PERFORMANCE ANALYSIS OF HYPERHEURISTICS AND THEIR USE WITH HILL-CLIMBERS

by
Burak Bilgin

Submitted to the Institute of Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science
in
Computer Engineering

Yeditepe University
2006

PERFORMANCE ANALYSIS OF HYPERHEURISTICS
AND THEIR USE WITH HILL-CLIMBERS

APPROVED BY:

Assist.Prof.Dr. Ender Özcan .....
(Thesis Supervisor)

Prof.Dr. Şebnem Baydere .....

Assoc.Prof.Dr. M. Kudret Yurtseven .....

DATE OF APPROVAL: 20/09/2006.

ACKNOWLEDGEMENTS

I want to thank Assist. Prof. Dr. Ender Özcan for his guidance throughout my M. Sc. studies, for sharing information and ideas and for his motivation and enthusiasm throughout the research; Assist. Prof. Dr. Erkan Korkmaz for his valuable feedback and for exchanging information and ideas throughout this research; Prof. Dr. Şebnem Baydere and Assoc. Prof. Dr. M. Kudret Yurtseven for their feedback and ideas on the final text of the thesis.

This research is funded by TUBITAK (The Scientific and Technological Research Council of Turkey) under grant number 105E027.

ABSTRACT

PERFORMANCE ANALYSIS OF HYPERHEURISTICS AND THEIR USE WITH HILL-CLIMBERS

Hyperheuristics are iterative approaches that are proposed as a higher level abstraction as compared to the metaheuristics. Hyperheuristic methods manage a set of heuristics for solving a problem. A typical iteration in a hyperheuristic framework consists of two phases: heuristic selection and move acceptance. After the selection and application of an appropriate heuristic to a single candidate solution at hand, a decision is made whether to keep the new candidate solution, or not. This decision is based on only nonproblem-specific data, such as, fitness change or heuristic execution time. In this thesis, the traditional framework is extended and three new frameworks are proposed in order to make better use of hill-climbers. These frameworks and several heuristic selection method and acceptance criterion combinations are analyzed in depth. Their performances are evaluated on well-known benchmark functions. The performance variances of the hyperheuristics are further investigated on the examination timetabling benchmark problem instances.

ÖZET

YARDIMLI BULUŞSAL ALGORİTMALARIN VE BUNLARIN TEPE TIRMANMA OPERATÖRLERİ İLE KULLANIMININ BAŞARIM ÇÖZÜMLEMESİ

Yardımlı buluşsal algoritmalar buluşsalüstü algoritmalara göre daha yüksek seviyede bir soyutlama olacak şekilde önerilen döngüsel yaklaşımlardır. Yardımlı buluşsal yöntemler problem çözümü için bir buluşsallar kümesini yönetir. Tipik bir yardımlı buluşsal döngüsü iki aşamadan oluşur: buluşsal seçme yöntemi ve hareket kabul etme. Uygun bir buluşsal seçildikten ve tek çözüm adayına uygulandıktan sonra yeni çözüm adayının kabul edilip edilmeyeceği kararı alınır. Bu karar sadece probleme özgü olmayan, uygunluk değeri, buluşsal işletim süresi gibi veriler kullanılarak alınır. Bu tezde geleneksel yardımlı buluşsal çerçeve modeli geliştirilmiş ve tepe tırmanıcıları daha iyi kullanmak için üç yeni çerçeve model önerilmiştir. Bu çerçeve modeller ve çeşitli buluşsal seçme yöntemi ve kabul kriteri çiftleri derinlemesine çözümlenmiştir. Bunların başarımları iyi bilinen matematiksel denektaşı fonksiyonlarında ölçülmüştür. Yardımlı Buluşsalların başarım varyansları ayrıca denektaşı sınav zaman çizelgeleme problem örnekleri üzerinde araştırılmıştır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS / ABBREVIATIONS

AM	All moves accepted
CF	Choice Function Heuristic Selection
DBHC	Davis' Bit Hill-Climbing Operator
DIMM	Dimensional Mutation Operator
GD	Great Deluge Acceptance Criterion
GR	Greedy Heuristic Selection
HYPM	Hyper-Mutation Operator
IE	Improving and equal moves accepted
MC	Monte Carlo Acceptance Criterion
NAHC	Next Ascent Hill-Climbing Operator
OI	Only improving moves accepted
SAHC	Steepest Ascent Hill-Climbing Operator
SR	Simple Random Heuristic Selection
SWPD	Swap Dimension Operator
RBHC	Random Bit Mutation Hill-Climbing Operator
RD	Random Descent Heuristic Selection
RP	Random Permutation Heuristic Selection
RPD	Random Permutation Descent Heuristic Selection
TABU	Tabu Search Heuristic Selection

1. INTRODUCTION

A heuristic is an algorithm that obtains a good solution in a reasonable amount of time for a given problem. Hopefully, the resulting solution will be the optimal and the running time complexity will be polynomial. A metaheuristic is a heuristic algorithm for solving a very general class of problems. They are generally used for solving problems that are NP-hard for which there exists no satisfactory heuristic, such as, combinatorial optimization problems. The term hyperheuristic refers to a recent approach used as a search methodology [1, 2, 3, 4, 5, 6, 7 and 8]. It is a higher level abstraction than metaheuristic methods. Hyperheuristics involve an iterative strategy that selects a heuristic to apply to a candidate solution of the problem at hand at each step. Cowling et al. defined properties of hyperheuristics in [1]. Each iteration of a hyperheuristic can be subdivided into two parts: heuristic selection and move acceptance. Hill-climbers can be deployed as heuristics in the traditional hyperheuristic framework. However additional steps in the traditional hyperheuristic framework can increase the efficiency of the hill-climbers and the efficiency of the hyperheuristic algorithm in general. Therefore three further hyperheuristic frameworks which involve additional steps to utilize hill-climbers are proposed in this thesis. The performance of the traditional and the proposed hyperheuristic frameworks are measured on benchmark functions. In the hyperheuristic literature, several heuristic selection methods and acceptance criteria are used [1, 2, 3, 4, 5, 6, 7 and 8]. However, no comprehensive study exists that compare the performances of these different methods and criteria in depth. In this thesis, seven heuristic selection methods and five different acceptance criteria are analyzed. Their performance is measured on well-known benchmark functions. Extensive experiments were performed on benchmark functions by coupling all heuristic selection methods and all acceptance criteria with each other.

Timetabling problems are real world constraint optimization problems. Due to their NP complete nature [9], traditional approaches might fail to generate a solution to a timetabling problem instance. Timetabling problems require assignment of *time-slots* (periods) and possibly some other resources to a set of events, subject to a set of constraints. Numerous researchers deal with different types of timetabling problems based on different types of constraints utilizing variety of approaches. *Employee timetabling*,

course timetabling and *examination timetabling* are the research fields that attract attention. The focus of this thesis is on examination timetabling. Hyperheuristics are applied to the examination timetabling problem. All heuristic selection methods combined with all of the acceptance criteria is tested on a set of Carter's benchmark examination timetabling problem instances [10].

This thesis is organized as follows. Literature survey on hyperheuristics, heuristic selection methods and acceptance criteria are presented in Chapter 2. Literature survey on benchmark functions, heuristics for benchmark functions, experimental settings and results on benchmark functions are presented in Chapter 3. Literature survey on examination timetabling, heuristics for examination timetabling, experimental settings and results on examination timetabling are presented in Chapter 4. Conclusions on the literature study, proposed hyperheuristic frameworks and combinations, experiments and results are presented in Chapter 5. The tables which present the experimental results of hyperheuristic frameworks on benchmark functions are presented on Appendix A. The tables which present the experimental results of hyperheuristic combinations on benchmark functions are presented on Appendix B. The tables which present the experimental results of hyperheuristic combinations on examination timetabling are presented on Appendix C.

2. HYPERHEURISTICS

Combinatorial optimization problems are typically NP-hard problems. Such problems encountered in practical situations are often characterized by large search spaces and a set of restricting constraints [2]. These facts make a convenient enumeration impossible. Therefore an exhaustive search is not applicable to most of these problems and if applicable an optimal solution cannot be found in a reasonable amount of computation time. Commercial organizations need solutions to such optimization problems that are "*good enough, soon enough, cheap enough*" and deploy simple heuristics to obtain such solutions. Although heuristics are easy to implement and reach satisfactory solutions in a reasonable amount of computation time, they lack the guarantee to find optimal or even near optimal solutions. The term heuristic refers to a single step which takes a candidate solution as an input and involves a decision criterion to modify it in a way. This single step of modification is considered as a move in the search space. Heuristics are called iteratively in search algorithms to reach satisfactory solutions [2 and 7].

Artificial intelligence research in the area of operations research is focused on the development of metaheuristics which are high-level algorithmic approaches to tackle optimization problems [2]. State-of-the-art solution methods for many optimization problems are metaheuristic methods. The high performance of metaheuristics is mostly due to their problem-specific and knowledge-rich nature. On the other side, metaheuristics have a variety of disadvantages. Most state-of-the-art metaheuristics are designed and fine tuned for a specific problem and commonly for a specific type or instance of a specific problem. They involve a variety of parameters which require to be adjusted for each problem type or instance. This fact prevents the reuse of a metaheuristic implementation on a different problem than it is designed for. This fact also makes the implementation and deployment of metaheuristics to require expert level knowledge and experience on both the optimization problem and the metaheuristic method applied. Metaheuristic methods require vast amounts of computation resources and long convergence durations. They also involve randomized processes which results in variances in performance. Theoretical and computational analysis on metaheuristics is not satisfactory and operation principles of most of these methods are not easy to understand which arouses doubts about these

methods. Due to these disadvantages metaheuristics are not commonly deployed in industry [2, 3 and 7].

