A Self-adaptive Multimeme Memetic Algorithm Co-evolving Utility Scores to Control Genetic Operators and Their Parameter Settings

Ender Özcan[a], John H. Drake[b], Cevriye Altıntaş[c], Shahriar Asta[a]

[a] *ASAP Research Group, School of Computer Science, University of Nottingham, NG8 1BB, Nottingham, UK*
[b] *The OR Group, Queen Mary University of London, Mile End Road, London, E1 4NS, UK*
[c] *Süleyman Demirel University, Isparta, Turkey*

Abstract

Memetic algorithms are a class of well-studied metaheuristics which combine evolutionary algorithms and local search techniques. A *meme* represents contagious piece of information in an adaptive information sharing system. The *canonical* memetic algorithm uses a fixed *meme*, denoting a hill climbing operator, to improve each solution in a population during the evolutionary search process. Given global parameters and multiple parametrised operators, adaptation often becomes a crucial constituent in the design of MAs. In this study, a *self-adaptive* self-configuring steady-state multimeme memetic algorithm (SSMMA) variant is proposed. Along with the individuals (solutions), SSMMA co-evolves memes, encoding the utility score for each algorithmic component choice and relevant parameter setting option. An individual uses tournament selection to decide which operator and parameter setting to employ at a given step. The performance of the proposed algorithm is evaluated on six combinatorial optimisation problems from a cross-domain heuristic search benchmark. The results indicate the success of SSMMA when compared to the static MAs as well as widely used self-adaptive Multimeme Memetic Algorithm from the scientific literature.

Keywords: Memetic Algorithms, Multimeme Memetic Algorithms, Reinforcement Learning, Hyper-heuristics, Combinatorial Optimisation

1. Introduction

Many real-world optimisation problems create a search space which is so large that it becomes impractical to exhaustively search all possible states. Heuristics, in the field of optimisation, seek to find good quality solutions to a problem given a 'rule of thumb' using some intuition, with no guarantee of solution quality. In essence, a *metaheuristic* is a high-level strategy which is designed to guide a computational search for any problem where heuristics exist to modify the current state of a given solution [64]. Traditional heuristics are liable to become trapped in local, suboptimal areas of the search space, never to discover the globally optimal solution. Metaheuristics are designed to escape local optima, often offering a good trade-off between computational effort and solution quality.

Memetic Algorithms (MAs) [41] are a hybrid metaheuristic approach to problem solving which combine evolutionary algorithms with local search techniques. The term Memetic Algorithm was first used to cover a number of techniques using the idea of a *meme* [18] as a 'unit of cultural transmission' which is considered analogous to the local search/hill climbing process. MAs have previously been used to solve problems in a large number of different problem domains. The taxonomy provided by Ong et al. [50] identifies three main categories of Memetic Algorithms, considering how the algorithmic components and their settings are decided based on feedback received during the evolutionary search process. The first category of MAs do not use any feedback, hence, they are referred to as *static* MAs. The *canonical* MA which utilises a single hill climbing operator or makes a random choice among multiple operators falls into this category. The second category consists of *adaptive* MAs which embed a method to receive feedback during the search process and influence the choice of a meme at a given decision point. The third category of MAs are *self-adaptive* algorithms that use evolution itself for adaptation and the selection of an appropriate meme. Adaptation strategies are crucial for the second and third types of MAs, determining the overall performance of a given algorithm.

The generic Multimeme Memetic Algorithm (MMA) of Krasnogor and Smith [35] is a *self-adaptive* MA, which co-evolves memetic material encoded into each individual at the same time as evolving solutions to a given problem. Although MMA was proposed as a general method, particular choices have been made regarding adaptation and inheritance. For example each meme option is directly encoded into the memetic material, referred to as a *memeplex*, and is perturbed with a certain probability. Özcan and Onbaşıoğlu [58] used an MMA to solve a parallel processing problem. The authors tested three different strategies to decide which meme to use as a hill climbing operator, showing that this algorithmic choice influences the performance of MMA on this problem.

There are a limited number of studies addressing the issue of how to select a meme in addition to choosing what to encode within a meme. In this study, we introduce a Steady-state Multimeme Memetic Algorithm (SSMMA), a new self-adaptive MA which employs a different encoding of options to a generic MMA. The utility score for each option is separately maintained within each individual in a population and gets updated using a strategy inspired by Reinforcement Learning [30, 57]. Based on these scores, a meme is selected when needed using tournament selection. To the best of authors' knowledge this approach, which diminishes the need for the innovation rate parameter, is used for the first time within MMAs.

One of the main limitations of self-adaptive MAs by Krasnogor and Smith is that the general framework was mostly theoretical and had no actual implementation. The performance of SSMMA is evaluated on six different problem domains including real-world problems from the HyFlex cross-domain heuristic search benchmark [48]. Overall, the empirical results show that the proposed MMA improves upon the original MMA presented by Krasnogor and Smith [35] in many cases.

The subsequent sections of this paper are organised as follows. In Section 2, an overview of Memetic Algorithms and the HyFlex benchmark is given. In Section 3, we describe the details of the Memetic Algorithms which are tested in this study. Section 4 presents the experimental design and parameter settings

used during experimentation before presenting the empirical results and analyses. Finally, some concluding remarks and potential future work are provided in Section 5.

2. Related Work

2.1. Memetic Algorithms

Evolutionary algorithms are a class of search techniques inspired by the natural process of evolution, of which by far the most well-known are Genetic Algorithms (GAs) [25]. A GA iteratively updates a population of solutions through the use of mutation and crossover operators, which modify a solution or recombine multiple solutions respectively. Hill climbing methods for local search are a relatively simple class of heuristics, utilising the concept of locality between candidate solutions for a given problem. A typical hill climbing algorithm moves through the search space from one solution to another non-worsening solution within a neighbourhood, where a neighbour is defined as any state that can be reached from the current solution through some modification. A core challenge when using such methods is the trade-off between intensification and diversification, where intensification refers to the exploitation of the current area of the search space and diversification refers the exploration of new regions of search space [4, 16].

Memetic Algorithms (MAs) were introduced by Moscato [41] as a set of evolutionary algorithms that make heavy use of hill climbing. A simple MA introduces a local search phase into a GA after crossover and mutation have been performed during the evolutionary process. Since their emergence, MAs and subsequent variants of MAs have been applied to a wide variety of problems, including educational timetabling [2, 10, 56, 60, 59], multi-objective optimisation problems [31, 24], permanent-magnet synchronous motor (PMSM) design [11], Optimal Placement of PMUs [37], permutation flow shop scheduling [28], the economic load dispatch problem [51, 52], protein folding [33], HIV therapy design [44], the quadratic assignment problem [39] and the travelling salesman problem

[42, 5, 26]. As well as the application of MAs to practical optimisation problems, a number of studies have sought to understand the concepts underpinning MAs and the behaviour of memes [34]. Indeed the performance of an MA has been observed to be strongly linked to the choice of local search mechanism used [58, 54], with competition and cooperation between different local search methods beneficial to to overall search process [49].

Previous studies have tried to address the issues in the design of MAs. Nguyen et al. [45] highlighted a number of design decisions within MAs, namely, the trade-off between intensification and diversification within the context of MAs, the choice of which individuals to improve if not all solutions undergo local search and choosing which meme to use when multiple memes are available depending on the problem currently being solved. A result of note from this work was that the choice of local search mechanism was found to be more critical to the overall performance of an MA than the choice of underlying population-based search strategy. Acampora et al. [1] tested a large number of MA configurations applied to ontology alignment, based on the issues outlined in Nguyen et al. [45], with the best MA found to be competitive with state-of-the-art ontology alignment systems.

In addition to the traditional MAs described above, a separate branch of evolutionary algorithms which are capable of adapting parameter settings and algorithmic component choices also exist [19, 14]. Self-adaptive Multimeme Memetic Algorithms (MMAs) [35, 33, 32] contain individuals that are made up of both genetic and memetic material. The memes denoting the choices for operators including hill climbing and their settings are co-evolved along with the genes during the search process.

Jakob [29] proposed a cost-benefit adaptation strategy for multi-meme algorithms in which adaptation is guided by the costs and benefits of a local search run. As a by-product, the strategy maintains a balance between intensification and diversification. Smith [63] described a co-evolutionary memetic algorithm in which the depth and resolution of the local search heuristics as well as the pivot function are co-evolved alongside the population of candidate solutions. This

study showed that a simple co-evolutionary model is able to maintain a balance between intensification and diversification. However, extensive experiments in the study failed to establish a logical and problem-independent link between the size of the meme pool and the evaluation accuracy of a meme's value at various points in the search space. Epitropakis et al. [23] used Separability Prototype for Automatic Memes (SPAM), initially proposed by Caraffini et al. [12], in which a success-based adaptation strategy is employed. This strategy, which is based on recent advances in hyper-heuristics, rewards promising memes and increases their chance to be used in subsequent stages of the search. Nogueras and Cotta [47] studied self-adaptation in spatially structured multi-memetic algorithms where populations use specific topologies within which operator interactions are constrained. The authors observed that these MMAs are sensitive to self-adaptation and suggest to use variable length memes to increase robustness for various population structures.

An overview of some adaptation schemes using multiple memes can be found in [50, 62]. Previous studies on learning, adaptation and evolution have led to the more general concept of Memetic Computing which introduces co-evolution, machine learning, cognitive observation of other individuals and memory utilisation for operator and parameter control, as well as the generation of the methods/rules used by them, into the evolutionary search process [13, 43]. In this study, we present a variant of the co-evolutionary MMA with a novel adaptation mechanism for operator selection and parameter setting.

2.2. Hyper-Heuristics and HyFlex

Hyper-heuristics form a class of high-level methods which search a space of low-level heuristics or components, rather than solutions, while solving computationally hard problems [7]. There are two main classes of hyper-heuristic methods: *selection* hyper-heuristics that mix and control a given set of low-level heuristics and *generation* hyper-heuristics that construct low-level heuristics from a set of given components [8]. Hyper-heuristics have been successfully applied to a range of domains including single objective timetabling problems

[9] and multi-objective scheduling problems [46].

