Drake, John H. and Özcan, Ender and Burke, Edmund K. (2015) A modified choice function hyper-heuristic controlling unary and binary operators. In: 2015 IEEE Congress on Evolutionary Computation (CEC2015), 25-28 May 2015, Sendai, Japan.

A Modified Choice Function Hyper-heuristic Controlling Unary and Binary Operators

John H. Drake
ASAP Research Group
School of Computer Science
University of Nottingham
Wollaton Road
Nottingham, NG8 1BB, UK
Email: drakejohnh@gmail.com

Ender Özcan
ASAP Research Group
School of Computer Science
University of Nottingham
Wollaton Road
Nottingham, NG8 1BB, UK
Email: ender.ozcan@nottingham.ac.uk

Edmund K. Burke
CHORDS Research Group
Computing Science and Mathematics
School of Natural Sciences
University of Stirling
Stirling, FK9 4LA, UK
Email: e.k.burke@stir.ac.uk

***Abstract*—Hyper-heuristics are a class of high-level search methodologies which operate on a search space of low-level heuristics or components, rather than on solutions directly. Traditional iterative selection hyper-heuristics rely on two key components, a heuristic selection method and a move acceptance criterion. *Choice Function* heuristic selection scores heuristics based on a combination of three measures, selecting the heuristic with the highest score. *Modified Choice Function* heuristic selection is a variant of the *Choice Function* which emphasises intensification over diversification within the heuristic search process. Previous work has shown that improved results are possible in some problem domains when using *Modified Choice Function* heuristic selection over the classic *Choice Function*, however in most of these cases crossover low-level heuristics (operators) are omitted. In this paper, we introduce crossover low-level heuristics into a *Modified Choice Function* selection hyper-heuristic and present results over six problem domains. It is observed that although on average there is an increase in performance when using crossover low-level heuristics, the benefit of using crossover can vary on a per-domain or per-instance basis.**

I. Introduction

The term 'hyper-heuristic' was defined by Burke et al. [2], [3] as: "...a search method or learning mechanism for selecting or generating heuristics to solve computational search problems". This definition covers the two main classes of hyper-heuristics, those methods which seek to select an appropriate heuristic to apply at a given stage of a search (e.g. [15]) and those which seek to generate new heuristics from an existing set of low-level heuristics or heuristic components (e.g. [7]). A typical selection hyper-heuristic iteratively selects a low-level heuristic to apply to a single solution, making a decision whether to accept the new solution generated at each step. Such hyper-heuristics are labelled $heuristic\ selection\ method$ - $move\ acceptance\ criteria$ in this paper hereafter. Although there has been a proliferation in hyper-heuristic research within the last decade, ideas exhibiting hyper-heuristic behaviour were around as early as 1961 [11]. Recently, hyper-heuristics have been used to solve a variety of problems including bin packing [19], dynamic environments [17], examination timetabling [25], the multidimensional knapsack problem [10] and the vehicle routing problem [12].

The HyFlex [1], [23] framework was introduced to support the first Cross-domain Heuristic Search Challenge (CHeSC2011) [22] and promote the development of general-purpose heuristic search algorithms. The HyFlex framework provides a software interface within which heuristic search methods can be defined and tested on a number of well-known problem domains. For each problem domain a set of low-level heuristics, including crossover low-level heuristics (operators), are implemented for a high-level search methodology to select from. Within the HyFlex framework, all crossover low level heuristics are $binary$ operators, which require two solutions as input each time they are selected, returning a new solution constructed from them. There remaining low-level heuristics are $unary$ operators. In the CHeSC2011 competition very few of the leading entrants provided a strategy for controlling the input for crossover low-level heuristics, with many choosing to omit them altogether.

This paper introduces a scheme to manage the input solutions for crossover low-level heuristics within an existing selection hyper-heuristic presented by Drake et al. [9]. This hyper-heuristic was shown to offer state-of-the-art results in the MAX-SAT problem domain within HyFlex, with mixed results in five other problem domains. We show that it is possible to vastly improve the performance of this hyper-heuristic in a number of problem domains by introducing crossover low-level heuristics.

II. Literature review

Selection hyper-heuristics can be decomposed into two key components [24], a heuristic selection method and a move acceptance criteria. In such hyper-heuristics, a low-level heuristic is selected and applied to a single solution at each step, before a decision is made whether or not to accept the newly generated solution. This framework is illustrated in Figure 1. The *Choice Function* is an elegant heuristic selection method which selects a heuristic to apply based on a weighted combination of three different measures [5]. Drake et al. [9] noted that in the context of cross-domain optimisation *Choice Function*-based hyper-heuristics performed poorly, leading to the proposal of the *Modified Choice Function*. The *Modified Choice Function* controls the intensification and diversification parameters of the *Choice Function* automatically, using a method inspired by Reinforcement Learning. As the focus of that paper was on the comparison of two different selection

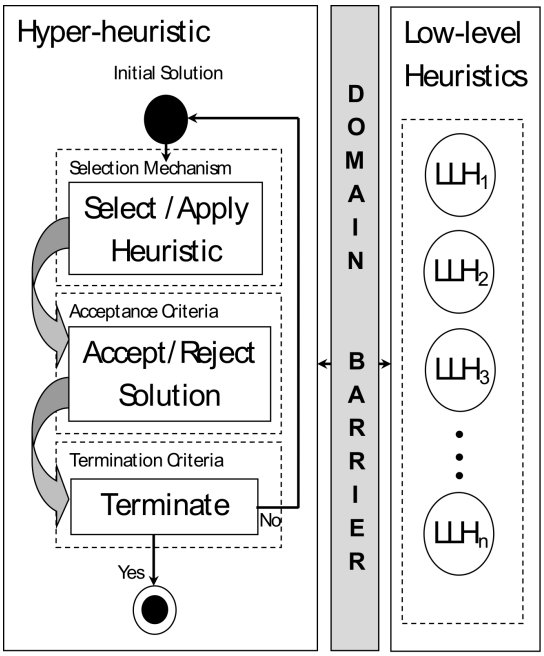


Fig. 1: A traditional selection hyper-heuristic framework

mechanisms, only simple *All Moves* acceptance criteria was used with crossover low-level heuristics omitted entirely.

