A Comparison of Crossover Control Mechanisms within Single-point Selection Hyper-heuristics using HyFlex

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 search methodologies which operate at a higher level of abstraction than traditional search and optimisation techniques. Rather than operating on a search space of solutions directly, a hyper-heuristic searches a space of low-level heuristics or heuristic components. An iterative selection hyper-heuristic operates on a single solution, selecting and applying a low-level heuristic at each step before deciding whether to accept the resulting solution. Crossover low-level heuristics are often included in modern selection hyper-heuristic frameworks, however as they require multiple solutions to operate, a strategy is required to manage potential solutions to use as input. In this paper we investigate the use of crossover control schemes within two existing selection hyper-heuristics and observe the difference in performance when the method for managing potential solutions for crossover is modified. Firstly, we use the crossover control scheme of *AdapHH*, the winner of an international competition in heuristic search, in a *Modified Choice Function - All Moves* selection hyper-heuristic. Secondly, we replace the crossover control scheme within *AdapHH* with another method taken from the literature. We observe that the performance of selection hyper-heuristics using crossover low-level heuristics is not independent of the choice of strategy for managing input solutions to these operators.**

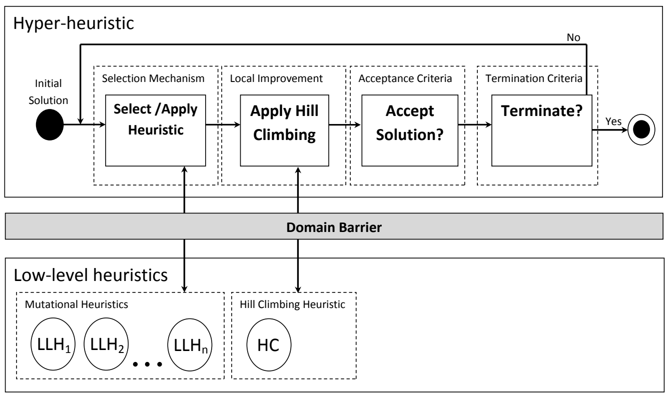
I. Introduction

Optimisation problems often explore a search space which is too large to enumerate and exhaustively search for an optimal solution. Various heuristics and metaheuristics have been applied successfully to problems of this nature. One drawback of such approaches is the necessity to manually adapt the method used to solve different problem domains or classes of problem. Hyper-heuristics are a class of high-level search techniques which operate at a higher level of abstraction than traditional search and optimisation techniques [3]. One of the key goals of hyper-heuristic research is the automation of the heuristic design process, minimising the amount of human effort needed to design effective problem solving methods. Unlike traditional strategies for search and optimisation, hyper-heuristics operate over a search space of low-level heuristics or heuristic components, rather than over a search space of solutions directly. Hyper-heuristics are broadly split into two main categories [4], those which select a low-level heuristic to apply from a set of existing heuristics [17] and those which create new heuristics from a set of low-level components [9]. This paper is concerned with the first category, those methodologies which select a low-level heuristic to apply at a given point in a search. Selection hyper-heuristics have been used to solve a large number of problems including examination timetabling [23], sports scheduling [15] and vehicle routing problems [14].

The objective of cross-domain heuristic search is to develop methods which are able to consistently find good quality solutions in multiple problem domains, using a given set of low-level heuristics. The HyFlex framework [1], [21], introduced chiefly to support the first Cross-domain Heuristic Search Challenge (CHeSC2011) [2], is used as a benchmark framework for the methods investigated in this paper. HyFlex provides a common software interface to test the performance of high-level search strategies over multiple problem domains, and an increasing body of associated research with which to compare. Currently HyFlex supports six problem domains and provides implementations of low-level heuristics from four categories: crossover, mutation, ruin-recreate and hill climbing for each domain. Unlike the other three categories, crossover low-level heuristics require more than one solution as input. As a typical selection hyper-heuristic is based on a single-point search framework (i.e. it operates on a single solution), some strategy is needed to manage the second input solution for this type of heuristic.