Originally hyper-heuristic research set out to develop methods which are more general than the traditional search and optimisation techniques, with recent research focussing considerably on *cross-domain heuristic search*. HyFlex v1.0 (Hyper-heuristics Flexible framework) [48] was introduced mainly to support the first Cross-domain Heuristic Search Challenge (CHeSC 2011) [6]. The results obtained from the selection hyper-heuristics competing in this challenge currently serve as a benchmark for comparison of selection hyper-heuristic methods. The HyFlex v1.0 framework provides means for rapid implementation and performance evaluation of high-level adaptive search methods. HyFlex v1.0 imposes a logical barrier between the problem domain implementation and the high-level method controlling a set of low-level operators operating over the domain. This barrier does not allow any problem domain specific information, such as solution representation, to be accessed by the high-level method. HyFlex v1.0 supports six problem domains, including Boolean Satisfiability (MAX-SAT), One Dimensional Bin Packing (BP), Permutation Flow Shop (PFS), Personnel Scheduling (PS), Travelling Salesman Problem (TSP) and Vehicle Routing Problem (VRP). Each domain comes with a set of low-level heuristics (operators) from four different categories: (i) mutation, (ii) local search/hill climbing, (iii) ruin-recreate, and (iv) crossover. Each operator implemented in a domain is given a unique ID and a high-level method can access a chosen operator using this ID. Table 1 provides the ID and type of each low-level heuristic for each Hyflex v1.0 problem domain, where C, M, R and L indicate a crossover, mutation, ruin-recreate, and hill climbing operator respectively. As an example, Bin Packing has eight low-level heuristics in total: one crossover, three mutation, two ruin-recreate and two hill climbing. These IDs will be used later in Section 3.1 to identify particular memeplexes. Permutation Flow Shop has the most low-level heuristics available, with 15 in total.

Ruin-recreate operators form a partial solution using a given complete solution, and then rebuild a complete solution. In this study, ruin-recreate operators are considered as mutation operators, since they do not guarantee a

Table 1: The nature of each low level heuristic with a given unique ID implemented in each HyFlex v1.0 problem domain.

Domain	LLH0	LLH1	LLH2	LLH3	LLH4	LLH5
MAX-SAT	M_1	M_2	M_3	M_4	M_5	M_6
BP	M_1	R_1	R_2	M_2	L_1	M_3
PS	L_1	L_2	L_3	L_4	L_5	R_1
PFS	M_1	M_2	M_3	M_4	M_5	R_1
TSP	M_1	M_2	M_3	M_4	M_5	R_1
VRP	M_1	M_2	R_1	R_2	L_1	C_1
Domain	LLH6	LLH7	LLH8	LLH9	LLH10	LLH11
MAX-SAT	R_1	L_1	L_2	C_1	C_2	
BP	L_2	C_1				
PS	R_2	R_3	C_1	C_2	C_3	M_1
PFS	R_2	L_1	L_2	L_3	L_4	C_1
TSP	L_1	L_2	L_3	C_1	C_2	C_3
VRP	C_2	M_3	L_2	L_3		
Domain	LLH12	LLH13	LLH14			
PFS	C_2	C_3	C_4			
TSP	C_4					

non-worsening solution as output. All crossover operators in HyFlex take two solutions as input, then recombine them to yield a new solution. Additionally, the mutation and hill climbing low-level heuristics have an adjustable parameter to modify their behaviour. In the case of mutation and ruin-recreate, the *intensity of mutation* parameter is a value in [0, 1] specifying the extent to which a low-level heuristic will perturb a solution. Setting this parameter to a higher value will result in a mutation low-level heuristic performing larger perturbations to the original solution. In the case of hill climbing, the *depth of search* parameter will dictate the number of steps executed during local search. This parameter also takes a value in [0, 1], with a higher value indicating a greater number of local search steps. More details on the implementation of problem domains, including specifics of the low-level heuristics, and the selection hyper-heuristics that competed in the challenge can be found at the CHeSC 2011 website[1].

Four different frameworks for selection hyper-heuristics were described by Özcan et al. [55]. Their work observed that a selection hyper-heuristic based on a framework distinguishing between mutation and hill climbing operators outperforms traditional approaches which use all operators together. This framework used the idea of explicitly enforcing diversification and intensification processes in hyper-heuristics, in a similar way to other existing approaches such as iterated local search [38] and MAs. Özcan et al. [53] evaluated the performance of two static MAs on the six problem domains of the HyFlex benchmark described above. The first approach, a Steady-State Memetic Algorithm (SSMA), is a traditional MA which applies a random crossover operator to two individuals from the population before applying a random mutation and hill-climbing operator to the resultant solution. The new individual then replaces the worst individual in the current population. The second, a Transgenerational Memetic Algorithm (TGMA), generates an entire new population at each generation rather than using steady-state replacement to update the existing population. Both SSMA

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

and TGMA generate the first population using the initialisation methods provided with the problem domains. They do not control the parameter settings for the operators and they make a random choice from five different discrete settings. Both approaches are *static* MAs with *external adaptation* under the taxonomy of [50]. The experimental results in [53] showed that SSMA performs better than TGMA overall, however most of the single-point-based search selection hyper-heuristics entrants to CHeSC 2011 outperform both MAs.

Although the hyper-heuristic and MA research communities have evolved separately, they are closely linked, with a number of shared goals. In the case of multiple operators and a range of values to choose from for parameter setting, adaptation is crucial. In this study, we propose a new steady state Multimeme Algorithm which employs a new encoding for the adaptation of genetic operators and their parameter settings. We compare its performance to MMA [35] on six problem domains of the HyFlex benchmark.

3. Self-Adaptive Memetic Algorithms for Cross-Domain Heuristic Search

As discussed in the previous sections, *self-adaptive* MAs inherit memes across generations [50]. The Multimeme Memetic Algorithm (MMA) of Krasnogor and Smith [35] is a canonical self-adaptive MA, which co-evolves a set of local-search heuristics acting as memes, at the same time as evolving solutions to computationally difficult problems. Each individual in an MMA consists of two parts, its own independent genetic material (chromosome) and memetic material (memeplex). In the context of hyper-heuristics, MMAs are able to adaptively select appropriate low-level heuristics to use for different problem instances, different stages of a given search or for different individuals in a population. Individual memes can adapt through changes to their parameter settings or the neighbourhoods in which they operate. Here we will present a new memeplex structure for MMAs and compare to the MMA of Krasnogor and Smith [35]. Due to the success of using a steady state approach as reported by Özcan et al. [53], the same replacement scheme is used in this study. The proposed MMA

variant, a Steady-state Multimeme Memetic Algorithm (SSMMA), supports the adaptation of genetic operators including hill climbing and their settings through the use of Reinforcement Learning [30] with the memeplex encoding. MMA and SSMA both make use of Lamarckian learning, placing an improved individual back into the population to compete within the next generation of evolution, and *local-level* adaptation as defined by Ong et al. [50]. Local-level adaptation indicates that some previous historical knowledge is used when deciding which meme to use among multiple memes. MMA and SSMMA are implemented as a high-level search method within HyFlex, enabling us to test their performance on six different problem domains. We also compare their performance to two other *static* MAs studied in [53] and a number of previously proposed single-point-based search selection hyper-heuristics from the scientific literature.

3.1. Multimeme Memetic Algorithm (MMA)

The MMA of Krasnogor and Smith [35], using steady state replacement, is implemented as shown in Algorithm 1. Each individual in the population consists of a candidate solution and a memeplex containing the memetic information associated with that individual. A memeplex, denoted as $\mathrm{C}_c\{\mathrm{M} \text{ or } \mathrm{R}\}_m\mathrm{I}_i\mathrm{L}_l\mathrm{D}_d$, represents the c^{th} crossover operator (C), m^{th} mutation (M) or ruin-recreate operator (R) using an *intensity of mutation* (I) parameter setting of $i \in [0, 1]$ and l^{th} hill climbing method (L) using a *depth of search* (D) parameter setting of $d \in [0, 1]$. As an example, the MAX-SAT domain of HyFlex contains two crossover, seven mutation (one of them being a ruin-recreate operator) and two hill climbing operators (labelled LLH0-LLH10 in Table 1 above). Assuming that there are five discrete choices for each parameter setting, i.e. $\{0.2, 0.4, 0.6, 0.8, 1.0\}$, a memeplex for this domain can encode $2 \times 7 \times 5 \times 2 \times 5 = 700$ different memes. Given $\mathrm{C}_1\mathrm{M}_2\mathrm{I}_{0.6}\mathrm{L}_1\mathrm{D}_{0.8}$, this meme denotes that crossover with low-level heuristic LLH9, mutation with low-level heuristic LLH1 having an *intensity of mutation* setting of 0.6 and hill climbing with low-level heuristic LLH7 having a *depth of search* setting of 0.8 are going to be used at a given decision point during the evolutionary search process.

The first generation of candidate solutions is created using the initialisation methods provided for each HyFlex domain, with values for each of the five components of the memeplex initialised randomly. At this point, a hill climbing heuristic is applied to each initial individual for improvement. Following the initialisation phase, the evolutionary cycle starts its execution. Tournament selection is used to choose two individuals from the population with which to perform crossover. Each parent is selected by comparing the fitness (objective value/quality of a solution) of *'tournament size'* randomly selected individuals from the population and choosing the one with the better fitness. In the case of equal quality individuals, an arbitrary individual is selected. The stronger of the two parents with the better fitness is designated Parent_1. A new Child solution is generated by performing crossover between the solutions of the two parents using the crossover operator from the meme of Parent_1. The *Simple Inheritance Mechanism* as proposed by Krasnogor and Smith [35] is utilised, which propagates memetic material from one generation to the next by directly passing the meme of a parent to a child. The mutation operator encoded in the inherited meme is then applied to the solution using the *intensity of mutation* parameter setting from the same memeplex. During the mutation process, memetic material is also mutated with a probability labelled as *Innovation Rate* (IR). A meme is mutated to another option value randomly. This mechanism is used to introduce 'new' memetic material into the evolutionary process. If IR is high then the memes will be perturbed in each generation and each algorithmic component choice will be almost random. If IR is low, then new memetic material might not be introduced and the search could stagnate. Following mutation, the hill climbing operator is applied to the Child solution with the given *depth of search* parameter setting from the meme. The evolutionary cycle continues until the termination criteria is satisfied.

3.2. Steady-state Multimeme Memetic Algorithm (SSMMA)

The MMA described above needs the IR parameter to ensure that every potential algorithmic choice has a non-negative chance of appearing during the

Algorithm 1 Pseudocode of Multimeme Memetic Algorithms (MMA) [35]

1: Create a population of $popSize$ random individuals.
2: Apply a random hill climber to each individual
3: **while** termination criteria is not satisfied **do**
4: Parent$_1$ ← Select-Parent(population, tour-size)
5: Parent$_2$ ← Select-Parent(population, tour-size)
6: **if** Parent$_2$ *has a better fitness than* Parent$_1$ **then**
7: Swap(Parent$_1$, Parent$_2$)
8: **end if**
9: Child ← ApplyC(Parent$_1 crossover$, Parent$_1$, Parent$_2$)
10: Copy the *meme* of Parent$_1$ to Child
11: Child ← ApplyM(Child$mutation$, Child$intensity$, Child)
12: *Mutate meme of Child with a probability of* IR
13: Child ← ApplyL(Child$hillclimber$, Child$depth$, Child)
14: Child replaces the worst individual in the population
15: **end while**

search process. Each meme directly encodes the choices of algorithmic components to use. The memetic material used during evolution is restricted to the material that the first population introduces, particularly if the IR setting is low. Increasing the innovation rate increases the likelihood of (re)introducing new memes into the search process. In previous studies, a fixed setting is suggested for IR, however, managing the interaction between IR and population size becomes crucial, especially if the number of potential memes is large which is the case in this study. Using the simple inheritance mechanism, the likelihood of good memes propagating thorough generations increases with the population size. As a result the population size arguably represents the memory size, with IR representing the probability of forgetting and refreshing that memory.