Crossover is a core operator in many evolutionary algorithms, inspired by its biological namesake, and is now included as a low-level heuristic in many general purpose hyper-heuristic frameworks such as HyFlex [23] and Hyperion [26]. In Genetic Algorithms, the canonical form of crossover combines two suitably fit solutions to yield a new solution which inherits genetic material from both. Jansen and Wegener [16] showed that it is possible to have a function which can be expected to be optimised in polynomial time using a Genetic Algorithm with crossover, whereas using evolution strategies based on only selection and mutation need expected exponential time. Watson and Jansen [27] introduced a function that was solvable by a Genetic Algorithm in polynomial time on average and exponential time for a mutation-based algorithm. Doerr et al. [6] provided the first theoretical proof of crossover being beneficial in a practical optimisation problem. Their work showed that introducing a crossover operator into a mutation-based evolutionary algorithm solving the all-pairs shortest path problem could reduce the expected optimisation time. As crossover is provably beneficial in some problem domains, it follows that it makes sense to use such operators when optimising over multiple problem domains.

As a traditional single-point selection hyper-heuristic operates on a single candidate solution, some method is required to control the additional input arguments required by crossover low-level heuristics. The management of crossover heuristics within selection hyper-heuristics was investigated at two levels of abstraction by Drake et al. [8]. This paper delineated the responsibility of managing the input arguments for crossover as belonging to either the high-level search methodology, or below the domain barrier at the problem-level with the low-level heuristics. Although against the traditionally accepted definition of hyper-heuristics, which strictly enforces the domain barrier, improved performance was observed on instances of the multidimensional knapsack problem by integrating problem domain-specific knowledge. Unfortunately it is not always the case that it is possible to choose the level at which crossover should operate. The HyFlex framework is one such case where crossover management can only be performed at the hyper-heuristic level. In the CHeSC2011 competition, very few of the leading entrants provided a strategy for controlling crossover. Only two of the top ten hyper-heuristics provide a description for managing the second input required by a crossover heuristic. The first uses the current best-of-run solution as the second input solution. The second gives a detailed explanation of a crossover management scheme and was the eventual CHeSC2011 competition winner, *AdapHH* [21]. This hyper-heuristic maintains a memory of the five best solutions seen so far, of which a random solution is used each time a crossover low-level heuristic is chosen. When a new best-of-run solution is found it replaces one of the five solutions in memory chosen at random.

III. A Modified Choice Function - All Moves hyper-heuristic with hyper-heuristic level crossover control

The *Modified Choice Function* is an elegant heuristic selection method which scores heuristics based on three different measures, which emphasises the intensification parameter of the original *Choice Function* [9]. At each iteration of a search, a heuristic is selected based on a weighted combination of these three measures. The first measure (f_1) reflects the previous performance of a given low-level heuristic, with the value of f_1 for a low-level heuristic h_j defined as:

$$f_1(h_j) = \sum_n \phi^{n-1} \frac{I_n(h_j)}{T_n(h_j)} \tag{1}$$

where $I_n(h_j)$ is the change in solution quality, $T_n(h_j)$ is the time taken to execute the heuristic for each previous invocation n of heuristic h_j and ϕ is a value between 0 and 1 giving greater importance to recent performance.

The second measure (f_2) rewards heuristics which are successful when applied consecutively. Values of f_2 are calculated for each heuristic h_j when applied immediately following h_k as follows:

$$f_2(h_k, h_j) = \sum_n \phi^{n-1} \frac{I_n(h_k, h_j)}{T_n(h_k, h_j)} \tag{2}$$

where $I_n(h_k, h_j)$ is the change in fitness, $T_n(h_k, h_j)$ is the time taken to call the heuristic for each previous application n of heuristic h_j following h_k and ϕ is the same value as in f_1.

The third measure (f_3) is the time ($\tau(h_j)$) since each heuristic was last selected by the *Choice Function*. This gives all heuristics at least a small chance of selection.

$$f_3(h_j) = \tau(h_j) \tag{3}$$

In order to rank heuristics, a score is given to each heuristic with *Modified Choice Function* F calculated as:

$$F_t(h_j) = \phi_t f_1(h_j) + \phi_t f_2(h_k, h_j) + \delta_t f_3(h_j) \tag{4}$$

where t is the current iteration. At each step, if the quality of the solution improves, ϕ is is rewarded heavily whilst δ is harshly punished. If the solution quality deteriorates after a low-level heuristic is applied, ϕ is reduced linearly and δ is increased in order to diversify the heuristic search process. This scheme intends to make the intensification component the dominating factor in the calculation of F. In the *Modified Choice Function*, the parameters ϕ_t and δ_t are defined as:

$$\phi_t = \begin{cases} 0.99, & \text{if quality improves} \\ \max\{\phi_{t-1} - 0.01, 0.01\}, & \text{if quality deteriorates} \end{cases} \quad (5)$$

$$\delta_t = 1 - \phi_t \quad (6)$$

Using 0.01 as the minimum weight ensures that ϕ always has some non-negative influence on the F value for each heuristic. Although each individual heuristic has an associated F value, all low-level heuristics use the same ϕ and δ values. The *Modified Choice Function* was shown to outperform the original *Choice Function* on average, over six different problem domains by Drake et al. [9]. In this paper, the *Modified Choice Function* was paired with *All Moves* acceptance criterion, however crossover low-level heuristics were not used. Here crossover low-level heuristics will be included in a *Modified Choice Function - All Moves* hyper-heuristic, with the second solutions for crossover managed at the hyper-heuristic level as defined by Drake et al. [8]. A memory of elite solutions will be maintained from which a second solution, necessary for crossover operators, is used each time a crossover low-level heuristic is selected.