The winner of the CHeSC2011 competition, *AdapHH* [20], is considered as one of the state-of-the-art selection hyper-heuristics in cross-domain search. *AdapHH* was the only method of the top 10 entrants to CHeSC2011 to provide a strategy to manage input for crossover low-level heuristics. Drake et al. [13] presented results over the CHeSC2011 benchmarks using a *Modified Choice Function - All Moves* hyper-heuristic, improving on the initial results of Drake et al. [11] by including a strategy to manage the input for crossover low-level heuristics. In this paper we present a set of experiments to investigate the use of crossover operators within these two previously proposed selection hyper-

Fig. 1: F_C single-point search hyper-heuristic framework



heuristics. Firstly, we replace the crossover control mechanism of Drake et al. [13] within a *Modified Choice Function - All Moves* hyper-heuristic with the mechanism from AdapHH, resulting in a significant deterioration in performance in some problem domains. Secondly, we use the crossover mechanism of Drake et al. [13] within *AdapHH*, with mixed results in different problem domains.

II. Literature Review

Özcan et al. [22] decomposed selection hyper-heuristics into two fundamental components, a heuristic selection method and a move acceptance criterion. A number of frameworks such hyper-heuristics can operate in was also defined. In a traditional selection hyper-heuristic framework operating on a single solution, low-level heuristics are iteratively selected and applied to the solution, with a decision made following each application as to whether to accept the modified solution. This is typically referred to as an F_A selection hyper-heuristic using the definitions of Özcan et al. [22]. A selection hyper-heuristic operating within an F_C framework is defined as one which first selects and applies a heuristic from a set of mutational low-level heuristics, before applying a hill climbing low-level heuristic. This is the framework that will be used for any *Modified Choice Function - All Moves* hyper-heuristics in this paper and is depicted in Figure 1.

Despite the inclusion of crossover operators in modern hyper-heuristic frameworks such as HyFlex [21] and Hyperion [24], limited research effort has been directed at managing the input for this type of low-level heuristic. Typically a selection hyper-heuristic operates within a single-point search framework, operating on a single solution. In the case of low-level heuristics which require more than one solution as input, a natural choice for one of the solutions is the current working solution in the hyper-heuristic. Unfortunately there is not necessarily a natural choice for which solution to use as the second input to such operators. Drake et al. [10] introduced the notion of crossover control at two different conceptual levels, raising a broader question regarding the responsibility of crossover management within selection hyper-heuristics. Although controlling crossover below the domain barrier has shown to be useful in some problem domains [10], depending on the framework used this is not always possible. In the case of the HyFlex framework, this responsibility lies entirely with the implementer of the high-level heuristic search methodology rather than the coder of the low-level problem domain. Doerr et al. [7], [8] showed that crossover is provably beneficial in at least some classes of practical optimisation problems. It follows that in order to be successful, methods operating over multiple problem domains need to manage this type of operator effectively.

The *Choice Function* was introduced as a heuristic selection method by Cowling et al. [6] in the first hyper-heuristic paper in the field of combinatorial optimisation. The *Choice Function* scores heuristics based on a combination of three different measures and applies the heuristic with the highest rank at each given step. Each measure is weighted to prioritise intensification and diversification in the heuristic search process at appropriate times during the search. A number of methods have been proposed in the literature to set appropriate parameter values to weight these measures. One such method is the *Modified Choice Function* heuristic selection method proposed by Drake et al. [11], designed to overcome some of the limitations of the original *Choice Function* within the context of cross-domain heuristic search. Placing a heavy emphasis on the intensification component of the original *Choice Function*, the *Modified Choice Function* showed increased performance on average over the CHeSC2011 benchmark instances. *Modified Choice Function* heuristic selection has also been used in the context of dynamic environments [18] and the multidimensional knapsack problem [12].

The winner of the CHeSC2011 competition, *AdapHH* [20], is a highly adaptive hyper-heuristic which contains a number of online learning mechanisms. Unlike many of the other leading entrants to CHeSC2011, *AdapHH* includes crossover operators in the available low-level heuristic set. Although the original description of this algorithm contained no mention of a method to manage the second input solution required by crossover operators, the code of *AdapHH* was later made publicly available with the crossover management scheme described by Misir [19] as:

> '...a population of five solutions including previously explored new best solutions is accommodated. They are applied using the current solution and a randomly selected one from these five solutions. Each time a new best solution is found, a randomly chosen solution from these solutions is replaced by the new solution.'