The term hyperheuristic refers to an optimization method which is developed to avoid the disadvantages of metaheuristics [1]. Hyperheuristics do not use problem-specific information [1, 2, 3 and 7]. They deploy a set of simple, easy to implement, low level heuristics and conduct the search using these heuristics. Due to this property hyperheuristics are called "*heuristics to choose heuristics*" [2]. Hyperheuristics operate at a higher level of abstraction than metaheuristics [1]. Hyperheuristics work on search spaces of heuristics where metaheuristics work on search spaces of candidate solutions [3 and 7]. Hyperheuristics take advantage of strengths and avoid the weaknesses of each heuristics [7]. A further advantage of hyperheuristics is that a set of well-performing heuristics are already handy for the practical problems. These heuristics can be deployed in a hyperheuristics framework easily [7]. These properties make the implementation and deployment of hyperheuristics fast, easy and cheap. Once the hyperheuristic method is implemented it can be applied to any problem with appropriate low level heuristics and a fitness function. The implementation and deployment of the hyperheuristic will not require high levels of knowledge or expertise in problem domain and heuristics [1]. There are further properties that a hyperheuristic method should satisfy. Hyperheuristics should have a comprehensible structure without the vast number of parameters and details as it is the case in metaheuristics. They should produce good quality solutions in a reasonable amount of time and exhibit satisfactory worst-case behavior. They should work robust and in a repeatable manner [1 and 7].

Hyperheuristic framework is a three layer approach [11]. Hyperheuristic algorithm operates at the lop layer and a set of heuristics in the middle. The problem tackled, is defined at the bottom layer. The communication between hyperheuristic algorithm and the low level heuristics is carried out on a common interface [1]. This interface also acts as a barrier between hyperheuristic algorithm and the low level heuristics by avoiding the passing of problem specific data from low level heuristics to hyperheuristic [2 and 7]. This interface is used to pass information about the performance of the low level heuristics and the selection and acceptance of each low level heuristic by hyperheuristic [1].

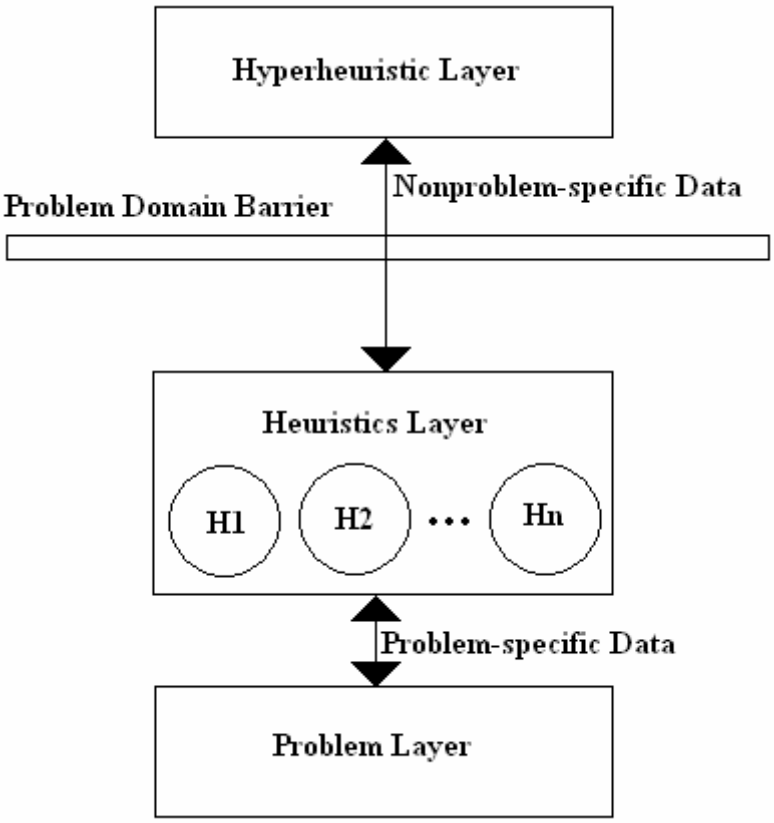

Figure 2.1. Three layer hyperheuristic approach

The execution of a hyperheuristic algorithm consists of an iterative step which is repeated continuously until a termination criterion is met. The termination criterion can be one or combination of the termination of execution time, termination of a given number of evaluations or iterations, or finding the optimum solution. Termination of the execution time is mostly used as the termination criterion in research [1 and 3]. If examined closely it can be observed that any hyperheuristic algorithm makes two decisions during a single iteration. First it selects a heuristic and applies it to the candidate solution and second it accepts or declines the modified candidate solution as the new candidate solution.

At the early stages of the hyperheuristic research the focus was on the first decisive step: the selection of low-level heuristics. At this stage hyperheuristics were named after their heuristic selection method. Examples are Simple, Choice Function, Tabu Search and Genetic Algorithm Hyperheuristics [1, 2, 3, 4 and 7]. These early examples used two simple decision criteria for the second step of the hyperheuristic algorithm: the acceptance of the modified candidate solution. These were All Moves Accepted (AM) and Only

Improving Moves Accepted (OI). The focus of the research moved to the second step of the algorithm with time. More complex acceptance criteria were introduced in [5 and 6]. However these complex acceptance criteria were used with the heuristic selection method Simple Random.

According to this survey several heuristic selection methods and acceptance criteria are introduced for hyperheuristics framework. Each one of the heuristic selection methods and acceptance criteria can be combined to one hyperheuristic algorithm. Despite this fact this kind of combinations are not studied in the literature. In this thesis various heuristic selection methods are combined with various acceptance criteria and resulting hyperheuristics are tested on a broad range of benchmark functions as well as on examination timetabling problem. Seven heuristic selection methods, which are Simple Random, Random Descent, Random Permutation, Random Permutation Descent, Choice Function, Tabu Search, and Greedy are implemented. For each heuristic selection method five acceptance criteria: *AM*, *OI*, *IE (Improving and Equal Moves Accepted)*, the *Great Deluge* and the *Monte Carlo* are implemented. As a result, a broad range of hyperheuristic combinations are obtained. These combinations are tested on benchmark functions and examination timetabling problems.

2.1. Heuristic Selection Methods

2.1.1. Simple Hyperheuristics

Simple Hyperheuristics utilize simple randomized processes to select the heuristic that will be applied next [1]. They are used in most of the hyperheuristic research and the performances of the newly proposed approaches are compared against these methods. There are four types of Simple Hyperheuristics: Simple Random, Random Descent, Random Permutation and Random Permutation Descent.

2.1.1.1. Simple Random. Simple Random is the most trivial heuristic selection method. It randomly selects a heuristic by assigning equal probability to each low level heuristic in the set [1]. This method is also used within other hyperheuristic methods which emphasize the acceptance criterion [5 and 6].

2.1.1.2. Random Descent. Random Descent selects a heuristic randomly like the Simple Random. However this method repeatedly applies the selected heuristic as long as it makes an improvement on the candidate solution [1].

2.1.1.3. Random Permutation. Random Permutation initially creates a random permutation of all low level heuristics in the set. At each iteration the next heuristic in the permutation gets selected. If the end of the permutation is reached the method cycles around the permutation and continues with the first low level heuristic in the permutation [11].

2.1.1.4. Random Permutation Descent. Random Permutation Descent operates similar to Random Permutation. The only difference is that the selected heuristic is reapplied each time it improves the candidate solution [1].

2.1.2. Greedy Hyperheuristic

Greedy Hyperheuristic calls all low level heuristics at each iteration. The low level heuristic that produces the most improvement is selected [1].

2.1.3. Choice Function Hyperheuristic

In this heuristic selection method a Choice Function is utilized that considers various data [1]. The Choice Function measures the previous performance of each single and each pair of low level heuristics. The performance measure depends on improvement and execution time. It also considers the time elapsed from the last call of each low level heuristic. Using these criteria Choice Function makes a selection among the low level heuristics [1].

In this method a linear combination is calculated for each low level heuristic at each step [1]. This linear combination consists of three different quantities which measure different properties of the low level heuristics under consideration. The first quantity is the measurement of the previous performance of the low level heuristic. The second quantity is the measurement of the previous performance of the consecutive calls of two low level

heuristics and this way the synergy between them. The last quantity is the time elapsed since the last call of the low level heuristic [1].

