Towards Many-objective Optimisation with Hyper-heuristics: Identifying Good Heuristics with Indicators

David J. Walker[1] and Ed Keedwell[1]

University of Exeter, United Kingdom {D.J.Walker, E.C.Keedwell}@exeter.ac.uk

Abstract. The use of hyper-heuristics is increasing in the multi-objective optimisation domain, and the next logical advance in such methods is to use them in the solution of *many-objective* problems. Such problems comprise four or more objectives and are known to present a significant challenge to standard dominance-based evolutionary algorithms. We incorporate three comparison operators as alternatives to dominance and investigate their potential to optimise many-objective problems with a hyper-heuristic from the literature. We discover that the best results are obtained using either the favour relation or hypervolume, but conclude that changing the comparison operator alone will not allow for the generation of estimated Pareto fronts that are both close to and fully cover the true Pareto front.

1 Introduction

As the field of hyper-heuristic research matures, attention is moving from solving problems requiring the optimisation of a single objective to those comprising two or more objectives. A recent paper [14] proposed a multi-objective extension of a single-objective algorithm that identifies good sequences of heuristics to apply to a given problem. The original single-objective algorithm updates the transition probabilities that govern the selection of the next heuristic using the raw fitness value, and in the multi-objective extension this was replaced with a dominance-based approach.

Though such *multi-objective* problems are prevalent, it is well known that optimisation problems often comprise a large set of objectives that must be simultaneously optimised [10]. Problems with four or more objectives are often called *many-objective* problems. Using dominance-based multi-objective algorithms to solve many-objective problems is generally problematic, as the dominance relation does not scale well to even relatively small numbers of objectives; solutions quickly become incomparable, as they are considered equivalent under dominance. A considerable amount of research in the evolutionary computation field has been devoted to the investigation of evolutionary algorithms that are capable of solving many-objective problems. These generally take one of two approaches: either some of the problem objectives must be discarded so that a standard multi-objective EA can be employed, or an alternative to the dominance relation must be found. To our knowledge, no work in the hyper-heuristic field has considered many-objective problems. In this paper, we begin to investigate many-objective test problems [4], considering problems comprising four, five and six objectives. We take inspiration from work on dominance alternatives and consider indicators to investigate how useful they are when incorporated into a recent multi-objective hyper-heuristic, such that the indicator replaces the dominance relation for comparing solutions.

The remainder of this paper is organised as follows. Some relevant background material is presented in Section 2 before the indicators we examine are introduced in Section 3. Section 4 presents our experimental setup, and results are discussed in Section 5. We discuss our conclusions and future work in Section 6

2 Background

2.1 Many-objective Optimisation

In the last decade research on *many*-objective optimisation has increased rapidly. A solution $\mathbf{x}$ to an arbitrary many-objective optimisation problem is described by an M-dimensional objective vector $\mathbf{y}$, such that $M \geq 4$:

$$\mathbf{y} = (f_1(\mathbf{x}), \ldots, f_M(\mathbf{x})). \tag{1}$$

Evolutionary algorithms are known to generate good solution sets to multi-objective problems. Such algorithms often use the *dominance* relation to compare the relative quality of two solutions. With the advent of research into many-objective optimisation it has been known that dominance does not scale well to compare many-objective solutions. As the number of objectives increases, so does the likelihood that two solutions will be equivalent; given just a 5-objective problem, and a uniform distribution of solutions, solutions residing in approximately 95% of objective space will be incomparable under dominance.

Various approaches have been taken to address the inability of dominance-based MOEAs to optimise many-objective problems. Generally, these approaches either involve finding an approach that can compare solutions described by a large number of objectives [3] or identifying redundant objectives that can be discarded so that a standard dominance-based MOEA can be used. This work takes the former option, and we consider three approaches to comparing many-objective solutions; these are described in Section 3.

2.2 Hyper-heuristics

Hyper-heuristics are techniques that identify low-level heuristics that generate good solutions to optimisation problems. They operate above the *domain barrier*, meaning that they optimise the heuristics, rather than the solutions to a given optimisation problem, and require no problem-specific information to function.