Here we propose a new variant of a *self-adaptive* MA, a Steady-state Multimeme Memetic Algorithm (SSMMA). In this approach we build on the original MMA by introducing a feedback loop, increasing the potential to remember previous performance in a way that is not limited by population size. Rather than simply encoding a specific operator or parameter setting in each meme, a set of *utility scores* for each possible option is maintained as a meme. The structure of this encoding is shown in Figure 1. The proposed approach increases the memory requirement for the memes, however this increase is reasonable considering that it will be within a factor of the number of algorithmic components. For example, for the MAX-SAT domain of HyFlex, a meme consists of $2+7+5+2+5=21$ entries where each entry holds the utility score (or option value) for each algorithmic component choice. The pseudocode for SSMMA is given in Algorithm 2.

Initially the first generation of candidate solutions is created using the methods in HyFlex and a hill climbing heuristic is applied to every individual, with the utility scores in the memeplex set to 0. The evolutionary cycle operates in almost the same manner as MMA. Tournament selection is again used to choose two individuals from the population with which to perform crossover, with the stronger of the two parents with the best fitness taken to be Parent_1. A crossover low-level heuristic, mutation low-level heuristic, local search low-level heuristic

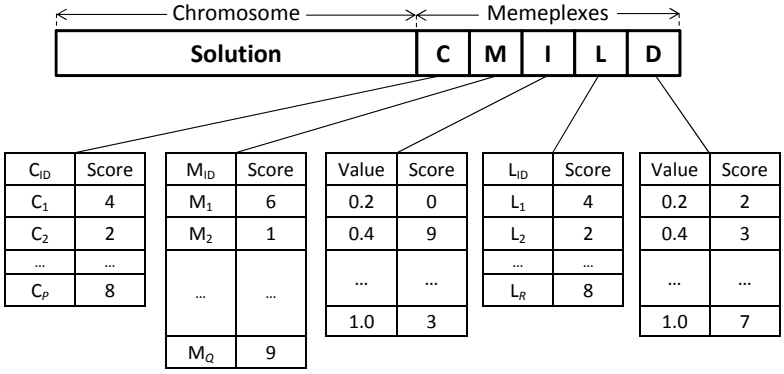


Figure 1: Encoding of an individual in SSMA

Algorithm 2 Pseudocode of Steady-state Multimeme Memetic Algorithm (SSMMA)

1: Create a population of *popSize* random individuals.
2: Apply a random hill climber to each individual
3: **while** termination criteria is not satisfied **do**
4: Parent$_1$ ← Select-Parent(population, tour-size)
5: Parent$_2$ ← Select-Parent(population, tour-size)
6: **if** Parent$_2$ *has a better fitness than* Parent$_1$ **then**
7: Swap(Parent$_1$, Parent$_2$)
8: **end if**
9: Tournament selection of Parent$_1$*crossover*, Parent$_1$*mutation*, Parent$_1$*hillclimber*, Parent$_1$*intensity* and Parent$_1$*depth*
10: Child ← ApplyC(Parent$_1$*crossover*, Parent$_1$, Parent$_2$)
11: Child ← ApplyM(Parent$_1$*mutation*, Parent$_1$*intensity*, Child)
12: Child ← ApplyL(Parent$_1$*hillclimber*, Parent$_1$*depth*, Child)
13: Copy the *meme* of Parent$_1$ to Child
14: Update the scores of the *meme* of Child
15: Child replaces the worst individual in the population
16: **end while**

and values for *intensity of mutation* and *depth of search* are selected (using tournament selection), based on the scores encoded in the meme of Parent$_1$. In each case, the operator or parameter setting with the highest value within the tournament size of options, indicating superior past performance, is favoured. In the case of equal scores, an arbitrary option is selected. For example, assume that the solver is operating on a MAX-SAT problem instance and it is time for mutation, and score for each one of the seven mutation operators is given as [1, 0, 4, 2, 3, 3, 5]. As an example, if the tournament size is 2 and 0^{th} and 6^{th} operators are chosen, then based on their scores which are 1 and 5 respectively, the 6th operator with the highest value option would be chosen. In the case of equal scores, an operator is chosen arbitrarily. The crossover low-level heuristic is applied to recombine the two selected parents and generate a new solution, the mutation low-level heuristic is applied using the *intensity of mutation* parameter and finally the local search low-level heuristic is applied with the selected *depth of search* parameter. Finally the *meme* of Parent$_1$ is copied to the Child individual. In the case that the Child solution is an improvement over the parent, the scores for each of the five memes within the memeplex of the Child are modified using an additive Reinforcement Learning scheme [57], with each value incremented by 1. In the case that the Child solution is of equal or poorer quality than the parent then no modification is made to the scores within the memeplex of the Child. This is similar to the *Simple Inheritance Mechanism* proposed by Krasnogor and Smith [35] which propagates memetic material from one generation to the next by directly passing the meme of a parent to a child.

4. Experimental Results

4.1. Experimental Design and Parameter Settings

In order to evaluate the performance of the proposed Steady-state Multi-meme Memetic Algorithm (SSMMA), we have used the CHeSC 2011 dataset which contains five instances per domain and six different problem domains as introduced in Section 2.2. All problems in HyFlex are modelled as minimisation

problems, with a lower objective value indicates a better fitness. The performance of SSMMA is compared to the self-adaptive MMA [35] and two static MAs from Özcan et al. [53] over 30 instances. In both SSMMA and MMA, population size is set to 10 in line with the MAs of Özcan et al. [53]. The *intensity of mutation* and *depth of search* parameters are able to take one of a set of five discrete values in $\{0.2, 0.4, 0.6, 0.8, 1.0\}$. In MMA, the *innovation rate* (IR) in MMA is fixed as 0.2 [35, 58, 54], parent selection is performed using tournament selection with a *tournament size* of 2 and all memetic values are initialised randomly. In SSMMA tournament selection with *tournament size* 2 is used for both parent selection, and meme selection. In both cases, the parent or meme with highest score is kept with the other discarded. The scores for each low-level heuristic and parameter setting of each meme are initially set to 0, giving each option an equal chance of selection.

All experiments were performed on an Intel Core Duo 3.16 GHz machine with 2 GB RAM. Each trial is run for a notional duration of 600 seconds and repeated 31 times, in line with the standard used by the CHeSC 2011 organisers. The original duration was determined based on the configuration of the machine used for ranking the competitors. The organisers provided a benchmarking tool to compute the equivalent time required for termination of a trial on another machine. The equivalent duration using our machine calculated by the benchmark tool is 576 seconds. Although this is not the primary goal of this study, using the same termination criterion as in CHeSC 2011 enables us to compare the performance of memetic algorithms to a range of adaptive heuristic optimisation methods which competed at CHeSC 2011, including the state of the art methods [48].

Using this termination criteria gives us a fair base for performance comparison of algorithms for cross domain search, where the run-time efficiency of heuristic implementations could vary not only for a given domain, but from one domain to another. It is worth noting that the execution time of a program does not only depend on the CPU speed. There are other influential factors ranging from memory to operating system of the computer used. Although benchmark-

ing tools make an attempt to take such factors into account, the use of time limit for termination could still pose some validity risk [17] on the precision of results obtained from a time-contract algorithm which terminates when the given time expires.

4.2. Performance comparison of SSMMA and MMA

Firstly we will compare the performance of SSMMA and MMA [35] using six HyFlex problem domains. Table 2 shows the average and best fitness values obtained by each method over 31 runs of each instance used in the CHeSC 2011 competition, with the better of the two methods given in **bold**. Where this table refers to a particular ID number, this represents the order in which this instance appears in the list of the instances tested for CHeSC 2011, rather than the HyFlex index of that problem instance. The *vs.* column indicates the results of Wilcoxon signed-rank tests within a 95% confidence interval, which is performed to assess the statistical significance of the performance variation of two algorithms using the results from 31 runs for each instance. Here, $>$ ($<$) denotes that SSMMA (MMA) performs statistically significantly better than MMA (SSMMA) on the given instance. Additionally, $\geq$ ($\leq$) denotes that there is no significant performance variation between the two methods, however SSMMA (MMA) performs slightly better on average.

SSMMA performs better than the original MMA in some problem domains, however each of the MAs have different strengths in different problem domains. SSMMA is very strong in the case of MAX-SAT, Bin Packing and Personnel Scheduling, outperforming MMA with a statistically significant difference in all five instances of MAX-SAT and four of the five instances of Personnel Scheduling in terms of both average performance and best solution found. For Bin Packing, SSMMA outperforms MMA in all five instances, however this difference is not statistically significant. Conversely, in the case of Permutation Flow Shop and the Travelling Salesman Problem, MMA is statistically significantly outperforming SSMMA in terms of average and best performance in all problem instances of these two domains. Although SSMMA is designed to be more

Table 2: Performance comparison of SSMMA and MMA based on best and average fitness obtained from 31 trials for each instance of CHeSC 2011