The first quantity, f_1 (N_j) (Formula 2.1), measures the previous performance of the low level heuristic N_j. In this formula I_n (N_j) is a positive quantity of the improvement accomplished by N_j at the n^{th} iteration, T_n (N_j) is the execution time of N_j at the n^{th} iteration, and α refers to a constant between 0 and 1.

$$f_1(N_j) = \sum_i \alpha^{i-1} \frac{I_i(N_j)}{T_i(N_j)} \tag{2.1}$$

f_1 (N_j) is calculated in the algorithm using the iterative Formula 2.2. In this formula the previous performance value is multiplied by the constant α which reduces the weight of the previous performance in the new quantity. The importance of current performance is emphasized this way.

$$f_1^{current}(N_j) = \frac{I(N_j)}{T(N_j)} + \alpha \cdot f_1^{previous}(N_j) \tag{2.2}$$

The second quantity, f_2 (N_j, N_k) (Formula 2.3), measures the previous performance of the low level heuristic N_j when it is called directly after the last called low level heuristic N_k. This quantity can be considered as to measure the synergy between the low level heuristics. In this formula I_n (N_j, N_k) is a positive quantity of the improvement accomplished by N_j when it is called directly after the last called low level heuristic N_k at the n^{th} iteration. T_n (N_j, N_k) is the execution time of N_j when it is called directly after the last called low level heuristic N_k at the n^{th} iteration, and β refers to a constant between 0 and 1.

$$f_2(N_k, N_j) = \sum_n \beta^{n-1} \left(\frac{I_n(N_k, N_j)}{T_n(N_{k,} N_j)}\right) \tag{2.3}$$

f_2 (N_j, N_k) is calculated in the algorithm similarly to f_1 (N_j) as in Formula 2.4. Again the previous performance value is multiplied by the constant β and the weight of the previous performance is reduced in the new quantity. And again the emphasis is given to the current performance this way.

$$f_2^{current}(N_k, N_j) = \frac{I(N_k, N_j)}{T(N_k, N_j)} + \beta \cdot f_2^{previous}(N_k, N_j) \tag{2.4}$$

The third quantity f_3 (N_j) is the time elapsed since the last call of the low level heuristic N_j. This quantity is measured in number of seconds of CPU time.

Each quantity has a different purpose in the method. The first and second quantities, f_1 (N_j) and f_2 (N_j, N_k), are there to intensify the search by giving the emphasis on low level heuristics which performed better currently. The second quantity f_2 (N_j, N_k) measures also the synergy between low level heuristics. The third quantity f_3 (N_j) has the purpose to diversify the search by giving the emphasis to low level heuristics which are not called lately [1].

2.1.4. Non-Stationary Reinforcement Learning Heuristic Selection Method

Nareyek introduced a heuristic selection method which uses Non-Stationary Reinforcement Learning in [12]. This method computes and maintains a "utility value" $w_j \geq 1$ for each heuristic j. Utility value is a measurement for the expected performance of the heuristic to be selected. The utility values are modified during the search according to the performance of the low level heuristic using the Non-Stationary Reinforcement Learning mechanism. Mostly the utility values are adapted to the specific region of the search space where the search is carried on. Two different approaches are proposed to make selection between low level heuristics using these utility values. The first approach is the Roulette Wheel Approach. The second approach selects the low level heuristic with the highest utility value and if there exists more than one low level heuristics with the highest utility value a random one is selected among these. Utility values are adapted according to various positive and negative reinforcement learning strategies [12].

Positive reinforcement learning strategies are Additive Adaptation, Escalating Additive Adaptation, Multiplicative Adaptation, Escalating Multiplicative Adaptation, and Power Adaptation. Negative reinforcement learning strategies are Subtractive Adaptation, Escalating Subtractive Adaptation, Divisional Adaptation, Escalating Divisional Adaptation, and Root Adaptation. The formula for each of the reinforcement learning is given in Table 2.1. A maximum and a minimum value for each of the utility values are also defined to keep the values in a useful range. In Escalating Adaptation strategies the escalation values are doubled each time if there is a consecutive improvement or worsening. They are set to 1 otherwise. The utility values are maintained as integers. If floating point values occur during reinforcement learning phase the values are rounded down to the next integer [12].

Table 2.1. Reinforcement learning strategies

Additive Adaptation	$w_a \longleftarrow w_a + 1$
Escalating Additive Adaptation	$w_a \longleftarrow w_a + m_{promotion}$
Multiplicative Adaptation	$w_a \longleftarrow w_a \times 2$
Escalating Multiplicative Adaptation	$w_a \longleftarrow w_a \times m_{promotion}$
Power Adaptation	$w_a \longleftarrow \begin{cases} w_a \times w_a \Leftarrow w_a > 1 \\ 2 \Leftarrow w_a = 1 \end{cases}$
Subtractive Adaptation	$w_a \longleftarrow w_a - 1$
Escalating Subtractive Adaptation	$w_a \longleftarrow w_a - m_{promotion}$
Divisional Adaptation	$w_a \longleftarrow w_a / 2$
Escalating Divisional Adaptation	$w_a \longleftarrow w_a / m_{promotion}$
Root Adaptation	$w_a \longleftarrow \sqrt{w_a}$

2.1.5. Tabu Search Hyperheuristic

Burke and Soubeiga proposed various hyperheuristic frameworks in [3]. These frameworks make use of the heuristic selection method introduced in [12]. They also improved the frameworks by utilizing a tabu list in the framework [3].

The proposed hyperheuristic frameworks involve a score for each low level heuristic. At each hyperheuristic iteration the low level heuristic with the highest score is selected. If there is a tie between the low level heuristics with the highest score one of them is selected randomly. The score of the low level heuristic is updated according to its performance at each time it is selected. This type of maintenance of a score for each low level heuristic is called reinforcement learning. At the beginning of the optimization process the score of each low level heuristic is equal to 0. The score is increased by adding a positive reinforcement rate to the score and decreased by subtracting a negative reinforcement rate from the score. A range is defined by a lower bound and upper bound and the score of low level heuristics cannot exceed this range [3].

In the first hyperheuristic framework positive and negative reinforcement rates are equal to 1. The lower bound is equal to 0 and the upper bound is equal to the number of the low level heuristics. A closer study of this framework exposes a weakness. Suppose that there exists a considerable amount of difference between the scores of the low level heuristic with the highest score and the remaining low level heuristics. If this low level heuristic reaches its local optimum it will start to worsen the candidate solution. Despite this fact the framework will continue to call it until there is no more difference between its score and the score of the remaining low level heuristics. Another weakness is that in some cases this framework will call a subset of heuristics cyclically by increasing and decreasing their scores [3].

The second hyperheuristic framework avoids the weakness of the first by involving a higher negative reinforcement rate which is equal to the number of low level heuristics. This framework uses the same positive reinforcement rate as the first. In this framework once a low level heuristic makes a non-improving move its score is set to 0. This way it is

avoided that the non-improving low level heuristic is called further and non-improving moves are made [3].

If analyzed closely there are two possible cases at heuristic selection phase of the second hyperheuristic framework. The first case is when the last move was a non-improving move. In this case all the low level heuristics have the same score of 0 and a random low level heuristic is selected among these. In the second case the last move was an improving move and low level heuristic selected at the last move will be repeatedly selected until it makes a non-improving move. This low level heuristic selection pattern is the same as the heuristic selection pattern of Random Descent Hyperheuristic. Another weakness of the second hyperheuristic framework is that although it tries to avoid the selection of the non-improving low level heuristic it assigns the same score to it as the remaining heuristics. This facts show that the learning potential of the framework is not fully utilized [3].

A tabu list is added to the first framework to overcome the weaknesses of both frameworks. Non-improving low level heuristics are added to the tabu list and are not called as long as they stay in this list. The immediate call of the non-improving low level heuristics are avoided this way. The length of the tabu list is variable but there is a maximum value which cannot be exceeded [3].

2.1.6. Graph-Based Hyperheuristic

Burke et al. [8] introduced a simple generic hyperheuristic which utilizes constructive heuristics (graph coloring heuristics) to tackle timetabling problems. A tabu search algorithm chooses among permutations of constructive heuristics according to their ability to construct complete, feasible and low cost timetables. At each iteration of the algorithm, if the selected permutations produce a feasible timetable, a deepest descent algorithm is applied to the obtained timetable. Burke et al. applied this hyperheuristic method to both examination and university course timetabling problem instances. The results of this experiments showed that the proposed method works well on benchmark examination and university course timetabling problem instances [8].

2.1.7. Case-Based Heuristic Selection

Burke et al. [13] propose a Case-Based Heuristic Selection Method. In this method a knowledge discovery method is employed to find the problem instances and situations where a specific heuristic works well. The proposed method also explores for similarities with the problem instance to be solved and the source cases, to predict the heuristic that will perform best. Burke et al. applied Case-Based Heuristic Selection Method to the examination and university course timetabling [13].

2.2. Acceptance Criteria

2.2.1. Simple Acceptance Criteria

Simple Acceptance Criteria consist of a single decisive step based on a simple rule. Two types of Simple Acceptance Criteria are defined in the literature [1] and a third type is attained intuitively. These are All Moves Accepted (AM), Only Improving Moves Accepted (OI), and Improving and Equal Moves Accepted (IE) criteria. The names of these acceptance criteria clearly define the rules they involve.

2.2.2. Monte Carlo Acceptance Criteria

Ayob and Kendall [5] emphasized the role of the acceptance criterion in the hyperheuristic. They introduced the Monte Carlo Hyperheuristic which has a more complex acceptance criterion than AM or OI criteria. In this criterion, all of the improving moves are accepted and the non-improving moves can be accepted based on a probabilistic framework. Ayob and Kendall defined three probabilistic approaches to accept the non-improving moves. First approach, named as Linear Monte Carlo (LMC), uses a negative linear ratio of the probability of acceptance to the fitness worsening. Second approach named as, Exponential Monte Carlo (EMC), uses a negative exponential ratio of the probability of acceptance to the fitness worsening. Third approach, named as Exponential Monte Carlo with Counter (EMCQ), is an improvement over Exponential Monte Carlo. The probability of acceptance of worsening moves decreases as the time passes and if there

is no improvement over a series of consecutive iterations then the probability increases. As the heuristic selection method, they all use the Simple Random [5].