An n-ary operator is a low-level heuristic which requires n solutions as input (assuming $n > 1$). In the HyFlex framework the only n-ary operators currently available are crossover operators. For the experiments in this paper where crossover low-level heuristics are included, each time a crossover heuristic is chosen the first input solution is the incumbent solution. For the second input solution a random solution is provided from a memory of elite solutions of length m, containing the best solutions found so far. At every m-th selection of a crossover heuristic the elite memory of solutions is not used. Instead a new solution is generated from scratch using the solution initialisation methods provided by the framework. If the application of a crossover operator yields an improvement in solution quality compared to the worst solution in the elite memory, the new solution replaces it, provided that this does not result in duplicate solutions appearing in the memory. For all experiments in this paper, the memory length m of potential solutions for n-ary operators is set to 10. This scheme intends to ensure that poor quality solutions found early in the search are quickly expunged from the memory, whilst still preserving a certain element of diversity. A generalised pseudocode of this mechanism is shown in Algorithm 1.

Algorithm 1 Scheme used to control input for crossover low level-heuristics

Inputs:
current solution ($curSoln$)
array of solutions in elite memory ($memSoln[m]$)
crossover operator selected ($crossOp$)
variable to count crossover calls ($calledCount$)
if crossover operator is selected **then**
  $calledCount$++
  if $calledCount$ mod m == 0 **then**
    generate a solution for crossover, $newCrossSoln$
    //apply crossover operator
    $newSoln \leftarrow crossOp(curSoln, newCrossSoln)$
  else
    $index \leftarrow$ random Int between 0 and m-1
    //apply crossover operator
    $newSoln \leftarrow crossOp(curSoln, memSoln[index])$
  end if
  if $newSoln$ is better than the worst solution in $memSoln[]$ **then**
    $newSoln$ replaces the worst solution in $memSoln[]$ if it is not already in $memSoln[]$
  end if
end if

IV. Experimental Framework and Results

This section compares the *Modified Choice Function - All Moves* hyper-heuristic of Drake et al. [9] which does not use crossover low-level heuristics, with the same hyper-heuristic using the crossover management scheme described in Section III. In Section IV-A an indirect comparison is performed, ranking each hyper-heuristic against the set of hyper-heuristics submitted to the CHeSC2011 competition over six benchmark problem domains. Whilst this gives a reasonable overview of performance generally and in some specific domains, little can be said of the performance difference in the domains where both methods perform relatively badly when compared to the competition entrants. As a result we will also provide a direct comparison between the objective function values obtained by both hyper-heuristics in Section IV-B.

A. Indirect comparison of Modified Choice Function - All Moves with and without crossover

Following CHeSC2011 the results for each of the competition entries were provided, over a set of 30 problems taken from six problem domains (MAX-SAT, Bin Packing, Personnel Scheduling, Permutation Flow Shop, Travelling Salesman Problem and Vehicle Routing Problem). These results were taken as the median of 31 runs of each hyper-heuristic on each instance. In each case, our results are also taken as the median of 31 runs in order to directly compare with the competition entries. They are ranked using the 'Formula One' scoring system, with the best performing hyper-heuristic for each instance awarded 10 points, the second 8 points and then each further hyper-heuristic is awarded 6, 5, 4, 3, 2, 1 and 0 points respectively. As this ranking system is based on relative performance, the *Modified Choice Function - All Moves* hyper-heuristics are compared to the competition entries independently. All experiments were carried out on machines allowed 576 seconds running time for a hyper-heuristic on each instance, as defined by the benchmarking tool provided by the competition organisers. Please note that in Figures 2 to 5, the number of hyper-heuristics may vary as methods which score 0 points are omitted from these plots.

Table I(a) shows the results of the *Modified Choice Function - All Moves* hyper-heuristic of Drake et al. [9] which did

TABLE I: Results of the median of 31 runs of the *Modified Choice Function - All Moves* hyper-heuristic (a) without crossover and (b) with crossover, compared to CHeSC2011 competitors using Formula One scores over all six domains

(a)

Rank	Name	Score
1	AdapHH	177.1
2	VNS-TW	131.6
3	ML	127.5
4	PHUNTER	90.25
5	EPH	88.75
6	NAHH	72.5
7	HAHA	71.85
8	ISEA	68.5
9	KSATS	61.35
10	HAEA	52
11	ACO-HH	39
12	**MCF - AM**	**38.85**
13	GenHive	36.5
14	DynILS	27
15	SA-ILS	22.75
16	XCJ	20.5
17	AVEG-Nep	19.5
18	GISS	16.25
19	SelfSearch	5
20	MCHH-S	3.25
21	Ant-Q	0

(b)

Rank	Name	Score
1	AdapHH	179.35
2	VNS-TW	129.35
3	ML	122
4	PHUNTER	86.75
5	EPH	84.75
6	**MCF - AM**	**73.7**
7	HAHA	73.6
8	NAHH	70.5
9	ISEA	65.5
10	KSATS	57.2
11	HAEA	49
12	ACO-HH	37
13	GenHive	33.5
14	SA-ILS	22.1
15	DynILS	22
16	XCJ	19.5
17	AVEG-Nep	18.5
18	GISS	16.6
19	SelfSearch	5.5
20	MCHH-S	3.6
21	Ant-Q	0

not use crossover low-level heuristics. Table I(b) shows the relative results of *Modified Choice Function - All Moves* including crossover management as described in Section III, using the same scoring metrics. From these tables, it can be seen that including crossover heuristics gives a marked improvement in performance when compared to the CHeSC2011 entrants. Where the *Modified Choice Function - All Moves* hyper-heuristic without crossover scores 38.85 points, ranking 12th out of 21 hyper-heuristics, the same hyper-heuristic including crossover scores 73.7 points and ranks 6th. The top five hyper-heuristics are unchanged from the original CHeSC2011 competition, with *AdapHH* [21] in first place with 177.1 points when ranked against *Modified Choice Function - All Moves* without crossover and 179.35 when compared to *Modified Choice Function - All Moves* with crossover. This is interesting as despite being outperformed by the hyper-heuristic containing crossover on average, the variant not including crossover performs better against the best hyper-heuristic overall. This suggests that in at least one problem domain, the *Modified Choice Function - All Moves* hyper-heuristic without crossover is outperforming the *Modified Choice Function - All Moves* hyper-heuristic with crossover.

