Memetic Algorithms for Cross-domain Heuristic Search

Ender Özcan*, Shahriar Asta*, Cevriye Altıntaş†
*Automated Scheduling, Optimisation and Planning Research Group
School of Computer Science, University of Nottingham, UK
Email: {exo,sba}@cs.nott.ac.uk
†Süleyman Demirel University, Isparta, Turkey
Email: cevriyealtintas@sdu.edu.tr

***Abstract*—Hyper-heuristic Flexible Framework (HyFlex) is an interface designed to enable the development, testing and comparison of iterative general-purpose heuristic search algorithms, particularly selection hyper-heuristics. A selection hyper-heuristic is a high level methodology that coordinates the interaction of a fixed set of low level heuristics (operators) during the search process. The Java implementation of HyFlex along with different problem domains was recently used in a competition, referred to as Cross-domain Heuristic Search Challenge (CHeSC2011). CHeSC2011 sought for the best selection hyper-heuristic with the best median performance over a set of instances from six different problem domains. Each problem domain implementation contained four different types of operators, namely mutation, ruin-recreate, hill climbing and crossover. CHeSC2011 including the competing hyper-heuristic methods currently serves as a benchmark for hyper-heuristic research. Considering the type of the operators implemented under the HyFlex framework, CHeSC2011 could also be used as a benchmark to empirically compare the performance of appropriate variants of the evolutionary computation methods across a variety of problem domains for discrete optimisation. In this study, we investigate the performance and generality level of generic steady-state and transgenerational memetic algorithms which hybridize genetic algorithms with hill climbing across six problem domains of the CHeSC2011 benchmark.**

I. Introduction

A metaheuristic can be considered as a template showing how to design a search algorithm to locate near optimal solutions for a given difficult combinatorial optimization problem. Although conceptually, metaheuristics are general, they require tailoring to each given specific problem domain. There are single point-based metaheuristics such as tabu search and simulated annealing, performing search using a single candidate solution (often remembering the best solution). There are also multi-point-based or population based metaheuristics, performing search using a set of candidate solutions, such as genetic programming (GP) and genetic algorithms (GA) [5]. GAs attempt to improve a population of candidate solutions through an evolutionary cycle by iteratively applying a set of genetic operators and creating new individuals then replacing the old individuals with the new ones. While moving from one *generation* to the next, *selection* operators together with *crossover* and *mutation* operators are used to form a new population as well as avoiding convergence to local optima [15]. The choice of these operators could influence the intensification/diversification balance, extremely.

Hybridizing single point local search methods with population based evolutionary algorithms achieves a better performance by introducing an additional intensification step in the evolutionary cycle. This is the main idea behind a family of evolutionary algorithms called Memetic Algorithms (MA). Memetic algorithms were first introduced by Moscato [30]. The term *meme*, however, was first coined by Dawkins [13] referring to a contagious piece of information which is processed, comprehended, adapted and passed on by the infected person. This adaptation process resembles the local refinement, hence the use of the term memetic algorithms that make extensive use of local search (hill climbing) methods. In fact, MA is a hybridization of GA and local search. That is, within the process of evolution, local search methods are applied to the individuals of the population in a certain stage of each cycle to improve their quality. Numerous variants of memetic algorithms have been proposed in the literature, such as Steady State Memetic Algorithm (SSMA) and Trans-Generational Memetic Algorithms (TGMA). A memetic algorithm is often designed specifically for a given problem domain [32].

Recently, there has been a growing number of studies on *hyper-heuristics* which are general high-level methodologies searching in the space of low-level heuristics for search and optimization [3]. Employing various heuristic selection mechanisms together with acceptance criteria to decide which heuristic to proceed with, while maintaining independence from the underlying problem domain, enables hyper-heuristic methods to be applicable to a range of problem domains without requiring any modification. Also, learning is sometimes employed to achieve an improved performance [40]. Hyper-heuristic methodologies are categorized as *heuristic selection* and *heuristic generation* methodologies in [6]. An overview of hyper-heuristics of different types and more can be found in [3]. A metaheuristic can be used as a hyper-heuristic to search the space of heuristics rather than solutions directly. An important challenge within hyper-heuristic research is to design automated search methodologies that perform well on the unseen instances from not only a given single problem domain, but also a range of domains without requiring expert intervention. This was the topic of a recent competition referred to as Cross-domain Heuristic Search Challenge (CHeSC2011). CHeSC2011 based on Hyper-heuristics Flexible Framework (HyFlex) [34] which currently serves as a benchmark to compare the performance of selection type of hyper-heuristics. HyFlex implementation includes six problem domain implementations, each with a fixed set of low level heuristics. These heuristics can be used as evolutionary algorithm components to perform direct search on the solution space as well, hence enable empirical performance comparison of memetic algorithms on real world discrete optimization problems, rather than benchmark functions [40].

The HyFlex framework as a benchmark provides a rich family of problem instances. This study has several goals. Our first goal is to integrate memetic algorithms into the HyFlex framework and test them across various problem domains, providing a baseline for performance comparison of different evolutionary computation methods in the future. Such a comparison will, at the same time, demonstrate the advantages and disadvantages of applying population based methods versus single point search methods. Our second goal is to analyze the generalizability of memetic algorithms over various problem domains without changing their implementation. We extended HyFlex with the implementation of two memetic algorithms Steady State Memetic Algorithm (SSMA) and Trans-Generational Memetic Algorithms (TGMA). Then we tested their performances on the CHeSC2011 benchmark.

The structure of the paper is organized as follows: In Section II a literature review on various approaches and recent advances in memetic algorithms and hyper-heuristics is given. The HyFlex framework has been described in Section III. In Section IV, our methodologies are discussed in detail. The experimental results and subsequent analysis are presented in Section V. Finally, concluding remarks and a glimpse of our future work is given in Section VI.