They have been applied in a wide range of problem domains, often solving combinatoric problems but also in the continuous domain, to great success. Hyper-heuristics are either *generative* or *selection*-based. A generative hyper-heuristic creates novel low-level heuristics, such as mutation or crossover operators, that are tailored to work on a specific type of problem. In this work we only consider selection-based methods, which operate with a pre-defined pool of low-level heuristics and identify those that are well suited to a specific problem domain.

A central part of a selection hyper-heuristic is the mechanism by which the next low-level heuristic to apply is chosen. Common methods are random selection, selection with choice function, and more recently Markov-based methods. In this work we employ an algorithm based on a hidden Markov model [14], which is described later. A recent survey of hyper-heuristic approaches is provided in [2].

The use of hyper-heuristics within many-objective optimisation has received very little attention. Some studies have used them to solve multi-objective problems. One recent example was [13], which employed a reinforcement learning-based Markov chain approach to solving continuous multi-objective problems. Another approach incorporated the hypervolume indicator [8] into the move acceptance strategy of a hyper-heuristic and applied it to solve multi-objective test problems [11] [12]. [9] presented a multi-objective hyper-heuristic designed to operate in the realm of search-based software engineering; their algorithm was based on NSGA-II, and used choice function in concert with a multi-armed bandit to select low-level heuristics. With the exception of the hypervolume example, these methods rely heavily on dominance; as discussed earlier, we hypothesise that these approaches will not scale well to deal with many-objective problems, and we discuss potential ways of addressing this issue in the next section.

3 Indicators

Since the discovery that standard, dominance-based, evolutionary optimisers do not provide sufficient selective pressure to locate an acceptable estimate of a many-objective problem's Pareto front [7] significant research efforts have been spent investigating alternatives to dominance. Three that are considered in this study are *hypervolume* [8], the *favour relation* [5], and an indicator based on the average rank method [1].

3.1 Hypervolume

An early contribution, which remains one of the principle indicators of solution quality is the hypervolume [8]. The hypervolume is the dominated space between a solution (or solutions) and a pre-defined reference point. Hypervolume has been used as an indicator in a range of studies, including a recent work in which it was incorporated into the acceptance strategy of a hyper-heuristic [11]. That work considered continuous multi-objective test problems, however was restricted to 2-objective problems only. Though the hypervolume scales to any number of objectives, it is often restricted to problems with low numbers of objectives because of its computational complexity when calculated exactly for a population (though it can be estimated accurately with Monte Carlo sampling [6]). This work is not hindered by such complexity issues, as the calculation is trivial for a single solution.

3.2 Favour relation

Given two solutions $\mathbf{y}_i$ and $\mathbf{y}_j$, the favour relation [5] determines which is the fitter solution in terms of which is dominant on the most objectives. More formally:

$$\mathbf{y}_i <_f \mathbf{y}_j \Leftrightarrow |\{m : y_{im} < y_{jm}\}| > |\{m : y_{jm} < y_{im}\}|. \tag{2}$$

We incorporate the favour relation as a direct replacement for dominance, such that transition probabilities and the parent solution for the next generation are updated if the current parent solution does not favour its child.

3.3 Average Rank

Many of the dominance alternatives that have been proposed in the literature are based on population ranking. The MOSSHH algorithm is a point-based approach, and thus has no population that can be ranked. That said, it does have an external archive of non-dominated solutions that can be ranked, we use it in combination with the average rank method [1].

Given a population $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^N$ of solutions, the solutions are ranked M times, once according to each objective such that r_{im} is the rank of the i-th solution on the m-th objective. The average rank $\bar{r}_i$ is then computed with:

$$\bar{r}_i = \frac{1}{M} r_{im}. \tag{3}$$

In order to use this formulation as an indicator we calculate the average rank of the elite archive. The indicator returns 1 if the rank of the new solution is superior to its parent, and 0 otherwise. It is necessary to ensure that the child has been added to the archive, however due to the formulation of the algorithm described shortly, it is not possible to evaluate the indicator if the child has not been added to the archive.