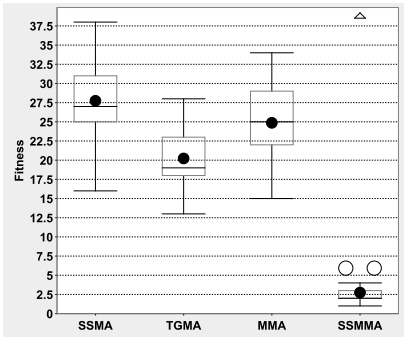
		SSMMA			MMA	
domain	ID	avr.	best	vs.	avr.	best
SAT	1	**3.129**	**1.0**	>	18.709	10.0
	2	**2.774**	**1.0**	>	51.258	24.0
	3	**2.225**	**0.0**	>	33.967	9.0
	4	**2.741**	**1.0**	>	24.870	15.0
	5	**8.387**	**7.0**	>	18.032	14.0
BP	1	**0.0345**	**0.0193**	≥	0.0551	0.0483
	2	**0.0031**	**0.0272**	≥	0.0097	0.007
	3	**0.0077**	**0.0038**	≥	0.0151	0.0112
	4	**0.1091**	**0.1087**	≥	0.109	0.1088
	5	**0.0194**	**0.0107**	≥	0.0262	0.0208
PS	1	**31.419**	**25.0**	>	32.451	26.0
	2	**10129.32**	**9815.0**	>	13829.45	10255.0
	3	**3232.806**	**3160.0**	>	3465.806	3242.0
	4	**1781.613**	**1585.0**	>	2045.226	1665.0
	5	380.225	345.0	≤	**353.903**	**321.0**
PFS	1	6342.581	6306.0	<	**6253.161**	**6238.0**
	2	26953.38	26914.0	<	**26840.52**	**26773.0**
	3	6387.903	6369.0	<	**6347.645**	**6303.0**
	4	11507.32	11472.0	<	**11401.45**	**11377.0**
	5	26753.65	26674.0	<	**26662.61**	**26603.0**
TSP	1	49032.385	48722.154	<	**48239.358**	**48194.921**
	2	2.138E7	2.1186E7	<	**2.111E7**	**2.0896E7**
	3	6996.01	6955.616	<	**6824.295**	**6800.543**
	4	70430.13	69284.88	<	**67608.168**	**66534.788**
	5	58200.41	55752.75	<	**54299.139**	**53010.404**
VRP	1	**69122.21**	66244.735	>	72159.779	**63373.559**
	2	**12740.57**	**12303.28**	>	13768.666	13335.578
	3	158509.3	148839.8	<	**148975.804**	**144089.757**
	4	**20670.185**	**20655.206**	>	21321.299	20656.798
	5	152092.27	**148151.9**	<	**150243.193**	148397.623

flexible and adaptive than the original MMA, it is clearly not performing well in these problem domains. The case of the Vehicle Routing Problem is slightly more complex with both SSMMA and MMA statistically significantly outperforming one another in different instances. It is also the case in Vehicle Routing Problem Instance 1 and Vehicle Routing Problem Instance 5 that the method performing best on average does not find the best result of the two over 31 runs. As MMA has a shorter memory length than SSMMA, these results could be an indication that SSMMA needs a better mechanism to forget and refresh its memory. However, such a mechanism would require the design of multiple components, including a method to decide when to forget and another to decide how to update which utility score, adding significant complexity to the proposed simple approach.

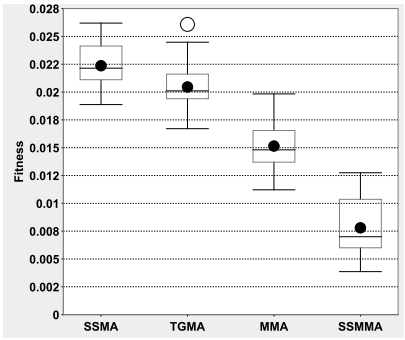
4.3. Performance comparison of static versus self-adaptive MAs

Figure 2 shows box and whisker plots of the best fitness values obtained from 31 runs on a selected instance from each of the six problem domains using four MAs. In addition to the self-adaptive SSMMA and MMA we also report the results for the static MAs, SSMA and TGMA, presented by Özcan et al. [53]. The results of a Freidman test [61] within a 95% confidence interval over the median results of each of the 30 problems tested indicate that MMA and SSMMA perform statistically significantly better than SSMA and TGMA. No statistically significant difference is shown between SSMMA and TGMA using this test. These box and whisker plots give us some indication of the performance consistency of a given MA over the 31 runs of each instance. Rather than provide all 30 figures for each individual instance, these six instances have been chosen as representative of each problem domain. Any significant variations within problem domains are highlighted and discussed below.

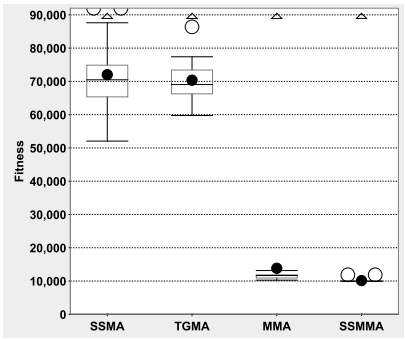
In the case of MAX-SAT, shown in Figure 2(a), SSMMA clearly outperforms the other three MAs on average. The relatively small distance between the first and third quartiles indicate that SSMMA is consistently achieving good quality solution values on this problem instance. This instance is typical of all five in-



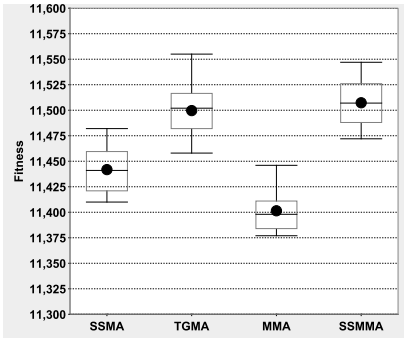
(a) MAX-SAT - Instance 4



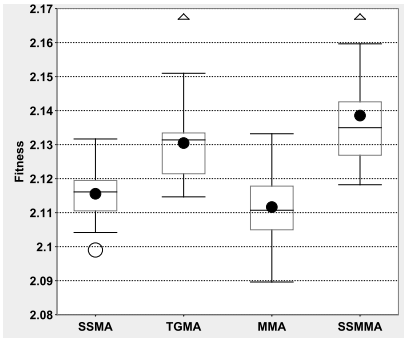
(b) BP - Instance 3



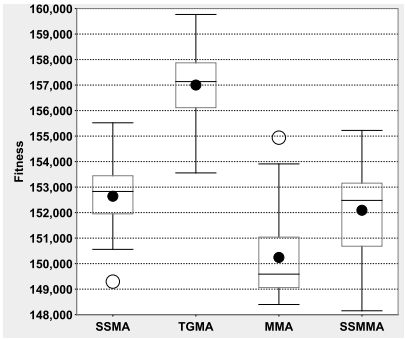
(c) PS - Instance 2



(d) PFS - Instance 4



(e) TSP - Instance 2



(f) VRP - Instance 5

Figure 2: Boxplot of the best fitness values obtained from 31 runs of Memetic Algorithms on a sample instance from each CHeSC 2011 problem domain

stances of MAX-SAT, with SSMMA clearly and consistently achieving the best results and TGMA the second best method ahead of MMA and SSMA. For Bin Packing, the relative performance of all four MAs is again similar in all five problem instances, with Bin Packing Instance 3 shown as a typical example. SSMMA is again the best performing MA on average in each of the five instances. On Bin Packing, self-adaptive MAs perform better than the static MAs. Similarly, Figure 2(c) shows the results for the Personnel Scheduling domain, in which a clear performance split between the self-adaptive MAs and static MAs can be observed. Learning through evolution seems to make a difference in these two cases, with the design of operators in Bin Packing and Personnel Scheduling enabling the more dynamic self-adaptive MAs to learn the best choice of algorithmic components during the search process.

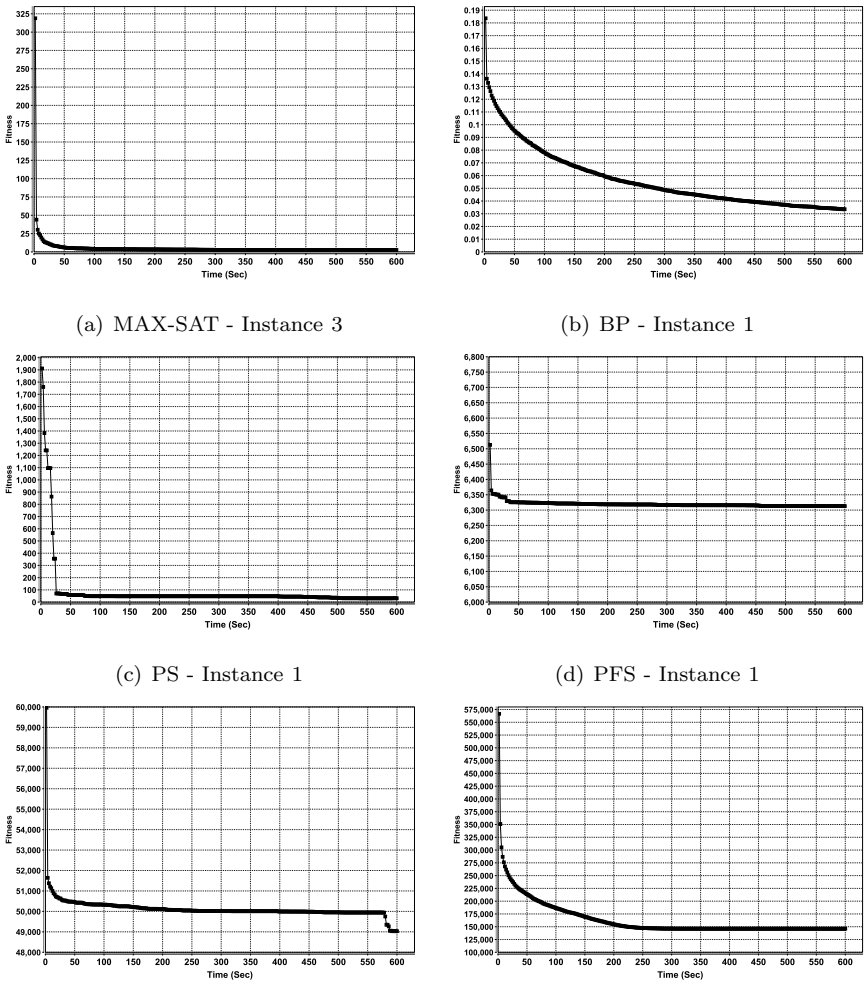
In both the Permutation Flow Shop and Travelling Salesman Problem domains, SSMMA is the worst of the four methods tested as can be seen in Figure 2(d) and Figure 2(e). Typically in both of these problem domains SSMA and MMA offer similar performance, ahead of TGMA. The poor performance of SSMMA in these problem domains may be an unintended consequence of the Reinforcement Learning scheme which does not 'forget' while controlling meme selection. SSMMA focusses on the improvement of solution quality and therefore intensification over diversification within the search. If at one point of the search a particular set of memes performs well, dominating the Reinforcement Learning scores, some other memes may have a very low chance of selection. It could be the case that those other memes are more beneficial to the search later on, however with such low probability of selection, those memes will struggle to improve their score and therefore their chance of selection within the time limit. This is not the case within SSMA and MMA, where when mutation occurs, each meme has equal probability of being reintroduced and applied. Finally in the Vehicle Routing Problem, we have included the box and whisker diagram for Instance 5 of the competition set. This instance is one of the cases where SSMMA was outperformed by MMA on average in Table 2. Again, in general the self-adaptive MAs outperform the static MAs of Özcan et al. [53] in this

domain, with SSMMA outperforming MMA in some instances and vice versa in others.

4.4. An analysis of the behaviour of SSMMA

This section provides an analysis of the performance and behaviour of SSMMA on a per-domain basis. Figure 3 shows the progress of the best fitness value observed with respect to time for a sample run of SSMMA on a single instance selected from each of the six CHeSC 2011 problem domains. In all cases the first instance from the competition test set is used.