II. Related Work

The term "Memetic Algorithm" was introduced by Moscato in [30]. Memetic algorithms (MAs), also referred to as hybrid genetic algorithms, represent a set of genetic algorithms that make heavy use of hill climbing. Ever since its emergence, memetic algorithms and subsequent variants of MAs have been applied to various problems such as the timetabling problems [7], [1], [42], [37], [9], [43], permutation flow shop scheduling [21], travelling salesman problem [31], [2], [16], quadratic assignment problem [28], multi-objective optimization problems [23], [11] and protein folding problem [25]. Apart from application of memetic algorithms on various problem domains, different meme considerations has also been well studied. An example of such studies can be found in [41] and [39] in which it has been shown that different meme structures might yield in different performances. Interested readers can find more details about memetic algorithms research in the comprehensive survey paper by Neri et al. [32].

Like many other approaches, memetic algorithms have also been subjected to constant improvements in various studies, evolving it to some complex variants such as multi-meme algorithms and memetic computing. In the excellent work of Nguyen et al. [33] this evolution of MAs has been covered extensively. Four generation of memetic algorithms has been introduced so far where in the first generation, population based methods are hybridized with local search methods to achieve improvements in the solution quality. In the second generation, the concept of evolving meme in which the meme is a part of the genotype is the focus of research. These type of algorithms which are usually referred to as multi-meme MAs [24] transfer memes to the next generation by using simple inheritance mechanisms. Also hyper-heuristic [22] and Meta-Lamarckian MA [35], [36] are investigated at this time suggesting the use of a pool of memes from which a candidate meme is selected based on the reward value associated to it. This reward value is computed considering the history of improvements that the meme has implied in the past. Co-evolution and self-generation MAs are considered to be concepts relevant to the third generation of MA's, where memetic information is passed to offspring with a learning strategy [45], [26]. Furthermore, the concept of Memetic Computing extends memetic algorithms as a whole to a framework in which machine learning, cognitive observation of other individuals and memory utilization are widely employed [10].

Similar to the hyper-heuristic research, adaptivity in co-evolution and self-generation MAs is achieved by exploring the space of local search heuristics and/or other parameters of MAs. Indeed, this is the basic idea behind hyper-heuristics, though in contrast to memetic algorithms, hyper-heuristics aims to perform this in a domain independent fashion. Covering hyper-heuristic approaches is out of the scope of this paper. However, it is the long-term intention of this study to investigate the two frameworks (hyper-heuristics and memetic computing) in order to establish a link between both research areas by applying the advances and ideas in one field of research to the other. Moreover, hyper-heuristic research seems to have much to offer to improve the performance of memetic algorithms, invoking ideas which lead to memetic computing and fourth generation of MAs where the main emphasis is on the utilization of machine learning techniques and cognition in MAs. Indeed, these ideas have long been employed in hyper-heuristic research resulting in algorithms with higher levels of generality and subsequently better performances. Thus, briefly discussing main ideas behind the hyper-heuristic approach and some recent advances in this field seems necessary.

Hyper-heuristics are known as high-level methods searching the space of low-level heuristics for search and optimization. Independence from the underlying problem domain is usually maintained through considering a domain barrier. A domain barrier restricts the amount of domain knowledge available to the higher level heuristic to the fitness value which is acquired by applying a low-level heuristic to the problem. Such a domain barrier is a means of providing the domain-independence which is necessary to achieve higher level of generality. Thus, during the search process, the hyper-heuristic uses some selection and acceptance mechanisms to choose a low-level heuristic and applies it on the problem. Online/offline learning techniques are sometimes used to achieve more generality through self-adaptation in hyper-heuristics. A detailed study of these approaches as well as a categorization of hyper-heuristic methods can be found in [6]. However, as mentioned above, one interesting point which correlates hyper-heuristic research and multi-meme MAs is their similarity in utilizing various local search methods to improve the solution quality. There are numerous examples in which local search heuristics have been embedded into hyper-heuristics. Simulated annealing [14], genetic algorithms [12], [19], [17], tabu search [18], [44], [4] and Variable Neighbourhood Search [8] are few examples. In fact, since hyper-heuristic methods provide considerably high quality approaches by utilizing various online and offline machine learning techniques such as reinforcement learning [38] and other data mining techniques, establishing a link and applying the same ideas to memetic algorithms might yield into MA approaches fitting into the category of memetic computing where such extensive use of machine learning methods are considered as promising approaches in the literature ([35], [36]).

III. First Cross-Domain Heuristic Search Challenge (CHeSC2011)

Hyperion [46] and Hyper-heuristics Flexible Framework (HyFlex) [34] are software libraries for the implementation and comparison of different hyper/meta-heuristics. The HyFlex framework allows the implementation of hyper-heuristic approaches and supports reusability of hyper-heuristic components, providing an interface for the problem domains. A Java implementation of HyFlex including six different problem domains was recently used at the Cross-Domain Heuristic Search Challenge (CHeSC2011) [1]. The goal of this competition was determining the best selection hyper-heuristic with the best median performance across thirty problem instances, five from each problem domain. 20 competitors reached the finals in the competition. CHeSC2011, including the HyFlex implementation and competing hyper-heuristics, currently serves as a benchmark to compare the performance of selection hyper-heuristics. HyFlex is designed imposing a domain barrier concept [12] between the hyper-heuristic and problem domain layers as illustrated in Figure 1. This barrier keeps the problem specific details, such as, low level heuristics, representation, fitness function hidden from the hyper-heuristic. Only problem independent information, such as the number of heuristics, fitness value of a solution are allowed to pass through this barrier from the problem domain.