4 Experiments

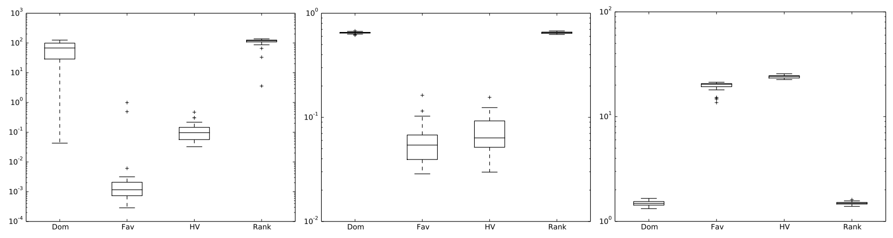
In order to determine the usefulness of the indicators outlined in Section 3 we now incorporate them into a hyper-heuristic to compare them against the dominance-based approaches that has been shown to work well for multi-objective problems comprising two or three objectives. We employ a selection hyper-heuristic called MOSSHH [14], in which sequences of low-level heuristics that lead to good solutions are identified. A transition probability matrix is

Algorithm 1 MOSSHH

```
1: $\mathbf{x}, h_c, A, B$ = initialise()
2: $E$ = initialise_archive()                         *Initialise the archive.*
3: repeat
4:     $h_p = h_c$
5:     $h_c$ = select($A, h_p$)                       *Choose the next heuristic*
6:     $AS$ = select($B, h_c$)                        *Set the acceptance strategy*
7:     record($h_p, h_c, AS$)                         *Record the current heuristic*
8:     $\mathbf{x}'$ = apply($\mathbf{x}, h_c$)       *Apply the heuristic to the current solution*
9:     if $AS == 1$ then
10:        $E$ = update_archive($E, \mathbf{f}(\mathbf{x}')$)   *If acceptance strategy was met update archive*
11:        if $\neg I(\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x}'))$ then
12:            $\mathbf{x} = \mathbf{x}'$             *If the indicator is true replace the parent with the child*
13:            if archived($\mathbf{f}(\mathbf{x}')$) then
14:                update_probabilities()
15:            end if
16:        end if
17:        clear_records()
18:    end if
19: until termination criterion met
```

maintained, which governs the transition from one low-level heuristic to another, and each heuristic has an *acceptance strategy*, which determines the likelihood that the solution generated by a given heuristic will be accepted. Both transition probabilities and acceptance strategies are learned using an online learning process.

The indicator-based multi-objective sequence-based hyper-heuristic (MOSSHH) [14] algorithm is described in Algorithm 1. The algorithm begins by initialising a random parent solution, choosing a starting low-level heuristic, and initialising the transition probability and acceptance strategy matrices uniformly (Line 1). An empty elite archive is initialised (Line 2). The first stage in each iteration of the iterative process is to select the next low-level heuristic and acceptance strategy using the current low-level heuristic (Lines 4-6). The chosen values are recorded (Line 7), in case the current sequence of low-level heuristics is identified as being useful, and the new low-level heuristic is applied to generate a new solution (Line 8). If the acceptance strategy is met ($AS == 1$) then the solution's objective values are evaluated and the archive is updated. Any solutions dominated by the new solution are discarded, and if the solution itself is not dominated by the archive then it is added to it (Line 10). At this point, the parent and child solutions are compared using one of the indicators. If the indicator deems that the child is superior to the parent, then the child solution succeeds the parent solution as the parent in the next generation. Otherwise, the current parent solution is retained. If the solution was added to the archive, then the sequence of low-level heuristics that led to it is complete, and transition probability and acceptance strategies are updated accordingly (Line 14).



(a) DTLZ1 (b) DTLZ2 (c) DTLZ6

Fig. 1: Generational distance results for 6-objective instances of DTLZ1, DTLZ2 and DTLZ6.

The problems we examine are drawn from the DTLZ suite of test problems [4]. Specifically, we investigate 4-, 5- and 6-objective instances of the DTLZ1, DTLZ2 and DTLZ6 test problems. Each problem has been selected to demonstrate the hyper-heuristic's ability to cope with specific problem features (such as deceptive fronts and a discontinuous Pareto front). The problems are parametrised as suggested in [4]. In the case of DTLZ1 and DTLZ6, the algorithm is run for 50,000 function evaluations. DTLZ2 is known to be a easier problem, and as such is run for just 5,000 function evaluations due to computational time constraints.