Although using the Formula One scoring system as a comparison method gives a good indication of performance over all six problem domains, it may be that one method excels in one or more different domains over another. Table II separates the information from Table I, giving the individual scores obtained in each problem domain by each hyper-heuristic. In the case of the MAX-SAT domain the proposed hyper-heuristic with crossover scores 21.2 points, with the original *Modified Choice Function - All Moves* [9] scoring 32.85 points. The *Modified Choice Function - All Moves* hyper-heuristic presented by Drake et al. [9] outperformed all CHeSC2011 entrants in this problem domain, offering state-of-the-art results. This indicates that crossover is in fact detrimental to performance in this domain. For instances of Bin Packing problems *Modified Choice Function - All Moves* with crossover scores 21 points. This is a big improvement on the 0 points scored by the version of this hyper-heuristic without crossover, indicating that crossover greatly improves the solution quality in this domain. In Personnel Scheduling, the *Modified Choice Function - All Moves* hyper-heuristic with crossover performs slightly better than *Modified Choice Function - All Moves* hyper-heuristic without crossover, with each method scoring 8.5 points and 6 points respectively. For Vehicle Routing Problem instances, using crossover again results in an a significant improvement in performance, obtaining 23 points compared to 0 points without crossover. In the case of both Permutation Flow Shop and Travelling Salesman Problem, both *Modified Choice Function - All Moves* variants score 0 points when compared with the CHeSC2011 entrants.

TABLE II: Number of Formula One points scored in each problem domain by *Modified Choice Function - All Moves* with and without crossover

Problem Domain	With Crossover	Without Crossover [9]
MAX-SAT	21.2	**32.85**
Bin Packing	**21**	0
Personnel Scheduling	**8.5**	6
Permutation Flow Shop	0	0
Travelling Salesman Problem	0	0
Vehicle Routing Problem	**23**	0
Total	**73.7**	38.85

Figure 2 shows the number of Formula One points of each of the CHeSC2011 entrants and the *Modified Choice Function - All Moves* hyper-heuristic with crossover in the MAX-SAT problem domain. Here the proposed hyper-heuristic with crossover scores 21.2 points and is the fifth best competitor. Crucially the *Modified Choice Function - All Moves* hyper-heuristic is no longer the highest scoring method as it was when no crossover low-level heuristics were included by Drake et al. [9]. Despite the fact that it is no longer the best method in this domain it still offers competitive performance, outperforming 16 of the other 20 hyper-heuristics. The best hyper-heuristic is *AdapHH* [21] which scores 34.1 points.

Fig. 2: Number of points scored in the MAX-SAT domain by each CHeSC2011 competitor and *Modified Choice Function - All Moves* (*MCF-AM*) hyper-heuristic with crossover

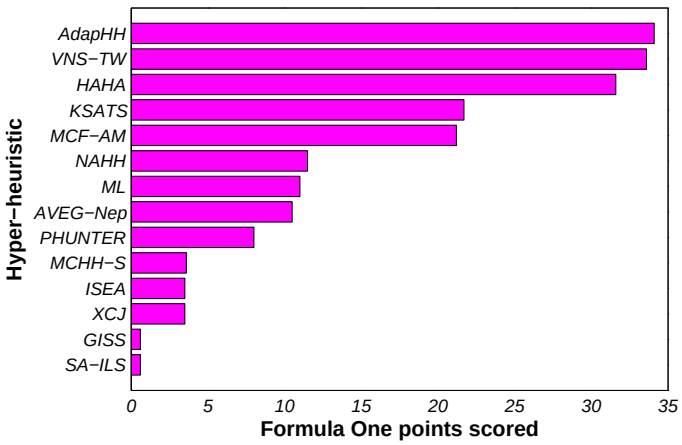


Figure 3 shows the results of the *Modified Choice Function - All Moves* hyper-heuristic with crossover over the Bin Packing problem instances. This hyper-heuristic ranks third in this domain with 21 points, beaten by only two other methods, *ISEA* [18] and *AdapHH*, which score 29 and 45 points respectively.

Fig. 3: Number of points scored in the Bin Packing domain by each CHeSC2011 competitor and *Modified Choice Function - All Moves* (*MCF-AM*) hyper-heuristic with crossover

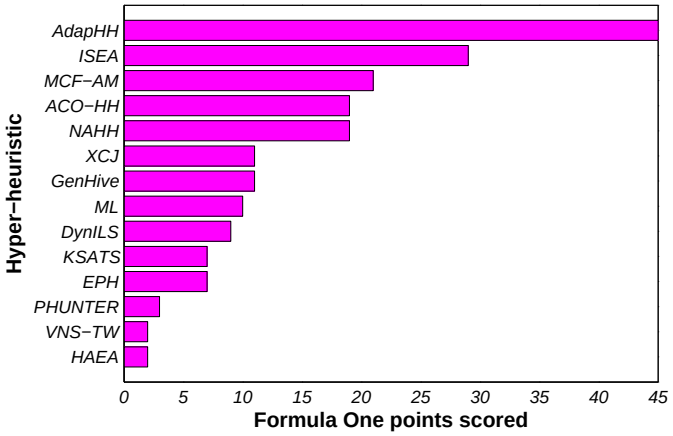


Figure 4 shows the performance of *Modified Choice Function - All Moves* hyper-heuristic with crossover and CHeSC2011 entrants on the Personnel Scheduling domain using the Formula One scoring system. The *Modified Choice Function - All Moves* hyper-heuristic with crossover performs almost as well as the winning CHeSC2011 entrant (*AdapHH*), with only 0.5 points separating these two methods. The best performing hyper-heuristic in Personnel Scheduling is *VNS-TW* [14] with 37.5 points.