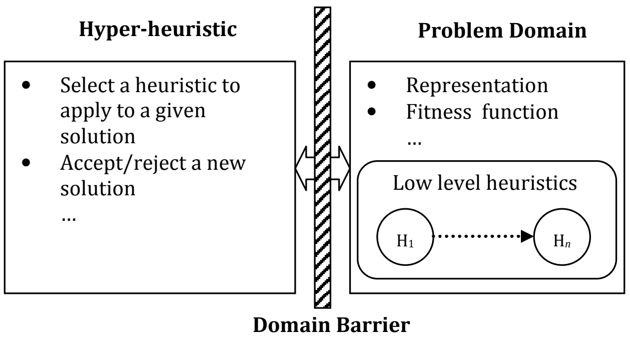


Fig. 1. A selection hyper-heuristic framework.

The CHeSC2011 problem domains include Boolean Satisfiability (SAT), One Dimensional Bin Packing (BP), Permutation Flow Shop (PFS), Personnel Scheduling (PS), Travelling Salesman Problem (TSP) and Vehicle Routing Problem (VRP). Each domain provides a set of The HyFlex low level heuristics are classified as mutation (MU), hill climbing (HC), ruin and re-create and crossover (XO) heuristics (operators). Ruin and re-create heuristics are considered as mutation operators in our experiments. The current implementation of HyFlex, not to a great extent but still restricts the evolutionary algorithms that can be implemented extending it. For example, a crossover in current version of the HyFlex interface definition returns a single offspring. All low level heuristics in HyFlex are perturbative. They process and return a complete solution at all times. Each low level heuristic in CHeSC2011 comes with a parameter setting that can be adjusted. The *mutation density* determines the extent of changes that the selected mutation operator yields on a solution. Higher values for mutation density indicate wider range of new values that the solution can take, relevant to its current value. Lower values for mutation density suggest a more conservative approach where changes are less influential. As for the *depth* of hill climbing, this value relates to the number of steps completed by the hill climbing heuristic. Higher values indicate that hill climbing approach searches more neighborhoods for improvement.

The number of the low level heuristics for each heuristic/operator type for each problem domain implemented in HyFlex is summarized in Table I. Due to the fact that CHeSC2011 provides a wide range of problem domains along with performances of 20 algorithms that competed, it is a viable benchmark to compare the performances of population based techniques within themselves and even to the other single point-based search approaches. All the algorithms in this paper are applied to the instances across six aforementioned problem domains and compared to each other to test their level of generality.

TABLE I. The number of different types of low level heuristics {mutation (MU) which includes ruin and re-create heuristics, hill climbing (HC), crossover (XO)} used in each CHeSC2011 problem domain.

Domain	MU	HC	XO	Total
SAT	7	2	2	11
BP	5	2	1	8
PS	4	5	3	12
PFS	7	4	4	15
TSP	6	3	4	13
VRP	5	3	2	10

The top three selection hyper-heuristics that generalize well across the CHeSC2011 problem domains are AdapHH [29], VNS-TW [20] and ML [27]. The winning algorithm, AdapHH is a multi-phase learning hyper-heuristic [29]. AdapHH adaptively determines the subset of low-level heuristics to apply at each phase. The duration with which each heuristic is applied is also dynamically determined during the search. The algorithm accepts only improving solutions in the absence of which the algorithm refuses to accept worsening solutions until no improvements are observed within an adaptively adjusted number of iterations. The parameters of each low-level heuristic are dynamically modified via a reinforcement learning method. This is while low-level heuristics are selected based on a *quality index* measure. This measure uses few weighted performance metrics to compute the quality index for each heuristic. Among these metrics are the number of new best solutions explored, the total improvement and worsening during the run and also the current phase and finally the remaining execution time. A heuristic with a quality index less than the average of the quality indexes of all the heuristics is excluded from the selection process in the corresponding phase. Using a Tabu style memory, the number of phases in which the heuristic is consecutively excluded is maintained. Whenever this number exceeds a threshold the heuristic gets excluded permanently. AdapHH also employs a relay hy-

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

bridization with which effective pairs of heuristics which are applied consecutively are identified.

The algorithm which ranked second in the competition, VNS-TW [20], is a double phase algorithm. The first phase consists of applying mutational or ruin and recreate type of low-level heuristics to a population of initial solutions. Subsequently, all the local search heuristics are applied until no more improvements are observed. In the next phase, the algorithm shifts from a population based method into a single solution approach in which the best solution achieved in the first phase is used. Iteratively, A circular priority queue of mutational heuristics is formed based on the severity of changes that they imply. Subsequent to the application of heuristics a local search is applied. This approach was ranked first in the Personnel Scheduling problem domain and second in the overall.

The hyper-heuristic proposed in [27], namely ML, relies explicitly on intensification and diversification components during the search process. A solution is generated initially which goes through a diversification stage by the application of a mutational or ruin-recreate low-level heuristic. The solution achieved at this stage is then subjected to a local search heuristic for further improvement. The local search heuristic is applied until no further improvements can be achieved. The acceptance mechanism, accepts improving solutions as well as the cases where the solution has not improved over the last 120 iterations. More on the rest of the algorithms can be found at the CHeSC2011 web-site.

IV. Methodologies

Two different memetic algorithm variants, namely steady-state memetic algorithm (SSMA) and transgenerational memetic algorithm (TGMA) are investigated in this study. Algorithms 1 and 2 present the pseudocode of these algorithms, respectively. SSMA and TGMA are both implemented as an extension to the HyFlex framework and they both utilize the low-level heuristics as their operators for each domain of CHeSC2011. Each domain in CHeSC2011 contains multiple crossover, mutation and hill climbing operators as illustrated in Table IV. The ruin and re-create operators are also considered as mutation operators in our study. We have performed simple random choice from each type of operator when needed.