Again in the context of the CHeSC2011 benchmark instances, Drake et al. [13] presented a *Modified Choice Function - All Moves* hyper-heuristic using crossover low-level heuristics managed using a control scheme inspired by Drake et al. [10]. This hyper-heuristic operated within an F_A framework and showed a vast improvement in solution quality over the six benchmark problem domains in HyFlex, compared to the results of Drake et al. [11]. Each time the heuristic selection mechanism selects a crossover low-level heuristic, a solution is selected randomly from a memory of elite solutions to use as a second input, with the first input being the incumbent solution within the single-point search framework. The memory of elite solutions is updated each time a new solution is found which is an improvement over any existing solution within the memory. In this case, the new solution replaces the poorest quality solution in the elite list. At every m-th application of a crossover low-level heuristic, where m is the length of the memory of elite solutions, a new randomly generated solution is used as the second input for crossover. This intends to preserve some element of diversity within the search.

III. Experimental Framework Definitions

The following sections of this paper explore the introduction of different crossover control schemes into different hyper-heuristics, using the instances from CHeSC2011 as a benchmark for comparison. Section IV-A modifies the hyper-heuristics of Drake et al. [13] to use the F_C framework as defined by [22] and compares the effect of using the original crossover control scheme to using the *AdapHH* crossover control scheme of Misir et al. [20] and omitting crossover low-level heuristics completely. Section IV-B compares the performance when replacing the original crossover control mechanism within the *AdapHH* hyper-heuristic with the method used in the *Modified Choice Function - All Moves* hyper-heuristic presented by Drake et al. [13].

The crossover control mechanism of Drake et al. [13] selects a second solution from a memory of elite solutions, with a random solution used every mth time a crossover low-level heuristic is selected. In the original work, the value of m was set to 10, in order to quickly expunge poor quality solutions early on in the search process. In all experiments using this crossover control mechanism within this paper we will also set this value to 10. In the case of the *AdapHH* crossover control mechanism, each time a second solution is required by a crossover operator, one is selected at random from a list of 5 solutions. In the case a superior solution is found after applying the crossover low-level heuristic, the new solution replaces one of the 5 solutions in the list at random. Initially the list of 5 solutions is comprised of copies of the starting solution.

In order to compare the hyper-heuristics submitted to the competition, CHeSC2011 used a points-based scoring system inspired by a system previously used by Formula One motor racing to rank performance. Here we will use the same method to compare hyper-heuristics. The Formula One ranking system (2003-2009) assigns a number of points to different competitors based on their performance. The first place competitor is 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 the Formula One ranking system allocates scores to the top 8 ranked contestants, all hyper-heuristics which are ranked $\geq$ 9th position are given a score of 0. A selection of five instances were selected from each of the six problem domains to use as testing instances, resulting in a total of 30 instances. The six problem domains are: Boolean satisfiability (MAX-SAT), one-dimensional bin packing (BP), personnel scheduling (PS) and permutation flow shop (FS), the vehicle routing problem (VRP) and the travelling salesman problem (TSP). Each hyper-heuristic was allowed to run for a notional 10 minutes per instance, as dictated by the benchmarking tool provided by the CHeSC2011 organisers. Runs are repeated 31 times in order to account for the stochastic nature of solving such optimisation problems. The median result of the 31 runs is reported as the score for a given hyper-heuristic applied to a given instance. Scores are then allocated using the Formula One system outlined above, based on the median results obtained by each hyper-heuristic for each instance. As the maximum number of points for each instance is 10, and there are 30 instances in total, the maximum possible score for any hyper-heuristic is 300.

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

(a)

Rank	Name	Score
1	AdapHH	179.1
2	VNS-TW	131.6
3	ML	127.5
4	PHunter	90.25
5	EPH	89.25
6	HAHA	72.85
7	NAHH	71.5
8	ISEA	68.5
9	KSATS-HH	61.35
10	HAEA	52
11	ACO-HH	39
12	GenHive	36.5
13	**MCF - AM**	**36.35**
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	176.85
2	VNS-TW	131.35
3	ML	122
4	PHunter	87.75
5	EPH	84.25
6	HAHA	74.1
7	**MCF - AM**	**73.2**
8	NAHH	69.5
9	ISEA	65.5
10	KSATS-HH	57.7
11	HAEA	49
12	ACO-HH	37
13	GenHive	33.5
14	SA-ILS	23.1
15	DynILS	22
16	XCJ	18.5
17	AVEG-Nep	18.5
18	GISS	16.6
19	SelfSearch	6
20	MCHH-S	3.6
21	Ant-Q	0

