An Investigation of Tuning a Memetic Algorithm for Cross-domain Search

Düriye Betül Gümüş, Ender Özcan, Jason Atkin
The University of Nottingham,
School of Computer Science,
ASAP Research Group,UK
Email: {betul.gumus,ender.ozcan,jason.atkin}@nottingham.ac.uk

***Abstract*—Memetic algorithms, which hybridise evolutionary algorithms with local search, are well-known metaheuristics for solving combinatorial optimisation problems. A common issue with the application of a memetic algorithm is determining the best initial setting for the algorithmic parameters, but these can greatly influence its overall performance. Unlike traditional studies where parameters are tuned for a particular problem domain, in this study we do tuning that is applicable to cross-domain search. We extend previous work by tuning the parameters of a steady state memetic algorithm via a 'design of experiments' approach and provide surprising empirical results across nine problem domains, using a cross-domain heuristic search tool, namely HyFlex. The parameter tuning results show that tuning has value for cross-domain search. As a side gain, the results suggest that the crossover operators should not be used and, more interestingly, that single point based search should be preferred over a population based search, turning the overall approach into an iterated local search algorithm. The use of the improved parameter settings greatly enhanced the cross-domain performance of the algorithm, converting it from a poor performer in previous work to one of the stronger competitors.**

I. Introduction

Memetic algorithms are well-known and highly effective population-based metaheuristics. They have been applied to a range of combinatorial optimisation problems, including educational timetabling [2], [21], parallel code optimisation [22], quadratic assignment [16], vehicle routing [18], the travelling salesman problem [13], protein structure prediction [12] and knapsack problems [10]. Parameter tuning, i.e. determining the best configuration for the algorithmic parameters, is crucial for improved metaheuristic performance. There are various parameter tuning methods available, including 'design of experiments' as well as automated approaches, which have already been applied to memetic algorithms [25], [8], [7]. Many studies which tuned the algorithm parameters focused on one problem domain and tailored the algorithm for that domain. This study considers the tuning of a memetic algorithm not for a single specific domain but across multiple domains using a 'design of experiments' approach, namely the Taguchi method [24]. It is known that parameter tuning has huge value within single domain. Our aim is to investigate whether similar tuning also has value for cross-domain search. To this end, parameter tuning experiments are performed using two sample instances from four problem domains in a training phase. The best setting obtained from the Taguchi method is then generalised for cross-domain search and used to solve the unseen instances from nine different domains. The best parameter setting is analysed further in order to verify the validity of the generality of the best setting which was detected via tuning.

There is a growing number of studies on *hyper-heuristics*, operating on a search space of heuristics (or heuristic components) rather than directly on the search space of potential solutions [5]. Selection hyper-heuristics perform improvement oriented iterative single point based search by managing and mixing a set of low level heuristics. One of the interfaces used in this area is referred to as HyFlex (Hyper-heuristics Flexible framework) [19], which provides a framework enabling the rapid development of general purpose metaheuristics, particularly selection hyper-heuristics for research. The initial version of HyFlex was implemented in Java and used in the first Cross-Domain Heuristic Search Challenge (CHeSC2011) to detect the best selection hyper-heuristic across multiple domains [4].

In this study, we have used HyFlex, which provides all of the genetic operators required by an evolutionary algorithm [20], for the implementation and performance evaluation of a memetic algorithm. The proposed memetic algorithm is a population based stochastic local search algorithm which can be considered as a hyper-heuristic, as it does not use any domain specific information and mixes multiple operators.

Özcan et al. [23] implemented a trans-generational and steady state memetic algorithm and investigated their performances across six problem domains. They applied a basic sequence of experiments for parameter tuning. The steady state approach was reported to perform better, however single point based search hyper-heuristics performed significantly better than memetic algorithms. In this study, we extend the previous work in [23] performing two sets of experiments. We have partitioned forty five instances across nine problem domains into training and test instances for each experiment. The problem instances consist of thirty instances from the six problem domains used at CHeSC2011 and fifteen instances from the three extended HyFlex problem domains as provided by Adriaensen et al. [1]. Firstly, Taguchi's design of experiments approach has been employed to tune the parameters of the steady state memetic algorithm from [23] considering 25 different configurations for overall performance improvement. Two training instances from each of the four HyFlex problem

domains, which were initially provided as public domains at CHeSC2011, are used during the tuning process. Then, the tuned steady state memetic algorithm with the best parameter setting is applied to all other 'unseen' instances across nine problem domains. The results show that the tuned steady state memetic algorithm delivers a significantly better performance for most of the instances. The parameter setting obtained through the tuning process generalises well from the limited number of training instances to unseen instances, even from the unseen domains. Moreover, surprisingly, the tuning process detects that crossover should not be used and single point based search should be preferred rather than population based search.

The subsequent sections of this paper are organised as follows. Section II describes HyFlex, including the details of the problem domains and CHeSC2011. Our methodology is discussed in Section III. The experimental results and analysis are presented in Section IV. Finally, we summarise our findings and explain our future work in Section V.

II. Hyper-Heuristics Flexible Framework (HyFlex)

Hyper-heuristics Flexible Framework (HyFlex) is an interface proposed to enable the development, testing and comparison of both single point and population-based meta/hyper-heuristics across different combinatorial optimisation problems [19]. The objective of the initial Java version of the software is to help algorithm designers to develop general purpose heuristic optimisation algorithms without requiring problem domain expertise. In HyFlex, the high-level method and problem domain layers are separated via a *domain barrier* [6]. Due to this barrier, the hyper-heuristic does not have access to problem specific information, such as the solution representation or details of the changes that a low-level heuristic will actually make/has made to the solution. However, problem independent information, such as the fitness (cost) value, can be passed to the algorithm [4].

