On Constructing Ensembles for Combinatorial Optimisation

Emma Hart e.hart@napier.ac.uk
School of Computing, Edinburgh Napier University, Edinburgh, EH10, UK

Kevin Sim k.sim@napier.ac.uk
School of Computing, Edinburgh Napier University, Edinburgh, EH10, UK

doi:10.1162/EVCO_a_00203

Abstract

Although the use of ensemble methods in machine-learning is ubiquitous due to their proven ability to outperform their constituent algorithms, ensembles of optimisation algorithms have received relatively little attention. Existing approaches lag behind machine-learning in both theory and practice, with no principled design guidelines available. In this article, we address fundamental questions regarding ensemble composition in optimisation using the domain of bin-packing as an example. In particular, we investigate the trade-off between accuracy and diversity, and whether diversity metrics can be used as a proxy for constructing an ensemble, proposing a number of novel metrics for comparing algorithm diversity. We find that randomly composed ensembles can outperform ensembles of high-performing algorithms under certain conditions and that judicious choice of diversity metric is required to construct good ensembles. The method and findings can be generalised to any metaheuristic ensemble, and lead to better understanding of how to undertake principled ensemble design.

1 Introduction

In the field of machine-learning, *ensemble-methods* that combine decisions from multiple learning algorithms to make accurate predictions have been the focus of intense research for well over a decade (Valentini and Masulli, 2002), with a broad range of empirical results underpinned by a sound theoretical understanding. Ensemble methods have also found favour within the constraint satisfaction and satisfiability domains (Kotthoff, 2014), where they are commonly referred to as *portfolio* methods. In the latter case, portfolios tend to be composed from exact solvers, and are evaluated according to runtime metrics.

On the other hand, research in ensemble-methods using metaheuristic algorithms (in which solution quality rather than runtime is the driving factor) lags behind machine-learning and satisfiability research in both theory and practice. Some recent work exists in which algorithm portfolios are constructed from multiple, well-known population-based optimisation algorithms (Vrugt et al., 2009; Peng et al., 2010; Tang et al., 2014; Yuen and Zhang, 2015); other recent work uses genetic programming to evolve ensembles that contain novel algorithms (Hart and Sim, 2016; Park et al., 2015), which cooperate to produce solutions. However, given the detailed theory from machine-learning that offers insight into *how* to design good ensembles, current approaches have a number of weaknesses.

Manuscript received: May 6, 2016; revised: October 27, 2016; accepted: November 26, 2016.

The first concerns size. The most effective classification ensembles contain hundreds of classifiers (Breiman, 2001), an explicit design decision substantiated by theory. In contrast, the population-based method listed above typically contains ≤ 5 algorithms. Limiting the size of the pool in this manner places an upper bound on potential performance. Second, a large number of theoretical studies in machine learning recognise *diversity* as a key characteristic of a successful ensemble (Kuncheva and Whitaker, 2003): with regard to classification, negatively correlated classifiers result in reduced error compared uncorrelated classifiers, which in turn reduce error in comparison to correlated classifiers. Note that the cited studies refer to *functional diversity*; that is, classifiers must exhibit differences in behaviour, as opposed to being structurally diverse in terms of their design. Investigating agent-based problem solving methods, Hong and Page (2004) go even further in showing that diversity can trump ability. However, the most common approach in metaheuristic optimisation is to construct the ensemble from well-known algorithms, typically those that are high-performing on common benchmarks (Vrugt et al., 2009; Grobler et al., 2015; Yuen and Zhang, 2015). This choice almost certainly restricts diversity: most "good" algorithms are labelled as such as they tend to perform well across the majority of problems in a benchmark set. More recent research has seen some effort directed towards selecting good combinations of algorithms for the portfolio, based on notions of algorithm *complementarity* (Peng et al., 2010; Yuen and Zhang, 2015). However, even these approaches still restrict the pool from which algorithms are chosen to the same small sets of well-known algorithms previously used, which as we have pointed out, inevitably comprises diversity. Furthermore, the concept of diversity or complementarity in metaheuristic optimisation algorithms is ill-defined. Peng et al. (2010) and Yuen and Zhang (2015) both note that current notions of the concept are vague and that more precise measures and understanding of complementarity are necessary. In addition, even if precise definitions are available, in a machine-learning context, Tang et al. (2006) illustrate that from a practical perspective, selecting the *right* diversity measure is not easy.

In order to move the use of ensembles in optimisation forward, some basic questions need to be answered to gain the necessary insights required to construct useful ensembles. These include what algorithms to consider for inclusion, and how better to select an optimal mix for the ensemble. Specifcally, we consider these questions:

- Is there a trade-off between the diversity and accuracy of the component algorithms in an ensemble? (That is, do we have to sacrifice accuracy in order to increase diversity?)
- Under what conditions (if any) does an ensemble composed of functionally diverse algorithms outperform an ensemble of high-quality algorithms?
- Is diversity an appropriate proxy for constructing an ensemble: if so, what diversity measure is best?

We develop a general methodology to answer these questions, involving the creation of a very large pool of algorithms of wide-ranging ability that can be used to construct an ensemble, and a large pool of benchmarks to evaluate performance against. Bin-packing is selected as a representative domain in combinatorial optimisation, and 1370 instances are collated from the literature to use as a test-bed. To facilitate generation of a very large set of algorithms, we define an algorithm as a sequence of well-known packing heuristics, which is applied repeatedly until a solution is created. Using a

sequence of length 3 created from 9 potential packing heuristics, we generate $9^3 = 729$ algorithms. Note that the method is not restricted to either bin-packing or algorithms created as just described. The latter are used to simply facilitate an in-depth experimental study, and can be replaced by any optimisation algorithm, whether they operate on a single individual or are population-based, and whether they are stochastic or deterministic. The goal is not to conduct a comparison of ensembles to other methods but instead to gain fundamental insights into how to construct ensembles for use in optimisation.

The main contributions are as follows. First, we provide empirical evidence of the conditions under which a random group of algorithms will outperform a group containing algorithms that are individually strong. We propose new methods for defining the behavioural diversity of an algorithm and use these to illustrate the trade-off between diversity and ability. Finally, we examine a set of metrics for measuring ensemble diversity and use these to construct novel ensembles, showing that as previously observed in machine-learning, judicious selection of an appropriate diversity metric is crucial. The article considerably extends current understanding of ensembles in optimisation by considering ensembles containing up to 100 members, a set of 729 candidate algorithms, and defining and evaluating explicit measures of diversity. Its method can be extended to include any type of algorithm in which behavioural diversity can be measured against a reference set of functions.

2 Background

Existing portfolio approaches fall into two categories: those in which the portfolio is used to solve a single instance, and those in which a large set of instances are solved.

In the former category, Grobler et al. (2015) describe a portfolio approach that uses multiple predefined population-based algorithms during the search for a solution to an individual instance of a problem. A budget of evaluations is divided between the ensemble of algorithms, with a control algorithm determining the most appropriate algorithm to allocate to each member of the population at a given iteration. At the start, all algorithms in the ensemble are simultaneously applied to the population, but the number is varied over time depending on the control strategy used. Four algorithms were used: a genetic algorithm, particle swarm optimisation algorithm (GCPSO), self-adaptive differential evolution algorithm with neighbourhood search (SaNSDE), and the covariance matrix adapting evolutionary strategy algorithm (CMAES). No specific rationale is given for the number of algorithms chosen other than that they are well-known high-performing algorithms.

Also in the category of solving single instances, Vrugt and Robinson (2007) propose an algorithm called AMALGAM that uses a simultaneous multimethod search to give computationally efficient solutions to multiobjective optimisation problems. Four algorithms are included in the ensemble (NSGA-II, particle swarm optimisation (PSO) (19), adaptive metropolis search (AMS), and differential evolution (DE)). The authors state that these choices are based on the outcome of numerical experiments which demonstrates that the four common algorithms are both mutually consistent and complementary, although no detail is given as to how these terms are defined in practice. They later created a variant for single-objective optimisation, AMALGAM-SO (Vrugt et al., 2009). This framework contained both stochastic and deterministic algorithms, including Covariance Matrix Adaptation (CMAES), an evolutionary strategy, a Genetic Algorithm (GA), Particle Swarm Optimiser (PSO), and Differential Evolution (DE). The ensemble was tested on a large set of multi-and unimodal functions with promising results.