2.2.3. Great Deluge Acceptance Criteria

Kendall and Mohamad [6] introduced another hyperheuristic method which also focuses on the acceptance criterion rather than the heuristic selection method. They used the Great Deluge Algorithm as the acceptance criterion and Simple Random as the heuristic selection method. In the Great Deluge Algorithm initial fitness is set as initial level. At each step, the moves which produce fitness values less than the level are accepted. At each step the level is also decreased by a factor [6].

3. HYPERHEURISTICS FOR BENCHMARK OPTIMIZATION

3.1. Benchmark Functions

Well-defined problem sets are useful to measure the performance of optimization methods such as genetic algorithms, memetic algorithms and hyperheuristics. Benchmark functions which are based on mathematical functions or bit strings can be used as objective functions to carry out such tests. If the benchmark function is based on a mathematical function than the search space is defined by lower and upper bounds.

The nature, complexity and other properties of the benchmark functions can be easily obtained from their definitions. Properties of benchmark functions give clues about the nature, complexity and difficulty levels of them. The term modality refers to the number of optima of a benchmark function in the defined search space. Unimodal benchmark functions have a single optimum in the search space where multimodal functions have more than one. Multimodal benchmark functions which have large Hamming distance between their local optima and the global optimum are categorized as deceptive functions. A benchmark function is continuous if it is based on a mathematical objective function which is continuous on the search space defined. Benchmark functions which are defined on bit strings and which are not continuous are called discrete.

The difficulty levels of most benchmark functions are adjustable by setting their parameters. The adjustable parameters of the benchmark functions are the number of the dimensions and number of the bits per dimension. The difficulty level of the benchmark function can be increased by increasing the number of the dimensions or the number of the bits per dimensions.

3.1.1. Sphere Function

Sphere Function, defined as a benchmark function in [14], is the most trivial function in the set. Any optimization algorithm is expected to easily locate the global optimum of this function. Sphere function is a unimodal, continuous function [14]. It is based on the

mathematical function defined in the Formula (3.1) on [-5.12, 5.12]. The global optimum is at the point (0, 0, … , 0) with the global optimum value 0.

$$f(\vec{x}) = \sum_{i=1}^{n} x_i^2 \tag{3.1}$$

3.1.2. Rosenbrock Function

Rosenbrock Function is defined as a benchmark function in [14]. Although this is a unimodal and continuous function it is not easy to find the global optimum due to the flat valley in the search space of this function. Rosenbrock Function is based on the mathematical function defined in the Formula (3.2) on [-2.048, 2.048]. The global optimum is at the point (1, 1, … , 1) with the global optimum value 0.

$$f(\vec{x}) = \sum_{i=1}^{n-1} 100 \cdot (x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \tag{3.2}$$

3.1.3. Step Function

Step Function, defined as a benchmark function in [14], is a unimodal and continuous function. The search space of Step Function consists of flat surfaces. Step Function is based on the mathematical function defined in the Formula (3.3) on [-5.12, 5.12]. The global optimum is on the surface ([-5.12, 5.0), [-5.12, 5.0), … , [-5.12, 5.0)) with the global optimum value 0.

$$f(\vec{x}) = 6 \cdot n + \sum_{i=1}^{n} \lfloor x_i \rfloor \tag{3.3}$$

3.1.4. Quartic Function with Random Noise

Quartic Function with Random Noise is defined as a benchmark function in [14]. This function is unimodal, continuous and dynamic. This function is based on the mathematical function defined in the Formula (3.4) on [-1.28, 1.28]. It involves a random noise $U(0, 1)$. $U(0, 1)$ returns a random value between 0 and 1 at each call. Due to its dynamic nature this function does not have a fixed global optimum point or value. A threshold value is defined in the experiments which utilize this function. If a value less then the threshold value is found then the optimization process is terminated.

$$f(\vec{x}) = \sum_{i=1}^{n} (i \cdot x_i^4 + U(0,1)) \tag{3.4}$$

3.1.5. Foxhole Function

Foxhole Function, defined as a benchmark function in [14], is a highly multimodal, continuous function. The dimension parameter of this function cannot be adjusted since it is defined as a two-dimensional function. It is based on the mathematical function defined in the Formula (3.5) on [-65.536, 65.536]. The global optimum is at the point (-32, -32, ... , -32) with the global optimum value 0.998004.

$$f(\vec{x}) = \frac{1}{0.002 + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6}},$$

$$a_{1j} = \begin{cases} -32 \rightarrow \mathrm{mod}(j,25) = 1 \\ -16 \rightarrow \mathrm{mod}(j,25) = 2 \\ 0 \rightarrow \mathrm{mod}(j,25) = 3 \\ 16 \rightarrow \mathrm{mod}(j,25) = 4 \\ 32 \rightarrow \mathrm{mod}(j,25) = 0 \end{cases}, \quad a_{2j} = \begin{cases} -32 \rightarrow j > 0 \wedge j \leq 5 \\ -16 \rightarrow j > 5 \wedge j \leq 10 \\ 0 \rightarrow j > 10 \wedge j \leq 15 \\ 16 \rightarrow j > 15 \wedge j \leq 20 \\ 32 \rightarrow j > 20 \wedge j \leq 5 \end{cases} \tag{3.5}$$

3.1.6. Rastrigin Function

Rastrigin Function, defined as a benchmark function in [15], is highly multimodal and continuous. It is based on the mathematical function defined in the Formula (3.6) on [-5.12, 5.12]. The global optimum is at the point (0, 0, … , 0) with the global optimum value 0.

$$f(\vec{x}) = 10 \cdot n + \sum_{i=1}^{n} (x_i^2 - 10 \cdot \cos(2\pi x_i)) \tag{3.6}$$

3.1.7. Schwefel Function

Schwefel Function, defined as a benchmark function in [16], is multimodal and continuous. The global optimum and the best local optimum lie geometrically distant on the search space. This fact makes this function a rather hard problem for optimization methods. Schwefel Function is based on the mathematical function defined in the Formula (3.7) on [-500, 500]. The global optimum is at the point (420.9687, 420.9687, … , 420.9687) with the global optimum value 418.9829.

$$f(\vec{x}) = 418.9829 \cdot n + \sum_{i=1}^{n} x_i \cdot \sin(\sqrt{|x_i|}) \tag{3.7}$$

3.1.8. Griewangk Function

Griewangk Function, defined as a benchmark function in [17], is highly multimodal and continuous. It is based on the mathematical function defined in the Formula (3.8) on [-600, 600]. The global optimum is at the point (0, 0, … , 0) with the global optimum value 0.

$$f(\vec{x}) = \sum_{i=1}^{n} \frac{X_i^2}{4000} - \prod \cos(\frac{x_i}{\sqrt{i}}) + 1 \tag{3.8}$$

3.1.9. Ackley Function

Ackley Function is defined as a benchmark function in [18]. This function is multimodal and continuous. Ackley Function is based on the mathematical function defined in the Formula (3.9) on [-32.768, 32.768]. The global optimum is at the point (0, 0, … , 0) with the global optimum value 0.

$$f(\vec{x}) = 20 + e - 20 \cdot e^{-0.2 \cdot \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)} \tag{3.9}$$

3.1.10. Easom Function

Easom Function, defined as a benchmark function in [19], is unimodal and continuous. This function involves flat surfaces which makes the optimization process harder. The degree of the flatness of the surfaces increases with the number of dimensions of the function. Easom Function is based on the mathematical function defined in the Formula (3.10) on [-100, 100]. The global optimum is at the point (π,π, … , π) with the global optimum value -1.

$$f(\vec{x}) = -(\prod_{i=1}^{n} \cos(x_i)) \cdot (e^{-\sum_{i=1}^{n} (x_i - \pi)^2}) \tag{3.10}$$

3.1.11. Rotated Hyper-Ellipsoid Function

Rotated Hyper-Ellipsoid Function, defined as a benchmark function in [14], is unimodal and continuous. Rotated Hyper-Ellipsoid Function is based on the mathematical function defined in the Formula (3.11) on [-65.536, 65.536]. The global optimum is at the point (0, 0, ... , 0) with the global optimum value 0.

$$f(\vec{x}) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j^2) \tag{3.11}$$

3.1.12. Royal Road Function

Royal Road Function is a unimodal, discrete benchmark function based on bit strings [20]. The fitness function of this benchmark function is defined in the Formula (3.12). The global optimum is the bit string (11… 1) with the global optimum value 0.

$$f(\vec{x}) = \sum_{s \in S} c_s \sigma_s(\vec{x})$$
$$\sigma_s(\vec{x}) = \begin{cases} 1 \rightarrow \vec{x} \subset s \\ 0 \rightarrow else \end{cases} \tag{3.12}$$

3.1.13. Goldberg's Deceptive Function

Goldberg's Deceptive Function is a discrete function based on bit strings [21 and 22]. This function can be used as both fitness minimizing and maximizing function. Specific fitness values are assigned to each permutation of 3 bit strings (Table 3.1). To make the problem multidimensional and increase the difficulty level, a sequence of 3 bit strings is treated as a single chromosome. The fitness of this chromosome is the sum of fitness value of each 3 bit string.