The following set of low-level heuristics is employed:

Ruin and recreate two versions; in the first, the entire solution is destroyed and replaced with a random feasible solution. In the second, a single parameter is chosen and replaced.

Mutation three additive mutation operators. In each, a parameter is chosen at random and mutated with an additive mutation drawn from one of three probability distributions (uniform, in the region $(-0.05, 0.05)$; Gaussian, with 0 mean and standard deviation 0.1; beta, in the region $(-0.05, 0.05)$).

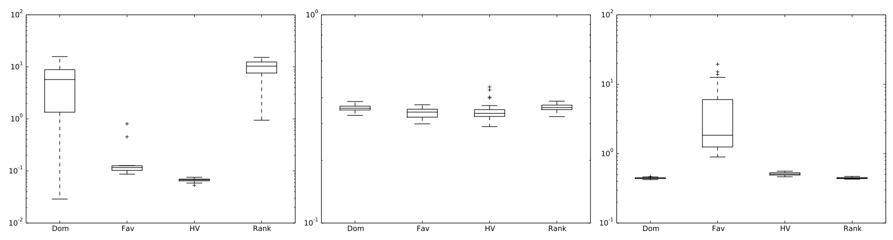
Archive selection two versions; one in which the entire solution is replaced with a solution drawn at random from the archive, and a second in which a parameter is replaced with an archived solution's corresponding parameter.

In total, $H = 7$ low-level heuristics are employed. To begin with, each has an equal probability of selection (transition probabilities are initialised to $1/H$).

Each problem is optimised with MOSSHH using each of the four indicators. In order to analyse the results, each instance of the problem is optimised 30 times for each of the four problems. We compare the results using the generational distance to examine the convergence properties of the algorithm, as well as using inverted generational distance to consider diversity.

5 Results

Figure 1 illustrates the generational distance results for 6-objective instances of DTLZ1, DTLZ2 and DTLZ6, while Figure 2 shows the corresponding inverted



(a) DTLZ1 (b) DTLZ2 (c) DTLZ6

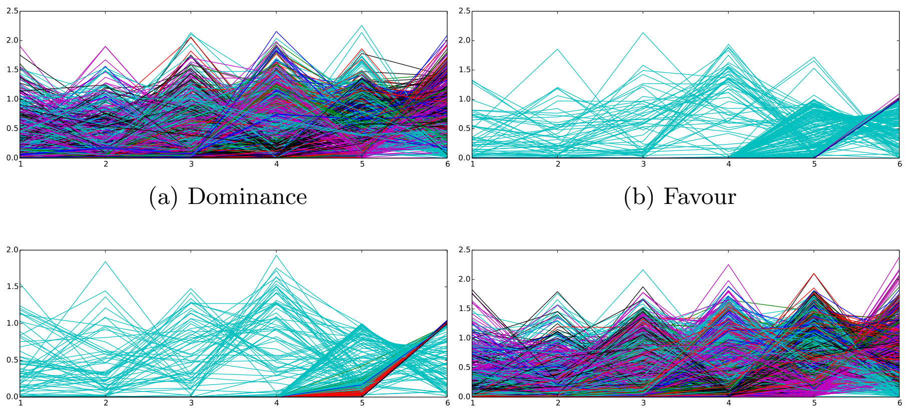
Fig. 2: Inverted generational distance results for 6-objective instances of DTLZ1, DTLZ2 and DTLZ6.

generational distance results. The corresponding 4- and 5-objective results are omitted for space. The results were generated by computing the mean distance from the final archive of solutions for each run to a set of Pareto optimal samples, in the case of generational distance, and the corresponding distance from the sample sets to the optimised solution sets, in the case of inverted generational distance.

In the case of DTLZ1, the results show that the dominance-based indicator has failed to converge to the Pareto front. Both the favour and hypervolume indicators have performed significantly better, converging much closer to the Pareto front, and covering it more extensively as can be seen from the IGD results. The rank-based indicator has not performed well, with at best comparable performance to that of the dominance indicator. The difference is less clear in the DTLZ2 case; though the favour and hypervolume indicators have again converged closer to the Pareto front, the difference is less significant. In terms of diversity, there is little to chose between the four alternatives; this is not surprising, as DTLZ2 is designed to be an easier problem for optimisers to solve.