In SSMA, the initial population of individuals is created randomly (Algorithms 1). Hill-climbing is then applied on each and every individual within the initial population. Subsequently, at each cycle of evolution, two parents (individuals) are chosen using tournament selection method. This method chooses the individual with the best fitness among a number of randomly chosen individuals as a parent, which is referred to as *tour size*. A new offspring is then created by applying crossover to those selected parents, which is mutated afterwards. Then a hill climbing method is used to improve the offspring further. Crossover, mutation and hill climbing operators are selected randomly whenever needed from the set of operators that each CHeSC2011 problem domain provides. The new offspring consequently substitutes the worst individual in the population.

Similar to SSMA, TGMA starts with a randomly generated initial population and applies hill-climbing on each individual of the population (Algorithms 2). However, in contrast to SSMA, in TGMA, an *offspring pool* is created by applying tournament selection twice to choose two parents, crossover, mutation and hill climbing, successively, for as many times as one less than the population size. At each time, a random crossover, mutation and hill climbing operator is selected. Weak elitism is employed by allowing only the best individual of the current generation to survive. The rest of the population is then filled with individuals from the offspring pool to form the next generation.

Algorithm 1 Pseudocode of a steady-state memetic algorithm

```
Create a population of $popSize$ random individuals.
// Apply hill-climbing on each individual
for $i = 1 : popSize$ do
    OP_ID = Random-Choice(Hill-Climbing-Operators)
    $Ind(i)$ ← Apply-Hill-Climbing(OP_ID, $Ind(i)$)
end for
while termination criteria is not satisfied do
    Parent1 ← Select-Parent(Population, tour-size)
    Parent2 ← Select-Parent(Population, tour-size)
    OP_ID = Random-Choice(Crossover-Operators)
    Offspr ← Apply-Crossover(OP_ID, Parent1, Parent2)
    OP_ID = Random-Choice(Mutation-Operators)
    Offspr ← Apply-Mutation(OP_ID, Offspr)
    OP_ID = Random-Choice(Hill-Climbing-Operators)
    Offspr ← Apply-Hill-Climbing(OP_ID, Offspr)
    Replacement: Replace the worst individual by Offspr
end while
```

Algorithm 2 Pseudocode of a transgenerational memetic algorithm

```
Create a population of $popSize$ random individuals.
// Apply hill-climbing on each individual
for $i = 1 : popSize$ do
    OP_ID = Random-Choice(Hill-Climbing-Operators)
    $Ind(i)$ ← Apply-Hill-Climbing(OP_ID, $Ind(i)$)
end for
while termination criteria is not satisfied do
    for $i = 1 : popSize - 1$ do
        Parent1 ← Select-Parent(Population, tour-size)
        Parent2 ← Select-Parent(Population, tour-size)
        OP_ID = Random-Choice(Crossover-Operators)
        Offspr ← Apply-Crossover(OP_ID, Parent1, Parent2)
        OP_ID = Random-Choice(Mutation-Operators)
        Offspr ← Apply-Mutation(OP_ID, Offspr)
        OP_ID = Random-Choice(Hill-Climbing-Operators)
        Offspr ← Apply-Hill-Climbing(OP_ID, Offspr)
        Add(Offspr, Offspring-Pool)
    end for
    Replacement: Keep the best individual in Population and
    replace the rest with Offspring-Pool
end while
```

Both SSMA and TGMA utilise Lamarckian learning, as they both put the individual modified by hill climbing back into the population. Furthermore, due to the simple random heuristic selection mechanism used, no feedback from the search process is used and no online knowledge is considered while choosing an operator. Thus, following the classification of adaptive MAs in [36], the adaptation type in both SSMA and TGMA is static and the level of adaptation is external.

V. Experimental Results

A. Experimental Settings

The performance of the memetic algorithms were tested on 5 instances from each of the six problem domains in CHeSC2011. Parameter settings for bot memetic algorithms used during the experiments are summarized in Table II. Population size is 10 for both SSMA and TGMA. Tournament (parent) selection mechanism uses a tour size of 2. The density of the mutation operator as well as the depth of the selected local search method is set to be the default value of 0.2. For a fair performance comparison to the competing hyper-heuristics in CHeSC2011, the same instances which were used in the competition are employed in our experiments. We have used a 2 Core Duo 3.16 GHz (2 GB RAM) machine during our experiments. In the competition, each algorithm was given 600 nominal seconds for a run which corresponds to 576 seconds on our machine determined using the benchmarking tool provided at the CHeSC website. Each experiment with an algorithm on a given instance is repeated for 31 times. We have performed Wilcoxon signed-rank test to compare the average performances of memetic algorithms.

TABLE II. Parameter settings

Parameter	**value**
Population Initialization	random
Population Size	10
Selection Mechanism	tournament selection
Tour Size	2
Genetic Operator Selection	uniform random
Mutation Density	0.2
Local Search Selection	uniform random
Local Search Depth	0.2
Termination Condition	600 nominal seconds

B. Performance Comparison of SSMA and TGMA

Table III provides performance comparison of SSMA and TGMA based on average fitness and best-of-run fitness over 31 trials for each instance. The results reveal that SSMA performs better than TGMA on two domains considering average and best-of-run results; flow shop and travelling salesman. Conversely, TGMA performs better than SSMA on satisfiability and bin packing problem domains on average, obtaining the best results for each instance on those domains. In both cases, these average performance differences between the memetic algorithms are statistically significant. SSMA delivers a slightly better performance over the personal scheduling and vehicle routing problem domains. SSMA obtains best solutions for all the personal scheduling instances and three out of five instances from the vehicle routing problem domain. SSMA is slightly better than TGMA considering their average and best-of-run performances on all instances.

Figures 2(a) to (f) provides average best fitness over 31 trials versus time plot for an arbitrarily chosen sample problem instance from each problem domain. These plots further validate the performance differences generated by SSMA and TGMA on those selected instances as presented in Table III. Moreover, in almost all cases, a sharp improvement has been observed within a couple of seconds, then the improvement slows down. After the initial tens of seconds into the search,

TABLE III. Average performance comparison of SSMA and TGMA based on cost averaged over 31 trials for each instance, where "i.no." is the instance number. Notation used in "vs.": $>$ / $<$ ($\geq$ / $\leq$) denotes that SSMA/TGMA performs (slightly) better than TGMA/SSMA and this performance difference is (not) statistically significant within a 95% confidence interval based on Wilcoxon signed-rank test.