IV. Experiments and Results

A. *The F_C Modified Choice Function - All Moves hyper-heuristic with and without crossover*

The *Modified Choice Function - All Moves* hyper-heuristics of Drake et al. [13] operated using an F_A framework. That is, the hyper-heuristics select and apply a low-level heuristic from the full set of available heuristics without discriminating between different heuristic types. This section will modify these hyper-heuristics to operate using an F_C framework. As discussed in detail by Drake et al. [13], the top three hyper-heuristics in CHeSC2011 are all capable of behaving as F_C selection hyper-heuristics. Table I presents the results of *Modified Choice Function - All Moves* with and without crossover, using an F_C selection hyper-heuristic framework over the CHeSC2011 benchmarks. As stated previously, the median of each of 31 ten minute runs are used to compare a hyper-heuristic with the 20 entrants to CHeSC2011 using the Formula One scoring system. From Table I(a) and Table I(b) it is clear that the F_C versions of these hyper-heuristics perform slightly worse than their F_A counterparts presented by Drake et al. [13].

Table II shows the breakdown of points scored in each problem domain, compared to the CHeSC2011 entrants, for each of the F_A and F_C *Modified Choice Function - All Moves* hyper-heuristics with and without crossover. In the case of *Modified Choice Function - All Moves* hyper-heuristics without crossover, very high scores are obtained in the MAX-SAT problem domain within both an F_A and an F_C framework. The F_C *Modified Choice Function - All Moves* without crossover scores 28.35 points in this domain whereas the F_A version of this hyper-heuristic scored 32.85 points. Despite performing well, the F_C hyper-heuristic is not the best method in this domain overall, unlike the F_A *Modified Choice Function - All Moves* of Drake et al. [11] which outperformed all 20 CHeSC2011 competitors. The performance of these two hyper-heuristics over all six problem domains is very similar in terms of Formula One score. The F_C and F_A *Modified Choice Function - All Moves* hyper-heuristics without crossover score 36.35 and 38.85 points in total respectively against the CHeSC2011 competitors. Both hyper-heuristics score 6 points in the personnel scheduling problem domain. Although the F_C hyper-heuristic scores fewer points in total, it does score 2 points in the vehicle routing problem where previously the F_A hyper-heuristic scored no points. This could be a direct result of the addition of a local improvement phase. The methods which are first, second and third in this domain all operate within frameworks which use local improvement following a perturbative move in the search space.

For *Modified Choice Function - All Moves* with crossover (Table II(c) and Table II(d)), again the performance of the F_A and F_C variants are similar in terms of total Formula One points scored (73.7 and 73.2 respectively). In the case of MAX-SAT, both variants score exactly the same number of Formula One points (21.2) so the inclusion of a local search phase has little effect in this domain. Both methods are also similar in bin packing and the vehicle routing problem, with the F_C hyper-heuristic slightly outperforming the F_A hyper-heuristic obtaining 24 and 25 points in each domain respectively, compared to 21 and 23 points gained by the F_A variant. The only notable difference in performance is observed in the personnel scheduling domain where the F_A hyper-heuristic scores 8.5 points and the F_C hyper-heuristic scores 2. This is unexpected as in general hyper-heuristics which use a local search phase performed well in this domain in the competition.

Table III shows the Formula One scores obtained by the *Modified Choice Function - All Moves* hyper-heuristics with the *AdapHH* crossover control scheme, compared to the 20 CHeSC2011 entrants. Overall the *Modified Choice Function - All Moves* with *AdapHH* crossover scores 34.6 points over the six problem domains, ranking 12th out of 21 when compared to the CHeSC2011 entries. This hyper-heuristic performs very well in the vehicle routing problem, scoring 26 points, second only to *PHunter* [5]. In the remaining five problem domains points are scored in four, with 6 points scored in bin packing being the only performance of note.

Figure 2 shows the number of Formula One points scored by an F_C *Modified Choice Function - All Moves* hyper-heuristic using the crossover control scheme described by Drake et al. [13], the crossover control scheme of *AdapHH* [20] or no crossover at all.

From this figure, we can see that the most striking difference between the hyper-heuristic using the *AdapHH* crossover control scheme and the two *Modified Choice Function - All Moves* hyper-heuristics from Section IV-A (With Crossover and Without Crossover), is the complete deterioration in performance in the MAX-SAT problem domain. Whereas the

TABLE II: Formula One scores for (a) F_A *Modified Choice Function - All Moves* without crossover [13], (b) F_C *Modified Choice Function - All Moves* without crossover, (c) F_A *Modified Choice Function - All Moves* with crossover [13] and (d) F_C *Modified Choice Function - All Moves* with crossover in each problem domain compared to CHeSC2011 competitors

(a)