Fig. 4: Number of points scored in the Personnel Scheduling domain by each CHeSC2011 competitor and *Modified Choice Function - All Moves* (*MCF-AM*) hyper-heuristic with crossover

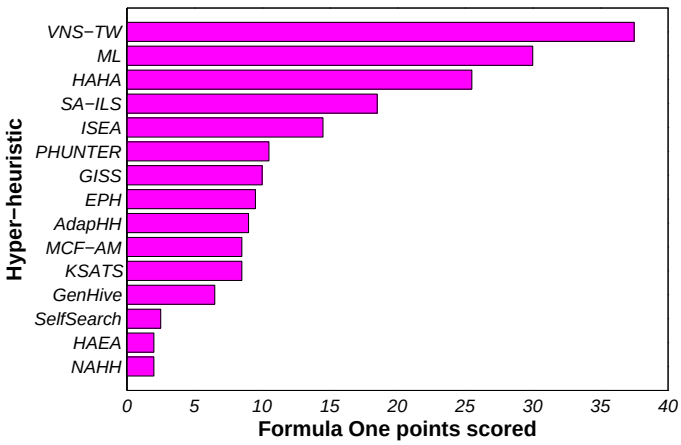
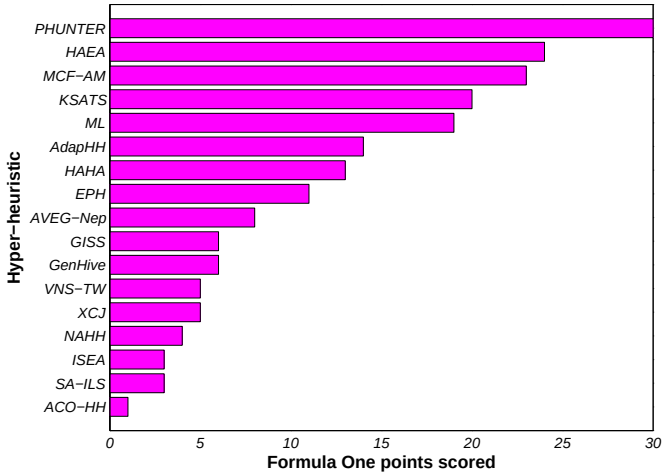


Figure 5 presents the results for the Modified Choice Function - All Moves hyper-heuristic with crossover over the instances of the Vehicle Routing Problem provided by HyFlex. Interestingly the other methods in the top three places also utilise crossover low-level heuristics. The first placed hyper-heuristic in this domain is *PHUNTER* [4] with 30 points and second is *HAEA* with 24 points. This suggests that using crossover may be desirable when trying to obtain solutions comparable with state-of-the-art hyper-heuristics in this problem domain.

Fig. 5: Number of points scored in the Vehicle Routing Problem domain by each CHeSC2011 competitor and *Modified Choice Function - All Moves* (*MCF-AM*) hyper-heuristic with crossover



Despite offering a great improvement in performance in terms of overall Formula One scores when crossover is added, *Modified Choice Function - All Moves* still scores 0 points in the Permutation Flow Shop and Travelling Salesman Problem domains.

The best performing hyper-heuristics in the Permutation Flow Shop domain are *ML*, *AdapHH* and *VNS-TW*. These are also the top three hyper-heuristics in the competition overall. Both *ML* and *VNS-TW* use an underlying Iterated Local Search [20] framework. Iterated Local Search consists of two phases, 'shaking' and 'local improvement'. The shaking phase is applied first and uses perturbation heuristics to modify the current solution and move the search into a different area of the search space. Following this, one or more local search heuristics are applied in the second phase to move the new solution to a local optimum. In both of these hyper-heuristics, only mutation low-level heuristics are used, all crossover low-level heuristics are omitted from the set of available heuristics. Although not strictly tied to an ILS framework, *AdapHH* contains a number of mechanisms which allow it to behave as if it were an ILS hyper-heuristic. It is possible that this hyper-heuristic is behaving in this way in order to be effective in this domain. It could be the case that it is necessary to enforce local search each time a modification is made, in order to reach a local minimum, to obtain strong performance in this domain.

In the case of the Travelling Salesman Problem the best three hyper-heuristics are *AdapHH*, *EPH* and *PHUNTER*. Again, these hyper-heuristics are all amongst the top entrants to the CHeSC2011 competition finishing first, fourth and fifth respectively. All of these hyper-heuristics are capable of selecting crossover low-level heuristics, indicating that crossover may be beneficial in this domain. The hyper-heuristics which finish second and third overall, *VNS-TW* and *ML* are fourth and sixth in this problem domain, with another ILS-based hyper-heuristic, *DynILS*, coming fifth. These three hyper-heuristics are all based on the iterative application of a perturbation operator, followed by a local search phase and do not select from the set of crossover low-level heuristics. This suggests that although crossover low-level heuristics are beneficial to the state-of-the-art methods, they are not necessary to obtain above average performance. Despite the fact that the leading entrants are all hyper-heuristics that use crossover, surprisingly *Modified Choice Function - All Moves* with crossover performs badly in this domain. This implies that it may not simply be a case of whether or not to include crossover, and that the best crossover management methods may in fact be domain-specific. It may also be the case that it is not the low-level heuristic set used which determines the quality of solutions found in this domain, but in fact the synergy between other hyper-heuristic components.

As with any ranking mechanism, there are issues with one method potentially gaining an advantage simply by the metrics of the comparison method used. Di Gaspero and Urli [13] used a normalised cost function value to compare the relative performance of hyper-heuristics. This can be generalised to compare hyper-heuristics over an arbitrary number of instances or domains. The median objective function value of the 31 runs for a given instance are normalised to a value $\in [0, 1]$, using the maximum and minimum fitness value obtained for all hyper-heuristics. The normalised objective function value obj_{norm} for a given problem instance $inst$ is calculated as:

$$obj_{norm}(inst) = \frac{obj_{actual}(inst) - obj_{best}(inst)}{obj_{worst}(inst) - obj_{best}(inst)} \quad (7)$$

where obj_{actual} represents the actual median objective achieved in this instance by a given hyper-heuristic and $obj_{best}(inst)$ and $obj_{worst}(inst)$ represent the best and worst median objective values obtained by any of the CHeSC2011 competitors. Figure 6 shows the normalised objective function values over all 30 instances for the 20 CHeSC2011 competitors and *Modified Choice Function - All Moves* of Drake et al. [9]. Figure 7 provides the same plot using the *Modified Choice Function - All Moves* with crossover and CHeSC2011 entrants. In these figures, the 21 hyper-heuristics being compared are sorted by median normalised objective function value with a lower value indicating better performance.