Recently, Yuen et al. (2016) proposed another novel multi-EA for solving a single instance in which multiple algorithms are executed independently with no exchange of information. Their algorithm predicts the fitness of each algorithm in the portfolio at some common future point by extrapolating the convergence curves of each algorithm. The winning algorithm is chosen and executed for one generation, before repeating the process of recalculating convergence curves for each algorithm. Similar to previous approaches, a portfolio of four algorithms is used, containing CMA-ES, a History driven Evolutionary Algorithm (HdEA), Particle Swarm Optimisation (PSO2011), and a Self-adaptive Differential Evolution (SaDE). The authors state that these algorithms are chosen because they represent current state-of-the-art methods, and that they have different strengths and weaknesses, though no further detail is given. Their approach is shown to perform better than the sum of its constituent parts in some instances, providing a synergistic effect. It also outperforms PAP and AMALAGAM-SO, as well as providing good results on a real-world problem.

In contrast to the above approaches that work in single instances, Peng et al. (2010) propose a portfolio approach which optimises performance over a set of problems. PAP (Population-based Algorithm Portfolio) uses parallel subpopulations, operated on by different algorithms from the portfolio, with migration between populations. The portfolio contains four similar algorithms to those described above, including self-adaptive differential evolution with neighbourhood search (SaNSDE), a particle swarm optimiser with inertia weight (wPSO), generalised generation gap (G3) model with generic parent-centric recombination (PCX) operator (G3PCX), and a covariance matrix adaptation evolution strategy (CMA-ES). They evaluate ensembles of size 2,3,4 that employ different combinations of algorithms. Analysis of results showed that some high-performing ensembles of size 2 contained algorithms that performed well on different problem instances, backing up their intuition that *"constituent algorithms should not only employ different operators, but also exhibit different behaviours on the problem set*." They also note that an ensemble of relatively "weak" algorithms can be stronger than a single state-of-the-art algorithm, for example CMA-ES. The authors also attempt to provide a theoretical analysis to clarify the term "complementary." Their treatment results in the insight that the performance of an algorithm A1 on a problem should be above its "average" performance over the problem set when the performance of algorithm A2 is below its "average" performance and vice versa. They note that selecting two appropriate algorithms in this way should be trivial, but it will become increasingly difficult to find new algorithms to add. The authors suggest this may place some upper bound on the size of the portfolio.

Tang et al. (2014) extend PAP in a new algorithm EPM-PAP, that has an explicit algorithm selection module for constructing the portfolio. An *estimated performance matrix* (EPM) is constructed for each potential candidate, obtained by running algorithm j on each of the n problems r times. The selection problem is then formulated as a minimisation problem in which the aim is to select a portfolio of EAs that when used in a PAP instantiation are least likely be outperformed by any of the other candidate EAs available for PAP, using the EPMs to calculate probability. The authors discuss the issue of selecting *complementary* algorithms, acknowledging that the use of the term is vague and that a more precise method of describing an algorithm's behaviour is required if we are to gain a clearer understanding of the concept of complementarity.

Addressing the issue posed by Peng et al. (2010) that it becomes increasingly hard to find new algorithms that complement others in the portfolio, Park et al. (2015) propose a portfolio approach for solving job-shop scheduling problems in which genetic

programming (GP) is used to evolve novel heuristics. A fixed-size ensemble of heuristics is evolved that *collaborate* to select the next operation to schedule. This contrasts to the above approaches in which portfolio algorithms tend to be executed independently. Their approach provides reusable heuristics that are subsequently shown to perform well on unseen test instances. This model bears close resemblance to many machine-learning methods in which an ensemble of classifiers vote to determine the class of an instance (Breiman, 2001).

Another approach that uses genetic programming to create ensembles is first described in Sim et al. (2015) in an application to bin-packing, and extended to apply to job-shop scheduling in Hart and Sim (2016). As with Park et al. (2015) their ensembles are applied to large sets of instances. Their method, dubbed NELLI, grows an ensemble by using GP to evolve new constructive heuristics. A new heuristic is added to the ensemble only if it provides a better result on at least one instance than any other heuristic currently in ensemble. Each heuristic constructs an entire solution, with a greedy selection process applied to choose the most appropriate for an instance. Their results show that the ensembles are self-limiting in size. One possible explanation for this is that the same difficulty noted by Peng et al. (2010) arises, in that it becomes increasingly hard to evolve a new heuristic that does better on any instance as the ensemble grows. Alternatively, the terminal set used in the GP may limit the number of behaviourally distinct heuristics that can arise.

Finally, another ensemble approach in which the ensemble is explicitly grown is that of Yuen and Zhang (2015). They make use of a ranking method in which the algorithm with the lowest average rank on a suite of benchmark functions is added to the ensemble each cycle, where rank is calculated based on the lowest average number of evaluations. Given q algorithms, the average rankings of each are calculated; then the q by q covariance matrix of rank is determined. The process determines the next algorithm n to be added to the current portfolio P, by choosing the algorithm that minimises the covariance(P, n). The approach is tested on 25 functions from the CEC competition (Liang et al., 2013). Five algorithms were available for portfolio selection: CMA-ES, composite differential evolution CoDE, self-adaptive differential evolution SaDE, PSO, and an artificial bee colony algorithm. The rationale behind this choice is that these algorithms would be the authors' algorithms of choice if they had to recommend them to outsiders.

3 Preliminaries

We use the 1-d bin-packing domain to conduct our investigation. A brief background to the domain and problem instances used are given below.

3.1 BinPacking

The Bin Packing Problem (BPP) is defined as follows (Garey and Johnson, 1979): given a finite set O of numbers (the object sizes), C (the bins' capacity), find a packing of objects into bins that minimises the number of bins used N. A trivial fitness function can be defined simply as N. However, this leads to a search landscape that lacks capacity to guide a search algorithm given that a very small number of optimal points in the space are lost in an exponential number of points where the fitness is just one unit above the optimum. In addition, the function does not discriminate between solutions in which the extra bin is packed to different extents. Hence, Falkenauer (1996) proposed a new cost function given in Equation 1: this yields a value between 0 and 1, in which higher values indicate better solutions. If all bins are filled exactly to capacity, the fitness is

Table 1: Benchmark bin-packing problems.

Data Set	capacity (c)	n	ω		#Problems
ds_1	100,120,150	50,100,200,500	[1,100],[20,100],[30,100]		$36 \times 20 = 720$
ds_3	100000	200	[20000,30000]		10
$FalU$	150	120,250,500,1000	[20,100]		$4 \times 20 = 80$
$FalT$	1	60,120,249,501	[0.25,0.5]		$4 \times 20 = 80$
Data Set	c	n	ϖ (avg weight)	δ(%)	# Problems
ds_2	1000	50,100,200,500	$\frac{c}{3}, \frac{c}{5}, \frac{c}{7}, \frac{c}{9}$	20,50,90	$48 \times 10 = 480$

exactly 1; if all bins *except* the final bin are filled to capacity, then the solution is optimal but has a value less than 1.

$$f_i = \frac{\sum_{b=1}^{i=N} (F_b/C)^k}{N}, \tag{1}$$

where $N$is the number of bins actually used in the solution, F_b is the sum of sizes of the objects placed in bin n, C is the bin capacity, and k is a constant, $k > 1$. Here, we set $k = 2$ in all experiments.

Many benchmark instances are available in the literature. Here, we use an aggregated set of 1370 instances as defined in Table 1 and previously described in Sim et al. (2015).

3.2 Bin-Packing Algorithms

Previous work in the hyper-heuristic domain has shown that novel bin-packing algorithms can be described by sequencing existing deterministic packing rules (Hart and Ross, 1998; Sim and Hart, 2014). Each rule is a *constructive* method that packs one or more items into a bin; superior performance is obtained by finding useful sequences when compared to packing all items with a single rule. The evolved sequence can be considered as a novel algorithm. In cases where the sequences contain deterministic rules, then it is common in the hyper-heuristic literature to refer to the sequence as a novel *heuristic*: we adopt this simpler terminology from now on in preference to the term algorithm, without loss of generality.