HyFlex was used for the first Cross-domain Heuristic Search Challenge (CHeSC2011). HyFlex supported six problem domains for the CHeSC2011 competition, namely Maximum Satisfiability (SAT), One Dimensional Bin Packing (BP), Permutation Flow Shop (PFS), Personnel Scheduling (PS), Traveling Salesman Problems (TSP) and Vehicle Routing Problems(VRP). Each domain contains a number of instances and problem specific components. Four of those domains (SAT, BP, PFS and PS) were made available to the participants when the competition started, while the other two domains were hidden.

HyFlex has four types of low-level heuristics (operators) to modify solutions, namely, *mutational (MU)*, *ruin and re-create (RR)*, *crossover (XO)* and *local search (LS)* [19]. Apart from crossover, all of the other low level heuristics are parameterised. For the mutation and ruin and re-create operators, the *intensity of mutation* parameter determines the extent of changes that the corresponding operator will make to the input solution. The *depth of search* parameter controls the number of steps that the local search heuristic will complete. Both of these parameters take values in the interval [0,1], thus the mutation, ruin and re-create and local search heuristics require parameter tuning.

TABLE I: The number of different types of low level heuristics for 9 problem domains

Domain	MU	RR	XO	LS	Total
SAT	6	1	2	2	11
BP	3	2	1	2	8
PS	1	3	3	5	12
PFS	5	2	4	4	15
TSP	5	1	4	3	13
VRP	3	2	2	3	10
0-1 KP	5	2	3	6	16
CUT	2	3	2	3	10
QAP	2	3	2	2	9

Adriaensen et al. [1] extended the HyFlex benchmark set with the addition of 3 problem domains after CHeSC2011: 0-1 Knapsack (0-1 KP), Max-Cut (CUT), and Quadratic Assignment (QAP). All of the extended set of HyFlex domains have been formulated as minimisation problems. In this study, we have used all nine domain implementations in our experiments. The number of low-level heuristics of each type in each domain is provided in Table I.

Twenty selection hyper-heuristics competed at CHeSC2011. The winning hyper-heuristic, with an overall score of 181, was AdapHH [17], which applies an adaptive heuristic selection combined with adaptive threshold move acceptance. A subset of heuristics are used and this subset is determined adaptively. Moreover, a low level heuristic from this set is chosen using an online learning mechanism. Then, a move acceptance method called "adaptive list-based threshold accepting mechanism" is used to decide whether to accept or reject the new solution at each step. This algorithm also determines the heuristic parameter settings, adaptively. For more information about CHeSC2011, including the tools used in the competition, a tutorial on those tools, details of the competing hyper-heuristics, and the results, readers can refer to the CHeSC2011 website[1].

Adriaensen et al. [1] applied six selection hyper-heuristics to all 9 of the extended HyFlex domains. Two of those hyper-heuristics were taken from CHeSC2011, namely Adap-HH and EPH (ranking the 5^{th}), which are publicly available. The remaining hyper-heuristics were proposed by the authors. FS-ILS follows the iterated local search logic, applying firstly a perturbative heuristic and then a local search heuristic in a tabu-portfolio. The solution is then accepted or rejected according to the acceptance criteria. Finally, the algorithm has two choices: either performing a new iteration, or restarting the search. The second algorithm is the same but without restarting and is labelled as NR-FS-ILS. They also provided 2 simple single point based search selection hyper-heuristics: AA-HH and ANW-HH. These randomly choose a heuristic in the categories of mutation, ruin-recreate and local search

[1] http://www.asap.cs.nott.ac.uk/external/chesc2011/

Algorithm 1 : Pseudocode of Steady-state Memetic Algorithm

```
Create a population of popsize random individuals
Set the parameter values for Intensity of Mutation (IoM),
Depth of Search (DoS) and tournament size (toursize)
Apply a random local search method (hill climbing) to each individual
while (termination criterion is not satisfied)
  Parent1 ← Select-Parent(population, toursize)
  Parent2 ← Select-Parent(population, toursize)
  Child   ← ApplyCrossover (Rand(1, MAX_CROSSOVER), Parent1,
Parent2)
  Child   ← ApplyMutation (Rand(1, MAX_MUTATION), IoM, Child)
  Child   ← ApplyLocalSearch(Rand(1, MAX_LOCALSEARCH), DoS,
Child)
  WorstOf(population) ← Child
end while
```

and apply that heuristic to the solution. All of the new solutions are accepted in AA-HH while only non-worsening solutions are accepted in ANW-HH. In this paper we compare the performance of our approach on the additional HyFlex domains to that of these 6 hyper-heuristics.

III. Methodology

In this study, we used a steady state memetic algorithm to solve a range of problems supported by HyFlex. The steady state approach iteratively creates new individuals and considers replacement of an individual from the population with the new one [27].

The steady state memetic algorithm is implemented based on HyFlex, utilising the mutation and crossover heuristics as genetic operators and the local search heuristics that already exist within the framework for each domain. Ruin and re-create heuristics are considered as mutation heuristics in this study, increasing the available number of mutation operators. The pseudocode of our approach for solving instances from each HyFlex domain is given in Algorithm 1 [23].