These box and whisker plots give an indication of relative variation in performance for each hyper-heuristic over all domains. Ranking hyper-heuristics by median normalised objective function value modifies the position of many of the top ten competitors from Table I when compared to both *Modified Choice Function - All Moves* variants. Effectively this metric measures the distance from the best performing hyper-heuristics in every single instance, tested relative to

Fig. 6: Box and whisker comparison of 21 CHeSC2011 entrants and *Modified Choice Function - All Moves* without crossover [9] using normalised objective function

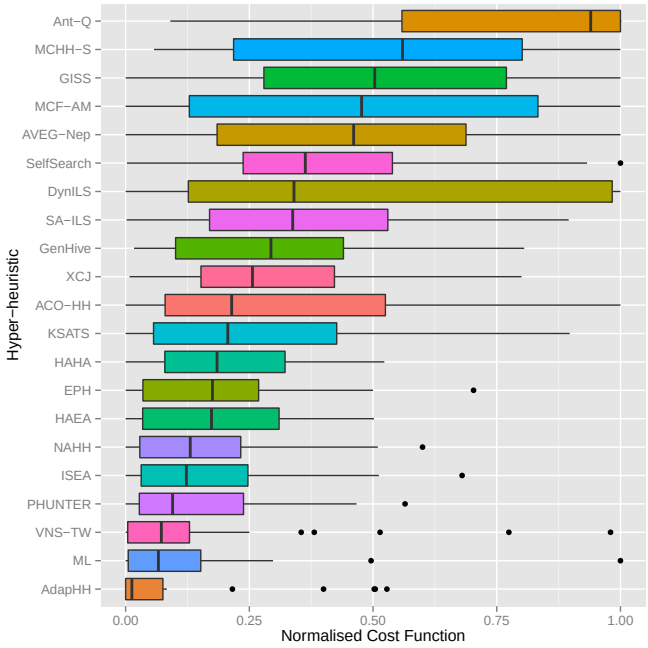
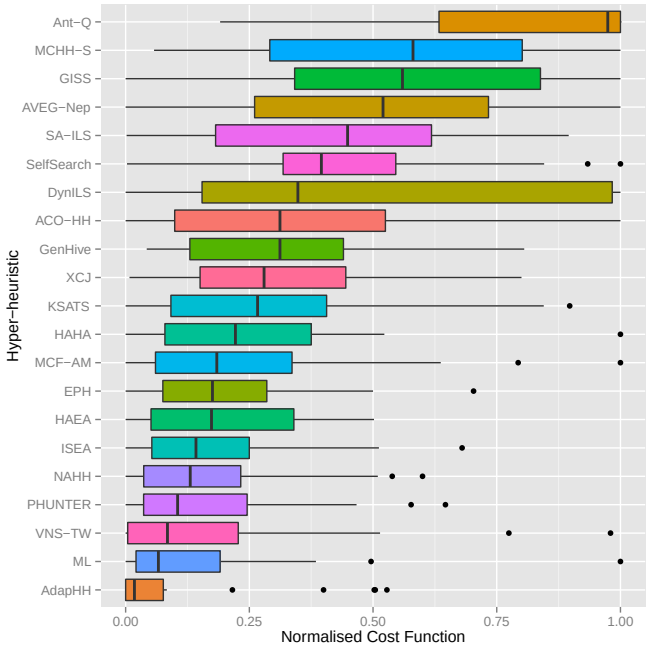


Fig. 7: Box and whisker comparison of 21 CHeSC2011 entrants and *Modified Choice Function - All Moves* with crossover using normalised objective function



the best and worst performing hyper-heuristics. This could arguably provide a better measure of average performance over all 30 instances than the Formula One scoring system. In any case, the best performing hyper-heuristic is still *AdapHH* using this scoring mechanism. It is likely that those hyper-heuristics which rank in a higher position using the Formula One system than using normalised objective function perform particularly well in some problem domains compared to others. The hyper-heuristics placing higher when using median normalised objective function value are likely to provide better performance on average over all domains. Using this metric, again adding crossover low-level heuristics to *Modified Choice Function - All Moves* is clearly beneficial, ranking 9th, where the original *Modified Choice Function - All Moves* [9] ranks 18th.

B. Direct comparison of Modified Choice Function - All Moves with and without crossover

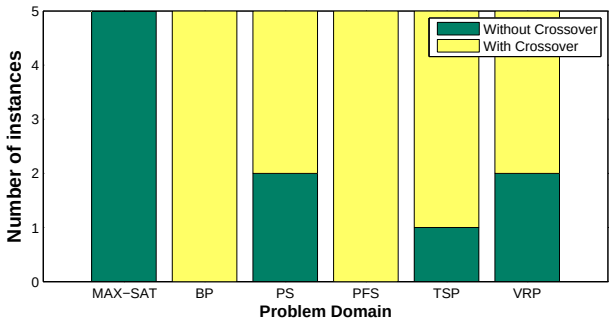
Table III shows the results of an independent Student's t-test within a 95% confidence interval on the objective function values obtained by *Modified Choice Function - All Moves* with and without crossover, for 31 runs of each instance. For each problem domain, five instances are tested. Each cell of the table provides the number of instances of a particular domain in which there is a variation in performance between the two hyper-heuristics. In this table, $>$ and $\gg$ denote the number of cases that *Modified Choice Function - All Moves* with crossover is outperforming *Modified Choice Function - All Moves* without crossover on average or statistically significantly respectively. Conversely, $<$ and $\ll$ denote the number of cases which the *Modified Choice Function - All Moves* without crossover is outperforming *Modified Choice Function - All Moves* with crossover on average or statistically significantly. From this table it becomes clear that there is a certain pattern in the performance in some problem domains with respect to whether or not crossover low-level heuristics are used.