Table 3.1. Fitness values for bit strings in Goldberg's Deceptive Function

String	*Fitness Maximizing*	*Fitness Minimizing*
000	7	1
001	5	3
010	5	3
011	0	8
100	3	5
101	0	8
110	0	8
111	8	0

3.1.14. Whitley's Deceptive Function

Whitley's Deceptive Function is a discrete function based on bit strings [23]. This function can be used as both fitness minimizing and maximizing function. Specific fitness values are assigned to each permutation of 4 bit strings (Table 3.2). By treating a sequence of 4 bit strings as a single chromosome, the problem is made multidimensional and the difficulty level is increased. The sum of fitness value of each 4 bit string is assigned as the fitness to the chromosome.

Table 3.2. Fitness values for bit strings in Whitley's Deceptive Function

String	*Fitness Maximizing*	*Fitness Minimizing*
0000	28	2
0001	26	4
0010	24	6
0011	18	12
0100	22	8
0101	16	14
0110	14	16
0111	0	30
1000	20	10
1001	12	18
1010	10	20
1011	2	28
1100	8	22
1101	4	26
1110	6	24
1111	30	0

3.2. Heuristics for Benchmark Function Optimization

Benchmark Functions that are used to evaluate the performance of the hyperheuristic combinations are encoded as bit strings. Therefore bit string modifying heuristics are required to search the global optimum of these functions in a hyperheuristic framework. There exist two types of bit string modifying heuristics which are hill-climbing and mutational heuristics. The details and versions of hill-climbing and mutational heuristics are explained in chapters 3.2.1 and 3.2.2 respectively.

3.2.1. Hill-Climbing Operators

There are four types of hill-climbing operators for bit strings defined in the literature which are Steepest Ascent, Next Ascent, Davis' Bit, and Random Bit Mutation Hill-Climbing Operators. These operators make bit inversions using predefined rules and accept the resulting candidate solution only if it is an improvement on the input candidate solution.

Steepest Ascent Hill-Climbing Operator (SAHC) checks each single bit inversion variant of the input candidate and accepts the one with the highest improvement [20].

Next Ascent Hill-Climbing Operator (NAHC) inverts a bit in the candidate solution and the resulting candidate solution is accepted only if there is an improvement on the previous candidate solution. This process is repeated starting from the first bit and selecting the next bit at each iteration [20].

Davis' Bit Hill-Climbing Operator (DBHC) functions similar to the Next Ascent Hill-Climbing Operator. The only difference is that this hill-climber creates a random permutation and inverts the bits in the candidate solution according to the sequence in this permutation [24].

Random Bit Mutation Hill-Climbing Operator (RBHC) selects a random bit at each iteration and inverts it. The resulting candidate solution is accepted only if there is an improvement on the previous candidate solution [20].

3.2.2. Mutational Heuristics

It is obvious that any approach utilizing only hill-climbing operators will fail to find global optimum solutions to multimodal or deceptive optimization problems. Therefore three further operators are defined for bit strings that can either improve or worsen the candidate solution. These operators involve random processes and due to their character they are called mutational heuristics.

Swap Dimension Operator (SWPD) can be deployed only if the problem tackled is a multidimensional problem. This operator randomly selects two dimensions and the bit strings representing these dimensions are swapped.

Dimensional Mutation Operator (DIMM), like the Swap Dimension, can be deployed only if the problem tackled is a multidimensional problem. This operator randomly selects a dimension and inverts all the bits representing this dimension with the probability 0.5. Practically this operator assigns new random values to the bits representing the selected dimension.

Hyper-Mutation Operator (HYPM) inverts all the bits in the candidate solution with the probability 0.5 which means that new random values are assigned to all the bits in the candidate solution.

3.3. Hyperheuristic Frameworks for Utilizing Hill-Climbers

A recent study shows that using a single and efficient hill-climber instead of a set of hill-climbers where the operator selection is done self adaptively might yield better solutions in evolutionary algorithms in [25 and 26]. As a result, different frameworks based on the general hyperheuristic approach can be defined in order to make better use of hill-climbers as heuristics. In this study, four different frameworks are used; F_A, F_B, F_C and F_D, as summarized in Figure 3.1.

F_A is the traditional framework and the others are the newly proposed ones. Hill-climbers are used together with the mutational heuristics. In some situations, after applying a mutational heuristic a hill-climbing might be desirable. For example, if IE acceptance criterion is used in the hyperheuristic, then most of the mutational heuristic moves will be declined. To avoid this phenomenon and to make better use of diversity provided by mutational heuristics, a hill-climber can be utilized additionally. F_B represents such a framework. If the hyperheuristic chooses a mutational heuristic, then a predefined single hill-climber is applied to the configuration. Notice that F_B still uses all heuristics together. In F_C, hill-climbers are separated from the mutational heuristics. Hyperheuristic chooses only an appropriate mutational heuristic. Application of a selected heuristic to a

configuration is followed by a hill-climbing. A single hill-climber is predefined by the user. F_D is a more general form of F_C. Two hyperheuristic modules are used; one for selecting an appropriate mutational heuristic and one for selecting an appropriate hill-climber. F_D can be implemented in two ways. The acceptance mechanism of the hyperheuristic for hill-climbers can get a feedback from the intermediate configuration (Figure 3.1 - F_D, marked solid lines) or from the input configuration (Figure 3.1 - F_D, dashed line).

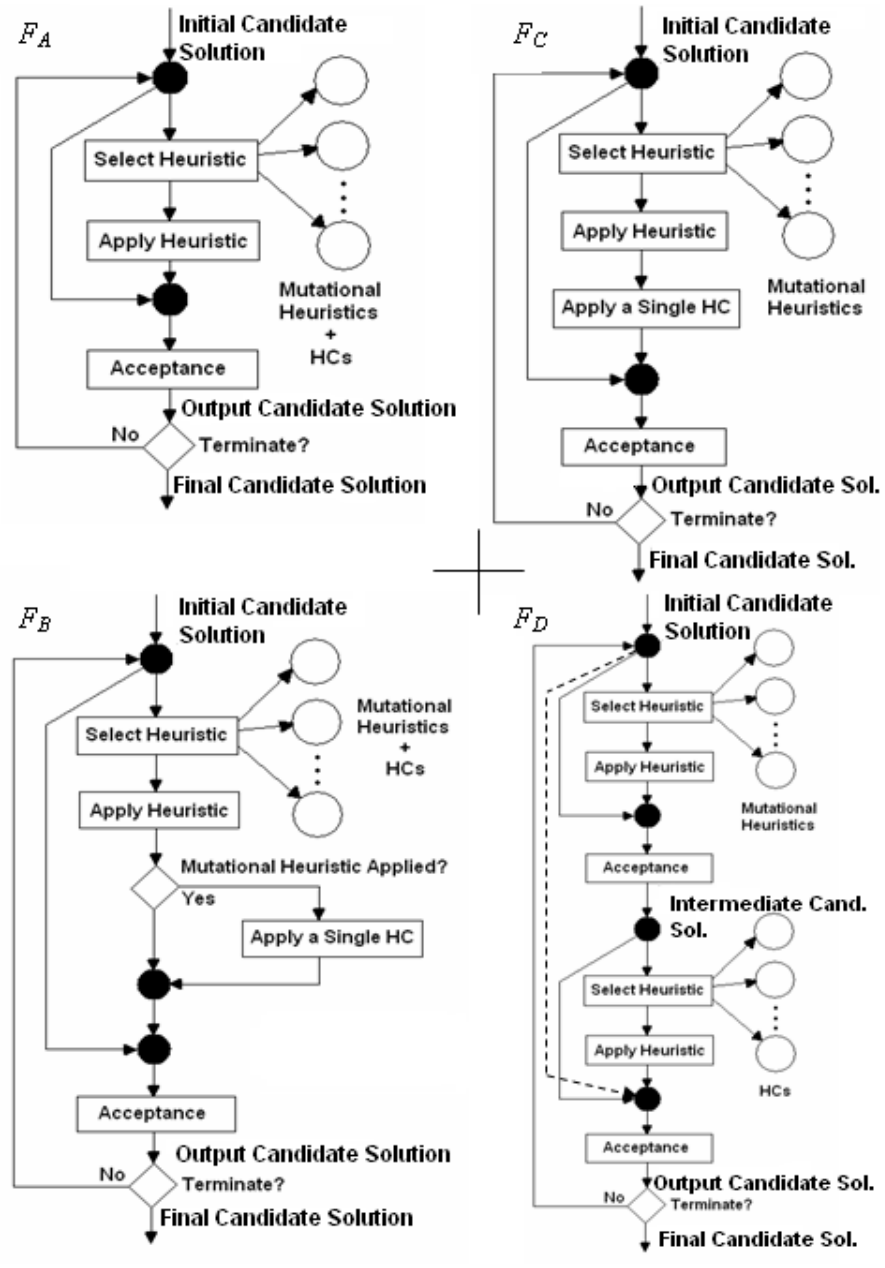

Figure 3.1. Different hyperheuristic frameworks combining mutational heuristics and hill-climbers

3.4. Experimental Settings

Two sets of experiments are executed on the benchmark functions. The performances of the hyperheuristic frameworks deploying various heuristic sets are evaluated in the first set of experiments. The performances of various heuristic selection method and acceptance criterion combinations are evaluated in the second set of experiments.