SAT	BP	PS	FS	TSP	VRP	Total
32.85	0	6	0	0	0	38.85

(b)

SAT	BP	PS	FS	TSP	VRP	Total
28.35	0	6	0	0	2	36.35

(c)

SAT	BP	PS	FS	TSP	VRP	Total
21.2	21	8.5	0	0	23	73.7

(d)

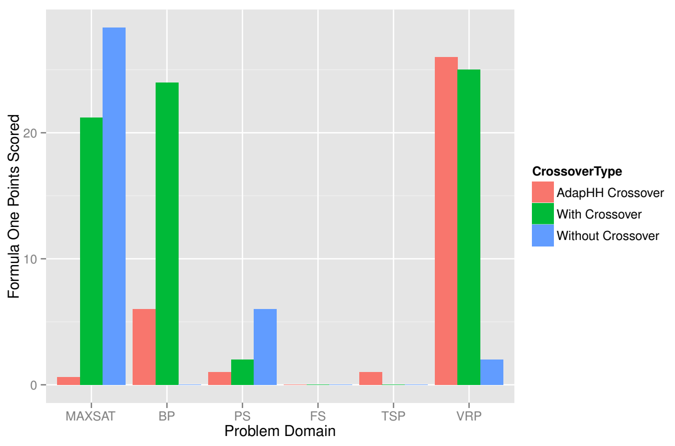
SAT	BP	PS	FS	TSP	VRP	Total
21.2	24	2	0	1	25	73.2

TABLE III: Formula One scores for *Modified Choice Function - All Moves* with the crossover control mechanism of the CHeSC2011 winner and CHeSC2011 competitors in each problem domain

Rank	Name	SAT	BP	PS	FS	TSP	VRP	Total
1	AdapHH	34.75	45	9	37	40.25	14	180
2	VNS-TW	34.25	2	39.5	34	17.25	5	132
3	ML	14.5	11	31	39	13	18	126.5
4	PHunter	10.5	3	11.5	9	26.25	30	90.25
5	EPH	0	10	10.5	21	35.25	11	87.75
6	HAHA	32.75	0	25.5	3.5	0	12	73.75
7	NAHH	14	19	2	22	12	4	73
8	ISEA	6	29	14.5	3.5	12	3	68
9	KSATS-HH	23.85	11	9.5	0	0	20	64.35
10	HAEA	0.5	3	2	10	11	24	50.5
11	ACO-HH	0	20	0	9	8	1	38
12	**MCF - AM**	**0.6**	**6**	**1**	**0**	**1**	**26**	**34.6**
13	GenHive	0	12	6.5	7	3	6	34.5
14	DynILS	0	12	0	0	13	0	25
15	SA-ILS	0.6	0	19.5	0	0	3	23.1
16	XCJ	5.5	12	0	0	0	4	21.5
17	AVEG-Nep	12	0	0	0	0	8	20
18	GISS	0.6	0	10	0	0	6	16.6
19	SelfSearch	0	0	3	0	3	0	6
20	MCHH-S	4.6	0	0	0	0	0	4.6
21	Ant-Q	0	0	0	0	0	0	0

decision to either include crossover operators or not in the low-level heuristic set did not affect the performance in this domain significantly, the management of the inputs for such operators can have a great effect on performance. This is an interesting result as it is not necessarily a simple case of whether or not crossover operators are useful in a particular problem class. In the case of the vehicle routing problem it is clear from Figure 2 that both of the hyper-heuristics which use crossover outperform the hyper-heuristic which doesn‘t. This result suggests that crossover is inherently beneficial in the case of the vehicle routing problem irrespective of method of managing the second solutions. Bin packing is another interesting case, where including crossover obviously provides benefits, however the strategy used to manage the

Fig. 2: Formula One points scored using each crossover control scheme within F_C *Modified Choice Function - All Moves* compared to CHeSC2011 entrants



input solutions also has a great bearing on performance. The crossover strategy of *AdapHH* scores 6 points in this domain, with the crossover control scheme of Drake et al. [13] scoring 24 points. For the travelling salesman problem and permutation flow shop domains, all three hyper-heuristics perform particularly poorly, scoring a single point between them. It is possible that there is another crossover control scheme which will perform well in these domains, however it is more likely that this is a result of the high-level *heuristic selection method - move acceptance criterion* combination used. The methods that perform particularly well in these two domains are often the hyper-heuristics which perform well overall. Not including crossover operators at all is the best scheme in personnel scheduling however even this strategy does not perform well when compared to the CHeSC2011 entrants.