Given k potential rules, and a sequence of length l, then $n = k^l$ possible sequences can be specified. To solve an instance with a heuristic $(k_1, k_2, \ldots, k_n)$ then rule k_1 is first applied, followed by k_2, and so on. After applying rule k_n, if there are still remaining items, then rule k_1 is reapplied, continuing in this manner until all items are packed.

We select $l = 9$ deterministic bin-packing rules to form heuristics. The 9 rules chosen were previously used in Sim and Hart (2014); note, however, that any deterministic rules could have been chosen without loss of generality. Rules 1 to 5 are of the type best-n and work as follows. For $n = 1$ to 5, best-n packs the best remaining n items into the available space, that is, the items that together best fill the space. Best-1 packs the best single item, and is equivalent to the well-known first-fit descending rule. Best-2 packs the best remaining 2 items into the available space, and so on. If there is no combination of n items that fit a bin, then the rule fails and a new bin is opened (no items are packed). The remaining 4 rules are of type best-of-n. These rules attempt to fill the bin as much as possible. For example, best-of-2 searches considers both single items *and* combinations of two items to best fill the bin. If there is a tie, that is, there are 2 items totalling size 5 or

1 item of size 5, then the single item is chosen, leaving 2 smaller items which are easier to pack. If a rule fails to place an item into an *empty* bin (e.g., best-3 cannot find any 3 items to fit the empty bin) then the first-fit descending rule is applied. In the case of ties, all algorithms give preference to combinations that include larger items, for example, two items of size (10,1) are preferred over two items of size (9,2).

We consider sequences of length 3, therefore generating $9^3 = 729$ possible heuristics. The length of 3 was selected as a reasonable compromise between generating a set of heuristics that was large enough to be diverse while being at the upper end of the number of heuristics that might be available in practice in a given problem domain.

4 Ensembles and Diversity Measures

In common understanding, differences in how people represent problems and how they go about solving them is known as *functional* diversity (Hong and Page, 2004). Similarly, in the field of Evolutionary Robotics, the term *behavioural* diversity has been used to describe differences in observed behaviours of robots with different controllers (Mouret and Doncieux, 2012). The authors note that on the one hand, several phenotypes can give rise to the same behaviours and on the other, very close phenotypes result in different behaviours. Thus they show that it is preferable to maximise the novelty of *behaviours* rather than a fitness-based objective function. Mouret and Doncieux (2012) suggest several approaches to defining a behaviour vector for a robot, with corresponding metrics for measuring diversity of behaviour.

In exactly the same way, we propose that behaviour is a more appropriate method of comparing algorithms: given a large set of problem instances, then algorithm similarity (or dis-similarity) can be better measured by a vector describing their behaviour across the set of problems rather than by similarity in terms of algorithm definition (e.g., operators, population size) or aggregated fitness metrics such as average solution quality.

4.1 Behavioural Vectors

Given a heuristic h, then assume the heuristic has a fitness $f_h(i)$ on each instance i of interest. For each instance, let $f^*(i)$ be its *best known fitness* (explained below). For any heuristic, we define two different behavioural vectors:

- The fitness vector $\mathcal{F}_a$, in which each component of the vector is the absolute fitness $f_h(i)$ of the heuristic on each instance i
- The binary vector $\mathcal{B}$ in which each component i has a value 1 if heuristic h achieves the best result on instance i; that is, $f_h(i) = f^*(i)$, and 0 otherwise

The fitness vector can be immediately written for any heuristic based on its performance over a set of instances. The second vector relies on a comparison fitness $f^*(i)$ for each instance, the *best-known fitness*. This can be obtained by considering an *oracle*, that is, the best-known result for the instance from any algorithm, or best-known lower bound. This is a common approach in machine-learning (Tang et al., 2006) and is used later in this article in Section 7 when we consider constructing ensembles based on diversity metrics. An alternative is to consider a specific subset of heuristics of interest, and set $f^*(i)$ equal to the best result obtained by the subset. This approach is useful when information regarding optimal solutions or performance of other approaches cannot be obtained. The subset approach is used in Section 6.

Euclidean distance is used to measure distance between any two fitness vectors. For binary vectors $\mathcal{B}$, two distance metrics are considered. *Ochiai's* distance is commonly used in ecological and biological studies (Champely and Chessel, 2002) when comparing diversity between species, in which presence/absence of a species at a site is represented in a binary matrix, with one row for each species. This is defined in Equation 2 (Legendre and Legendre, 2012), and is the geometric mean of the ratios of the number of shared species to the total number of species in each site. For two heuristics (i, j) with vectors $\mathcal{B}_i, \mathcal{B}_i$ as described above, then a is the number of heuristics which both win an instance i, that is, (1,1) pairs; b is the number of instances in which only $\mathcal{B}_1$ wins, and c the the number of instances in which only $\mathcal{B}_2$ wins, that is, (0,1) pairs. The equation given represents *dissimilarity* (and is therefore equivalent to distance).

$$d_{ij} = \sqrt{1 - s_{ij}} \text{ where } s_{ij} = \frac{a}{\sqrt{(a+b)(a+c)}}. \tag{2}$$

Note that d= (0,0) pairs denoting that neither heuristic wins an instance is not counted by this metric.

An alternative metric suitable for binary data is the *disagreement measure*, proposed by Skalak (1996) in order to evaluate the diversity between two base classifiers, and later used by Ho (1998) to construct decision-tree forests. The measure is defined based on the intuition that two diverse classifiers perform differently on the same training data (Tang et al., 2006) and is given in Equation (3)

$$d_{ij} = \frac{b + c}{(a + b + c + d)}. \tag{3}$$

4.2 Ensembles

As should be clear from Section 2, ensembles can be applied in a number of forms. Simultaneous execution of each member is common in population-based methods (e.g., Peng et al. 2010). Another form of simultaneous application is the majority-voting method of Park et al. (2015). In contrast, Sim et al. (2015) select a single member from the ensemble to solve each problem instance in a large set, while Yuen et al. (2016) select a different member at each iteration to solve a single instance. Finally, in an agent-based problem-solving task, Hong and Page (2004) adopt a sequential approach in which each agent in the ensemble sequentially executes its strategy, although they note their findings did not depend on whether simultaneous or sequential selection was used.

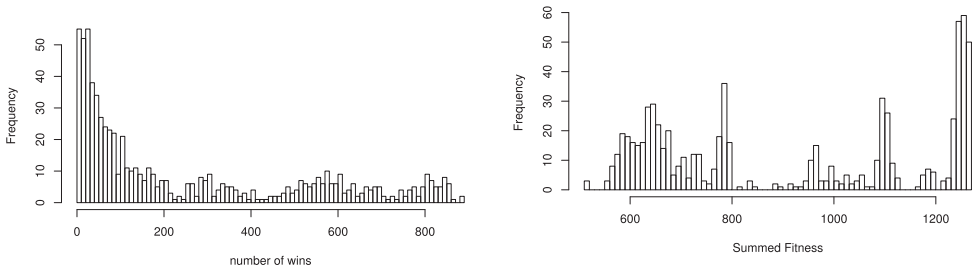
We adopt the selective approach previously shown to be effective in Sim et al. (2015) and subsequent publications, in which each member of the ensemble solves a subset of the instances in a large class. A formal definition of the ensemble is provided below, alongside definitions of heuristic and ensemble fitness which are used in the remainder of the article.

4.2.1 Definition and Metrics

An ensemble E contains e heuristics, and is typically applied to a collection of p problem instances. Each heuristic in the ensemble has an individual fitness f_h defined by Equation 4

$$f_h = \sum_{i=1}^{i=p} f_{h,i}, \tag{4}$$

where $f_{h,i}$ is the fitness of heuristic h on instance i as specified by the objective function for the problem, and in this case given by Equation (1).



(a) Frequency of wins over the 1370 instances

(b) Frequency of summed fitness over 1370 instances

Figure 1: Contrasting behavioural diversity in terms of number of wins (a) and summed fitness (b) across the set of 729 heuristics.