The performances of four hyperheuristic frameworks are evaluated on fourteen benchmark functions. Choice Function IE is used during the performance evaluations of F_A, F_B and F_C. Simple Random AM is used over the mutational heuristic set and Choice Function IE is used over the hill-climber set during the performance evaluation of F_D. Various heuristic and hill-climber sets are deployed within the hyperheuristic frameworks F_A and F_B to evaluate the contribution of each heuristic and hill-climber. The performance of five different heuristic and hill-climber sets are evaluated within the hyperheuristic framework F_A: hill-climbers only, all hill-climbers and one of the mutational heuristics at a time and all hill-climbers with all of the mutational heuristics. The performance of four different heuristic and hill-climber sets are evaluated within the hyperheuristic framework F_B: all hill-climbers and one of the mutational heuristics at a time and all hill-climbers with all of the mutational heuristics. The single hill-climber applied after the mutational heuristic in the framework F_B is DBHC. All mutational heuristics are deployed and the DBHC is used as the single hill-climber during the performance evaluations of the framework F_C. All mutational heuristics and all hill-climbers are deployed during the performance evaluations of the framework F_D.

Table 3. 3. Heuristic sets and the frameworks used in each hyperheuristic pattern H1-H11 (+ and * indicate that the corresponding heuristic is controlled by the same hyperheuristic)

Sets	H1	H2	H3	H4	H5	H6	H7	H8	H9	H10	H11
NAHC	+	+	+	+	+	+	+	+	+		+
DBHC	+	+	+	+	+	+	+	+	+		+
RBHC	+	+	+	+	+	+	+	+	+		+
SAHC	+	+	+	+	+	+	+	+	+		+
SWPD		+			+			+	+	+	*
DIMM			+			+		+	+	+	*
HYPM				+			+	+	+	+	*
Framework	F_A	F_A	F_A	F_A	F_B	F_B	F_B	F_B	F_A	F_C	F_D

Another series of experiments are executed to evaluate the performance of various heuristic selection method and acceptance criterion combinations within the hyperheuristic framework F_B. Seven heuristic selection methods and five acceptance criteria are combined to 35 hyperheuristic algorithms. The performance of these algorithms are evaluated on benchmark functions. The heuristic selection methods used in the experiments are Simple Random, Random Descent, Random Permutation, Random Permutation Descent, Choice Function, Tabu Search and Greedy. The acceptance criteria used in the experiments are AM, OI, IE, Monte Carlo and Great Deluge. The heuristic set deployed in these experiments consists of three hill-climbers and three mutational heuristics. Hill-climbers are NAHC, DBHC and RBHC. Mutational heuristics are SWPD, DIMM, HYPM. The single hill-climber used after mutational heuristics is DBHC.

The candidate solutions to all the continuous functions are encoded as bit strings using Gray Code. The discrete functions have their own specific representation.

Table 3.4. *Dimensions* and *Bits per Dimension* parameters used in experiments

Name	*dim*	*bits*
Sphere	10	30
Rosenbrock	10	30
Step	10	30
Quartic *with noise*	10	30
Foxhole	2	30
Rastrigin	10	30
Schwefel	10	30
Griewangk	10	30
Ackley	10	30
Easom	10	30
Rotated Hyper-ellipsoid	10	30
Royal Road	8	8
Goldberg	30	3
Whitley	6	4

3.5. Experimental Results

Each hyperheuristic pattern (hyperheuristic framework with a specific heuristic set) is run on each benchmark function 50 times. The average number of fitness evaluations to

converge to global optimum is used as the performance criterion for all benchmark functions except the Quartic Function. No hyperheuristic pattern converged to the global optimum on the Quartic Function. Therefore average best fitness value is used as the performance criterion on this function. The results are statistically evaluated using t-test with 95 per cent confidence level. The results for each benchmark function are presented in Appendix A. A general table on the performance of hyperheuristic patterns on benchmark functions is given in Table 3.5. In this table the patterns which performed better then the rest are marked with the sign +.

The hyperheuristic pattern H10, which deploys all mutational heuristics with the DBHC as the single hill-climber within the framework F_C, performed significantly better then the rest of the hyperheuristic patterns on Step, Rastrigin, Schwefel, Easom, Royal Road, Goldberg and Whitley. Step and Easom are unimodal benchmark functions which involve flat surfaces in their search space. Rastrigin and Schwefel are highly multimodal benchmark functions. Royal Road, Goldberg and Whitley Functions are discrete functions where Goldberg and Whitley Functions are also deceptive.

The hyperheuristic pattern H11, which involves the framework F_D and deploys all of the mutational heuristics and the entire hill-climbers separately, performed significantly better then the rest of the hyperheuristic patterns on Foxhole and Griewangk functions. Foxhole and Griewangk are both highly multimodal benchmark functions.

The hyperheuristic pattern H5, which deploys all the hill-climbers and SWPD mutational heuristic within the framework F_B, performed significantly better then the rest of the hyperheuristic patterns on Rosenbrock Function. Rosenbrock Function is a unimodal benchmark function which involves flat surfaces in its search space.

The hyperheuristic patterns H5 and H10 performed significantly better then the rest of the hyperheuristic patterns on Rotated Hyper-Ellipsoid Function, which is a unimodal benchmark function and involves flat surfaces in its search space.

The hyperheuristic patterns H5, H7, H8 and H10 performed significantly better then the rest of the hyperheuristic patterns on Quartic Function, which is a unimodal and dynamic benchmark function.

The hyperheuristic patterns H3, H6, H7, H8, H9 and H10 performed significantly better then the rest of the hyperheuristic patterns on Ackley Function. The hyperheuristic patterns H1, H2, H3, H4, H5, H6, H7, H8 and H9 performed significantly better then the rest of the hyperheuristic patterns on Sphere Function.

Table 3.5. Hyperheuristic patterns – benchmark functions performance

	H1	H2	H3	H4	H5	H6	H7	H8	H9	H10	H11
Sphere	+	+	+	+	+	+	+	+	+		
Rosenbrock					+						
Step										+	
Quartic					+		+	+		+	
Foxhole											+
Rastrigin										+	
Schwefel										+	
Griewangk											+
Ackley			+			+	+	+	+	+	
Easom										+	
Rotated H.					+					+	
Royal Road										+	
Goldberg										+	
Whitley										+	

Each heuristic selection method and acceptance criterion combination is run on each benchmark function 50 times. The runs where the global optimum of the benchmark function is found by the optimization method is considered to be successful. The average number of fitness evaluations to converge to global optimum is used as the performance criterion for the experiments with 100 per cent success rate. The average best fitness

reached is used as the performance criterion for the experiments with success rates lower than 100 per cent. The results are statistically evaluated using t-test with 95 per cent confidence level. The results for each benchmark function are presented in Appendix B.

All heuristic selection methods except Greedy combined with the acceptance criteria OI and IE performed significantly better than the rest of the hyperheuristic combinations on Step, Rastrigin, Schwefel, Royal Road, Goldberg, and Whitley Functions.

All heuristic selection methods except Greedy combined with the acceptance criteria OI and IE and the heuristic selection method Greedy combined with all acceptance criteria except OI performed significantly better than the rest of the hyperheuristic combinations on Rosenbrock and Easom Functions.

The heuristic selection methods Simple Random, Random Descent, Random Permutation, Random Permutation Descent and Choice Function combined with the acceptance criteria OI and IE performed significantly better than the rest of the hyperheuristic combinations on Rotated Hyper-Ellipsoid Function.

The heuristic selection method Choice Function combined with the acceptance criteria AM, Monte Carlo and Great Deluge performed significantly better than the rest of the hyperheuristic combinations on Griewangk Function.

The heuristic selection methods Simple Random and Choice Function combined with all acceptance criteria and the heuristic selection method Random Permutation combined with the acceptance criteria OI, IE, and Monte Carlo performed significantly better than the rest of the hyperheuristic combinations on Sphere Function.

All heuristic selection methods except Greedy combined with the acceptance criterion Monte Carlo and all heuristic selection methods except Random Permutation and Greedy combined with the acceptance criteria AM and Great Deluge performed significantly better than the rest of the hyperheuristic combinations on Foxhole Function.

All heuristic selection methods except Greedy combined with all acceptance criteria except AM and the heuristic selection method Greedy combined with all acceptance criteria except OI performed significantly better than the rest of the hyperheuristic combinations on Quartic Function.

All hyperheuristic combinations except Random Permutation combined with the acceptance criteria AM and Great Deluge and the heuristic selection method Greedy combined with the acceptance criterion OI performed significantly better than the rest of the hyperheuristic combinations on Ackley Function.