		SSMA			TGMA	
domain	i.no.	avr.	best	vs.	avr.	best
SAT	1	21.161	8.0	$<$	14.323	**3.0**
	2	52.484	37.0	$<$	41.806	**11.0**
	3	35.0	10.0	$<$	27.129	**5.0**
	4	27.742	26.0	$<$	20.226	**13.0**
	5	19.194	14.0	$<$	17.290	**13.0**
BP	1	0.083	0.074	$<$	0.075	**0.066**
	2	0.015	0.115	$<$	0.012	**0.007**
	3	0.022	0.018	$<$	0.020	**0.016**
	4	0.111	0.110	$<$	0.110	**0.109**
	5	0.043	0.036	$<$	0.037	**0.032**
PS	1	51.129	**37.0**	$>$	66.419	50.0
	2	72015.613	**52056.0**	$\leq$	70356.161	59736.0
	3	13203.710	**5581.0**	$\geq$	14028.710	10027.0
	4	2942.419	**1820.0**	$\leq$	2923.710	2003.0
	5	435.645	**385.0**	$>$	498.032	430.0
PFS	1	6257.806	**6231.0**	$>$	6301.968	6273.0
	2	26884.935	**26813.0**	$>$	26944.548	26889.0
	3	6351.871	**6318.0**	$>$	6366.677	6342.0
	4	11441.806	**11410.0**	$>$	11499.645	11458.0
	5	26699.226	**26626.0**	$>$	26724.742	26668.0
TSP	1	48227.747	**48194.920**	$>$	48337.598	48194.921
	2	2.116E7	**2.096E7**	$>$	2.130E7	2.114E7
	3	6825.552	**6800.708**	$>$	6893.822	6858.803
	4	68123.369	**67423.655**	$>$	69778.135	68922.954
	5	53810.138	**52685.992**	$>$	55463.600	54052.398
VRP	1	71768.053	**67820.589**	$>$	76331.070	71560.119
	2	14324.522	13358.611	$<$	13869.191	**13333.0914**
	3	176206.081	**167704.512**	$>$	200838.192	193416.831
	4	21647.018	20678.096	$<$	21412.571	**20659.896**
	5	152642.040	**149032.551**	$>$	157001.253	153557.350

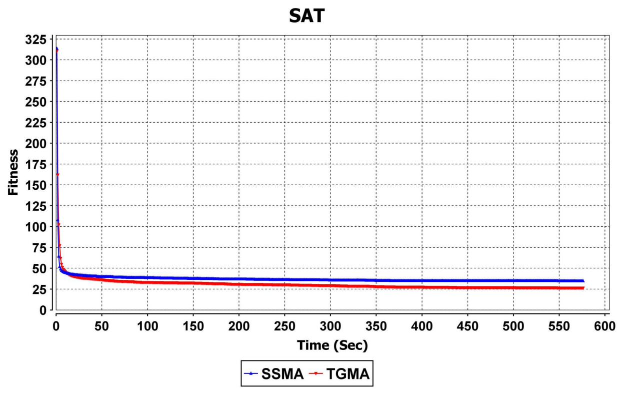
the improvement continues gradually but at even a much slower rate. In the personnel scheduling domain, improvement occurs time to time with sudden jumps to a better solution as it can be observed in Figure 2(c). This could be due to the fact that low level heuristics for this domain always return a *feasible* solution.

C. Performance Comparison of Memetic Algorithms to Selection Hyper-heuristics

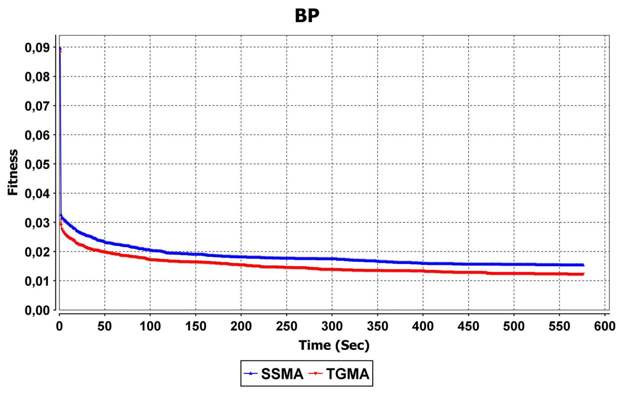
Table IV provides the ranking of the memetic algorithms among the competing selection hyper-heuristics in CHeSC2011 based on Formula 1 scoring system as used in the competition. The ranking and overall sum of scores is obtained using the median of the 31 trials for each instance. SSMA delivers a better median performance when compared to TGMA and performs better than some previously proposed selection hyper-heuristics in the overall and in particular on flow shop and travelling salesman problems (Figure 3). However, they are both among the lowest ranking algorithms. Single point based search methods outperform the implemented population based metaheuristics. Simple memetic algorithms do not generalize well across different problem domains as the selection hyper-heuristics do. This was somewhat expected, considering that the memetic algorithms used in our experiments are very simple and static lacking any type of the adaptation [36].