Figure 3 shows representative estimated Pareto fronts obtained by optimising a 6-objective instance of DTLZ2 using the four indicators. To colour the solutions, the population was ranked to identify the objective on which each solution has the best rank. This information is used to colour the line representing each solution. As can be seen, the dominance and rank indicators have a spread of preferred objectives, whereas the favour and hypervolume indicators have optimised a specific objective (objective 6). The improved performance of these two indicators can be explained by this, as large numbers of solutions that optimise this objective (and objective 5) have been included in the estimated Pareto set, which means the overall mean distance between the estimated front and the true front is reduced.

The algorithm has managed to optimise DTLZ6, though, interestingly, the generational distance results are the reverse of those for DTLZ1 and DTLZ2. This is likely to be because of the available heuristics; given the propensity for the indicators to optimise specific regions of the Pareto front, as discussed above,



(a) Dominance (b) Favour

(c) Hypervolume (d) Rank

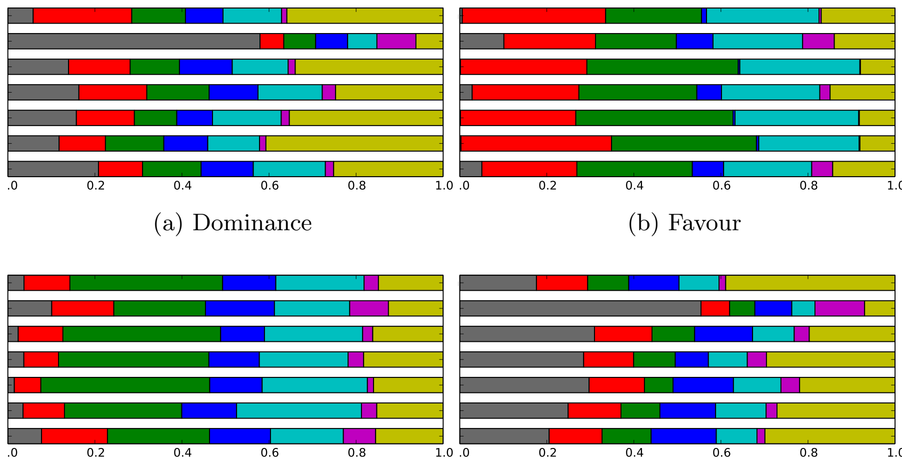
Fig. 3: Parallel coordinate plots showing the estimated Pareto front obtained by optimising a 6-objective instance of DTLZ2 with various indicators.

a problem with discontinuities will present difficulties for an optimiser that does not have crossover heuristics available to it. Once the algorithm has converged to a specific region of the Pareto front, the mutation heuristics used herein appear to lack the ability to cross discontinuities and, though the archive heuristics were intended to ameliorate this lack of crossover, they only allow the algorithm to return to areas of the space that have previously been explored. Crossover heuristics will be required to make this algorithm scale to problem features such as discontinuous Pareto fronts.

Figure 4 presents the transition probability matrices for the 6-objective instances of DTLZ1. In each case, the transition probability matrices from the thirty runs have been averaged. As can be seen, in the case of the dominance and rank-based indicators, which performed less well, there is a higher propensity to use the ruin and recreate and archive heuristics. In contrast, the more successful favour and hypervolume indicators have preferred mutational heuristics; this aligns with known results for multi-objective instances of these problems [14].

6 Conclusion

This paper has presented an analysis of the use of a selection hyper-heuristic to solve many-objective optimisation problems. This is, as far as we are aware, the first study of its type, and as such have evaluated the algorithm's performance on a small number of test problems with relatively small numbers of objectives; future work will expand this approach to a wider range of problems and consider many more objectives. The work presented three approaches to comparing many-objective solutions. Of the three, we consider the favour and hypervolume



(a) Dominance (b) Favour

(c) Hypervolume (d) Rank

Fig. 4: Transition probability matrices for 6-objective instances of DTLZ1 (top). Key to low-level heuristics (bottom-top) – grey: ruin and recreate (solution); red: ruin and recreate (parameter); green: uniform mutation; blue: Gaussian mutation; cyan: beta mutation; magenta: archive replacement (solution); yellow: archive replacement (parameter). A large block indicates a large probability of transitioning to that heuristic from the current heuristic.

indicators to be the most successful, though we note that these were less useful when optimising discontinuous Pareto fronts. A wider range of low-level heuristics, including crossover heuristics, would likely address this issue, though that would require conversion to a population-based approach.