In all six problem domains there is a large improvement made in the best-of-run solution quality within the early stages of the search. This indicates that typically upon solution initialisation, the starting point for each search is in an area of the search space containing very poor quality solutions. The rate of improvement in solution quality appears to be more drastic in some cases than others depending on the problem domain. In MAX-SAT and Personnel Scheduling, SSMMA converges to a high quality solution rapidly, while in Permutation Flow Shop, in which SSMMA does not perform well on average, the search seems to get stuck at a local optimum very quickly. In these domains, very little improvement in the best solution found occurs after the first 100 seconds of running time. In the remaining three problem domains, Bin Packing, Travelling Salesman Problem and Vehicle Routing Problem, the convergence to the final best-of-run solution is somewhat more gradual, with improvements observed later on in the search. In the case of Bin Packing in particular, a large number of small improvements are observed almost continually throughout the run. Arguably in this domain the search has not converged on the best solution it is capable of achieving within the time limit. For the Travelling Salesman Problem, after an initial sharp improvement in best-of-run solution, the amount of improvement levels off to a more gradual rate, however better quality solutions are again observed later on in the search. This shows that in this domain, the best quality solutions that this method is capable of finding are again not necessarily found within the fixed time limit.



(a) MAX-SAT - Instance 3

(b) BP - Instance 1

(c) PS - Instance 1

(d) PFS - Instance 1

(e) TSP - Instance 1

(f) VRP - Instance 3

Figure 3: Best-of-run fitness value plotted against time from a sample run of SSMMA on a sample instance from each CHeSC 2011 problem domain

An *evolutionary activity* plot provides the progress of the cumulative utility score of a meme in a population at each time step. If a meme is not used, then it is ignored in the plots. In an evolutionary activity plot, slope denotes the rate of increase in the utility score of the related meme among the individuals in the population during the evolutionary process. Figures 4–9 provide the evolutionary activity plot of memes based on their utility scores averaged over 31 trials for each HyFlex problem domain separately. Part (a) in each figure focusses on the memes encoding combination of crossover, mutation and hill climbing operators while part (b) provides the evolutionary activity plot of the best performing operator combination taking into account its parameter settings. For the ease of visualisation, instead of the ID of a parameter setting, its exact value is used in the plots. The evolutionary activity plots show that two memeplexes differing in one of the genetic operators are favoured over the others overall. The parameter settings for *intensity of mutation* and *depth of search* vary depending on the problem instance.

Figure 4(a) shows that the meme $C_1M_4L_1$ performs the best on the Instance 3 in MAX-SAT, however, this figure ignores the parameter settings of the mutation and hill climber operators. Figure 4(b) provides the evolutionary activity plot of all memes including their parameter settings for that combination of operators, which shows that adaptation preferred low values (0.2) for *intensity of mutation* and *depth of search* for this particular instance. In general, these figures provide some insight into particular operators and parameter settings that are performing well in the instances considered. Figures 5(a) and 5(b) show that in the case of Instance 1 of the Bin Packing domain, L_1 (ID#4) is a preferable hill climbing heuristic in combination with most other memes and for the best set of operators, the *intensity of mutation* setting is best at a low value of 0.2. For the Personnel Scheduling problem and Travelling Salesman Problem (Figure 6 and 8), whilst there are clearly better operator choices evidenced by their high likelihood of appearing in high quality memeplexes, the choice of *intensity of mutation* and *depth of search* does not seem to influence performance. This is particularly true in the case of the Travelling Salesman

Problem, a domain that SSMMA seems to perform poorly in overall. In the case of Permutation Flow Shop and the Vehicle Routing Problem the combination of two operators is particularly effective ($C4_1$ and R_1 for Permutation Flow Shop and C_1 and L_2 for the Vehicle Routing Problem). Also for these two domains there is a clear distinction in performance related to *depth of search* parameter setting, with lower values clearly performing better but little difference can be observed between different values for *intensity of mutation*.

As seen in the previous subsections, the proposed adaptation method with long-term memory outperforms previous MAs which either contain no learning or learning on a more short-term basis. Despite this, SSMMA does not particularly perform well on the Permutation Flow Shop and Traveling Salesman Problem domains. The sample evolutionary activity plots from these two domains reveal that there could be several reasons for this poor performance. All memes in the Traveling Salesman Problem seem to have been used and the approach cannot figure out the best parameter setting for the relevant operators either. Although SSMMA detects several useful memes in Permutation Flow Shop, those memes cause the search to get stuck. Remembering that SSMA using random choice with a fixed parameter setting performs better than SSMMA in both of these domains, the proposed adaptation method is struggling to detect a similar strategy. The search process turns into almost a random walk for Traveling Salesman Problem, while the adaptation cannot recover from initially detected extremely successful memes which seem to mislead the search process causing it to get stuck for Permutation Flow Shop within the given limited time.

4.5. Performance comparison of SSMMA and MMA to the CHeSC 2011 entrants

The goal of this study is to provide a novel Multimeme Memetic Algorithm using a novel encoding, with the potential to outperform generic Multimeme Algorithms, not to provide a hyper-heuristic approach which performs well across different domains. However as we have utilised the HyFlex cross-domain search framework as a benchmark and testbed, this enables us also to compare SSMMA

and MMA in terms of relative performance against the selection hyper-heuristics which competed in CHeSC 2011. Following the competition, median results of 31 runs for each of the 30 problem instances were made available for each of the 20 competitors. SSMMA and MMA are compared independently to the 20 CHeSC 2011 entrants, based on the median score obtained in 31 runs of each instance, using the scoring metric used in the competition. CHeSC 2011 used a points-based scoring system inspired by a system used previously by Formula One motor racing to rank the performance of hyper-heuristics entered to the competition. This ranking system assigns a number of points to each competitor based on their performance for each of the 30 instances tested. The method which obtains the best objective value for a given instance is awarded 10 points, the method in second is given 8 points and then each subsequent method is awarded 6, 5, 4, 3, 2, 1 and 0 points respectively. As this system only allocates scores to the top 8 ranked contestants, all entrants which are ranked $\geq$ 9th position are allocated a score of 0. It is worth noting that the Formula One ranking system cannot be used to compare for statistically significant differences between methods, unlike other comparison methods such as Null Hypothesis Significance Testing (NHST) and Chess Rating System for Evolutionary Algorithms (CRS4EAs) [65]. However as the Formula One ranking system was used in the original CHeSC 2011 competition, using it here allows us to easily compare with a wide variety of existing methods from the literature.

Table 3(a) and 3(b) provide the relative ranking of MMA and SSMMA, respectively compared to the selection hyper-heuristics entered into CHeSC 2011 based on the Formula 1 scoring system described above. Please note that here we are comparing SSMMA and MMA to the CHeSC2011 competitors independently. In each case, the 21 hyper-heuristics compared compete for a total of 1170 points, with a maximum of 300 points available to a single hyper-heuristic. This accounts for the slightly different values for some methods in this table.

From these tables, we can clearly see that SSMMA outperforms MMA, with each method scoring 75.85 and 26.25 points respectively. The static MAs discussed previously, TGMA and SSMA, were shown to generalise poorly over dif-

Table 3: Ranking of (a) MMA, and (b) SSMMA with respect to the CHeSC 2011 competitors based on Formula 1 scores.

(a)

Rank	Name	Score
1	AdapHH	178
2	VNS-TW	132
3	ML	130.5
4	PHunter	90.25
5	EPH	85.25
6	NAHH	75.1
7	HAHA	75
8	ISEA	68
9	KSATS-HH	66.5
10	HAEA	50
11	GenHive	36.5
12	ACO-HH	36.4
13	**MMA**	**26.25**
14	SA-ILS	24.25
15	DynILS	24
16	XCJ	22.5
17	AVEG-Nep	21
18	GISS	16.75
19	SelfSearch	7
20	MCHH-S	4.75
21	Ant-Q	0

(b)

Rank	Name	Score
1	AdapHH	175.1
2	VNS-TW	126.6
3	ML	123
4	PHunter	89.25
5	EPH	84.75
6	**SSMMA**	**75.85**
7	NAHH	70.5
8	HAHA	68.85
9	ISEA	64.5
10	KSATS-HH	58.35
11	HAEA	51
12	ACO-HH	37
13	GenHive	34.5
14	SA-ILS	22.25
15	DynILS	25
16	XCJ	20.5
17	AVEG-Nep	17.5
18	GISS	16.25
19	SelfSearch	6
20	MCHH-S	3.25
21	Ant-Q	0

ferent problem domains compared to the selection hyper-heuristics in the competition. If these two MAs were entered into the original competition, TGMA would have scored 0 points, and SSMA would have scored 7.5 points [53]. Here, both self-adaptive MAs outperform these methods by some distance. The ability to 'learn' which memes are useful in an adaptive manner is clearly beneficial in the context of cross-domain optimisation. SSMMA ranks 6th overall compared to the CHeSC 2011 competitors.

Table 4 separates the results given in Table 3(a) and Table 3(b), breaking down performance in each problem domain. The most striking figure within this table is that SSMMA is performing extremely well in the MAX-SAT problem domain, in fact outperforming all 20 CHeSC 2011 competitors. These results are also superior to the previous state-of-the-art results of Drake et al. [20], which scored 32.85 points in this problem domain. Despite extremely encouraging results overall, SSMMA still performs relatively poorly in the Travelling Salesman Problem and Permutation Flow Shop problem domains, two domains in which MMA does score some points. Although scoring 0 points in a single domain indicates poor performance using this ranking system, it should be highlighted that this corresponds to a method not being one of the top 8 competitors. This does not necessarily mean that it is one of the worst of the 21 methods compared.

Interestingly, the performance of SSMMA is extremely similar to the *Modified Choice Function - All Moves* (MCF-AM) hyper-heuristic presented by Drake et al. [21] both in terms of overall performance (MCF-AM scored 73.7 points compared to the CHeSC 2011 competitors) and the individual problem domains in which it is stronger and weaker. This version of MCF-AM used a framework introduced by Drake et al. [22] to manage crossover low-level heuristics, which were omitted from the set of low-level heuristics in the original MCF-AM [20]. They suggested that in the case of the Travelling Salesman Problem, where the top three methods ML [36], AdapHH [40], and VNS-TW [27] were also the top three in the competition overall, a hyper-heuristic framework which enforces hill climbing was desirable in order to perform well in this domain. This

Table 4: Number of Formula One points scored in each CHeSC 2011 problem domain by SSMMA and MMA

Domain	SSMMA	MMA
SAT	**36.85**	0
BP	**21**	0
PS	**10**	0
PFS	0	**10.25**
TSP	0	**12**
VRP	**8**	4
Total	**75.85**	26.25

doesn't seem to be the case here, where SSMMA uses a hill climbing operator at each step through the use of memes, with relatively poor performance observed in this problem domain.