VI. Conclusion

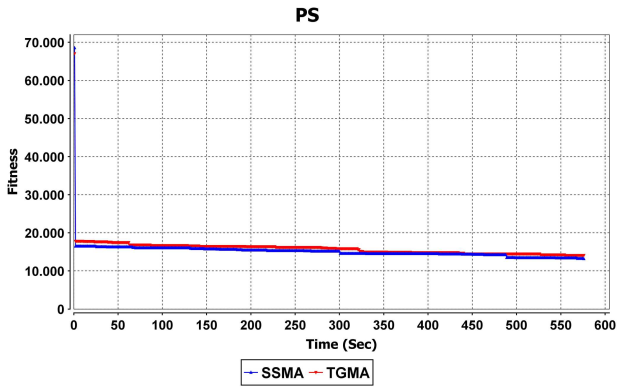
In this paper, we have successfully integrated two memetic algorithms, namely steady-state and transgenerational memetic



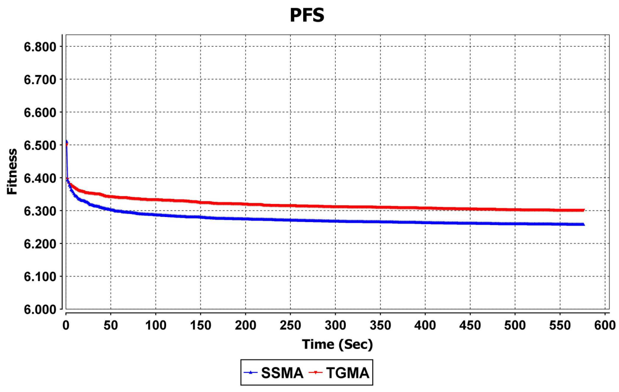
(a) Instance 3



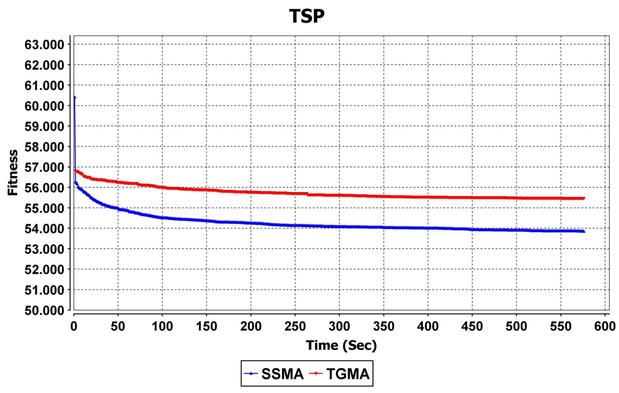
(b) Instance 2



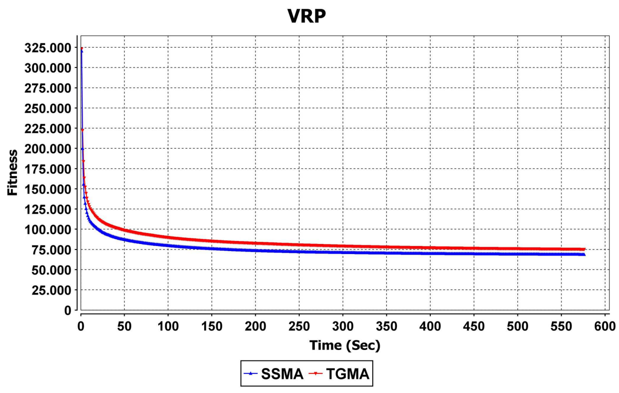
(c) Instance 3



(d) Instance 1



(e) Instance 5



(f) Instance 1

Fig. 2. Average best fitness versus time plot based on the results from 31 trials on an arbitrarily selected instance from each CHeSC2011 problem domain using SSMA and TGMA.

algorithms into the HyFlex framework. The CHeSC2011 problem domains provide memetic algorithms with a rich family of genetic operators as well as hill climbing methods. The two memetic algorithms studied in this paper have relatively simple structure with no adaptation capacity. While this partly explains the reason why many of the selection hyper-heuristics outperformed them in the comparative study, it further encourages us to integrate further learning mechanisms into the memetic framework to handle multiple evolutionary operators. Moreover, in our experiments, we used the default settings for the operator parameters enforcing them to use *small* step sizes while performing search. We intend to implement, adaptive and self-adaptive memetic algorithms for choosing operators and controlling their parameter values and compare their performances using CHeSC2011 as a benchmark in our future work.

References

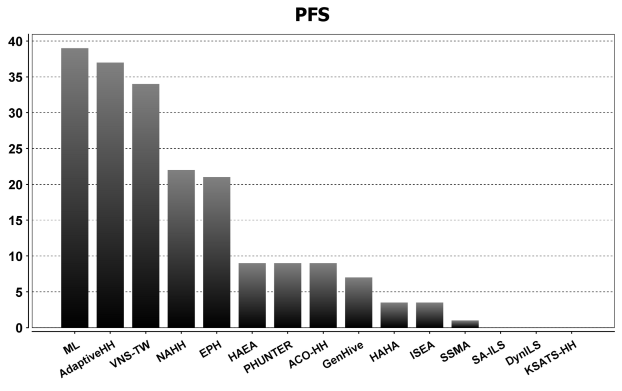
[1] A. Alkan and E. Özcan, “Memetic algorithms for timetabling,” in *Congress on Evolutionary Computation, CEC ’03.*, vol. 3, 2003, pp. 1796–1802.

[2] L. Buriol, P. M. França, and P. Moscato, “A new memetic algorithm for the asymmetric traveling salesman problem,” *Journal of Heuristics*, vol. 10, no. 5, pp. 483–506, Sep. 2004.

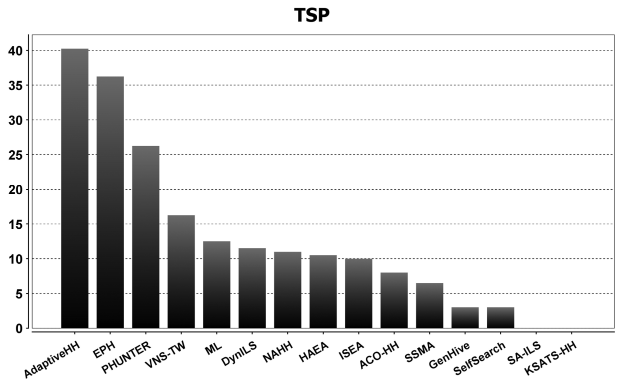
[3] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu, “Hyper-heuristics: A survey of the state of the art,” *Journal of the Operational Research Society*, to appear.