B. *Introducing a different crossover control scheme into the CHeSC2011 winner*

Where the previous section introduced the crossover control scheme of the CHeSC2011 winner into an F_C *Modified Choice Function - All Moves hyper-heuristic*, this section will reverse these roles and introduce the crossover control scheme defined by Drake et al. [13] into the CHeSC2011 winner. Following the CHeSC2011 competition the source code for the winning hyper-heuristic (*AdapHH*) was made available[1]. For our experiments, no modifications were made to this source code with the exception of modifying the mechanism to provide second solutions for crossover. Here the results of an independent set of 31 runs of AdapHH over the CHeSC2011 benchmarks are presented along with 31 runs of this hyper-heuristic with the crossover control scheme of Drake et al. [13]. As an indirect comparison, Table IV(a) and Table IV(b) present the number of Formula One points scored by each *AdapHH* hyper-heuristic when compared with the other 19 competition entrants. These figures are presented visually in Figure 3. Please note that in each case the results obtained replace the results of the original CHeSC2011 *AdapHH* hyper-heuristic when calculating the Formula One scores for each instance.

[1]Available at: http://code.google.com/p/generic-intelligent-hyper-heuristic/downloads/list

TABLE IV: Formula One scores for (a) an independent run of *AdapHH* and (b) *AdapHH* with modified crossover management scheme in each problem domain compared to CHeSC2011 competitors

(a)

Rank	Name	SAT	BP	PS	FS	TSP	VRP	Total
1	AdapHH	35.75	32	13.5	42	45	15	183.25
2	ML	14.5	12	31	37	13	22	129.5
3	VNS-TW	33.25	3	37.5	32	16.33	6	128.083
4	PHunter	10.5	4	11.5	9	25.33	33	93.33
5	EPH	0	11	10.5	20	33.33	12	86.83
6	NAHH	14	22	2	22	12	6	78
7	ISEA	6	34	14.5	3.5	12	5	75
8	HAHA	32.75	0	24.5	3.5	0	14	74.75
9	KSATS-HH	24	11	9.5	0	0	22	66.5
10	HAEA	0.5	3	2	10	11	27	53.5
11	ACO-HH	0	23	0	9	8	2	42
12	GenHive	0	14	7	7	3	6	37
13	DynILS	0	14	0	0	13	1	28
14	SA-ILS	0.75	0	19.5	0	0	4	24.25
15	XCJ	5.5	12	0	0	0	5	22.5
16	AVEG-Nep	12	0	0	0	0	9	21
17	GISS	0.75	0	8	0	0	6	14.75
18	SelfSearch	0	0	4	0	3	0	7
19	MCHH-S	4.75	0	0	0	0	0	4.75
20	Ant-Q	0	0	0	0	0	0	0

(b)

Rank	Name	SAT	BP	PS	FS	TSP	VRP	Total
1	AdapHH	34.75	31	14	39	44	13	175.75
2	VNS-TW	34.25	3	37.5	34	16.33	6	131.083
3	ML	14.5	12	31	38	13	22	130.5
4	PHUNTER	10.5	4	11.5	9	25.33	32	92.33
5	EPH	0	11	10.5	21	33.33	12	87.83
6	ISEA	6	34	14.5	3.5	14	6	78
7	NAHH	14	22	2	21	11	6	76
8	HAHA	32.75	0	24.5	3.5	0	13	73.75
9	KSATS-HH	24	11	9.5	0	0	22	66.5
10	HAEA	0.5	4	2	10	11	27	54.5
11	ACO-HH	0	23	0	9	8	3	43
12	GenHive	0	14	6.5	7	3	6	36.5
13	DynILS	0	14	0	0	13	1	28
14	SA-ILS	0.75	0	19.5	0	0	4	24.25
15	XCJ	5.5	12	0	0	0	5	22.5
16	AVEG-Nep	12	0	0	0	0	9	21
17	GISS	0.75	0	8	0	0	8	16.75
18	SelfSearch	0	0	4	0	3	0	7
19	MCHH-S	4.75	0	0	0	0	0	4.75
20	Ant-Q	0	0	0	0	0	0	0

Interestingly the results of 31 independent runs of *AdapHH* yield slightly different results to those achieved in CHeSC2011, with 183.25 points scored overall compared to 181 in the original competition. This has the knock-on effect that the second placed competitor in the original competition *VNS-TW* [16], is now third having been overtaken by *ML*. Comparing the results of Table IV it can be seen that the original crossover management scheme of *AdapHH* is able to yield better results than modifying the crossover management scheme to the method described in Drake et al. [13]. The original *AdapHH* outperforms the version with a modified crossover management scheme in all problem domains with the exception of personnel scheduling. There does not appear to be a significant difference in general, with both hyper-heuristics retaining first place when compared to the other 19 competition entrants.

