respective generator agent. Thus, it is possible to evaluate the MOEA agent performance;
- The *Voter agent* contains an instance of a quality indicator. This agent works as follows: (i) First, each *Voter agent* reads the current population from Population Artifact and splits it into subpopulations, associating each of them with the MOEA agent that has generated it; (ii) then, this agent evaluates each subpopulation according to its instanced quality indicator and (iii) the agent votes, sending a ranking of its preferences following the quality values to the *Voting pool artifact*;
- The *HH agent*, the hype-heuristic agent, uses information available on the *Voting Pool Artifact* to employ a voting method. With the voting outcome, this agent defines how many solutions each MOEA can generate, giving to the best MOEA more solutions to generate in the next cycles.

4.3 Algorithm

Algorithm 1 shows MOABHH steps. First, all agents and artifacts are initialized using MOABHH parameters, the optimization problem is instantiated and all global variables set in *System Variables* artifact. The *Problem manager* agent then generates a random population of solutions. The execution continues performing the training phase where all MOEA agents receive the same number of solutions to generate. This is necessary because it is hard to evaluate MOEA at the beginning of the search. Thus, the task of comparing algorithm performance becomes harder. So, MOABHH does not perform any quality evaluation until the training phase finishes, after δ generations. After finishing the training phase, the voting process starts by *Voter agents* evaluating MOEAs related solutions, ranking preferences and setting the ranking to *Voting pool* artifact. After *voter agents* vote, the *HH* agent performs the voting method and assign a bigger number of solutions to generate in the next generations to the election winner and a lower number of solutions to the election losers. After updating the number of solutions to generate, all MOEA agents execute over γ generations and add new solutions to *Population* artifact.

Algorithm 1: MOABHH Pseudocode.

```
1  Input: Problem, γ - generations for each LLH execute, δ - The training size
2  begin
3      Initialize agents and artifacts;
4      Generate a random population of solutions;
5      while Training do
6          Uniformly share the number of solutions to generate among Low-Level Heuristic
             agents;
7          Low-Level Heuristic Agents execute for γ generation;
8          Update the main population artifact;
9      end
10     while Executing do
11         Voters evaluate Low-Level Heuristics outcome and vote;
12         HH agent performs the voting method;
13         HH agent shares the number of solutions to generate among Low-Level Heuristic
             agents according to voting results;
14         Low-Level Heuristic agents execute for γ generations;
15         Update the main population;
16     end
17     return Main population
18 end
```

Surviving solutions are defined by each MOEA agent, that means if a metaheuristic agent has to generate n new solutions, the resulting population also will have n solutions.

In [5], we had defined fixed rules for the increment in the participation of generating solutions. A increment of $0.75 * \beta$ in the participation pool was given to the election winner, and an increment of $0.25 * \beta$ to the second best rated MOEA, in case of three MOEA agents running. For just two MOEA agents, we augment a rate of β to the winner. The worst rated MOEA aways lost a rate of β. On the other hand, fixed rules makes the approach harder to adapt to a bigger set of MOEAs. Thus, we defined in this work a function (Equation 25) in

order to give us similar values, but allowing different sizes of MOEA set. In this function, n is the number of candidates.

$$fx(pos, n) = \left\{ \begin{array}{l} 2^n \text{ if } pos = 1 \\ 0 \text{ if } pos = n \\ 2^{n-pos} \text{ otherwise} \end{array} \right\} \quad (24) \qquad \forall pos \in rank \frac{fx(pos, n)}{\sum_{i=1}^{n} fx(i, n)} * \beta \quad (25)$$

4.4 Voting Method

The Copeland Voting method [7] is a Condorcet method. The Condorcet's principle says that if a candidate defeats every other candidate in pairwise comparisons (a Condorcet winner), it must be elected [17]. Thus, after all voters vote, a pairwise comparison is performed. In Figure 2, we have three different candidates and voters. Then we perform the one-on-one contest for each candidate in all voters. Thus, the Condorcet's principle has the fundamental idea that the opinion of the majority should prevail, at least when majority comparisons pinpoint an unambiguous winner [17].