4. HYPERHEURISTICS FOR EXAMINATION TIMETABLING

4.1. Examination Timetabling

Examination timetabling involves a search for a solution, where values from domains (timeslots) are assigned to all variables while satisfying all the constraints. Examination timetabling problem tackled in this thesis can be formulated as a linear integer program. The parameters, vectors and the matrices involved in the problem are presented in the Formula (4.1). The Formula (4.2) enforces that each exam is scheduled one and only one time. The student-exam clash constraint is stated in the Formula (4.3). The capacity constraint for each slot is expressed in the Formula (4.4). Another constraint enforces that there must be at least an empty slot between two exams taken by a student at a day. This constraint is expressed in the Formula (4.5). The number of constraint violations are used to calculate the evaluation function in the Formula (4.6). In this formula w_i indicates the weight associated to the constraint i, g_i indicates the number of violations of the constraint i. The value 0.4 is used as the weight for the constraints in the Formulae (4.3) and (4.4). The value 0.2 is used as the weight for the constraint in the Formula (4.5).

$$M = \text{number of slots}$$

$$N = \text{number of exams}$$

$$C = \text{total capacity for a slot}$$

$$a_{ij} = \begin{cases} 1 & \text{if } j^{th} \text{ exam is on } i^{th} \text{ slot} \\ 0 & \text{else} \end{cases} \tag{4.1}$$

$$b_j = \text{number of students taking exam } j$$

$$c_{jk} = \text{number of students taking both exams } j \text{ and } k$$

$$\forall j, \sum_{i=1}^{M} a_{ij} = 1 \tag{4.2}$$

$$\forall i, \sum_{j=2}^{N} a_{ij} \sum_{k=1}^{j-1} a_{ik} c_{jk} = 0 \tag{4.3}$$

$$\forall i, \sum_{j=1}^{N} a_{ij} b_j \leq C \tag{4.4}$$

$$\forall i \text{ , if } i \text{ not the last slot in the day, } \sum_{j=1}^{N} a_{ij} \sum_{k=1}^{N} a_{i+1,k} c_{jk} = 0 \tag{4.5}$$

$$F(T) = \frac{-1}{1 + \sum_{\forall i} w_i g_i(T)} \tag{4.6}$$

Burke et al. [27 and 28] applied a light or a heavy mutation, randomly selecting one, followed by a hill-climbing. Investigation of various combinations of Constraint Satisfaction Strategies with Genetic Algorithms for solving examination timetabling problems can be found in [29]. Paquete et al. [30] applied a Multiobjective Evolutionary Algorithm (MOEA) based on Pareto Ranking for solving the examination timetabling problem in the Unit of Exact and Human Sciences at University of Algarve. Two objectives were determined as to minimize the number of conflicts within the same *group* and between *groups*. Wong et al. [31] used a Genetic Algorithm utilizing a Non-Elitist Replacement Strategy to solve a single examination timetabling problem at École de Technologie Supérieure. After genetic operators were applied, violations were fixed in a hill-climbing procedure.

Carter et al. [10] applied Different Heuristic Orderings based on Graph Coloring. Their experimental data became one of the commonly used benchmark examination timetabling problem instances. Gaspero and Schaerf [32] analyzed Tabu Search Approach using Graph Coloring based Heuristics. Merlot et al. [32] explored a hybrid approach for solving the examination timetabling problem that produces an initial feasible timetable via Constraint Programming, and then apply Simulated Annealing with Hill-Climbing to improve the solution. Petrovic et al. [34] introduced a Case Based Reasoning System to create initial solutions to be used by Great Deluge Algorithm. Burke et al. [35] proposed a general and fast adaptive method that arranges the heuristic to be used for ordering exams to be scheduled next. Their algorithm produced comparable results on a benchmark of problems with the current state of the art. Özcan and Ersoy [36] used a Violation Directed

Adaptive Hill-Climber within a Memetic Algorithm to solve the examination timetabling problem at Yeditepe University, Faculty of Engineering and Architecture. A Java tool named FES is introduced that utilizes XML as input/output format, proposed by Özcan in [37].

4.2. Heuristics for Examination Timetabling

Candidate solutions are encoded as an array of timeslots where the indices are final exams. Four heuristics are implemented to be used with hyperheuristics on examination timetabling problems. Three of these utilize tournament selection and assignment methods to improve the candidate solution, and the other one is a mutation operator. Each improving heuristic targets a different conflict. Heuristics which targets the conflicts stated Formulae (4.3) and (4.5) work in a similar way. They randomly choose a predetermined number of exams and select the exam with the highest number of targeted conflict among these. They randomly choose a predetermined number of timeslots and check the number of targeted conflicts if the exam was assigned to that timeslot. The timeslot with the minimum number of targeted conflict is then assigned to the selected exam.

The heuristic which targets the capacity conflict randomly chooses a predetermined number of timeslots and selects the timeslot with the maximum capacity conflict violations among these. A predetermined number of exams that are scheduled to this timeslot are chosen randomly and the exam that has the most attendants is selected among these. A predetermined number of timeslots are chosen randomly and the timeslot with the minimum number of attendants among these is assigned to the selected exam. Mutational heuristic passes over each exam in the array and assigns a random timeslot to the exam with a predetermined probability (1/*number of courses*).

4.3. Experimental Data

Hyperheuristic combinations are tested on Carter's Benchmark and the examination timetabling problem data of Yeditepe University, Faculty of Engineering. Properties and parameters for each problem instance are presented in the Table 4.1. Three slots are allocated for each day.

The candidate solutions are arrays of integers. Each item in the array represents an exam and the value of the array is the timeslot assigned to that exam. So the candidate solutions have the length of the number of exams.

Table 4.1. Properties and parameters of the examination timetabling problem instances used in the experiments

Instance	Exams	Students	Enrollment	Density	Days	Capacity
Carf92	543	18419	54062	0.14	12	2000
Cars91	682	16925	59022	0.13	17	1550
Earf83	181	941	6029	0.27	8	350
Hecs92	81	2823	10634	0.20	6	650
Kfus93	486	5349	25118	0.06	7	1955
Lsef91	381	2726	10919	0.06	6	635
Purs93	2419	30032	120690	0.03	10	5000
Ryes93	486	11483	45051	0.07	8	2055
Staf83	139	611	5539	0.14	4	3024
Tres92	261	4360	14901	0.18	10	655
Utas92	622	21267	58981	0.13	12	2800
Utes92	184	2749	11796	0.08	3	1240
Yorf83	190	1125	8108	0.29	7	300
Yue20011	140	559	3488	0.14	6	450
Yue20012	158	591	3706	0.14	6	450
Yue20013	30	234	447	0.19	2	150
Yue20021	168	826	5757	0.16	7	550
Yue20022	187	896	5860	0.16	7	550
Yue20023	40	420	790	0.19	2	150
Yue20031	177	1125	6716	0.15	6	550
Yue20032	210	1185	6837	0.14	6	550

4.4. Experimental Results

Each heuristic selection method and acceptance criterion combination is run on each problem instance 50 times. The average best fitness reached is used as the performance criterion for all experiments. The results for each benchmark function are presented in the Appendix C. There are two rows for each heuristic selection method and acceptance criterion combination in the tables in the Appendix C. The first row is the average value for the given performance criterion and the second row is the standard deviation for this average. The performances are evaluated statistically using t-test. Confidence interval is set to 95 per cent in t-test to determine significant performance variance. For each problem instance the heuristic selection method and acceptance criterion combinations which performed significantly better than the rest of the combinations are typed in boldface characters in the tables.

The heuristic selection method Choice Function combined with the acceptance criterion Monte Carlo performed significantly better than the rest of the hyperheuristic combinations on Ear-f-83, Lse-f-91, Rye-s-93, Ute-s-91, Yor-f-83, Yue20022, Yue20031 and Yue20032. The heuristic selection methods Choice Function and Random Descent combined with the acceptance criterion Monte Carlo performed significantly better than the rest of the hyperheuristic combinations on Yue20023. The heuristic selection methods Choice Function and Greedy combined with the acceptance criterion Monte Carlo performed significantly better than the rest of the hyperheuristic combinations on Hecs92.

The heuristic selection method Simple Random combined with the acceptance criterion Great Deluge performed significantly better than the rest of the hyperheuristic combinations on Kfu-s-93, Tre-s-92, Yue20011, Yue20012 and Yue20021. The heuristic selection method Simple Random combined with the acceptance criterion IE performed significantly better than the rest of the hyperheuristic combinations on Pur-s-93. All heuristic selection methods combined with the acceptance criterion Monte Carlo performed significantly better than the rest of the hyperheuristic combinations on Yue20013. All heuristic selection methods combined with the acceptance criterion Monte Carlo and the heuristic selection method Simple Random combined with the acceptance criterion Great

Deluge performed significantly better than the rest of the hyperheuristic combinations on Sta-f-83.

All heuristic selection methods except Greedy combined with the acceptance criterion IE performed significantly better than the rest of the hyperheuristic combinations on Uta-s-92. All heuristic selection methods except Simple Random and Greedy combined with the acceptance criterion IE performed significantly better than the rest of the hyperheuristic combinations on Car-s-91. All heuristic selection methods except Simple Random combined with the acceptance criteria IE and Monte Carlo and the heuristic selection method Simple Random combined with the acceptance criteria IE and Great Deluge performed significantly better than the rest of the hyperheuristic combinations on Car-f-92.

Average best fitness values for best performing heuristic selection method and acceptance criterion combinations are provided in Table 4.2. If several hyperheuristics share the same ranking, than only one of them appears in the table, marked with *. Rankings are assigned to each hyperheuristic combination to indicate their performance variances on examination timetabling problems. The combinations that do not have significant performance variances according to the t-test are assigned the same ranking. The combinations with better performance are placed higher in the ranking and therefore lower ranking numbers indicate better performance. The rankings are given in the Table 4.3. Seven combinations that have the top average rankings are presented in Figure 4.1.