The Taguchi orthogonal arrays experimental design method is employed for parameter tuning to decide the most appropriate combination of parameter values for the steady state memetic algorithm. The following steps are executed [24]:

1) Planning of experiments
   - Determine the control parameters (design factors)
   - Determine the levels of each control parameter
   - Select a suitable orthogonal array based on the number of parameters and their levels, forming an experimental design table
2) Conduct the experiments based on the design table
3) Analyse the results and determine the optimum levels for the individual control parameters
4) Perform a confirmation run using a combination of the optimum individual parameter levels

In this study, the parameter combinations consist of four control parameters: *population size*, *intensity of mutation* (used for the mutation and ruin and re-create heuristics), *depth of search* (used for the local search heuristics) and *tour size* for the tournament selection to choose parents for operators. These are problem independent parameters and common across all of the problem domains which respect the HyFlex API. The potential values (levels) for each parameter are listed in Table II. Testing all of the combinations for all settings would mean testing 500 configurations, however, this has been reduced to testing 25 settings using the L_{25} Taguchi orthogonal array.

TABLE II: Parameter setting options used during the parameter tuning of the steady state memetic algorithm

Parameters	Value Options
Intensity of Mutation (IoM)	{0.2, 0.4, 0.6, 0.8, 1.0}
Depth of Search (DoS)	{0.2, 0.4, 0.6, 0.8, 1.0}
Population size (PopSize)	{5, 10, 20, 40, 80}
Tournament size (TourSize)	{2, 3, 4, 5}

In related work, Sun [26] used the same technique for parameter tuning of a genetic algorithm to solve a job shop scheduling problem. In that study, parameters, including population size, crossover rate, mutation rate and stopping condition, were considered as design factors. The differences between our study and that of Sun [26] are not only that we use different design factors, but, importantly, that we also tune the parameters for cross-domain search rather than for a particular domain.

IV. Experimental Results

In this study, two sets of experiments are performed with the goal of (i) obtaining the best parameter setting for the steady state memetic algorithm, and (ii) assessing the generalisation capability of the setting obtained from the training phase to see whether parameter tuning has value for cross-domain search. In the first set of experiments, parameter tuning of the steady state memetic algorithm is performed. 25 different parameter settings, indicated by the L_{25} Taguchi orthogonal array, were tested to obtain the best configuration for the steady state memetic algorithm, using two training instances from each of the four HyFlex problem domains, which were accessible prior to CHeSC2011. Then the best setting obtained through the tuning process as well as the same 25 settings were tested using the unseen instances from each of the 9 domains.

A. Experimental Settings

Five problem instances from each of the nine HyFlex domains, including the original six and additional three problem domains, are employed during the experiments. The same competition instances as in CHeSC2011 are used for the original six domains, while arbitrarily chosen instances with IDs 1, 3, 5, 7 and 9 are used from the three extended HyFlex problem domains.

All experiments were performed on an Intel Core 3.60 GHz machine with 16 GB RAM. According to the benchmarking tool which the organisers of CHeSC2011 provided, 415 seconds on that machine are equivalent to the 600 nominal seconds used on the competition machines, so this time limit was used as the termination criterion for the algorithm, to ensure a fair comparison.

Each trial was repeated 31 times, the same as for the CHeSC2011 competition rules. The competition used 'Formula 1' scoring, where the algorithms were assessed according

TABLE III: Formula 1 score for each steady state memetic algorithm run, with a particular parameter configuration based on the L_{25} Taguchi orthogonal array.

Experiment number	IoM	DoS	Pop size	Tour size	Score
1	0.2	0.2	5	2	24.88
2	0.2	0.4	10	3	14
3	0.2	0.6	20	4	17.50
4	0.2	0.8	40	5	17.88
5	0.2	1.0	80	3	15.50
6	0.4	0.2	10	4	9.88
7	0.4	0.4	20	5	2.50
8	0.4	0.6	40	2	8.88
9	0.4	0.8	80	2	22.50
10	0.4	1.0	5	3	27.25
11	0.6	0.2	20	5	0
12	0.6	0.4	40	2	1
13	0.6	0.6	80	3	12
14	0.6	0.8	5	4	24.38
15	0.6	1.0	10	5	22.25
16	0.8	0.2	40	3	0
17	0.8	0.4	80	4	25.88
18	0.8	0.6	5	5	30.88
19	0.8	0.8	10	3	16.25
20	0.8	1.0	20	2	18.38
21	1.0	0.2	80	5	4
22	1.0	0.4	5	4	0
23	1.0	0.6	10	2	1.88
24	1.0	0.8	20	3	1.88
25	1.0	1.0	40	4	2.50

to the median objective values from 31 trials for each instance, and points were awarded to only the top eight algorithms for each of the instances. 10, 8, 6, 5, 4, 3, 2 and 1 point(s) were allocated for first, second, third, fourth, etc, to eighth place algorithm, respectively. The same scoring system was utilised here as a performance indicator for the algorithms and configurations tested. Ties were broken by taking the mean of the scores that the tied algorithms would take for the tied positions and assigning that value as the score for each of the tied algorithms. e.g., algorithms tied for first and second place would each get a score of 9 (i.e., the mean of 10 and 8). At the end, the sum of the scores awarded to each algorithm across all of the instances was totalled to obtain the final score.

B. Parameter Tuning

25 combinations of parameter settings for the 'intensity of mutation', 'depth of search', 'population size' and 'tournament size' of the steady state memetic algorithm were tested on 8 'training' instances, consisting of 2 arbitrarily chosen instances from 4 HyFlex domains. The L_{25} orthogonal array was used to determine the parameter values for each experiment. Since the tour size has 4 levels (settings/values), one of the possible settings was assigned randomly in place of the $5th$ level. The parameter setting and total score of each setting calculated using Formula 1 scoring system are given in Table III.