Overall, single-point-based adaptive search methods outperform self-adaptive multi-point evolutionary algorithms for cross domain search. However, considering the advances in CPUs and multi-core/multi-threaded hardware technologies, it is not difficult to see the underlying potential in this area. There has been a large increase in the number of parallel processing methods proposed in the literature, at least partially as a result of the availability of affordable multi-core computers as well as general purpose GPUs. Combining MAs with hyper-heuristics which exploit the underlying computer architecture has the potential to lead to more efficient and effective solvers. Asta et al. [3] described such an approach which was the winner of an international challenge solving a project scheduling problem.

5. Conclusion

Adaptation and control are crucial in many metaheuristics considering that many applications often require implementation of multiple operators, potentially with various internal system setting options, such as parameters defining

the neighbourhood move or number of steps for repeating a chosen process. In this study, we propose a new *self-adaptive* Memetic Algorithm [50], referred to as Steady-state Multimeme Memetic Algorithm (SSMMA) with a new adaptation mechanism having a long-term memory for controlling genetic operators and their parameter settings during the search process. SSMMA co-evolves utility scores, indicating the success of producing an improved solution for all algorithmic component choices encoded as a meme in each individual. The simple inheritance mechanism is used for transmitting memes from parents to offspring, however memetic material does not get mutated as is the case in the Multimeme Memetic Algorithm (MMA) of Krasnogor and Smith [35]. The utility scores are maintained based on a *simple* Reinforcement Learning strategy, increasing the score of a meme if an improved solution is produced.

The performance of SSMMA is evaluated using a software library, referred to as HyFlex, containing six problem domains. The empirical results show that SSMMA performs significantly better than MMA on 12 out of 15 instances across three of the domains. It is slightly better than MMA on 5 instances on another domain, while MMA performs significantly better than SSMMA on 10 instances from the remaining two domains on average. Based on Formula 1 ranking used in CHeSC2011, SSMMA performs better than MMA with a score of 75.85 against 26.25 and memetic algorithm variants without operator control as well as fifteen other previously proposed algorithms.

Both methods were shown to be vastly superior to traditional static Memetic Algorithms. Additionally, SSMMA is able to offer results that are competitive with some of the state-of-the-art methods from CHeSC 2011, outperforming all 20 entrants in the five instances of the MAX-SAT problem domain used in the competition. The results in this domain are also an improvement over the previously best results reported in the literature for these instances.

Looking into the cases for which SSMMA performs poorly, there seems to be an issue with the use of long term memory, i.e., memory length. The proposed approach might need a 'forgetting' mechanism, however this will require design of multiple additional algorithmic components, covering when to forget, what

to forget and how to forget. We intend to continue our work in this area in future by analysing the effect of the Reinforcement Learning mechanism used to select a low-level heuristic of a given type and memory length. Currently SSMMA relies on tournament selection to choose a heuristic to apply based on previous performance, however a different component can be used instead, such as Roulette Wheel or Choice Function [15], and even this choice could be co-evolved. A more 'intelligent' selection mechanism could lead to a more reactive system, capable of quickly learning which heuristics are effective at a particular point in the search.

Considering the success of selection hyper-heuristics performing single-point-based search in the control of operators, Memetic Algorithms embedding selection hyper-heuristics is another research direction we plan to take. There is already empirical evidence that such approaches could yield improved performance, particularly when implemented in a distributed processing setting exploiting the underlying computer architecture with multiple cores [3].

References

[1] G. Acampora, V. Loia, A. Vitiello, Enhancing ontology alignment through a memetic aggregation of similarity measures, Information Sciences 250 (2013) 1–20.

[2] A. Alkan, E. Özcan, Memetic algorithms for timetabling, in: Proceedings of the IEEE Conference on Evolutionary Computation (CEC 2003), vol. 3, IEEE Press, Canberra, Australia, 2003, pp. 1796–1802.

[3] S. Asta, D. Karapetyan, A. Kheiri, E. Özcan, A. J. Parkes, Combining Monte-Carlo and hyper-heuristic methods for the multi-mode resource-constrained multi-project scheduling problem, CoRR abs/1511.04387.
URL `http://arxiv.org/abs/1511.04387`

[4] C. Blum, A. Roli, Metaheuristics in combinatorial optimization: Overview

and conceptual comparison, ACM Computing Surveys 35 (3) (2003) 268–308.

[5] L. Buriol, P. M. França, P. Moscato, A new memetic algorithm for the asymmetric traveling salesman problem, Journal of Heuristics 10 (5) (2004) 483–506.

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

[7] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, R. Qu, Hyper-heuristics: A survey of the state of the art, Journal of the Operational Research Society 64 (12) (2013) 1695–1724.

[8] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, J. R. Woodward, A classification of hyper-heuristic approaches, in: M. Gendreau, J.-Y. Potvin (eds.), Handbook of Metaheuristics, vol. 146 of International Series in Operations Research & Management Science, Springer US, 2010, pp. 449–468.

[9] E. K. Burke, G. Kendall, M. Mısır, E. Özcan, Monte carlo hyper-heuristics for examination timetabling, Annals of Operations Research 196 (1) (2012) 73–90.

[10] E. K. Burke, J. D. L. Silva, The design of memetic algorithms for scheduling and timetabling problems, in: W. E. Hart, J. Smith, N. Krasnogor (eds.), Recent Advances in Memetic Algorithms, vol. 166 of Studies in Fuzziness and Soft Computing, Springer Berlin Heidelberg, 2005, pp. 289–311.

[11] A. Caponio, G. L. Cascella, F. Neri, N. Salvatore, M. Sumner, A fast adaptive memetic algorithm for online and offline control design of pmsm drives, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 37 (1) (2007) 28–41.

[12] F. Caraffini, F. Neri, L. Picinali, An analysis on separability for memetic computing automatic design, Information Sciences 165 (2014) 1–22.

[13] X. Chen, Y.-S. Ong, M.-H. Lim, K. C. Tan, A multi-facet survey on memetic computation, IEEE Transactions on Evolutionary Computation 15 (5) (2011) 591–607.

[14] H. G. Cobb, An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environment, Tech. Rep. NRL Memorandum Report 6760, U. S. Navy Center for Applied Research in Artificial Intelligence Information Technology Division (1990).

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

[16] M. Črepinšek, S.-H. Liu, M. Mernik, Exploration and exploitation in evolutionary algorithms: a survey, ACM Computing Surveys 45 (3) (2013) 35.

[17] M. Črepinšek, S.-H. Liu, M. Mernik, Replication and comparison of computational experiments in applied evolutionary computing: common pitfalls and guidelines to avoid them, Applied Soft Computing 19 (2014) 161–170.

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

[19] K. A. De Jong, An analysis of the behavior of a class of genetic adaptive systems., Ph.D. thesis, University of Michigan, Ann Arbor, MI, USA, aAI7609381 (1975).

[20] J. H. Drake, E. Özcan, E. K. Burke, An improved choice function heuristic selection for cross domain heuristic search, in: C. A. C. Coello, V. Cutello,

K. Deb, S. Forrest, G. Nicosia, M. Pavone (eds.), Proceedings of Parallel Problem Solving from Nature (PPSN 2012), Part II, vol. 7492 of LNCS, Springer, Taormina, Italy, 2012, pp. 307–316.

[21] J. H. Drake, E. Özcan, E. K. Burke, A modified choice function hyper-heuristic controlling unary and binary operators, in: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2015), IEEE Press, Sendai, Japan, 2015, pp. 3389–3396.

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

[23] M. Epitropakis, F. Caraffini, F. Neri, E. Burke, A separability prototype for automatic memes with adaptive operator selection, in: Foundations of Computational Intelligence (FOCI), 2014 IEEE Symposium on, 2014, pp. 70–77.

[24] C.-K. Goh, Y.-S. Ong, K. C. Tan, Multi-Objective Memetic Algorithms, Springer Berlin Heidelberg, 2009.

[25] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, Boston, MA, USA, 1989.

[26] G. Gutin, D. Karapetyan, A memetic algorithm for the generalized traveling salesman problem, Natural Computing 9 (1) (2010) 47–60.

[27] P.-C. Hsiao, T.-C. Chiang, 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), IEEE Press, Brisbane, Australia, 2012, pp. 1–8.

[28] H. Ishibuchi, T. Yoshida, T. Murata, Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling, IEEE Transactions on Evolutionary Computation 7 (2) (2003) 204–223.

[29] W. Jakob, A general cost-benefit-based adaptation framework for multimeme algorithms, Memetic Computing 2 (3) (2010) 201–218.

[30] L. P. Kaelbling, M. L. Littman, A. W. Moore, Reinforcement learning: a survey, Journal of Artificial Intelligence Research 4 (1996) 237–285.

[31] J. Knowles, D. Corne, Memetic algorithms for multiobjective optimization: Issues, methods and prospects, in: W. E. Hart, J. Smith, N. Krasnogor (eds.), Recent Advances in Memetic Algorithms, vol. 166 of Studies in Fuzziness and Soft Computing, Springer Berlin Heidelberg, 2005, pp. 313–352.

[32] N. Krasnogor, Studies on the theory and design space of memetic algorithms, Ph.D. thesis, University of the West of England, Bristol, UK (2002).

[33] N. Krasnogor, B. Blackburne, E. Burke, J. Hirst, Multimeme algorithms for protein structure prediction, in: J. Guervós, P. Adamidis, H.-G. Beyer, H.-P. Schwefel, J.-L. Fernández-Villacañas (eds.), Proceedings of Parallel Problem Solving from Nature (PPSN 2002), vol. 2439 of LNCS, Springer, Granada, Spain, 2002, pp. 769–778.

[34] N. Krasnogor, S. Gustafson, A study on the use of "self-generation" in memetic algorithms, Natural Computing 3 (1) (2004) 53–76.

[35] N. Krasnogor, J. Smith, Emergence of profitable search strategies based on a simple inheritance mechanism, in: L. Spector, E. Goodman, A. Wu, W. B. Langdon, H. M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, E. Burke (eds.), Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001), Morgan Kaufmann, San Francisco, CA, US, 2001, pp. 432–439.

[36] M. Larose, A hyper-heuristic for the chesc 2011, in: CHeSC2011 Competition, 2011.

[37] O. Linda, D. Wijayasekara, M. Manic, M. McQueen, Optimal placement of phasor measurement units in power grids using memetic algorithms, in:

Proceedings of the IEEE International Symposium on Industrial Electronics (ISIE 2014), IEEE Press, Istanbul, 2014, pp. 2035 – 2041.

[38] H. R. Lourenço, O. C. Martin, T. Stützle, Iterated local search: framework and applications, in: M. Gendreau, J.-Y. Potvin (eds.), Handbook of Metaheuristics, vol. 146 of International Series in Operations Research and Management Science, Springer US, 2010, pp. 363–397.