TABLE IV. RANKING OF THE MEMETIC ALGORITHMS AMONG THE SELECTION HYPER-HEURISTICS THAT COMPETED IN CHeSC2011 WITH RESPECT TO THEIR FORMULA 1 SCORES.

Rank	Name	Score
1	AdapHH	181
2	VNS-TW	134
3	ML	131.5
4	PHunter	93.25
5	EPH	89.25
6	HAHA	75.75
7	NAHH	75
8	ISEA	71
9	KSATS-HH	66
10	HAEA	53.5
11	ACO-HH	39
12	GenHive	36.5
13	DynILS	27
14	SA-ILS	24.25
15	XCJ	22.5
16	AVEG-Nep	21
17	GISS	16.75
18	**SSMA**	**7.5**
19	SelfSearch	7
20	MCHH-S	4.75
21	Ant-Q	0
22	**TGMA**	**0**



(a) Flow



(b) TSP

Fig. 3. Performance comparison of CHeSC2011 hyper-heuristics and SSMA based on their Formula 1 scores for the problem domains of (a) permutation flow shop and (b) travelling salesman.

[4] E. K. Burke, G. Kendall, and E. Soubeiga, “A tabu-search hyperheuristic for timetabling and rostering,” *Journal of Heuristics*, vol. 9, no. 6, pp. 451–470, 2003.

[5] E. Burke and G. Kendall, *Search methodologies: introductory tutorials in optimization and decision support techniques*. Springer Science+ Business Media, 2005.

[6] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. R. Woodward, “A classification of hyper-heuristics approaches,” in *Handbook of Metaheuristics*, 2nd ed., ser. International Series in Operations Research & Management Science, M. Gendreau and J.-Y. Potvin, Eds. Springer, 2010, vol. 57, ch. 15, pp. 449–468.

[7] E. K. Burke, J. P. Newall, and R. F. Weare, “A memetic algorithm for university exam timetabling.” Springer-Verlag, 1996, pp. 241–250.

[8] E. Burke, A. Eckersley, B. McCollum, S. Petrovic, and R. Qu, “Hybrid variable neighbourhood approaches to university exam timetabling,” *European Journal of Operational Research*, vol. 206, no. 1, pp. 46–53, 2010.

[9] E. Burke and J. Landa Silva, “The design of memetic algorithms for scheduling and timetabling problems,” in *Recent Advances in Memetic Algorithms*, ser. Studies in Fuzziness and Soft Computing, W. E. Hart, J. Smith, and N. Krasnogor, Eds. Springer Berlin Heidelberg, 2005, vol. 166, pp. 289–311.

[10] X. Chen, Y.-S. Ong, M.-H. Lim, and K. C. Tan, “A multi-facet survey on memetic computation,” *Evolutionary Computation, IEEE Transactions on*, vol. 15, no. 5, pp. 591–607, 2011.

[11] K. C. T. Chi-Keong Goh, Yew-Soon Ong, *Multi-Objective Memetic Algorithms*. Springer Berlin Heidelberg, 2009.

[12] P. Cowling, G. Kendall, and L. Han, “An investigation of a hyper-heuristic genetic algorithm applied to a trainer scheduling problem,” in *Evolutionary Computation, 2002. CEC ’02. Proceedings of the 2002 Congress on*, vol. 2, 2002, pp. 1185–1190.

[13] R. Dawkins, *The Selfish Gene*. New York City: Oxford University Press, 1976.

[14] K. A. Dowsland, E. Soubeiga, and E. Burke, “A simulated annealing based hyperheuristic for determining shipper sizes for storage and transportation,” *European Journal of Operational Research*, vol. 179, no. 3, pp. 759 – 774, 2007.

[15] M. Gendreau and J.-Y. Potvin, “Metaheuristics in combinatorial optimization,” *Annals of Operations Research*, vol. 140, no. 1, pp. 189–213, 2005.

[16] G. Gutin and D. Karapetyan, “A memetic algorithm for the generalized traveling salesman problem,” vol. 9, no. 1, pp. 47–60, Mar. 2010.

[17] L. Han and G. Kendall, “Guided operators for a hyper-heuristic genetic algorithm,” in *AI 2003: Advances in Artificial Intelligence*, ser. Lecture Notes in Computer Science, T. Gedeon and L. Fung, Eds. Springer Berlin Heidelberg, 2003, vol. 2903, pp. 807–820.

[18] ——, “An investigation of a tabu assisted hyper-heuristic genetic algorithm,” in *Evolutionary Computation, 2003. CEC ’03. The 2003 Congress on*, vol. 3, 2003, pp. 2230–2237.

[19] L. Han, G. Kendall, and P. Cowling, “An adaptive length chromosome hyperheuristic genetic algorithm for a trainer scheduling problem,” in *Proceedings of the fourth Asia-Pacific Conference on Simulated Evolution And Learning, (SEAL’02), Orchid Country Club, Singapore, 18-22 Nov 2002*, 2002, pp. 267–271.

[20] P.-C. Hsiao, T.-C. Chiang, and L.-C. Fu, “A vns-based hyper-heuristic with adaptive computational budget of local search,” in *Evolutionary Computation (CEC), 2012 IEEE Congress on*, 2012, pp. 1–8.

[21] H. Ishibuchi, T. Yoshida, and T. Murata, “Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling,” *Evolutionary Computation, IEEE Transactions on*, vol. 7, no. 2, pp. 204–223, 2003.

[22] G. Kendall, P. Cowling, and E. Soubeiga, “Choice function and random hyper-heuristics.” in *in Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution And Learning*. SEAL, 2002, pp. 667–671.