Given an instance i and an ensemble E, let V_i be a set of size $|E|$ containing the fitness of each heuristic in the ensemble on the instance. The ensemble returns a fitness $f_i^* = max(V_i)$, that is, a greedy selection method assigns the fitness of an instance as the best fitness obtained from applying each heuristic in the ensemble. The collective fitness of the ensemble f_E over the complete dataset is given by Equation (5)

$$f_E = \sum_{i=1}^{i=p} f_i^*, \ \textit{where} \ \ f_i^* = max\{f_{h,i} : h \in E\}. \tag{5}$$

Average fitness of ensemble: If the fitness of an individual heuristic h averaged across an entire dataset P containing p instances is $\overline{f_{h,P}}$, then for a given ensemble of E, the average fitness of the heuristics in the ensemble is:

$$\overline{F_E} = \frac{1}{e} \sum_{i=1}^{i=e} \overline{f_{h_i,P}}. \tag{6}$$

5 Heuristic Analysis

Before conducting an in-depth study of ensembles, it is useful to understand the nature of the 729 heuristics that will be used to define each ensemble. We consider the individual performance of each heuristic across the whole instance set of 1370 instances.

A heuristic can be said to *win* an instance i if its fitness $f_h(i)$ is equal to the best fitness obtained by any of the 729 heuristics on that instance. Figure 1 shows a histogram giving the frequency of "total wins" across each of the 729 heuristics. Two heuristics do not win any instances, and therefore can never contribute to an ensemble, given the definition in Section 4.2. At the other end of the spectrum, one heuristic wins 883 of the 1370 instances. The median number of wins is 115, that is, approximately 8% of the total. The figure broadly indicates that many heuristics win a small number of instances, while few heuristics win a large number. The distribution has a long tail, denoting a large spread in performance. In contrast, Figure 1b indicates frequency in terms of summed fitness, that is, according to Equation (4). From this, we observe that a large number of heuristics perform well (although not optimally, as seen in Figure 1a), achieving a high summed fitness. Heuristics appear to fall into 4 or 5 distinct groups, centered around the peaks, in which similar summed fitness is observed for members of each group.

The figures provide sufficient assurance that at a qualitative level, the heuristics themselves are diverse. Therefore, we proceed to evaluate the extent to which combining them into ensembles provides advantage.

6 Diversity vs Quality in Ensemble Composition

Given the set of heuristics defined above, we conduct empirical experiments to answer the question *"Do groups of functionally diverse heuristics outperform groups of high-ability heuristics?"* with respect to the bin-packing optimisation problem defined. The extent to which any performance discrepancy is related to heuristic diversity and the trade-off between individual ability and the contribution a heuristic makes to the collective performance of an ensemble is then analysed. Given a pool H containing the 729 heuristics already described, two types of ensemble are created:

- The **random** ensemble $\mathcal{E}_r$ contains e heuristics randomly selected from H (uniformly, with replacement)
- The **elite** ensemble $\mathcal{E}_e$ contains the best e heuristics from H, assuming heuristics are ranked according to individual heuristic performance as given by Equation (4)

For both types of ensemble, we create ensembles in which the number of heuristics e varies from 10–729 in steps of 5. For the *random* ensembles, 50 random ensembles are selected. This is not necessary for the elite ensembles as the selected process is deterministic.

6.1 Comparison of Random vs Elite Ensembles

The performance of each ensemble is evaluated on the full set of 1370 instances according to Equation (5). Figure 2 compares the performance of the randomly composed ensembles $\mathcal{E}_r$ and the elite ensembles $\mathcal{E}_e$. The figure shows the median fitness value obtained for the randomly composed ensembles from the 50 randomly selected ensembles. The figure shows only ensembles from size 10–100 at values of $e \geq 100$, the results from the two approaches converge.

It is clear that the randomly composed ensembles outperform those composed of the best-ranked heuristics for $n < 95$. A Mann–Whitney Wilcoxon Test is used to test for significance at the 95% confidence level. No significant differences are observed between the ensembles $\mathcal{E}_r$ and $\mathcal{E}_e$ at $e = 10$, and when $e \geq 95$. At all other values of e, significant differences are observed with $p \ll 0.05$ in each case.

The randomly composed ensembles show a smooth increase in fitness as the size of ensemble is increased. In contrast, the ensembles composed from the best-ranked heuristics show a stepped pattern; at specific points, adding one or more lower-ranked heuristics to the ensemble causes a sudden jump in fitness. Figure 2 suggests that the heuristics fall into distinct "behaviour" groups. Within each group, heuristics have similar average fitness, due to working well on the *same* instances. A lower-ranked heuristic in a different group might perform poorly on those instances on which the higher-ranked heuristics do well, but better than those heuristics on other instances. Thus, adding a new heuristic from a different group into the ensemble results in a sudden large increase in fitness. This is shown in Figure 3 in which the same fitness graph as in Figure 2 is superimposed on a graph showing the fitness of each individual heuristic sorted by rank (where rank 1 is best). The green lines show that the jumps in

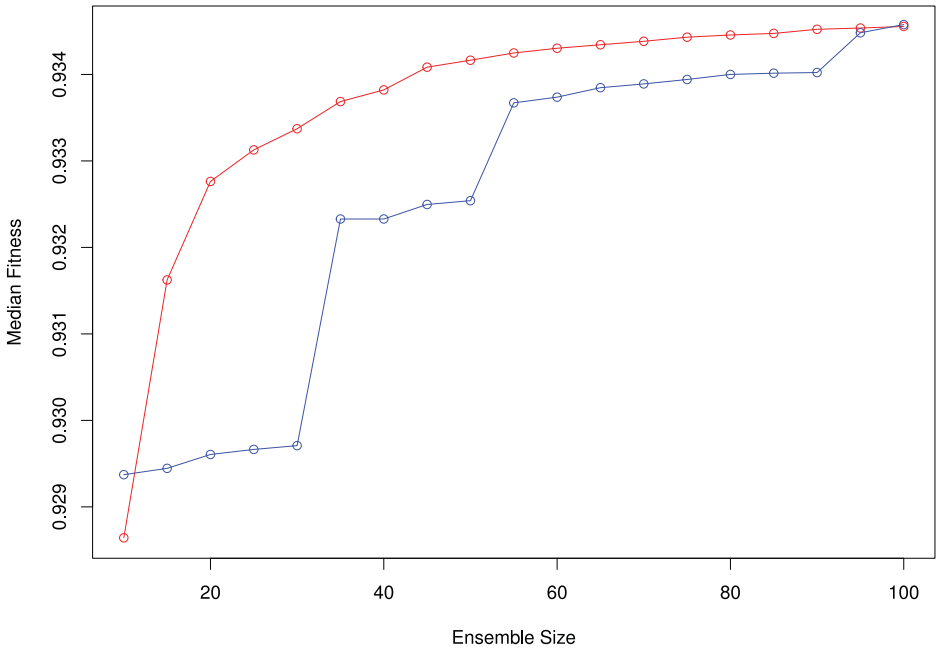


Figure 2: Median ensemble fitness for ensembles of size E. The figure shows ensembles constructed randomly (red) and the elite ensembles constructing using the best E heuristics (blue).

ensemble fitness roughly correspond to a sudden drop in individual fitness between heuristics that are adjacent in the ranking. Finally, Figure 4 shows box plots of fitness obtained from the randomly composed ensembles. At very low e, for example, an ensemble of size 5, unsurprisingly we see a wide variation in performance across the 50 experiments suggesting that a minimum size of ensemble is required before the benefits of random selection outweigh the risks. Conversely, for $e \geq 35$, there is no significant variation in performance across the 50 experiments, suggesting a single random ensemble can be selected with confidence.

6.2 Trade-Off Average vs Individual Quality

Next, we consider the question, *"Is it necessary to sacrifice quality of an individual heuristic in order to favour ensemble performance?"* Heuristics with very high average fitness (i.e., close to the maximum value) must perform well on the majority instances. Therefore, two heuristics with similar high average fitness are likely to have similar performance on any given instance. Thus, the two heuristics would have low behavioural diversity and are unlikely to complement each other in an ensemble in which collective fitness is defined in a greedy manner. In contrast, low average performance can be obtained if a heuristic performs well on a few instances, but poorly on most. If we pick two poor heuristics at random, there is reasonable chance that the instances on which each performs well will be different.