[39] P. Merz, B. Freisleben, Fitness landscape analysis and memetic algorithms for the quadratic assignment problem, IEEE Transactions on Evolutionary Computation 4 (4) (2000) 337–352.

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

[41] P. Moscato, On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms, Caltech concurrent computation program, C3P Report 826 (1989) 1989.

[42] P. Moscato, 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 1 (1992) 177–186.

[43] F. Neri, C. Cotta, Memetic algorithms and memeting computing optimization: A literature review, Swarm and Evolutionary Computation 2 (2012) 1–14.

[44] F. Neri, J. Toivanen, G. L. Cascella, Y.-S. Ong, An adaptive multimeme algorithm for designing hiv multidrug therapies, IEEE/ACM Transactions on Computational Biology and Bioinformatics 4 (2) (2007) 264–278.

[45] Q. H. Nguyen, Y.-S. Ong, N. Krasnogor, A study on the design issues of memetic algorithm, in: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2007), IEEE Press, Singapore, 2007, pp. 2390 – 2397.

[46] S. Nguyen, M. Zhang, M. Johnston, K. C. Tan, Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming, IEEE Transactions on Evolutionary Computation 18 (2) (2014) 193–208.

[47] R. Nogueras, C. Cotta, On meme self-adaptation in spatially-structured multimemetic algorithms, in: I. Dimov, S. Fidanova, I. Lirkov (eds.), Numerical Methods and Applications, vol. 8962 of LNCS, Springer International Publishing, 2015, pp. 70–77.

[48] G. Ochoa, M. Hyde, T. Curtois, J. Vazquez-Rodriguez, J. Walker, M. Gendreau, G. Kendall, B. McCollum, A. Parkes, S. Petrovic, E. Burke, Hyflex: A benchmark framework for cross-domain heuristic search, in: J.-K. Hao, M. Middendorf (eds.), Proceedings of the European Conference on Evolutionary Computation in Combinatorial Optimisation (EvoCOP 2012), vol. 7245 of LNCS, Springer, Malaga, Spain, 2012, pp. 136–147.

[49] Y.-S. Ong, A. Keane, Meta-lamarckian learning in memetic algorithms, IEEE Transactions on Evolutionary Computation 8 (2) (2004) 99–110.

[50] Y.-S. Ong, M.-H. Lim, N. Zhu, K.-W. Wong, Classification of adaptive memetic algorithms: a comparative study, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 36 (1) (2006) 141–152.

[51] S. Orike, D. W. Corne, Improved evolutionary algorithms for economic load dispatch optimization problems, in: Proceedings of the 12th Annual Workshop on Computational Intelligence (UKCI 2012), 2012, pp. 1–8.

[52] S. Orike, D. W. Corne, A memetic algorithm for dynamic economic load dispatch optimization, in: Proceedings of the IEEE Symposium on Com-

putational Intelligence in Dynamic and Uncertain Environments (CIDUE 2013), IEEE Press, Singapore, 2013, pp. 92 – 99.

[53] E. Özcan, S. Asta, C. Altıntaş, Memetic algorithms for cross-domain heuristic search, in: Y. Jin, S. A. Thomas (eds.), Proceedings of the 13th Annual Workshop on Computational Intelligence (UKCI 2013), IEEE Press, Surrey, UK, 2013, pp. 175–182.

[54] E. Özcan, C. Başaran, A case study of memetic algorithms for constraint optimization, Soft Computing 13 (8-9) (2009) 871–882.

[55] E. Özcan, B. Bilgin, E. E. Korkmaz, A comprehensive analysis of hyper-heuristics, Intelligent Data Analysis 12 (1) (2008) 3–23.

[56] E. Özcan, E. Ersoy, Final exam scheduler - fes, in: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2005), vol. 2, IEEE Press, 2005, pp. 1356–1363.

[57] E. Özcan, M. Mısır, G. Ochoa, E. K. Burke, A reinforcement learning - great-deluge hyper-heuristic for examination timetabling, International Journal of Applied Metaheuristic Computing 1 (1) (2010) 39–59.

[58] E. Özcan, E. Onbaşıoğlu, Memetic algorithms for parallel code optimization, International Journal of Parallel Programming 35 (1) (2007) 33–61.

[59] E. Özcan, A. J. Parkes, A. Alkan, The interleaved constructive memetic algorithm and its application to timetabling, Computers & Operations Research 39 (10) (2012) 2310–2322.

[60] D. Qaurooni, A memetic algorithm for course timetabling, in: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2011), ACM, Dublin, Ireland, 2011, pp. 435–442.

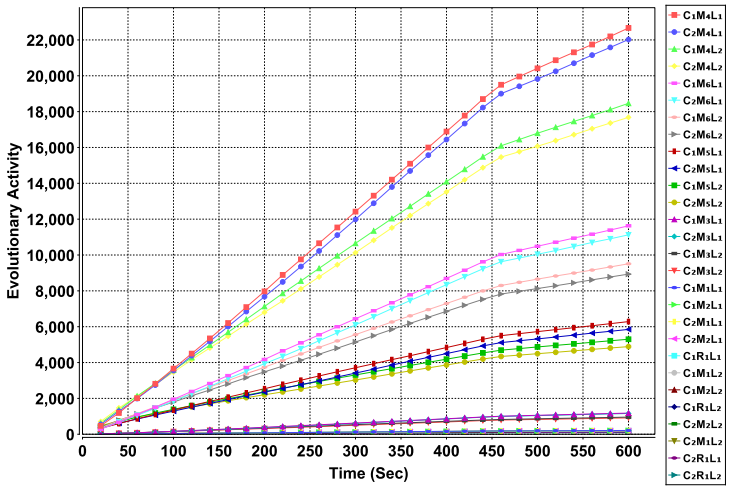
[61] D. J. Sheskin, Handbook of parametric and nonparametric statistical procedures, CRC Press, 2003.

[62] J. Smith, Coevolving memetic algorithms: A review and progress report, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 37 (1) (2007) 6–17.

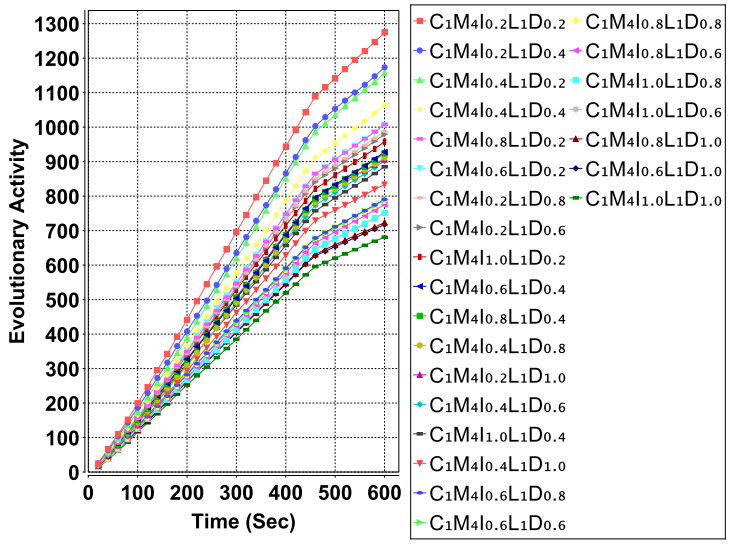
[63] J. Smith, Meme fitness and memepool sizes in coevolutionary memetic algorithms, in: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2010), IEEE Press, 2010, pp. 1–8.

[64] K. Sörensen, F. W. Glover, Metaheuristics, in: S. I. Gass, M. C. Fu (eds.), Encyclopedia of Operations Research and Management Science, Springer US, 2013, pp. 960–970.

[65] N. Veček, M. Mernik, M. Črepinšek, A chess rating system for evolutionary algorithms: A new method for the comparison and ranking of evolutionary algorithms, Information Sciences 277 (2014) 656–679.

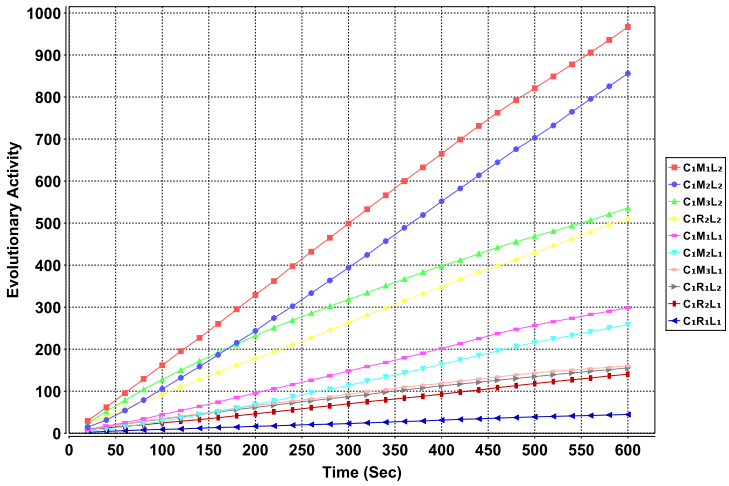


(a)

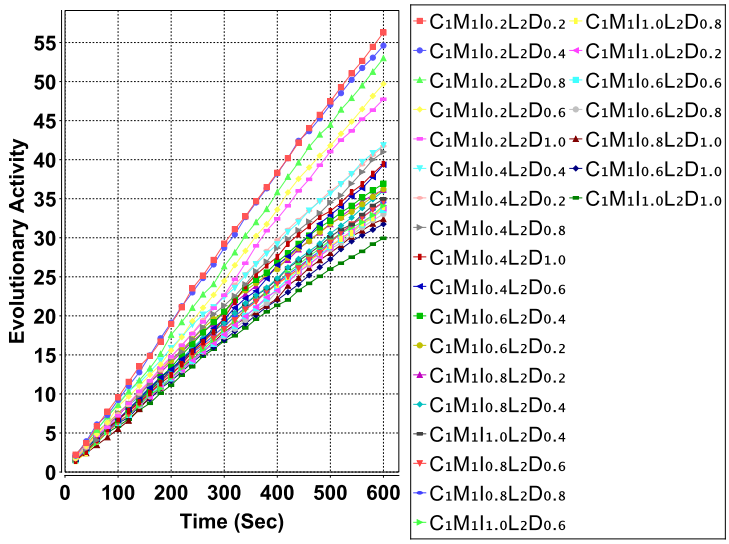


(b)

Figure 4: Evolutionary activity versus time plot averaged over 31 runs on Instance 3 of the MAX-SAT problem domain.