Using the results from this initial set of experiments, the average effect of each parameter setting was computed. The score of all steady state memetic algorithms with each parameter setting is averaged over the number of configurations with the same particular parameter setting. For example, the average effect of a population size of 20 (at Level 3) is calculated as $(17.5 + 2.5 + 0 + 18.38 + 1.88)/5 = 8.052$, since the number of configurations is 5 out of 25. The main effects plot summarising the average effect of each parameter is provided in Figure 1. Since a higher Formula 1 score would be preferred, the parameter setting that has the highest value would be predicted as the best setting for each parameter from the main effects plot. Hence, the best configuration for the steady state memetic algorithm parameters is found to be 0.2 for IoM, 1.0 for DoS and 5 for popsize. The values 2 and 5 have the same average effect for toursize parameter. However, the toursize of 5 using the best setting for the remaining parameters performs better than 2 resulting with a Formula 1 score of 75 as opposed to 69, so the toursize of 5 is included as a part of the best configuration for the steady state memetic algorithm.

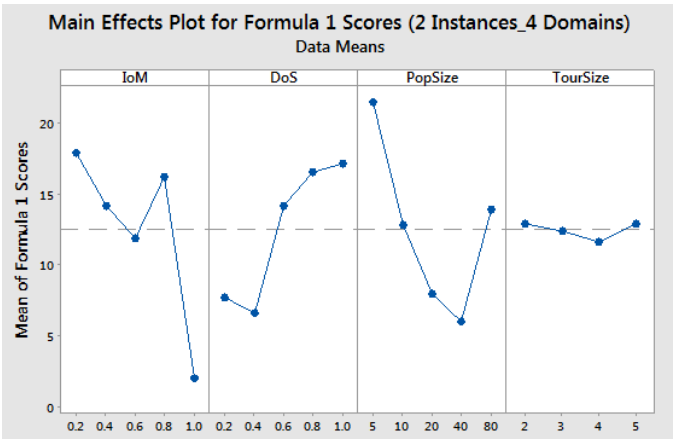

Fig. 1: Main effects plot with mean Formula 1 scores based on 2 training instances from 4 domains

Anova analysis was performed with the total normalised mean objective values over 31 runs for each instance to determine the significant factors and percentage contribution of each factor. Table IV shows that the population size and depth of search parameters significantly contribute to the performance with a confidence level of 95%. Depth of search has the highest percentage contribution of 50.68%, followed by population size, intensity of mutation and tour

TABLE IV: Anova test results for identifying the contribution of each parameter (DoF: degrees of freedom, SS: sum of squares, MS: mean squares, F: variance ratio).

Parameters	DoF	SS	MS	F	p-value	Percent cont.
IoM	4	3.25	0.81	3.04	0.0766	14.65
DoS	4	11.26	2.81	10.51	0.0019	50.68
PopSize	4	4.54	1.13	4.24	0.0336	20.42
TourSize	3	0.76	0.25	0.94	0.4601	3.40
Residual	9	2.41	0.27			10.85
Total	24	22.22	5.27			100%

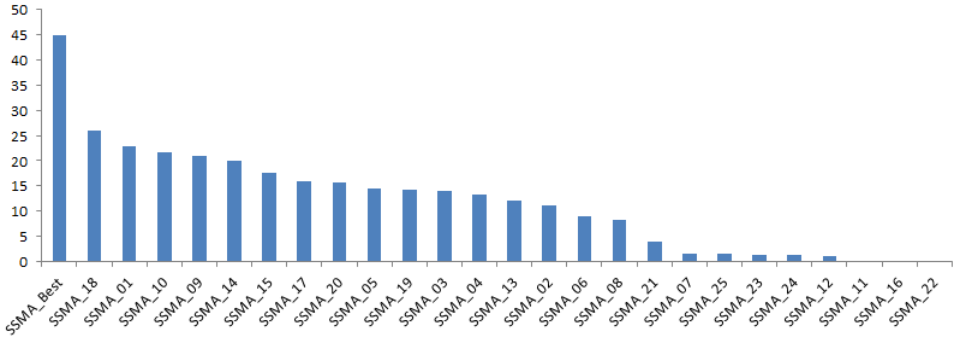

Fig. 2: Performance comparison of each steady state memetic algorithm configuration run on two selected instances from each of the four public HyFlex domains based on their Formula 1 scores.

size respectively.

A confirmation test, using the best parameter values, was performed on the same training instances. The purpose of this validation step is to verify that the optimum parameter values obtained from the Taguchi method provide improvement over the other tested settings. We put all of the 25 configurations along with the best configuration obtained through the Taguchi method into a competition and evaluated them using the Formula 1 scoring system. The results are illustrated in Figure 2. The tuned steady state memetic algorithm with the best identified parameter setting outperforms all of the other settings with a score of 45, followed by the setting S18 with a score of 25.88.

C. Parameter Tuning Validation with Unseen Instances

The Taguchi method was applied using all fourty five instances across nine problem domains considering 25 parameter settings in order to assess the generality of the discovered best setting, and to validate the outcome of the parameter tuning experiments. Figure 3 shows the main effects plot of the parameter levels based on the experimental results. As illustrated, the best parameter combination obtained via tuning on all instances across nine domains is consistent, being 0.2 for IoM, 1.0 for DoS, 5 for popsize and 5 for toursize. This is exactly the parameter configuration which was identified using the training instances. This gives evidence for parameter tuning with the Taguchi method having value for cross-domain search.