TABLE III: Pairwise comparison of *Modified Choice Function - All Moves* with and without crossover using independent Student's t-test

Problem Domain	$\ll$	$<$	$>$	$\gg$
MAX-SAT	3	2	0	0
Bin Packing	0	0	0	5
Personnel Scheduling	1	1	3	0
Permutation Flow Shop	0	0	3	2
Travelling Salesman Problem	1	0	0	4
Vehicle Routing Problem	1	1	0	3

In the previous section it was shown that *Modified Choice Function - All Moves* is no longer the best hyper-heuristic in the MAX-SAT domain compared to CHeSC2011 entrants when crossover low-level heuristics are introduced. A direct comparison between the objective function values shows that the *Modified Choice Function - All Moves* hyper-heuristic without crossover performs better on average in all 5 instances of the competition set, with the difference being statistically significant in 3 instances. Conversely, in the Bin Packing and Permutation Flow Shop problem domains, *Modified Choice Function - All Moves* with crossover outperforms *Modified Choice Function - All Moves* without crossover on average in all 5 instances. This difference is statistically significant in all 5 Bin Packing instances and in 2 of the 5 Permutation

Fig. 8: Number of competition instances in which *Modified Choice Function - All Moves* hyper-heuristic with and without crossover perform best on average for each CHeSC2011 problem domain



Flow Shop instances. In the case of Bin Packing the difference in performance was noted in the previous section, as there is a clear improvement in relative performance against the CHeSC2011 competitors in this problem domain. For Permutation Flow Shop this difference was less clear in Section IV-A, as both methods scored 0 points using the Formula One scoring system.

Differentiating between the performance of each hyper-heuristic in the other three problem domains is more difficult. In Personnel Scheduling, both methods outperform each other on average in at least one of the instances, with the variant not using crossover obtaining statistically significantly better results in one instance. In the case of both the Travelling Salesman Problem and the Vehicle Routing Problem it is the case that either including or omitting crossover low-level heuristics can provide statistically significantly better results depending on the instance in question. This presents a problem when trying to generalise methods, as performance does not only vary on a per-domain basis but also a per-instance basis. Figure 8 shows some of the information of Table III visually, giving the number of instances in which each hyper-heuristic performs best on average. With the exception of MAX-SAT, the problem domains have been abbreviated in this figure as follows: Bin Packing (BP), Personnel Scheduling (PS), Permutation Flow Shop (PFS), the Travelling Salesman Problem (TSP) and the Vehicle Routing Problem (VRP). In terms of the total number of instances in which each hyper-heuristic performed better on average, *Modified Choice Function - All Moves* with crossover is better in 20 cases and *Modified Choice Function - All Moves* without crossover better in 10 cases.

V. Conclusion

Crossover low-level heuristics have been added to a *Modified Choice Function - All Moves* hyper-heuristic, managed using a hyper-heuristic level crossover control scheme. The inclusion of crossover low-level heuristics results in a large improvement in performance on average over the six benchmark problem domains provided in HyFlex for CHeSC2011. It has been observed that crossover seems to provide a greater benefit in some problem domains or instances than others. In the case of MAX-SAT, Bin Packing and Permutation Flow Shop it

seems that explicitly including or removing crossover low-level heuristics from the set of available heuristics could potentially lead to improved performance. With the remaining three domains, particularly the Travelling Salesman Problem and the Vehicle Routing Problem, performance can vary significantly depending on the instance being solved so making this decision is less clear cut. Five instances is a small sample from which to provide general comments on the performance of a hyper-heuristic, however it is clear that crossover heuristics are beneficial in some problem domains and instances and not others. An interesting question this raises is that if crossover is only beneficial in some circumstances, can methods be designed to recognise when crossover is helpful or not and include it appropriately in a selection hyper-heuristic framework when necessary? This is an interesting future research direction that we intend to pursue further.

REFERENCES

[1] E. K. Burke, T. Curtois, M. Hyde, G. Kendall, G. Ochoa, S. Petrovic, and J. A. Vázquez-Rodríguez, "Hyflex: A flexible framework for the design and analysis of hyper-heuristics," in *Proceedings of the Multidisciplinary International conference on Scheduling: Theory and Applications (MISTA 2009)*, Dublin, Ireland, 2009, pp. 790–797.

[2] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu, "Hyper-heuristics: A survey of the state of the art," *Journal of the Operational Research Society*, vol. 64, no. 12, pp. 1695–1724, 2013.

[3] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. Woodward, *Handbook of Metaheuristics 2nd ed.* Springer, 2010, ch. A Classification of Hyper-heuristic Approaches, pp. 449–468.

[4] C.-Y. Chan, F. Xue, W. Ip, and C. Cheung, "A hyper-heuristic inspired by pearl hunting," in *Proceedings of Learning and Intelligent Optimization (LION 2012)*, ser. LNCS, Y. Hamadi and M. Schoenauer, Eds., vol. 7219. Paris, France: Springer, 2012, pp. 349–353.

[5] P. Cowling, G. Kendall, and E. Soubeiga, "A hyperheuristic approach to scheduling a sales summit," in *Proceedings of the International Conference on the Practice and Theory of Automated Timetabling (PATAT 2000)*, ser. LNCS, E. K. Burke and W. Erben, Eds., vol. 2079. Konstanz, Germany: Springer, 2001, pp. 176–190.

[6] B. Doerr, E. Happ, and C. Klein, "Crossover can provably be useful in evolutionary computation," *Theoretical Computer Science*, vol. 436, pp. 71–86, 2012.

[7] J. H. Drake, N. Kililis, and E. Özcan, "Generation of vns components with grammatical evolution for vehicle routing," in *Genetic Programming - 16th European Conference (EuroGP 2013)*, ser. LNCS, K. Krawiec, A. Moraglio, T. Hu, A. S. Etaner-Uyar, and B. Hu, Eds., vol. 7831. Vienna, Austria: Springer, 2013, pp. 25–36.