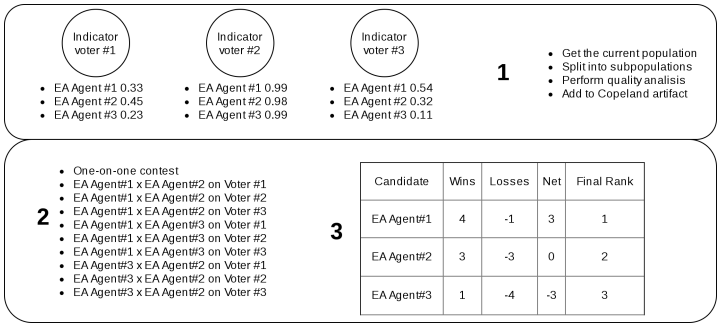

Fig. 2. Voting procedure, adapted from [5]

5 Experiments

5.1 Setup

In our experiments, we used four independent evolutionary algorithms as MOEAs: NSGA-II, SPEA2, IBEA and GDE3, described in section 3. These four algorithms have been extensively applied in several multi-objective optimization problems, so they are suitable for our approach. IBEA, NSGA-II and SPEA2 were set according to [8], using as heuristics SBX Crossover (with distribution 30 and rate 1.0) and Polynomial Mutation (with distribution 20 and rate de $1/n$, where n =number of problems variables). The population size was defined as

100. GDE3 also employed the same mutation configuration and used *rand/1/bin* differential evolution using $Cr = 0.2, F = 0.2$ and $K = 0.5$ as parameters.

MOABHH parameters was set as $\lambda = \delta = 40$, $\beta = 3$. As *Voter Agents* we employed all six quality indicators mentioned in Section 3.6. As *MOEA agents* we employed the four mentioned MOEAs. All experiments were executed 40 times, over 1000 generations.

The four studied problem don't have a true Pareto Front P, due to the fact they are aren't benchmarks. However, to properly evaluate a multi-objective algorithm we need a Pareto Front. With this purpose, we created an approximated Pareto Front P_{approx}.

In order to create a P_{approx}, first we generated a set of solutions SS composed by all found solutions, for a give problem, considering all algorithms (four MOEA and MOABHH). Then we generated $P_{approx} = ND(SS)$, selecting just non-dominated solutions to compose P_{approx}.

Posteriorly, all algorithm performance (40 values for each algorithm) were calculated using Hypervolume, IGD [31] and Epsilon [32]. For Hypervolume, we defined the reference point using the worse point found in P_{approx}. Finally, all indicator averages values for each problem were taken and statistically compared using Kruskal-Wallis test with 1% of significance.

5.2 Results

Table 1 presents Hypervolume, IGD and Epsilon averages for all algorithms. Bold values mean statistical equivalence and highlighted cells denotes the best values. Lower IGD and Epsilon values and higher Hypervolumes are preferred.

For the Car Side Impact problem, MOABHH found statistically tied results with IBEA considering hypervolume, and has overcome all others algorithms using IGD;however, it was beaten by GDE3 considering the Epsilon indicator. If we analyze just the individual MOEAs outcome, we can see that IBEA overcomes all other MOEAs (GDE3, NSGA-II, and SPEA2) in hypervolume and IGD, as GDE3 does considering the Epsilon indicator. Hence, MOABHH was rated better than most individual MOEAs, and it was just overcome in our experiments considering the Epsilon values.

For the CrashWorthiness problem, MOABHH found the best value in all the quality indicators, but statistically tied results with GDE3, which is considered in th literature so far as the best MOEA for this problem.

For the Water problem, MOABHH found the best value in all the quality indicators, with statistical difference considering Hypervolume and IGD indicators; it tied with GDE3 with respect to the Epsilon indicator. In in our experiments, GDE3 has shown to be the best MOEA in this problem.

For the Machining problem, MOABHH statistically tied results with IBEA, which is the best MOEA, considering the three quality indicators; MOABHH found the best IGD average while IBEA found the best solutions considering Hypervolume and Epsilon indicators.

Considering all results, MOABHH statistically overcomes all individual MOEAs three times. In the other eight times, MOABHH statistically tied with

them. Within these eight times, MOABHH found higher hypervolumes and lower IGD/Epsilon five times. There was just once MOABHH had statistically overcome by individual MOEAs.