Figure 5 plots average individual fitness against average ensemble fitness for the elite ensembles (blue) and random ensembles (black) from size 10–100. Note the graph is plotted on two different x-scales due to the much lower average fitness values obtained

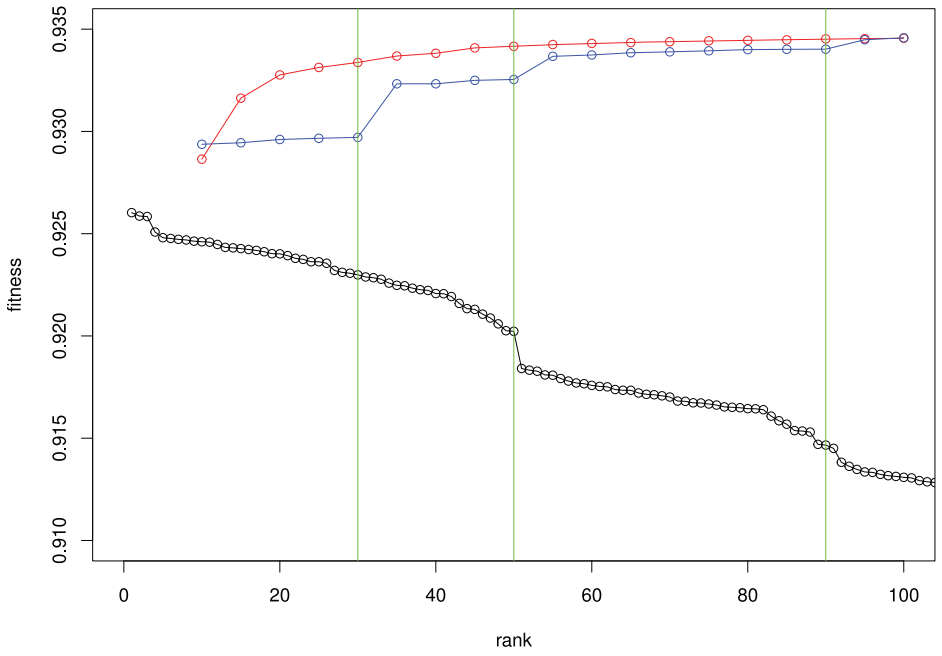


Figure 3: The figure shows the average fitness of each heuristic calculated from Equation 4 plotted against its rank, where rank 1 corresponds to the best heuristic. This is overlaid with the data from Figure 2. Note that an elite ensemble of size e contains the e best-ranked heuristics.

from the random ensembles. A clear pattern is observed from the elite ensembles in which increasing average fitness leads to lower ensemble fitness, illustrating the trade-off between individual quality and ensemble quality and reinforcing the message that functionally diverse heuristics outperform groups of high-ability heuristics. Although it is clear that the individual heuristics in high-performing randomly selected ensembles have lower fitness than individuals in corresponding elite ensembles, the overall pattern is less clear. This is due to the stochasticity associated with random selection of ensembles and the variation in individual heuristic quality. However, a cluster of points is observed in the top-left quadrant corresponding to low-individual but high-ensemble fitness.

6.3 Ensemble Diversity

As mentioned previously, Hong and Page (2004) noted that in agent-based computational experiments examining functional diversity, groups composed of high-ability agents have strikingly lower diversity than randomly selected groups of agents. We investigate whether the same result holds for the ensembles of bin-packing heuristics.

The two behavioural vectors described in Section 4.1 are calculated for each heuristic in each ensemble, for both random and elite ensembles. The diversity of each ensemble was calculated as the median pair-wise Euclidean distance in the case of the fitness vectors F_a and as the median pairwise Disagree distance between the binary vectors, as defined in Equation (3). The results, shown in Figures 6a and 6b, are plotted as the ratio of diversity (random ensemble):diversity (elite ensemble), therefore showing the

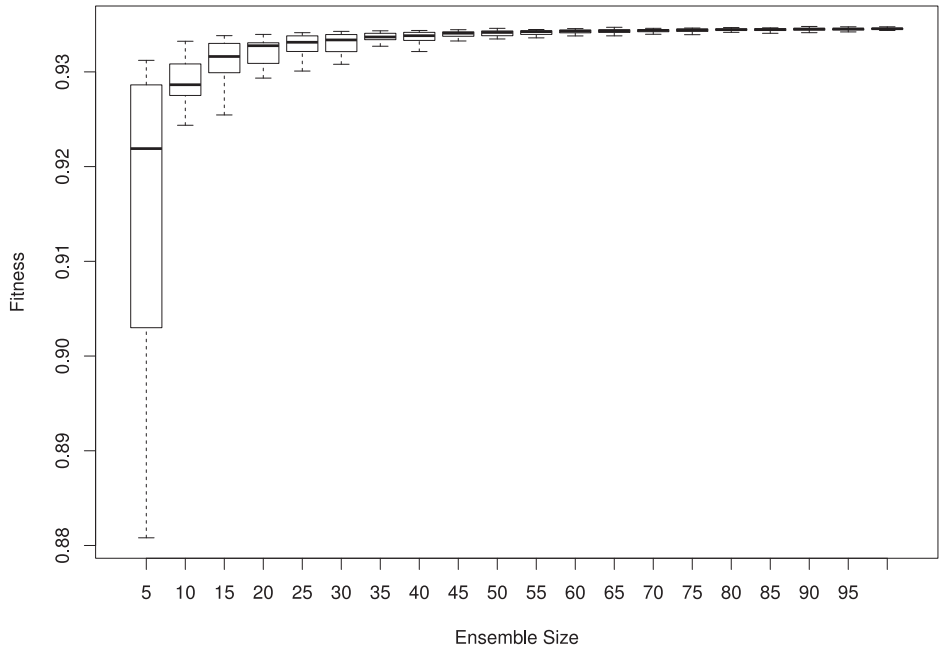


Figure 4: Box plots indicating the ensemble fitness from the random ensembles of varying size. Each box refers to 50 experiments.

relative increase in diversity gained by using a random ensemble. (Note that the scales on the two graphs cannot be directly compared but that the ratio should converge to 1 as the size approaches the maximum of 729. Values greater than 1 indicate the random ensemble is more diverse.)

Using the Euclidean metric, it is clear that the random ensembles are more diverse. As expected, diversity decreases as the size of the ensemble increases, with small random ensembles significantly more diverse than elite ones of the same size. The ratio obtained from Disagree metric (which is a coarser metric) converges to approximately 1 at an ensemble size of 100. Although the general pattern is decreasing diversity with ensemble size, an anomalous region is observed when the size of the ensemble $e = 35$. Closer analysis shows that for random ensembles, the trend is decreasing diversity with increasing size. However, for the elite ensembles, although diversity decreases from 10 to approximately 30, the first 35 ranked heuristics have very similar profiles. However, the next group of heuristics have a very different profile than the earlier ones: this is seen clearly in Figure 3 with a jump in fitness, and causes an increases in diversity as shown in Figure 6b. As the ensemble increases in size, the expected decrease in diversity counters any increase from adding heuristics with different profiles, producing a gradual overall decrease.

7 Ensemble Construction

In the previous section, we have shown that a randomly constructed ensemble can deliver better performance that an ensemble consisting of individually strong optimisers. Results given in the previous section attribute this difference to the increased *diversity* found in random ensembles, in line with the results from Hong and Page (2004).