Fig. 3: Formula One points scored by each crossover control mechanism within *AdapHH* compared to CHeSC2011 competitors

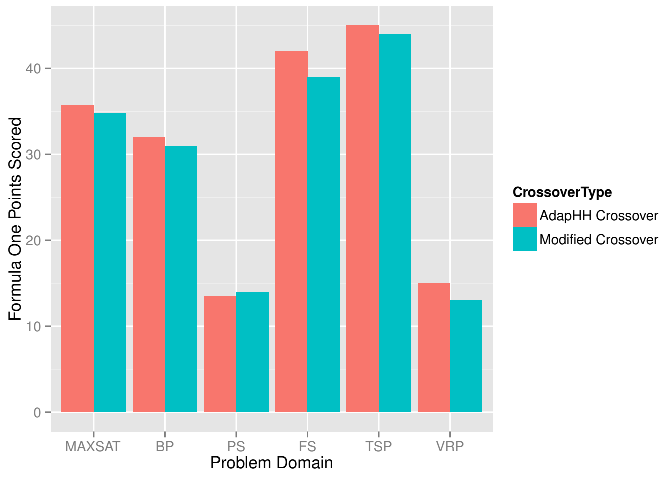


TABLE V: Pairwise comparison of *AdapHH* with differing crossover control schemes using independent Student's t-test

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

As a direct comparison, Table V shows the results of an independent Student's t-test within a 95% confidence interval on the objective function values for 31 runs of each instance. 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 $\gg$ show the number of cases that the original *AdapHH* is outperforming performing *AdapHH* with a modified crossover control scheme on average or statistically significantly respectively. Conversely, $<$ and $\ll$ denote the number of cases which the *AdapHH* with modified crossover control scheme outperforms the original *AdapHH* on average or statistically significantly.

In terms of average performance, the original *AdapHH* outperforms *AdapHH* with modified crossover management in 18 of the 30 instances. There are very few cases in which the difference is statistically significantly different. A notable case is the vehicle routing problem where it is possible for both *AdapHH* hyper-heuristics to statistically significantly outperform each other in different instances. If it is indeed the case that the benefit of crossover varies on a per instance basis, this effectively reduces the size of each problem class for which crossover is useful to one, rendering generalisation irrelevant. It could also be the case that in the problem domains where crossover is useful in some cases but not others, there is a subset of instances which share certain features for which the use of crossover leads to gains in performance. This raises a larger question regarding the tuning of hyper-heuristics, particularly in an offline manner. If in advance of a full run, a hyper-heuristic is made aware of instance specific properties (e.g. size, nature of the search landscape), it would be possible to make choices regarding which hyper-heuristic components or parameter settings to use in different cases. A counter argument to this type of tuning is that in effect, as the hyper-heuristic is able to distinguish between problem domains based on these properties and therefore alter behaviour for different domains, the domain barrier is effectively broken.

The same argument can be made regarding offline tuning based on the set of low-level heuristics available, using properties such as heuristic type (i.e. mutation, crossover etc.) or expected runtime. As an example, in the case of the personnel scheduling domain the expected runtime of each low-level heuristic is exceptional in length compared to the other problem domains in HyFlex. Some of the competitors to CHeSC2011 take advantage of this fact, modifying behaviour at execution time based on the observed runtimes of low-level heuristics, often through schemes designed based on previous experience. This is a slippery slope when the intention is to develop general methods which are able to perform well over a large number of domains. If the behaviour of a hyper-heuristic is adjusted in such a way for each available problem domain, potentially it is reduced to a set of if-then clauses, with individual behaviours specified for each problem domain. This reduces the hyper-heuristic design process to a software engineering task, with the intention of 'winning' a competition, rather than focusing on algorithm design for multiple problem domains.