As we move toward optimising problems comprising larger numbers of objectives, we expect the conditions experienced in this work to become more pronounced. The dominance relation will become less able to provide selection pressure, and the favour and hypervolume indicators will likely continue to optimise specific regions of the Pareto front well, at the expense of other regions. It is therefore unlikely that considering alternative comparison methods alone will allow us to successfully optimise many-objective problems, and as such we are currently investigating additional ways in which hyper-heuristics can be modified so that they can be used to optimise such problems.

7 Acknowledgements

This work was funded under EPSRC grant EP/K000519/1.

References

1. P. J. Bentley and J. P. Wakefield. Finding Acceptable Solutions in the Pareto-optimal Range Using Multiobjective Genetic Algorithms. In P. K. Chawdhry, R. Roy, and R. K. Pant, editors, *Soft Computing in Engineering Design and Manufacturing*, pages 231–240. Springer-Verlag, 1998.
2. E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and R. Qu. Hyper-heuristics: a survey of the state of the art. *Journal of the Operations Research Society*, 64(12):1695–1724, 2013.
3. D. Corne and J. Knowles. Techniques for Highly Multiobjective Optimisation: Some Nondominated Points are Better than Others. In *Genetic and Evolutionary Computation Conference*, pages 773–780, London, UK, 2007.
4. K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable Multi-Objective Optimization Test Problems. In *Proceedings of IEEE Congress on Evolutionary Computation*, volume 1, pages 825–830, May 2002.
5. N. Drechsler, R. Drechsler, and B. Becker. Multi-objective optimisation based on relation favour. *Lecture Notes in Computer Science*, 1993:154–168, 2001.
6. R. M. Everson, J. E. Fieldsend, and S. Singh. Full Elite Sets for Multi-Objective Optimisation. In I.C. Parmee, editor, *Proceedings of the Fifth International Conference on Adaptive Computing Design and Manufacture (ACDM 2002)*, volume 5, pages 343–354, University of Exeter, Devon, UK, April 2002. Springer-Verlag.
7. M. Farina and P. Amato. On the Optimal Solution Definition for Many-criteria Optimization Problems. In *2002 Annual Meeting of the North American Fuzzy Information Processing Society Proceedings*, pages 233–238, 2002.
8. M. Fleischer. The Measure of Pareto Optima. Applications to Multi-objective Metaheuristics. In *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003*, pages 519–533. Springer, 2003.
9. Giovani Guizzo, Gian Mauricio Fritsche, Silvia Regina Vergilio, and Aurora Trinidad Ramirez Pozo. A hyper-heuristic for the multi-objective integration and test order problem. In *Proceedings Genetic and Evolutionary Computation Conference*, GECCO 2015, pages 1343–1350, 2015.
10. H. Ishibuchi, N. Tsukamoto, and Y. Nojima. Evolutionary Many-objective Optimization: A short Review. In *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 2419–2426, June 2008.
11. Mashael Maashi, Graham Kendall, and Ender Ozcan. Choice function based hyper-heuristics for multi-objective optimization. *Applied Soft Computing*, 28:312 – 326, 2015.
12. Mashael Maashi, Ender Ozcan, and Graham Kendall. A multi-objective hyper-heuristic based on choice function. *Expert Systems with Applications*, 41(9):4475 – 4493, 2014.
13. K. McClymont and E. Keedwell. Markov chain hyper-heuristic (mchh): an online selective hyper-heuristic for multi-objective continuous problems. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2011)*, pages 2003–2010. ACM, 2011.
14. D. J. Walker and E. Keedwell. Multi-objective optimisation with a sequence-based selection hyper-heuristic. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2016)*, 2016.