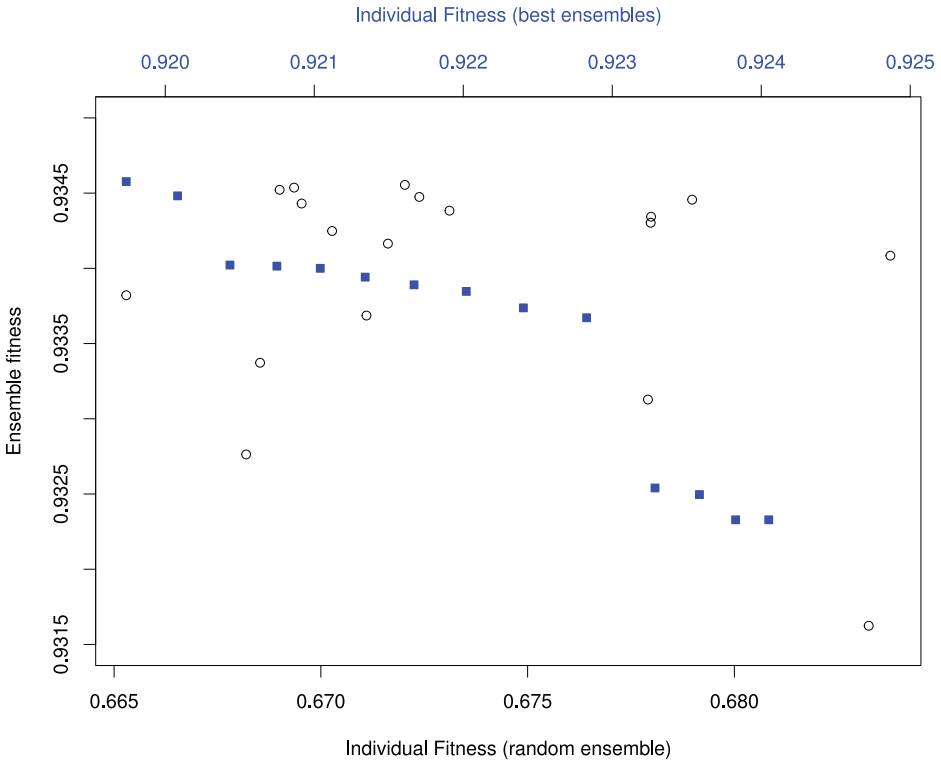
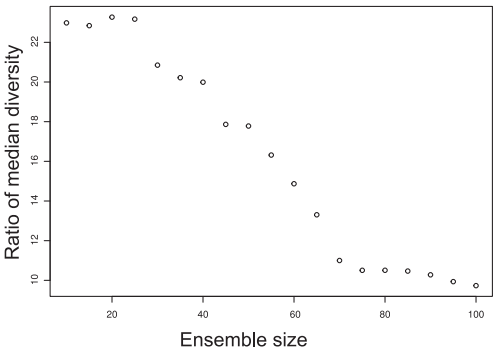
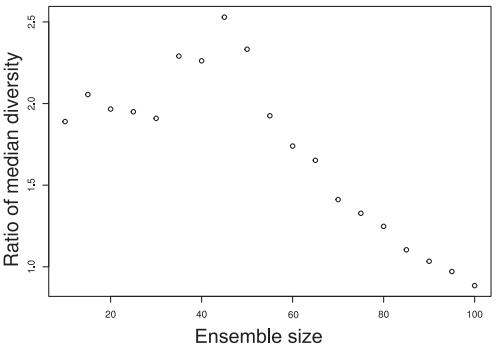


Figure 5: Average Individual fitness vs. Ensemble fitness. Note the graph is plotted on two different x-scales. The blue square (top scale) refers to values obtained from the ensembles composed of the best-ranked heuristics. The bottom scale and black circles refer to values obtained from the randomly composed ensembles.



(a) Diversity (Euclidean Distance)



(b) Diversity (Disagree Metric)

Figure 6: Ratio of diversity of random ensembles: diversity of elite ensemble for two different diversity metrics.

In this section, we pursue this more deeply and ask the question, *"Does explicitly seeking diversity while constructing an ensemble result in consistently good performance?"* and the related question, *"What is a good diversity measure for designing an ensemble learning algorithm?"*

Algorithm 1 Construct ensemble based on diversity

$\mathcal{H} \leftarrow getAllAvailableHeuristics()$
$d_{train} \leftarrow$ training set # randomly selected from P
$d_{test} \leftarrow$ test set
$M \leftarrow$ diversity metric
$h^* \leftarrow$ heuristic with maximum f_h on d_{train}
$s \leftarrow 1$
$\mathcal{E} \leftarrow \{h^*\}$
while s $< ensembleSize$ **do**
  for j in all unselected heuristics **do** # greedy selection of heuristic
    $\mathcal{E}_j \leftarrow \mathcal{E} \cup h_j$
    $d(\mathcal{E}_j) \leftarrow calculateAverageDiversity(M, \mathcal{E}_j)$
  end for
  $h^* \leftarrow$ heuristic with maximum $d(\mathcal{E}_j)$
  $\mathcal{E} \leftarrow \mathcal{E} \cup h^*$ # add new heuristic to ensemble
  $s \leftarrow s + 1$
end while
$f_{\mathcal{E}} = calculateFitness(\mathcal{E}, d_{test})$

In order to answer this empirically, we use the process defined in Algorithm 1. In essence, an ensemble of size 1 is first created using the heuristic with best individual fitness (Equation (4)). New heuristics are then added one at a time until the required ensemble size is reached by greedily selecting the heuristic that will maximise the average diversity of the ensemble according to a diversity metric M, based on a set of *training* instances. Following construction, the ensemble is applied to an unseen *test* set and its performance recorded.

The diversity metric is calculated relative to a behavioural vector, as previously discussed. In addition to the binary vector $\mathcal{B}$ and the absolute fitness vector $\mathcal{F}_a$ described in Section 4.1, we consider two further candidate vectors, the *relative* fitness vector $\mathcal{F}_r$ and the *ranked* vector $\mathcal{R}$, described below.

7.1 Diversity Metrics

The behaviour vectors $\mathcal{B}, \mathcal{F}_a$, and $\mathcal{R}$ are calculated relative to an *oracle* matrix that denotes the best-known solution for each instance in P. This is a common approach in machine-learning, where many authors discuss ensemble diversity in terms of oracle outputs (e.g., Kuncheva (2003), Tang et al. (2006)), and is also used in an optimisation scenario by Yuen and Zhang (2015). Any method can be used to generate the oracle, for example, using known optima or lower bounds, an exact algorithm or any specialised metaheuristic. Here, we run each of the 729 heuristics on each of the instances in the suite, and set the oracle value $\mathcal{O}_i$ of an instance to the best value found. In addition, for each instance, we rank the performance of each of the 729 heuristics on each instance (such that rank 1 is best), and assign $r_{h,i}$ as the rank of heuristic h on instance i.

Thus for each training set t containing a subset of instances, 4 *behaviour vectors* containing t components are created:

- The binary vector $\mathcal{B}$ in which each component i is 1 if the heuristic achieves the $oracle$ result on instance i, and 0 otherwise

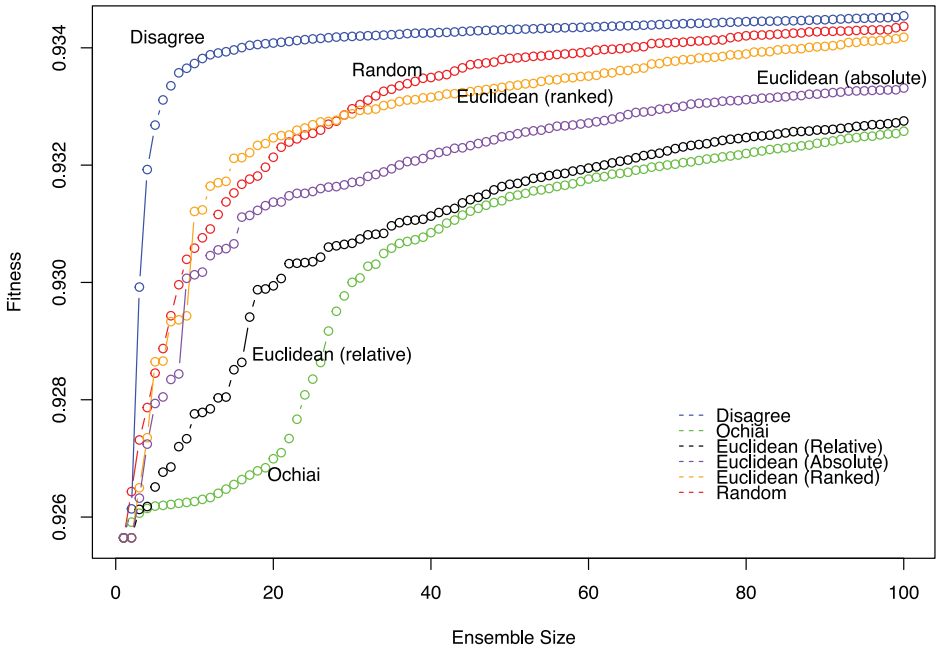


Figure 7: Average fitness of constructed ensembles per diversity metric compared to randomly selected ensemble (training set contains 685 instances, i.e., 50% of full set).