D. Performance Comparison of SSMA-Best to Previously Proposed SSMA and TGMA

The results obtained from SSMA with the best parameter setting denoted as *SSMA–Best* are here compared to the memetic algorithms, denoted as SSMA and TGMA from Özcan et al. [23]. The parameter settings for both SSMA and TGMA in [23] were: (i) population size of 10, (ii) tour size for tournament selection of 2, and (iii) intensity of mutation as well as depth of search of 0.2. Table V shows the mean, median and best fitness values for SSMA-Best, SSMA, TGMA obtained for each CHeSC2011 instance. The same instances

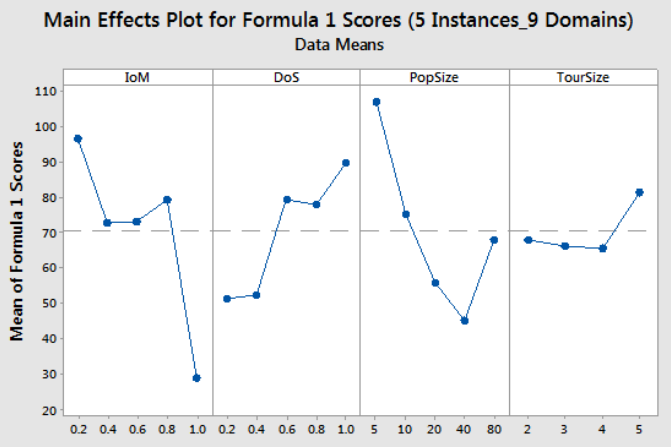

Fig. 3: Main effects plot with mean Formula 1 scores based on five instances from each of the nine HyFlex domains.

which were used in the competition and in [23] are used here, for a fair comparison.

Table V provides a comparison of the performance of SSMA-Best compared to SSMA and TGMA. The "vs." column of this table shows the results of paired one-tail Wilcoxon signed ranked test results. "$<$" or "$>$" indicate that the result on the left is "worse" or "better", respectively, than the result to the right and that this performance difference is statistically significant within a 95% confidence interval. "$\leq$" or "$\geq$" indicate that the difference is slightly worse or better, respectively, but that the difference is not statistically significant. For example, SSMA-Best has a better performance than SSMA for Instance 1 of SAT problem and this performance difference is statistically significant; however, SSMA-Best has a slightly better performance than TGMA for Instance 2 of VRP but this performance difference is not statistically significant.

Considering the mean performance, SSMA-Best performs better than SSMA in 27 out of the 30 CHeSC2011 instances and 26 of those cases have a statistically significant performance difference. There are only 3 cases where SSMA performs better than SSMA-Best. Similarly, SSMA-Best performs better than TGMA in 28 CHeSC2011 instances and in 25 of

TABLE V: Performance comparison of SSMA, SSMA-Best and TGMA based on mean, median and best fitness values obtained from 31 trials for each instance of CHeSC2011. The algorithm achieving the best fitness is marked in bold style for each instance.

		SSMA				SSMA-Best				TGMA		
PD	ID	mean	median	best	vs.	mean	median	best	vs.	mean	median	best
SAT	1	21.161	22	8	$<$	**9.484**	**9**	**3**	$>$	14.323	14	**3**
	2	52.484	53	37	$<$	**30.065**	**36**	**9**	$>$	41.806	45	11
	3	35	37	10	$<$	**15.387**	**10**	**3**	$>$	27.129	31	5
	4	27.742	27	26	$<$	**13.613**	**13**	**9**	$>$	20.226	19	13
	5	19.194	18	14	$<$	**12.742**	**13**	**9**	$>$	17.29	17	13
BP	1	0.083	0.082	0.074	$<$	**0.063**	**0.063**	**0.058**	$>$	0.075	0.074	0.066
	2	0.015	0.013	0.011	$<$	**0.011**	**0.012**	0.008	$\leq$	0.012	**0.012**	**0.007**
	3	0.022	0.022	0.018	$>$	0.034	0.034	0.029	$<$	**0.02**	**0.02**	**0.016**
	4	0.1115	0.1112	0.1105	$<$	**0.1102**	**0.11**	**0.1098**	$>$	0.1108	0.1106	0.1099
	5	0.043	0.041	0.036	$>$	0.061	0.06	0.054	$<$	**0.037**	**0.037**	**0.032**
PS	1	51.129	50	37	$<$	**21.548**	**21**	**16**	$>$	66.419	66	50
	2	72015.613	70477	52056	$<$	**9705.129**	**9677**	**9477**	$>$	70356.16	69061	59736
	3	13203.71	11859	5581	$<$	**3211.452**	**3219**	**3146**	$>$	14028.71	12338	10027
	4	2942.419	2655	1820	$<$	**1596.871**	**1589**	**1344**	$>$	2923.71	2550	2003
	5	435.645	431	385	$<$	**318.065**	**315**	**290**	$>$	498.032	490	430
PFS	1	6257.806	6258	6231	$<$	**6249.581**	**6251**	**6219**	$>$	6301.968	6303	6273
	2	26884	26884	26813	$<$	**26812.452**	**26811**	**26754**	$>$	26944.55	26945	26889
	3	6351.871	6363	6318	$<$	**6336.387**	**6333**	**6303**	$>$	6366.677	6369	6342
	4	11441.806	11441	11410	$<$	**11376.613**	**11375**	**11333**	$>$	11499.65	11502	11458
	5	26699.226	26703	26626	$<$	**26632.548**	**26640**	**26515**	$>$	26724	26725	26668
TSP	1	48227.747	**48194.92**	**48194.92**	$\leq$	**48221.583**	**48194.92**	**48194.92**	$>$	48337.6	48286.76	**48194.92**
	2	21155458.17	21160875.64	20969185.56	$<$	**20912006.397**	**20885233.01**	**20789116.98**	$>$	21304532.63	21313786.27	21146401.19
	3	6825.552	6825.663	6800.708	$<$	**6811.145**	**6811.518**	**6799.111**	$>$	6893.822	6893.809	6858.803
	4	68123.369	68059.971	67423.655	$<$	**67029.810**	**67043.255**	**66518.735**	$>$	69778.14	69811.893	68922.954
	5	53810.138	53748.537	52685.992	$<$	**53503.576**	**53457.422**	**52247.568**	$>$	55463.6	55463.992	54052.398
VRP	1	**71768.053**	71480.081	67820.589	$\leq$	71946.305	**70776.497**	**65967.938**	$>$	76331.07	76181.913	71560.119
	2	14324.522	14411.658	13358.611	$<$	**13826.295**	**13384.024**	**13328.791**	$\geq$	13869.19	13411.284	13333.091
	3	176206.081	177131.584	167704.512	$<$	**147512.686**	**148001.434**	**143921.208**	$>$	200838.2	201457.298	193416.831
	4	21647.018	21675.195	20678.096	$<$	**21275.493**	**21648.051**	**20654.219**	$\geq$	21412.57	21659.585	20659.896
	5	152642.04	152829.839	149032.551	$<$	**147372.783**	**147228.056**	**145266.409**	$>$	157001.3	157136.619	153557.35