A further complication is the relatively small number of problem domains being considered. In some cases the choice of whether or not to include crossover operators at all will affect performance, however in others it can be affected by the method used to manage the inputs for such operators. In a more general sense, using only six problem domains could be seen as a very small sample. It is accepted that in order to assess the effectiveness of a stochastic method, multiple runs are needed to provide a reasonable picture of average performance. It is also the case that to assess how effective a method is in a given problem domain, a reasonable size sample of problem instances should be tested to average performance. Therefore it could be considered unreasonable to expect any generalisations made using these benchmarks to hold true if there were 50 or 5000 problem domains.

V. Conclusions

In this paper we have compared the hyper-heuristic level crossover control scheme introduced by Drake et al. [13] with the crossover management scheme of the CHeSC2011 winner, *AdapHH*. This was done in two ways, firstly the crossover control scheme defined by Drake et al. [13] was replaced with the crossover management scheme of *AdapHH* within a *Modified Choice Function - All Moves* hyper-heuristic. Secondly, this crossover control scheme was used within *AdapHH*, replacing the existing crossover control scheme defined for this hyper-heuristic. When comparing different crossover control schemes, it becomes clear that there are some domains or instances for which the choice of crossover control scheme is crucial in determining performance. Unfortunately despite having a large impact in some instances within a *Modified*

Choice Function - All Moves hyper-heuristic previously, the choice of crossover management scheme does not make a great deal of difference in the case of *AdapHH*. This provides a challenge when trying to generalise certain hyper-heuristic components, such as the management of solutions for n-ary operators, as dependencies clearly exist between different hyper-heuristic components. Although these conclusions may seem inconclusive, this is somewhat a result of performing such a wide study over multiple domains. If a single domain was studied, it may have been possible to derive a stronger conclusion however this would not have survived future studies. Future work will focus on identifying which situations certain crossover control schemes are effective in, and adjusting the method used based on the requirements of the problem currently under consideration.

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. Gendreau, M. Hyde, G. Kendall, B. McCollum, G. Ochoa, A. J. Parkes, and S. Petrovic, "The cross-domain heuristic search challenge - an international research competition," in *Proceedings of Learning and Intelligent Optimization (LION 2011)*, ser. LNCS, C. A. C. Coello, Ed., vol. 6683. Rome, Italy: Springer, 2011, pp. 631–634.

[3] E. K. Burke, E. Hart, G. Kendall, J. Newall, P. Ross, and S. Schulenburg, "Hyper-heuristics: An emerging direction in modern search technology," in *Handbook of Metaheuristics*, F. Glover and G. Kochenberger, Eds. Kluwer Academic Publishers, 2003, ch. 16, pp. 457–474.

[4] 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.

[5] 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.

[6] 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.

[7] B. Doerr, E. Happ, and C. Klein, "Crossover can provably be useful in evolutionary computation," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2008)*, C. Ryan and M. Keijzer, Eds. Atlanta, Georgia, USA: ACM, 2008, pp. 539–546.

[8] ——, "Crossover can provably be useful in evolutionary computation," *Theoretical Computer Science*, vol. 436, pp. 71–86, 2012.

[9] J. H. Drake, M. Hyde, K. Ibrahim, and E. Özcan, "A genetic programming hyper-heuristic for the multidimensional knapsack problem," *Kybernetes*, vol. 43, no. 9-10, pp. 1500–1511, 2014.

[10] 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*, 2015. [Online]. Available: http://dx.doi.org/10.1162/EVCO_a_00145

[11] ——, "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.

[12] ——, "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.

[13] ——, "A modified choice function hyper-heuristic controlling unary and binary operators," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2015)*, 2015.

[14] 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.

[15] J. Gibbs, G. Kendall, and E. Özcan, "Scheduling english football fixtures over the holiday period using hyper-heuristics," in *Proceedings of Parallel Problem Solving from Nature (PPSN 2011)*, ser. LNCS, R. Schaefer, C. Cotta, J. Kolodziej, and G. Rudolph, Eds., vol. 6238. Kraków, Poland: Springer, 2011, pp. 496–505.

[16] 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.

[17] 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.

[18] B. Kiraz, A. Ş. Etaner-Uyar, and E. Özcan, "An ant-based selection hyper-heuristic for dynamic environments," in *Proceedings of the International Conference on the Applications of Evolutionary Computation (EvoApplications 2013)*, ser. LNCS, A. I. Esparcia-Alcázar, Ed., vol. 7835. Vienna, Austria: Springer, 2013, pp. 626–625.

[19] M. Misir, "Intelligent hyper-heuristics: A tool for solving generic optimisation problems," Ph.D. dissertation, Department of Computer Science, KU Leuven, 2012.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.