- The ranked vector $\mathcal{R}$ in which each component i is the *oracle* ranking of the heuristic $r_{h,i}$ on each instance i
- The fitness vector $\mathcal{F}_a$ containing the absolute fitness of the heuristic on each instance i
- The fitness vector $\mathcal{F}_r$ containing the relative fitness of the heuristic on each instance i, that is, the absolute fitness divided by the *oracle* value for i

For the binary vectors $\mathcal{B}$, diversity between any pair is calculated using the disagree metric 3 and the Ochiai distance measure 2. For the remaining vectors $\mathcal{R}$, $\mathcal{F}_a$, and $\mathcal{F}_r$, diversity is calculated as the Euclidean distance between any two vectors.

7.2 Results

Three sizes of training set are considered, containing 50%, 60%, and 70% of the full dataset of 1370 instances. Training sets are selected at random from the complete set of problem instances P. A test set is formed from the remaining data in each case. For each size of training set, the four behavioural vectors are created and an ensemble constructed using the appropriate diversity metrics in line with Algorithm 1. Each experiment is repeated 50 times with a random training set selected each time, and the performance on the unseen test sets recorded.

Figure 7 shows the mean fitness on the test set over the 50 experiments from ensembles of size 2–100. Results are shown from ensembles constructed using 5 different diversity metrics, and from ensembles constructed randomly. Training sets containing 50% of the full dataset were used to generate this figure. Table 2 (a, b, c) provides further

Table 2: Average fitness on test set of ensembles constructed using 5 different diversity metrics and at random. Experiments are averaged over 50 test sets, and values shown for ensembles of size 10, 20, and 30. Bold text used to show results that are significant at the 5% level compared to the *random* construction method.

(a) Ensemble size = 10

Training Set	Ochiai	Disagree	Euclidean ($\mathcal{F}_r$)	Euclidean ($\mathcal{F}_a$)	Euclidean ($\mathcal{R}$)	Random
50%	0.9263	**0.9337**	0.9278	0.9301	0.9312	0.9306
60%	0.9263	**0.9337**	0.9278	0.9303	0.9313	0.9304
70%	0.9262	**0.9339**	0.9279	0.9303	0.9314	0.931

(b) Ensemble size = 20

Training Set	Ochiai	Disagree	Euclidean ($\mathcal{F}_r$)	Euclidean ($\mathcal{F}_a$)	Euclidean ($\mathcal{R}$)	Random
50%	0.927	**0.9341**	0.9299	0.9314	0.9325	0.9321
60%	0.9269	**0.9341**	0.9299	0.9314	0.9325	0.9324
70%	0.9271	**0.9342**	0.9297	0.9315	0.9326	0.9327

(c) Ensemble size = 30

Training Set	Ochiai	Disagree	Euclidean ($\mathcal{F}_r$)	Euclidean ($\mathcal{F}_a$)	Euclidean ($\mathcal{R}$)	Random
50%	0.93	**0.9342**	0.9307	0.9317	0.9329	0.933
60%	0.9301	**0.9342**	0.9308	0.9317	0.9329	0.9332
70%	0.9305	0.9344	0.9308	0.9318	0.933	0.9335

detail, showing the mean fitness obtained on the test set for ensembles of size 10, 20, and 30. For each size of ensemble, the ensemble is constructed repeatedly using each diversity metric. The results shown give the average performance on the corresponding test sets (averaged over 50 repeated runs) in each case.

Figure 7 shows that the *Disagree* metric based on differences between the binary vectors $\mathcal{B}$ appears to construct the most effective ensembles for all size of ensemble e. Using this metric, small but effective ensembles can be constructed: for values $17 \leq e \leq 99$, applying a Wilcoxon test shows that there is no significant difference (95% confidence level) between the result obtained from constructed ensembles of size $e \in$ (17–99) and the constructed ensemble of size 100. As the ensemble increases in size, results from the disagree-constructed ensembles converge with the random ensembles: the disagree-constructed ensembles provide significantly different results from the randomly constructed ensembles for ensembles of size 3–42. Interestingly, none of the other metrics are able to compete with the random construction method. For low values of e (≤ 11), the metric based on Euclidean distance between the $\mathcal{F}_r$ vectors produces broadly similar performance to random, after which its performance deteriorates. The ensembles constructed using the diversity metric based on Euclidean distance between the Ranked vectors $\mathcal{R}$ do not show significantly different performance to the randomly constructed ensembles at any value of e. Finally, the *Ochiai* metric, also based on binary vectors $\mathcal{B}$, performs the most poorly.

The data in Table 2 (a–c) provides a more detailed comparison of results for three different ensembles of size 10, 20, and 30. Results are highlighted where a Wilcoxon test (95% confidence level) applied to an ensemble constructed using a diversity metric shows a significant difference from the randomly selected ensembles (for cases where the constructed ensemble outperforms the random). As the size of the training set increases, the benefit of using the *Disagree* construction metric for larger size of ensemble decreases. As mentioned above, using a training set containing 50% of the data, the disagree-constructed ensembles provide significantly different results from the randomly constructed ensembles for ensembles of size 3–42. Using training sets containing 60% of the data, significantly different performance is obtained for ensembles of size 3–35 and for ensembles of size 3–24 using training sets containing 70% of the data.

8 Discussion

We summarise our findings with respect to the research questions identified in the introductory section below.

Diversity trumps individual ability. Hong and Page (2004) used a general framework for modelling functionally diverse problem-solving agents, and studied the tension between the individual abilities of members of a group and its functional diversity. Their investigation led them to identify conditions under which the gain in individual abilities within the group is more than offset by the functional diversity of the group, leading to the claim that "diversity trumps ability."[1] Experience from machine-learning also demonstrates this, going back to work from Schapire (1990) who showed that a set of weak learners could be combined into a strong learner. Note it is important to recognise that the adjective *weak* means that accuracy on the training set is only slightly better than random guessing, just over 50%.

In optimisation, Peng et al. (2010) show that an ensemble containing two algorithms—SaNSDE and G3PC—outperforms CMA-ES, with the claim that "this clearly demonstrates that a combination of some relatively weak algorithms can be stronger than a state-of-the-art algorithm." Although the ensemble performance is clearly superior, we suggest that this claim is possibly exaggerated, given that for the particular results obtained on 27 functions given in their paper, it is unclear to what extent SaNSDE can be described as weak with respect to CMA-ES. Certainly the term *weak* does not have the same conviction as when used in machine-learning: the base algorithms are superior in some instances and at least competitive on others.

However, in Section 6.1 we clearly show that ensembles composed of randomly selected heuristics—inevitably including heuristics with a range of performance—can outperform those composed of high-performing heuristics, over a specific range of ensemble size. From Figure 1, we can claim with confidence that many of these heuristics are weak: individual heuristic scores range from winning 0 instances to 883 (Figure 1), with heuristics in the bottom quartile winning fewer than 36 problems. Figure 5, which plots average individual fitness against ensemble fitness, adds further weight to this point. The average fitness of heuristics in the random ensembles ranges between 0.66 and 0.69, yet leads to ensembles with fitness >0.93. Lack of ability is clearly offset

[1]Strictly speaking, the claim is better worded as *group diversity trumps individual ability.*

by diversity: this is backed up in Figure 6, which shows that diversity in the random ensembles is much higher than in elite ensembles, particularly at low ensemble sizes.

Precise mechanisms for defining behaviour are required. Yuen and Zhang (2015) noted that a clearer understanding of the concept of complementary is required in order to better construct portfolios. We have proposed four different behavioural vectors, with corresponding diversity metrics. The behaviours can be calculated relative to a subset of algorithms (heuristics) under consideration or relative to an *oracle*, defining best-known solutions or relevant bounds.