those cases this performance difference is significant. There are 2 instances for which TGMA performs better than SSMA-S18.

SSMA-Best achieves the best fitness values for 27 out of 30 instances (being joint best in one case). SSMA performs the best for Instance 1 of TSP (joint best) and TGMA finds the best fitness values for Instances 2, 3, 5 of BP and Instance 1 of SAT and TSP (joint best).

Overall, SSMA-Best was the best performing configuration for cross-domain combinatorial optimisation (Table V). SSMA-Best was the most successful approach, particularly for SAT, PS and PFS, significantly outperforming the others on all instances. However, it did not perform well on BP.

E. Performance Comparison of Memetic Algorithms to the CHeSC2011 Competitors

The performances of SSMA, TGMA and SSMA-Best are here compared to the selection hyper-heuristics which competed in CHeSC2011, based on the Formula 1 scoring which was used there. According to the study in [23], SSMA and TGMA were among the lowest ranking algorithms, with scores of 7.5 and 0, respectively.

Figure 4 provides the relative ranking of these 3 algorithms with respect to the previously described selection hyper-heuristics from the competition.

SSMA-Best ranked 4^{th} overall among the 23 algorithms while SSMA and TGMA ranked 19^{th} and 23^{nd}, respectively. Even though SSMA-Best and SSMA have the same pseudocode, the performance improvement that was obtained from parameter tuning is remarkable. Indeed, when the total score was considered for each of the domains independently, SSMA-Best was the 2^{nd} for TSP and PS. SSMA-Best performed particularly poorly in the SAT and BP domains, compared to the competing hyper-heuristics, receiving no points for those domains. The overall score for SSMA-Best was 103.

F. Performance Comparison of SSMA-Best to Previously Proposed Hyper-heuristics on Additional HyFlex Domains

In order to further assess the performance and the level of generality of SSMA-Best, it was tested on the 3 additional HyFlex problem domains[2] provided by Adriaensen et al. [1]. The authors compared the performance of six selection hyper-heuristics in their study. Two of these hyper-heuristics were taken from the competition, namely Adap-HH and EPH. The rest of the hyper-heuristics were proposed by the authors.

Table VI summarises the results and gives a performance comparison of SSMA-Best to the hyper-heuristics in [1], based on the median fitness values for each instance. For the 0-1 KP domain, AdapHH, which was the winning algorithm at

[2]http://https://github.com/Steven-Adriaensen/hyflext

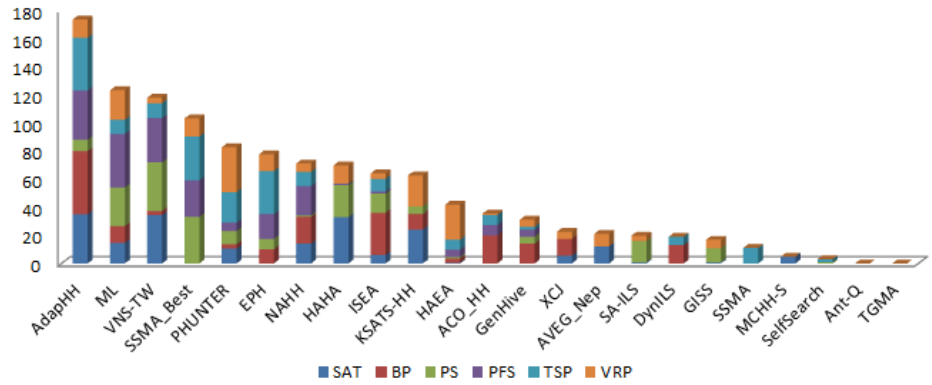

Fig. 4: Performance comparison of SSMA-Best, SSMA, TGMA and selection hyper-heuristics competed at CHeSC2011 across six HyFlex problem domains based on their Formula 1 scores.