These are interesting and promising results, since MOABHH's goal is to diminish the user effort to choose a MOEA without losing performance. In some cases, our approach has also overcomes individual MOEAs results, showing that the HH agent has chosen better solutions generated by all MOEAs.

Table 1. Hypervolume, IGD and Epsilon Result Table

	Problem	GDE3	IBEA	NSGAII	SPEA2	MOABHH
Hyp.	Car Side Impact	4.4342E-01	**4.7710E-01**	3.7671E-01	4.4507E-01	**4.7161E-01**
	CrashWorthiness	**7.3603E-01**	7.0594E-01	6.6108E-01	7.2210E-01	**7.3985E-01**
	Water	5.6227E-01	5.0439E-01	4.3440E-01	4.9700E-01	**5.8632E-01**
	Machining	1.8393E-01	**2.7348E-01**	1.7288E-01	1.7705E-01	**2.7118E-01**
IGD	Car Side Impact	7.8878E-04	8.1957E-04	1.2878E-03	7.5318E-04	**6.6803E-04**
	Crash Worthiness	**6.9652E-04**	2.6247E-03	1.2639E-03	7.5822E-04	**4.2570E-04**
	Water	1.4869E-03	3.5247E-03	2.1495E-03	1.9045E-03	**8.9055E-04**
	Machining	1.6902E-03	**5.1369E-04**	1.6953E-03	1.7521E-03	**5.0530E-04**
Ep.	Car Side Impact	**1.6403E-01**	9.6482E-02	1.9015E-01	1.8226E-01	1.3509E-01
	Crash Worthiness	**5.3299E-02**	1.4667E-01	1.1723E-01	6.4985E-02	**4.3900E-02**
	Water	**1.4684E-01**	2.5247E-01	2.5750E-01	2.1015E-01	**1.1912E-01**
	Machining	4.8167E-01	**1.6378E-01**	4.9150E-01	5.0770E-01	**1.9654E-01**

6 Conclusions

This paper evaluated MOABHH (Multi-Objective Agent-Based Hyper-Heuristic) in real-world applications, by searching solutions for four multi-objective engineering optimization problems. As we intended to solve real-world problems, we enhanced the proposal initially published in [5] by applying a different quality indicator set and also employing GDE3 as MOEA agent, making MOABHH more powerful.

The results showed that MOABHH was very competitive against the best known MOEAs, in most of the cases, it has found better Hypervolume, IGD and Epsilon averages, sometimes with statistical difference. We believe that this makes this approach interesting for engineers to solve their real-world problems, without having to test all other algorithms in order to find which is the best one.

As for further work, we intend to evaluate different voting methods and to use a larger number of MOEAs in future experiments.

Acknowledgments

Vinicius de Carvalho is supported by CNPq, Brazil, grant 140974/2016-4.