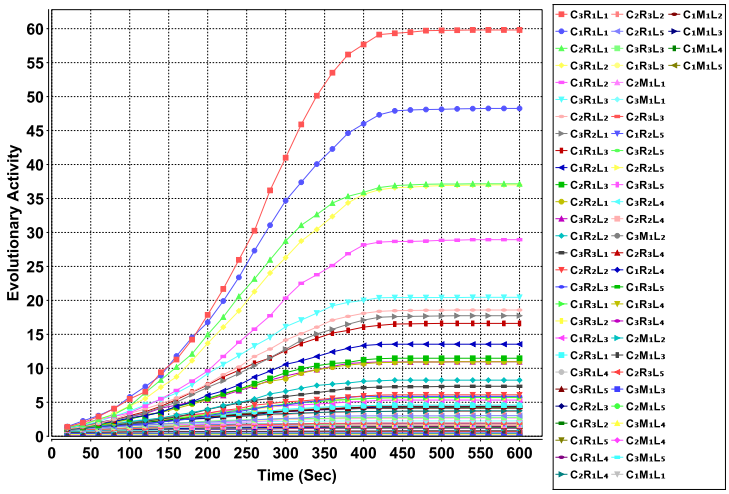


(a)

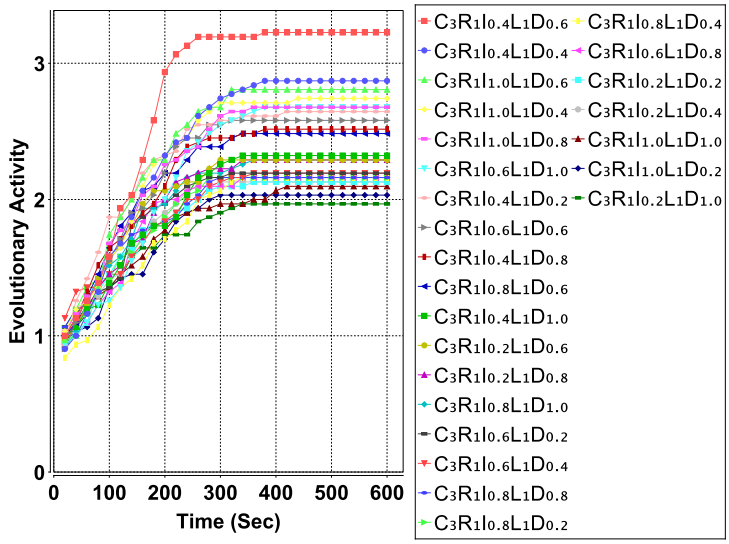


(b)

Figure 5: Evolutionary activity versus time plot averaged over 31 runs on Instance 1 of the Bin Packing problem domain.

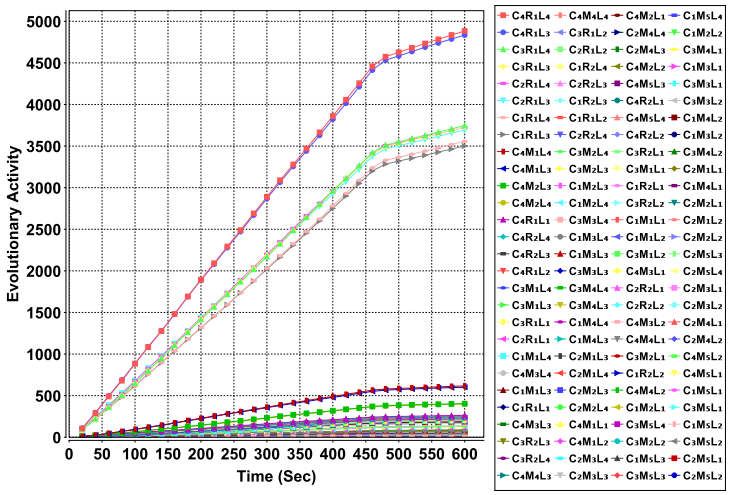


(a)

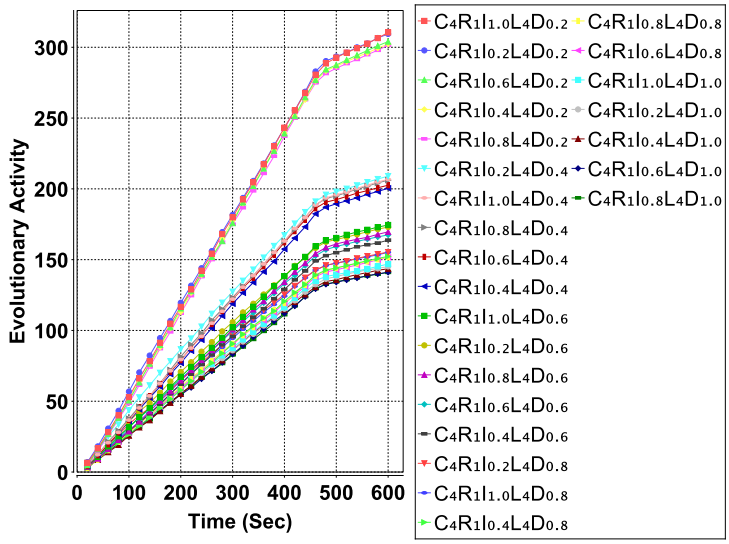


(b)

Figure 6: Evolutionary activity versus time plot averaged over 31 runs on Instance 1 of the Personnel Scheduling problem domain.

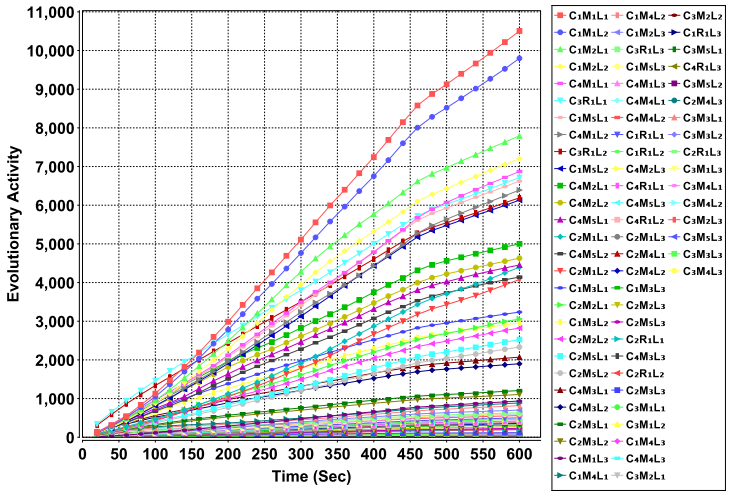


(a)

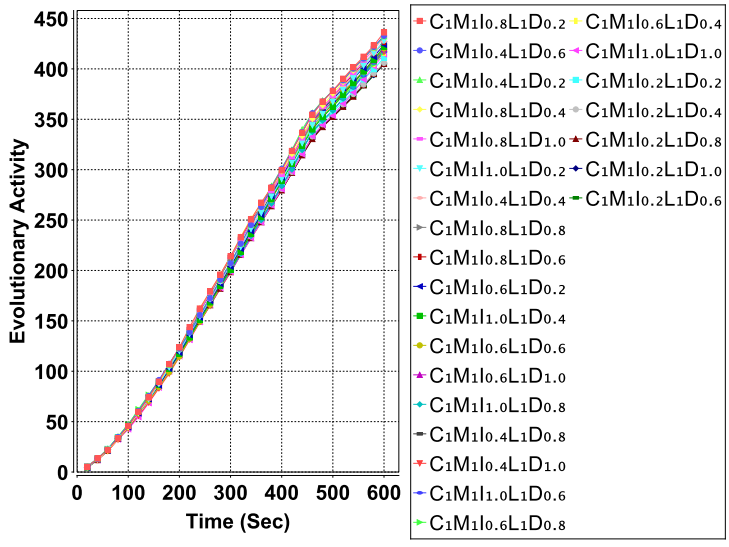


(b)

Figure 7: Evolutionary activity versus time plot averaged over 31 runs on Instance 1 of the Permutation Flow Shop problem domain.

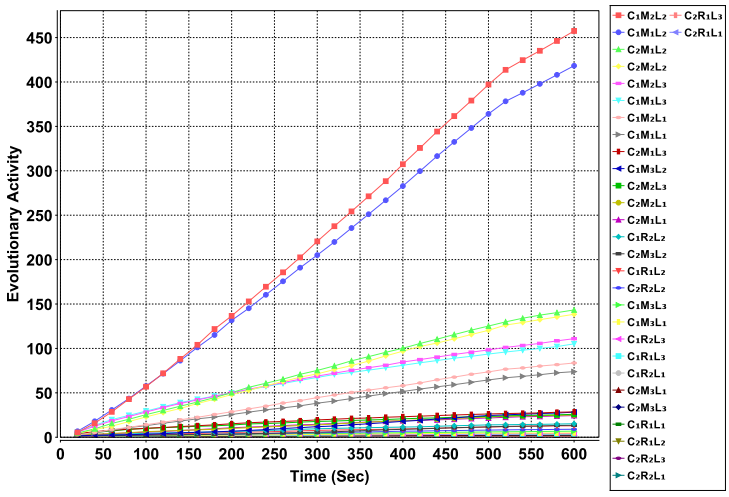


(a)

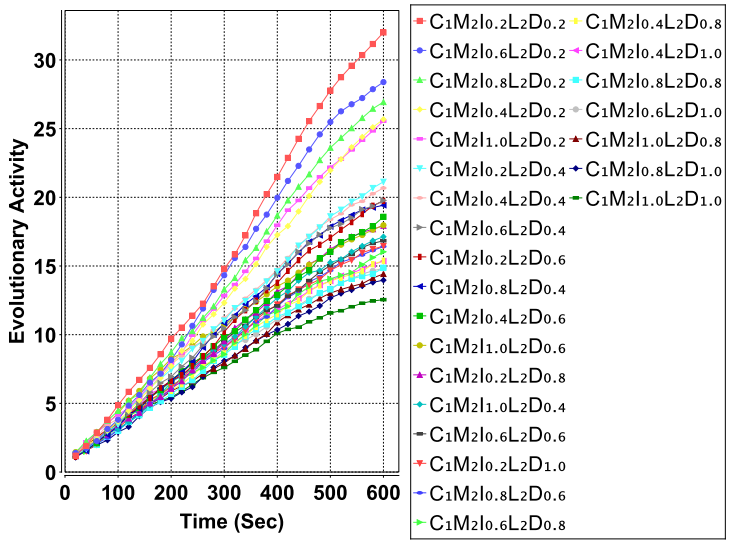


(b)

Figure 8: Evolutionary activity versus time plot averaged over 31 runs on Instance 1 of the Travelling Salesman Problem.



(a)



(b)

Figure 9: Evolutionary activity versus time plot averaged over 31 runs on Instance 3 of the Vehicle Routing Problem.