[23] J. Knowles and D. Corne, “Memetic algorithms for multiobjective optimization: Issues, methods and prospects,” in *Recent Advances in Memetic Algorithms*, ser. Studies in Fuzziness and Soft Computing, W. E. Hart, J. Smith, and N. Krasnogor, Eds. Springer Berlin Heidelberg, 2005, vol. 166, pp. 313–352.

[24] N. Krasnogor, “Studies on the theory and design space of memetic algorithms,” Ph.D. dissertation, University of the West of England, 2002, supervisor: Dr. J.E. Smith.

[25] N. Krasnogor, B. Blackburne, E. Burke, and J. Hirst, “Multimeme algorithms for protein structure prediction,” in *Parallel Problem Solving from Nature - PPSN VII*, ser. Lecture Notes in Computer Science,

J. Guervós, P. Adamidis, H.-G. Beyer, H.-P. Schwefel, and J.-L. Fernández-Villacañas, Eds. Springer Berlin Heidelberg, 2002, vol. 2439, pp. 769–778.

[26] N. Krasnogor and S. Gustafson, “A study on the use of ”self-generation” in memetic algorithms,” *Natural Computing*, vol. 3, no. 1, pp. 53–76, 2004.

[27] M. Larose, “A hyper-heuristic for the chesc 2011,” in *LION6*, 2011.

[28] P. Merz and B. Freisleben, “Fitness landscape analysis and memetic algorithms for the quadratic assignment problem,” *Evolutionary Computation, IEEE Transactions on*, vol. 4, no. 4, pp. 337–352, 2000.

[29] M. Mısır, K. Verbeeck, P. De Causmaecker, and G. V. Berghe, “An intelligent hyper-heuristic framework for chesc 2011,” in *Learning and Intelligent Optimization*. Springer, 2012, pp. 461–466.

[30] P. Moscato, “On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms,” *Caltech concurrent computation program, C3P Report*, vol. 826, p. 1989, 1989.

[31] P. Moscato and M. G. Norman, “A memetic approach for the traveling salesman problem implementation of a computational ecology for combinatorial optimization on message-passing systems,” *Parallel Computing and Transputer Applications*, vol. 1, pp. 177–186, 1992.

[32] F. Neri and C. Cotta, “Memetic algorithms and memeting computing optimization: A literature review,” *Swarm and Evolutionary Computation*, vol. 2, pp. 1–14, 2012.

[33] Q. H. Nguyen, Y. S. Ong, and M. H. Lim, “Non-genetic transmission of memes by diffusion,” in *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, ser. GECCO ’08. New York, NY, USA: ACM, 2008, pp. 1017–1024.

[34] G. Ochoa, M. Hyde, T. Curtois, J. Vazquez-Rodriguez, J. Walker, M. Gendreau, G. Kendall, B. McCollum, A. Parkes, S. Petrovic, and E. Burke, “Hyflex: A benchmark framework for cross-domain heuristic search,” in *European Conference on Evolutionary Computation in Combinatorial Optimisation, EvoCOP ’12.*, ser. LNCS, J.-K. Hao and M. Middendorf, Eds., vol. 7245. Heidelberg: Springer, 2012, pp. 136–147.

[35] Y.-S. Ong and A. Keane, “Meta-lamarckian learning in memetic algorithms,” *Evolutionary Computation, IEEE Transactions on*, vol. 8, no. 2, pp. 99–110, 2004.

[36] Y.-S. Ong, M.-H. Lim, N. Zhu, and K.-W. Wong, “Classification of adaptive memetic algorithms: a comparative study,” *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 36, no. 1, pp. 141–152, 2006.

[37] E. Özcan and A. Alkan, “A memetic algorithm for solving a timetabling problem: An incremental strategy,” in *Proceedings of the third Multidisciplinary International Conference On Scheduling: Theory and Applications*, 2007, pp. 394–401.

[38] E. Özcan, M. Mısır, 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.

[39] E. Özcan, “An empirical investigation on memes, self-generation and nurse rostering,” in *Proc. of the 6th International Conference on the Practice and Theory of Automated Timetabling*. Citeseer, 2006, pp. 246–263.

[40] 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.

[41] E. Özcan and E. Onbaşıoğlu, “Memetic algorithms for parallel code optimization,” *International Journal of Parallel Programming*, vol. 35, no. 1, pp. 33–61, 2007.

[42] E. Özcan, A. J. Parkes, and A. Alkan, “The interleaved constructive memetic algorithm and its application to timetabling,” *Computers & Operations Research*, vol. 39, no. 10, pp. 2310–2322, 2012.

[43] D. Qaurooni, “A memetic algorithm for course timetabling,” in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, ser. GECCO ’11. New York, NY, USA: ACM, 2011, pp. 435–442.

[44] P. Ross, J. Marín-Blázquez, S. Schulenburg, and E. Hart, “Learning a procedure that can solve hard bin-packing problems: A new ga-based approach to hyper-heuristics,” in *Genetic and Evolutionary Computation - GECCO 2003*, ser. Lecture Notes in Computer Science, E. Cantú-Paz, J. Foster, K. Deb, L. Davis, R. Roy, U.-M. O’Reilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. Potter, A. Schultz, K. Dowsland, N. Jonoska, and J. Miller, Eds. Springer Berlin Heidelberg, 2003, vol. 2724, pp. 1295–1306.

[45] J. Smith, “Coevolving memetic algorithms: A review and progress report,” *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 37, no. 1, pp. 6–17, 2007.

[46] J. Swan, E. Özcan, and G. Kendall, “Hyperion - a recursive hyper-heuristic framework.” in *LION*, ser. Lecture Notes in Computer Science, C. A. C. Coello, Ed., vol. 6683. Springer, 2011, pp. 616–630.