References

1. Acan, A., Lotfi, N.: A multiagent, dynamic rank-driven multi-deme architecture for real-valued multiobjective optimization. Artificial Intelligence Review **48**, 1–29 (2016)
2. Boussaid, I., Lepagnot, J., Siarry, P.: A survey on optimization metaheuristics. Information Sciences **237**, 82 – 117 (2013)
3. Bradstreet, L., Barone, L., While, L., Huband, S., Hingston, P.: Use of the wfg toolkit and pisa for comparison of moeas. In: 2007 IEEE Symposium on Comput. Intell. in Multi-Criteria Decision-Making. pp. 382–389 (April 2007)
4. Burke, E.K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E.and Qu, R.: Hyper-heuristics: a survey of the state of the art. Journal of the Operational Research Society **64**(12), 1695–1724 (Dec 2013)
5. de Carvalho, V.R., Sichman, J.S.: Applying Copeland Voting to Design an Agent-Based Hyper-Heuristic. In: Proc. of the 16th Conference on Autonomous Agents and MultiAgent Systems. pp. 972–980 (2017)
6. Coello, C.: Evolutionary algorithms for solving multi-objective problems. Springer, New York (2007)
7. Copeland, A.H.: A reasonable social welfare function. In: Mimeographed notes from a Seminar on Applications of Mathematics to the Social Sciences, University of Michigan (1951)
8. Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. IEEE Trans. on Evol. Comput. **18**(4), 577–601 (Aug 2014)
9. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. on Evol. Comput. **6**(2), 182–197 (Apr 2002)
10. Ghiassi, M., DeVor, R., Dessouky, M., Kijowski, B.: An application of multiple criteria decision making principles for planning machining operations. IIE Transactions **16**(2), 106–114 (1984)
11. Goh, C.K., Tan, K.C.: A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization. IEEE Transactions on Evolutionary Computation **13**(1), 103–127 (Feb 2009)
12. Jain, H., Deb, K.: An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part ii: Handling constraints and extending to an adaptive approach. IEEE Trans. Evolutionary Computation **18**(4), 602–622 (2014)
13. Kukkonen, S., Lampinen, J.: Gde3: The third evolution step of generalized differential evolution. In: Evolutionary Computation, 2005. The 2005 IEEE Congress on. vol. 1, pp. 443–450. IEEE (2005)
14. Li, B., Li, J., Tang, K., Yao, X.: Many-objective evolutionary algorithms: A survey. ACM Comput. Surv. **48**(1), 13:1–13:35 (Sep 2015)
15. Liao, X., Li, Q., Yang, X., Zhang, W., Li, W.: Multiobjective optimization for crash safety design of vehicles using stepwise regression model. Structural and multidisciplinary optimization **35**(6), 561–569 (2008)
16. Maashi, M., Özcan, E., Kendall, G.: A multi-objective hyper-heuristic based on choice function. Expert Systems with Applications **41**(9), 4475–4493 (2014)
17. Moulin, H.: Condorcet's principle implies the no show paradox. Journal of Economic Theory **45**(1), 53–64 (1988)
18. Pareto, V.: Manual of political economy. AM Kelley (1971)

19. RICCI, A. and VIROLI, M. and OMICINI, A.: Give agents their artifacts: the a&a approach for engineering working environments in MAS. In: Durfee, E.H., Yokoo, M., Huhns, M.N., Shehory, O. (eds.) 6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007), Honolulu, Hawaii, USA, May 14-18, 2007. p. 150. IFAAMAS (2007)
20. Srinivas, N., Deb, K.: Multiobjective optimization using nondominated sorting in genetic algorithms. IEEE Trans. on Evol. Comput. **2**, 221–248 (1994)
21. Storn, R., Price, K.: Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization **11**(4), 341–359 (Dec 1997)
22. Tan, K.C., Lee, T., Khor, E.: Evolutionary algorithms for multi-objective optimization: Performance assessments and comparisons. Artificial Intelligence Review **17**(4), 251–290 (2002)
23. Tapabrata, R., Kang, T., Seow, K.C.: Multiobjective design optimization by an evolutionary algorithm. Engineering Optimization **33**(4), 399–424 (2001)
24. Van Veldhuizen, D.A.: Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations. Ph.D. thesis, Air Force Institute of Technology, Wright Patterson AFB, OH, USA (1999), aAI9928483
25. Vázquez-Rodríguez, J.A., Petrovic, S.: A mixture experiments multi-objective hyper-heuristic. Journal of the Operational Research Society **64**(11), 1664–1675 (2012)
26. While, L., Bradstreet, L., Barone, L.: A fast way of calculating exact hypervolumes. IEEE Trans. on Evol. Comput. **16**(1), 86–95 (Feb 2012)
27. Wooldridge, M.: An introduction to multiagent systems. John Wiley & Sons (2009)
28. Zitzler, E., Künzli, S.: Indicator-based selection in multiobjective search. In: PPSN. Lecture Notes in Computer Science, vol. 3242, pp. 832–842. Springer (2004)
29. Zitzler, E., Laumanns, M., Thiele, L.: SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In: Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems. pp. 95–100. International Center for Numerical Methods in Engineering (2001)
30. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. Trans. on Evol. Comput. **3**(4), 257–271 (nov 1999)
31. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., da Fonseca, V.G.: Performance assessment of multiobjective optimizers: An analysis and review. Trans. on Evol. Comput. **7**(2), 117–132 (Apr 2003)
32. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., da Fonseca, V.G.: Performance assessment of multiobjective optimizers: an analysis and review. IEEE Transactions on Evolutionary Computation **7**(2), 117–132 (April 2003)