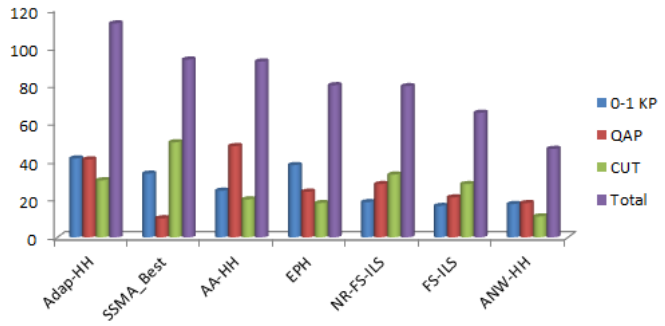

Fig. 5: Performance comparison of SSMA-Best and each of the previously proposed selection hyper-heuristics covered in [1] across the three extended HyFlex domains based on their Formula 1 scores.

CHeSC2011, performs the best overall, by producing the best median fitnesses on 3 instances (joint best in one case). For the QAP domain, NR-FS-ILS outperforms the other algorithms for 3 out of 5 instances. SSMA-Best is the best algorithm on the CUT domain. SSMA-Best manages to outperform the other algorithms in terms of median fitness on 3 CUT instances.

Figure 5 illustrates the ranking of all selection hyper-heuristics and SSMA-Best across the 3 additional HyFlex domains. SSMA-Best ranks second, just after AdapHH. AdapHH combines many sophisticated ideas in its design and makes use of online learning and adaptation mechanisms, thus, being the second algorithm after AdapHH is a success for SSMA-Best, since it is a generic, although tuned, metaheuristic. We emphasize at this point, that it was not tuned by considering the instances upon which it was evaluated, which would obviously be an unfair comparison.

G. Further consideration of SSMA-Best

The SSMA-Best setting considers a population size of 5 and tournament size of 5 for parent selection. The success of this configuration is surprising, since these settings mean that SSMA-Best always selects the best individual (best fitness value) in the entire population as both Parent 1 and 2. Since both parents are exactly the same individuals, after the application of a chosen crossover, the new offspring will be identical to the parents, thus the crossover does nothing. Hence, SSMA under the best setting actually performs a single point based search rather than multipoint. This implies that, for the HyFlex domains and instances, the crossover operators do not seem to have much of a positive influence on the overall performance of SSMA. Under the best settings, none of the individuals other than the best solution are used during the search process, turning the overall algorithm into an Iterated Local Search [15], where mutation is followed by local search at each step.

V. Conclusion and Future Work

In this study, the steady state memetic algorithm presented in [23] is applied to 3 additional domains and is exposed to parameter tuning experiments. The Taguchi orthogonal arrays method is utilised for configuring the parameters of the memetic algorithm for cross-domain search and the performance of the tuned steady state memetic algorithm is evaluated across nine problem domains. The empirical results show that the best setting obtained from the parameter tuning experiments on two instances from four problem domains also performed the best on all instances across nine domains, validating the generality of the tuned parameters. These results support our hypothesis that parameter tuning has actual value in cross-domain search.

The tuning experiments indicate the success of the steady state memetic algorithm with a specific configuration denoted as SSMA-Best, which, significantly, outperforms the two memetic algorithm variants in [23] and even some of the competing hyper-heuristics in CHeSC2011. That metaheuristic approaches are sensitive to parameter settings is a well known fact, which is reinforced by these experiments, which also show that tuning has benefits even across domains.

The cross-domain performance improvement achieved via design of experiments is substantial. Moreover, the key observations from the empirical results are that crossover should not

TABLE VI: Median fitness values over 31 trials achieved by SSMA-Best and previously proposed selection hyper-heuristics for each instance of the additional HyFlex domains. The best values are marked in bold style.

PD	ID	AdapHH	FS-ILS	NR-FS-ILS	EPH	AA-HH	ANW-HH	SSMA-Best
0-1 KP	1	**-1258634**	-1220103	-1231767	-1253074	-1209914	-1208666	-1218057
	3	-431351	-431297	-431312	-431333	-431311	-431304	**-431357**
	5	**-4328770**	-3756992	-3697266	-4283926	-4248962	-4252143	-4259569
	7	**-1577175**	-1572999	-1572999	**-1577175**	**-1577175**	-1572999	-1572999
	9	-1467353	-1463681	-1462759	-1467357	-1467353	-1466892	**-1467362**
QAP	1	154164	**154088**	154166	1543390	154290	156642	154472
	3	149850	149858	**149828**	150144	149992	152090	150330
	5	1187876000	1187491000	**1187383000**	1189221000	1189321000	1237911000	1189780533
	7	44858390	44874030	44873020	44860940	44866880	44929370	**44853276**
	9	273414	273362	**273336**	273630	273512	275644	273726
CUT	1	-269692927	-255265025	-257764081	-260608752	-263151470	-258016734	**-274024670**
	3	-3025	-3020	-3025	-3004	**-3033**	-2976	-3018
	5	-13126	-13083	-13091	-13065	**-13177**	-12964	-13128
	7	-9823	-9632	-9668	-9794	-9878	-9657	**-9965**
	9	-2786	-2676	-2680	-2648	-2814	-2578	**-2822**

be used in this case and that single point based search should be preferred. The steady state memetic algorithm without the crossover operator becomes an Iterated Local Search (ILS) [15] approach, in which the solution is perturbed using a mutation operator and then local search is employed at each step. Interestingly, there are a number of previously proposed 'successful' selection hyper-heuristics that also do not benefit from the usage of crossover operators, such as [3], [9], [11], [14]. In future work, we plan to further investigate this situation, with the further use of automated parameter tuning methods.