These explicit definitions of behaviour and diversity represent a significant step beyond the intuitive notions of complementarity previously referred to (Tang et al., 2014), which the authors themselves acknowledge to be vague. The proposed methods are applicable for defining behaviour of an algorithm with respect to a set of problem instances, and therefore for ensembles which are used to solve classes of problems. Although solution fitness has been used to define vectors in this instance, this can easily be replaced by (for example) speed or worst-case behaviour, depending on the application. Alternative definitions are necessary for approaches in which the members of an ensemble collaborate to solve a single instance (e.g., Grobler et al., 2015). An interesting future direction here would be to define a reference set of instances for a problem domain containing a wide range of diverse instances, as a gold-standard against which diversity of any algorithm could be measured.

Choice of diversity metric matters. In relation to the construction of ensembles using diversity as a metric, Tang et al. (2006) pointed out (in a machine-learning context) that "if one exploits diversity measures as criteria to select the base classifiers, then the diversity measure is required to be precise, since the choice of diversity measure will directly influence the final ensemble . . . this is a problem that is important for practical implementation that needs to be noted."

Empirical results in Section 7.2 that evaluated five different metrics showed this also holds in optimisation, with the *disagree* metric operating on the binary vector providing the best results on an unseen test-set, and the Ochiai metric performing particularly poorly. However, in contrast to Tang et al. (2006), who are unable to use explicit diversity measures to construct any ensemble that outperforms common single algorithms from the literature, we find that the disagree metric leads to ensembles that are better than randomly composed ones and elite ensembles over a certain range. Thus, it appears with judicious choice, using diversity metrics in construction is a potential way forward for the optimisation community.

9 Conclusion

The algorithm-selection problem—identifying a specific algorithm to apply to an instance—is nontrivial (Rice, 1976). Ensemble methods offer a solution to this, by including a range of algorithms and with the potential for achieving a result that is better than any of the constituent algorithms. However, unlike in machine-learning, the theory of ensemble design is not well advanced in optimisation, particularly with respect to using metaheuristic methods within the ensemble, rather than exact solvers.

Using a large set of algorithms in conjunction with a large set of problem instances, we have conducted the first in-depth investigation into the factors that underpin ensemble performance. We show that diversity trumps individual ability in certain cases, precise mechanisms for defining behaviour can be found, and that careful choice of diversity metric is required when using diversity to explicitly construct an ensemble.

The new insights shed light on how to create good ensembles from existing algorithms. However, by harnessing the power of genetic programming to generate novel algorithms, those insights could additionally be used to evolve even better ensembles, in which diversity is maximised.

Although the investigation has been conducted purely in the bin-packing domain and using ensembles composed of a specific type of algorithm (i.e., a sequence of rules), we suggest that its findings can be generalised to other domains and to other types of algorithms. With respect to the former point, the bin-packing rules used to form sequences are simply constructive heuristics: they can be directly replaced by constructive heuristics appropriate to another domain (e.g, TSP, JSSP) and we would expect the results to be similar. In terms of the latter, sequences could also be formed from population-based algorithms or other more complex heuristics to permit evaluation in domains that are not amenable to the use of simple heuristics. Fruitful directions in which this work can be pursued are likely to exploit genetic programming which can generate the large numbers of algorithms that might be required to maximise the potential of the ensemble, or use of hyper-heuristic methods which already are capable of generating "new" algorithms by combining heuristics. Future work should also consider the application of ensembles to very large instance sets, themselves containing instances which are as diverse as possible; diversity in instances will encourage diversity in algorithms and vice versa.

Acknowledgments

Emma Hart would like to thank The Leverhulme Trust for funding a personal Research Fellowship (RF-2015-092) during 2015–2016 which supported the entirety of this work.

References

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Champely, S., and Chessel, D. (2002). Measuring biological diversity using Euclidean metrics. *Environmental and Ecological Statistics*, 9(2):167–177.

Falkenauer, E. (1996). A hybrid grouping genetic algorithm for bin packing. *Journal of Heuristics*, 2:5–30.

Garey, M. R., and Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of NP-completeness*. *A series of books in the mathematical sciences*. San Francisco: W.H. Freeman.

Grobler, J., Engelbrecht, A. P., Kendall, G., and Yadavalli, V. (2015). Heuristic space diversity control for improved meta-hyper-heuristic performance. *Information Sciences*, 300:49–62.

Hart, E., and Ross, P. (1998). A heuristic combination method for solving job-shop scheduling problems. In A. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel (Eds.), *Parallel problem solving from nature PPSN V*, pp. 845–854. Lecture Notes in Computer Science, vol. 1498.

Hart, E., and Sim, K. (in press). A hyper-heuristic ensemble method for static job-shop scheduling. *Evolutionary Computation*, doi:10.1162/EVCO_a_00183

Ho, T. K. (1998). The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844.

Hong, L., and Page, S. E. (2004). Groups of diverse problem solvers can outperform groups of high-ability problem solvers. *Proceedings of the National Academy of Sciences of the United States of America*, 101(46):16385–16389.

Kotthoff, L. (2014). Algorithm selection for combinatorial search problems: A survey. *AI Magazine*, 35(3):48–60.

Kuncheva, L. I. (2003). That elusive diversity in classifier ensembles. *Pattern Recognition and Image Analysis*, 1126–1138.

Kuncheva, L. I., and Whitaker, C. J. (2003). Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, 51(2):181–207.

Legendre, P., and Legendre, L. F. (2012). *Numerical ecology*, volume 24. New York: Elsevier.

Liang, J., Qu, B., and Suganthan, P. (2013). *Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization*. Technical Report 201212. Computational Intelligence Laboratory, Zhengzhou University. Singapore: Nanyang Technological University.

Mouret, J.-B., and Doncieux, S. (2012). Encouraging behavioral diversity in evolutionary robotics: An empirical study. *Evolutionary Computation*, 20(1):91–133.

Park, J., Nguyen, S., Zhang, M., and Johnston, M. (2015). A single population genetic programming based ensemble learning approach to job shop scheduling. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation, GECCO Companion '15*, pp. 1451–1452.

Peng, F., Tang, K., Chen, G., and Yao, X. (2010). Population-based algorithm portfolios for numerical optimization. *IEEE Transactions on Evolutionary Computation*, 14(5):782–800.

Rice, J. R. (1976). The algorithm selection problem. *Advances in Computers*, 15:65118.

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning*, 5(2):197–227.

Sim, K., and Hart, E. (2014). An improved immune inspired hyper-heuristic for combinatorial optimisation problems. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, GECCO '14*, pp. 121–128.

Sim, K., Hart, E., and Paechter, B. (2015). A lifelong learning hyper-heuristic method for bin packing. *Evolutionary Computation*, 23(1):37–67.

Skalak, D. B. (1996). The sources of increased accuracy for two proposed boosting algorithms. In *Proceedings of the American Association for Artificial Intelligence, AAAI-96, Integrating Multiple Learned Models Workshop*, p. 1133.

Tang, E. K., Suganthan, P. N., and Yao, X. (2006). An analysis of diversity measures. *Machine Learning*, 65(1):247–271.

Tang, K., Peng, F., Chen, G., and Yao, X. (2014). Population-based algorithm portfolios with automated constituent algorithms selection. *Information Sciences*, 279:94–104.

Valentini, G., and Masulli, F. (2002). Ensembles of learning machines. In *Proceedings of Italian Workshop on Neural Nets*, pp. 3–20.

Vrugt, J. A., and Robinson, B. A. (2007). Improved evolutionary optimization from genetically adaptive multimethod search. *Proceedings of the National Academy of Sciences*, 104(3):708–711.

Vrugt, J. A., Robinson, B. A., and Hyman, J. M. (2009). Self-adaptive multimethod search for global optimization in real-parameter spaces. *Transactions on Evolutionary Computation*, 13(2):243–259.

Yuen, S. Y., Chow, C. K., Zhang, X., and Lou, Y. (2016). Which algorithm should I choose? An evolutionary algorithm portfolio approach. *Applied Soft Computing*, 40:654–673.

Yuen, S. Y., and Zhang, X. (2015). On composing an algorithm portfolio. *Memetic Computing*, 7(3):203–214.