[8] J. H. Drake, E. Özcan, and E. K. Burke, "A case study of controlling crossover in a selection hyper-heuristic framework using the multidimensional knapsack problem," *Evolutionary Computation*, To appear.

[9] ——, "An improved choice function heuristic selection for cross domain heuristic search," in *Proceedings of Parallel Problem Solving from Nature (PPSN 2012), Part II*, ser. LNCS, C. A. C. Coello, V. Cutello, K. Deb, S. Forrest, G. Nicosia, and M. Pavone, Eds., vol. 7492. Taormina, Italy: Springer, 2012, pp. 307–316.

[10] ——, "Modified choice function heuristic selection for the multidimensional knapsack problem," in *Proceedings of the International Conference on Genetic and Evolutionary Computing (ICGEC2014)*, ser. Advances in Intelligent Systems and Computing, H. Sun, C.-Y. Yang, C.-W. Lin, J.-S. Pan, V. Snásel, and A. Abraham, Eds. Nanchang, China: Springer, 2014, pp. 225–234.

[11] H. Fisher and G. Thompson, "Probabilistic learning combinations of local job-shop scheduling rules," in *Factory Scheduling Conference*, Carnegie Institute of Technology, 1961.

[12] P. Garrido and C. Castro, "Stable solving of cvrps using hyperheuristics," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2009)*, F. Rothlauf, Ed. Québec, Canada: ACM, 2009, pp. 255–262.

[13] L. D. Gaspero and T. Urli, "Evaluation of a family of reinforcement learning cross-domain optimization heuristics," in *Proceedings of Learning and Intelligent Optimization (LION 2012)*, ser. LNCS, Y. Hamadi and M. Schoenauer, Eds., vol. 7219. Paris, France: Springer, 2012, pp. 384–389.

[14] P.-C. Hsiao, T.-C. Chiang, and L.-C. Fu, "A vns-based hyper-heuristic with adaptive computational budget of local search," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2012)*. Brisbane, Australia: IEEE Press, 2012, pp. 1–8.

[15] W. G. Jackson, E. Özcan, and J. H. Drake, "Late acceptance-based selection hyper-heuristics for cross-domain heuristic search," in *Proceedings of the 13th Annual Workshop on Computational Intelligence (UKCI 2013)*, Y. Jin and S. A. Thomas, Eds. Surrey, UK: IEEE Press, 2013, pp. 228–235.

[16] T. Jansen and I. Wegener, "Real royal road functions - where crossover provably is essential," *Discrete Applied Mathematics*, vol. 149, no. 1-3, pp. 111–125, 2005.

[17] B. Kiraz, A. S. Uyar, and E. Özcan, "Selection hyper-heuristics in dynamic environments," *Journal of the Operational Research Society*, vol. 64, no. 12, pp. 1753–1769, 2013.

[18] J. Kubalik, "Hyper-heuristic based on iterated local search driven by evolutionary algorithm," in *Proceedings of Evolutionary Computation in Combinatorial Optimization (EvoCOP 2012)*, ser. LNCS, J.-K. Hao and M. Middendorf, Eds., vol. 7245. Malaga, Spain: Springer, 2012, pp. 148–159.

[19] E. López-Camacho, H. Terashima-Marín, and P. Ross, "A hyper-heuristic for solving one and two-dimensional bin packing problems," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2011)*, P. L. L. Natalio Krasnogor, Ed. Dublin, Ireland: ACM, 2011, pp. 257–258.

[20] H. R. Lourenço, O. Martin, and T. Stützle, *Handbook of Metaheuristics 2nd ed.* Springer, 2010, ch. Iterated Local Search: Framework and Applications, pp. 363–397.

[21] M. Misir, K. Verbeeck, P. D. Causmaecker, and G. V. Berghe, "An intelligent hyper-heuristic framework for chesc 2011," in *Proceedings of Learning and Intelligent Optimization (LION 2012)*, ser. LNCS, Y. Hamadi and M. Schoenauer, Eds., vol. 7219. Paris, France: Springer, 2012, pp. 461–466.

[22] G. Ochoa and M. Hyde. (2011) The cross-domain heuristic search challenge (CHeSC 2011). ASAP Research Group, The University of Nottingham. [Online]. Available: http://www.asap.cs.nott.ac.uk/chesc2011/

[23] G. Ochoa, M. Hyde, T. Curtois, J. A. Vázquez-Rodríguez, J. D. Walker, M. Gendreau, G. Kendall, B. McCollum, A. J. Parkes, S. Petrovic, and R. Qu, "Hyflex: A benchmark framework for cross-domain heuristic search," in *Proceedings of Evolutionary Computation in Combinatorial Optimization (EvoCOP 2012)*, ser. LNCS, J.-K. Hao and M. Middendorf, Eds., vol. 7245. Malaga, Spain: Springer, 2012, pp. 136–147.

[24] E. Özcan, B. Bilgin, and E. E. Korkmaz, "A comprehensive analysis of hyper-heuristics," *Intelligent Data Analysis*, vol. 12, no. 1, pp. 3–23, 2008.

[25] E. Özcan, M. Misir, G. Ochoa, and E. K. Burke, "A reinforcement learning - great-deluge hyper-heuristic for examination timetabling," *International Journal of Applied Metaheuristic Computing*, vol. 1, no. 1, pp. 39–59, 2010.

[26] J. Swan, E. Özcan, and G. Kendall, "Hyperion - a recursive hyper-heuristic framework," in *Proceedings of Learning and Intelligent Optimization (LION 2011)*, ser. LNCS, C. A. C. Coello, Ed., vol. 6683. Rome, Italy: Springer, 2011, pp. 616–630.

[27] R. A. Watson and T. Jansen, "A building-block royal road where crossover is provably essential," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2007)*, H. Lipson, Ed. London, UK: ACM, 2007, pp. 1452–1459.