REFERENCES

[1] Steven Adriaensen, Gabriela Ochoa, and Ann Nowé. A benchmark set extension and comparative study for the hyflex framework. In *IEEE Congress on Evolutionary Computation, CEC 2015, Sendai, Japan, May 25-28, 2015*, pages 784–791. IEEE, 2015.

[2] Alpay Alkan and Ender Özcan. Memetic algorithms for timetabling. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, volume 3, pages 1796–1802. IEEE, 2003.

[3] Shahriar Asta and Ender Özcan. A tensor-based selection hyper-heuristic for cross-domain heuristic search. *Information Sciences*, 299:412–432, 2015.

[4] Edmund K Burke, Michel Gendreau, Matthew Hyde, Graham Kendall, Barry McCollum, Gabriela Ochoa, Andrew J Parkes, and Sanja Petrovic. The cross-domain heuristic search challenge–an international research competition. In *Learning and Intelligent Optimization*, pages 631–634. Springer, 2011.

[5] Edmund K Burke, Michel Gendreau, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and Rong Qu. Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724, 2013.

[6] Peter Cowling, Graham Kendall, and Eric Soubeiga. A hyperheuristic approach to scheduling a sales summit. In *Practice and Theory of Automated Timetabling III*, pages 176–190. Springer, 2001.

[7] A.E. Eiben and S.K. Smit. Evolutionary algorithm parameters and methods to tune them. In Youssef Hamadi, Eric Monfroy, and Frederic Saubion, editors, *Autonomous Search*, pages 15–36. Springer Berlin Heidelberg, 2012.

[8] Agoston E Eiben and Selmar K Smit. Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, 1(1):19–31, 2011.

[9] Ping-Che Hsiao, Tsung-Che Chiang, and Li-Chen Fu. A variable neighborhood search-based hyperheuristic for cross-domain optimization problems in chesc 2011 competition.

[10] Hisao Ishibuchi and Shiori Kaige. Implementation of simple multiobjective memetic algorithms and its applications to knapsack problems. *Int. J. Hybrid Intell. Syst.*, 1(1):22–35, 2004.

[11] Ahmed Kheiri and Ender Özcan. An iterated multi-stage selection hyper-heuristic. *European Journal of Operational Research*, 2015.

[12] Natalio Krasnogor, BP Blackburne, Edmund K Burke, and Jonathan D Hirst. Multimeme algorithms for protein structure prediction. In *Parallel Problem Solving from NaturePPSN VII*, pages 769–778. Springer, 2002.

[13] Natalio Krasnogor, Jim Smith, et al. A memetic algorithm with self-adaptive local search: Tsp as a case study. In *GECCO*, pages 987–994, 2000.

[14] Mathieu Larose. A hyper-heuristic for the chesc 2011, 2011.

[15] Helena R Lourenço, Olivier C Martin, and Thomas Stützle. Iterated local search: Framework and applications. In *Handbook of Metaheuristics*, pages 363–397. Springer, 2010.

[16] P. Merz and B. Freisleben. A comparison of memetic algorithms, tabu search, and ant colonies for the quadratic assignment problem. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 3, pages 2063–2070, 1999.

[17] Mustafa Mısır, Katja Verbeeck, Patrick De Causmaecker, and Greet Vanden Berghe. An intelligent hyper-heuristic framework for chesc 2011. In *Learning and Intelligent OptimizatioN*, pages 461–466. Springer, 2012.

[18] Sandra Ulrich Ngueveu, Christian Prins, and Roberto Wolfler Calvo. An effective memetic algorithm for the cumulative capacitated vehicle routing problem. *Computers & Operations Research*, 37(11):1877–1885, 2010.

[19] Gabriela Ochoa, Matthew Hyde, Tim Curtois, Jose A Vazquez-Rodriguez, James Walker, Michel Gendreau, Graham Kendall, Barry McCollum, Andrew J Parkes, Sanja Petrovic, et al. Hyflex: A benchmark framework for cross-domain heuristic search. In *Evolutionary Computation in Combinatorial Optimization*, pages 136–147. Springer, 2012.

[20] Gabriela Ochoa, James Walker, Matthew Hyde, and Tim Curtois. Adaptive evolutionary algorithms and extensions to the hyflex hyper-heuristic framework. In *Parallel Problem Solving from Nature-PPSN XII*, pages 418–427. Springer, 2012.

[21] E. Özcan and E. Ersoy. Final exam scheduler - fes. In *The 2005 IEEE Congress on Evolutionary Computation*, volume 2, pages 1356–1363, 2005.

[22] E. Özcan and E. Onbasioglu. Memetic algorithms for parallel code optimization. *International Journal of Parallel Programming*, 35(1):33–61, 2007.

[23] Ender Özcan, Shahriar Asta, and Cevriye Altintas. Memetic algorithms for cross-domain heuristic search. In *Computational Intelligence (UKCI), 2013 13th UK Workshop on*, pages 175–182. IEEE, 2013.

[24] R.K. Roy. *A primer on the Taguchi method*. Competitive manufacturing series. Van Nostrand Reinhold, 1990.

[25] Selmar K Smit and Agoston E Eiben. Comparing parameter tuning methods for evolutionary algorithms. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 399–406. IEEE, 2009.

[26] Ji Ung Sun. A taguchi approach to parameter setting in a genetic algorithm for general job shop scheduling problem. *IEMS*, 6(2):119–124, 2007.

[27] Gilbert Syswerda. A study of reproduction in generational and steady state genetic algorithms. *Foundations of genetic algorithms*, 2:94–101, 1991.