Table 4.2. Average best fitness values for the best performing heuristic selection method and acceptance criterion combinations on each problem instance

Instance	Average Best Fitness	Standard Deviation	Algorithm
Carf92	-1.02E-02	1.18E-03	TABU_IE *
Cars91	-1.93E-01	1.20E-01	TABU_IE *
Earf83	-7.27E-03	4.94E-04	CF_MC
Hecs92	-2.19E-02	2.43E-03	CF_MC *
Kfus93	-3.40E-02	4.30E-03	SR_GD
Lsef91	-1.42E-02	1.38E-03	CF_MC
Purs93	-1.41E-03	6.98E-05	SR_IE
Ryes93	-1.08E-02	1.37E-03	CF_MC
Staf83	-2.68E-03	1.04E-05	SR_MC *
Tres92	-6.79E-02	1.08E-02	SR_GD
Utas92	-1.87E-02	1.79E-03	TABU_IE *
Utes92	-2.27E-03	8.64E-05	CF_MC
Yorf83	-8.32E-03	4.57E-04	CF_MC
Yue20011	-9.02E-02	1.07E-02	SR_GD
Yue20012	-7.54E-02	9.38E-03	SR_GD
Yue20013	-2.50E-01	0.00E+00	SR_MC *
Yue20021	-3.45E-02	4.55E-03	SR_GD
Yue20022	-1.26E-02	9.08E-04	CF_MC
Yue20023	-1.52E-02	2.69E-04	CF_MC *
Yue20031	-1.59E-02	1.65E-03	CF_MC
Yue20032	-5.42E-03	3.68E-04	CF_MC

Table 4.3. The performance rankings of each heuristic selection method and acceptance criterion combination on all problem instances

(a)

H.Heuristic	Carf92	Cars91	Earf83	Hecs92	Kfus93	Lsef91	Purs93
SR_AM	30.5	26.5	26.0	26.0	26.0	26.0	26.0
SR_OI	19.5	19.0	12.5	16.0	19.0	16.0	8.0
SR_IE	7.5	7.5	12.5	16.0	9.0	11.5	1.0
SR_MC	15.0	15.0	7.0	7.5	15.0	11.5	23.0
SR_GD	7.5	6.0	8.0	7.5	1.0	4.5	9.0
RD_AM	30.5	31.5	30.0	31.0	31.0	29.5	31.5
RD_OI	19.5	19.0	20.0	16.0	19.0	20.0	12.5
RD_IE	7.5	3.0	12.5	16.0	9.0	11.5	4.0
RD_MC	7.5	11.5	3.5	4.5	9.0	4.5	20.5
RD_GD	30.5	31.5	30.0	31.0	31.0	29.5	31.5
RP_AM	30.5	31.5	34.5	31.0	31.0	34.5	34.5
RP_OI	19.5	19.0	20.0	16.0	19.0	20.0	12.5
RP_IE	7.5	3.0	12.5	16.0	9.0	11.5	4.0
RP_MC	7.5	11.5	3.5	4.5	9.0	4.5	20.5
RP_GD	30.5	31.5	34.5	31.0	31.0	34.5	34.5
RPD_AM	30.5	31.5	30.0	31.0	31.0	29.5	31.5
RPD_OI	19.5	19.0	20.0	16.0	19.0	20.0	12.5
RPD_IE	7.5	3.0	12.5	16.0	9.0	11.5	4.0
RPD_MC	7.5	11.5	3.5	4.5	9.0	4.5	20.5
RPD_GD	30.5	31.5	30.0	31.0	31.0	29.5	31.5
CF_AM	30.5	26.5	30.0	31.0	31.0	33.5	27.0
CF_OI	19.5	19.0	20.0	16.0	19.0	20.0	12.5
CF_IE	7.5	3.0	12.5	16.0	9.0	11.5	4.0
CF_MC	7.5	9.0	1.0	1.5	3.0	1.0	16.5
CF_GD	19.5	19.0	20.0	16.0	19.0	20.0	12.5
TABU_AM	30.5	31.5	30.0	31.0	31.0	29.5	28.5
TABU_OI	19.5	19.0	20.0	16.0	19.0	20.0	12.5

TABU_IE	7.5	3.0	12.5	16.0	9.0	11.5	4.0
TABU_MC	7.5	11.5	3.5	4.5	9.0	4.5	20.5
TABU_GD	30.5	31.5	30.0	31.0	31.0	29.5	28.5
GR_AM	24.5	24.5	24.0	24.5	24.5	24.5	24.5
GR_OI	19.5	23.0	20.0	16.0	23.0	20.0	16.5
GR_IE	7.5	7.5	12.5	16.0	9.0	11.5	7.0
GR_MC	7.5	14.0	6.0	1.5	2.0	4.5	18.0
GR_GD	24.5	24.5	25.0	24.5	24.5	24.5	24.5

(b)

H.Heuristic	Ryes93	Staf83	Tres92	Utas92	Utes92	Yorf83	Y011
SR_AM	26.0	31.0	26.0	26.0	26.0	26.0	26.0
SR_OI	19.5	16.0	19.5	15.0	16.0	19.5	19.5
SR_IE	8.0	16.0	8.5	3.5	16.0	12.0	12.0
SR_MC	15.0	4.5	15.0	19.0	7.0	7.0	6.0
SR_GD	8.0	4.5	1.0	9.0	8.0	8.0	1.0
RD_AM	31.0	31.0	31.0	32.5	31.0	29.5	31.0
RD_OI	19.5	16.0	19.5	19.0	16.0	19.5	19.5
RD_IE	8.0	16.0	8.5	3.5	16.0	12.0	12.0
RD_MC	8.0	4.5	8.5	11.5	4.0	3.5	6.0
RD_GD	31.0	31.0	31.0	32.5	31.0	29.5	31.0
RP_AM	31.0	31.0	31.0	32.5	31.0	34.5	31.0
RP_OI	19.5	16.0	19.5	19.0	16.0	19.5	19.5
RP_IE	8.0	16.0	8.5	3.5	16.0	12.0	12.0
RP_MC	8.0	4.5	8.5	11.5	4.0	3.5	6.0
RP_GD	31.0	31.0	31.0	32.5	31.0	34.5	31.0
RPD_AM	31.0	31.0	31.0	32.5	31.0	29.5	31.0
RPD_OI	19.5	16.0	19.5	19.0	16.0	19.5	19.5
RPD_IE	8.0	16.0	8.5	3.5	16.0	12.0	12.0
RPD_MC	8.0	4.5	8.5	11.5	4.0	3.5	6.0
RPD_GD	31.0	31.0	31.0	32.5	31.0	29.5	31.0
CF_AM	31.0	26.0	31.0	27.0	31.0	33.0	31.0

CF_OI	19.5	16.0	19.5	19.0	16.0	19.5	19.5
CF_IE	8.0	16.0	8.5	3.5	16.0	12.0	12.0
CF_MC	1.0	4.5	2.0	8.0	1.0	1.0	3.0
CF_GD	19.5	16.0	19.5	19.0	16.0	19.5	19.5
TABU_AM	31.0	31.0	31.0	28.5	31.0	29.5	31.0
TABU_OI	19.5	16.0	19.5	19.0	16.0	19.5	19.5
TABU_IE	8.0	16.0	8.5	3.5	16.0	12.0	12.0
TABU_MC	8.0	4.5	8.5	11.5	4.0	3.5	6.0
TABU_GD	31.0	31.0	31.0	28.5	31.0	29.5	31.0
GR_AM	24.5	24.5	24.5	24.5	24.5	24.5	24.5
GR_OI	19.5	16.0	19.5	23.0	16.0	19.5	19.5
GR_IE	8.0	16.0	8.5	7.0	16.0	12.0	12.0
GR_MC	8.0	4.5	8.5	14.0	4.0	6.0	2.0
GR_GD	24.5	24.5	24.5	24.5	24.5	24.5	24.5

(c)

H.Heuristic	Y012	Y013	Y021	Y022	Y023	Y031	Y032
SR_AM	26.0	22.5	26.0	26.0	9.5	26.0	28.5
SR_OI	19.5	31.5	19.5	16.0	17.5	16.0	17.5
SR_IE	11.5	14.0	12.0	12.0	17.5	16.0	9.0
SR_MC	11.5	4.0	8.0	7.5	3.5	7.5	6.5
SR_GD	1.0	8.0	1.0	7.5	7.0	7.5	8.0
RD_AM	31.0	22.5	30.0	29.5	9.5	30.0	28.5
RD_OI	19.5	31.5	19.5	20.0	17.5	16.0	17.5
RD_IE	11.5	14.0	12.0	12.0	17.5	16.0	17.5
RD_MC	5.0	4.0	4.5	4.0	1.5	4.0	3.5
RD_GD	31.0	22.5	30.0	29.5	9.5	30.0	28.5
RP_AM	31.0	22.5	34.5	34.5	34.5	34.5	34.5
RP_OI	19.5	31.5	19.5	20.0	28.0	16.0	17.5
RP_IE	11.5	14.0	12.0	12.0	17.5	16.0	17.5
RP_MC	5.0	4.0	4.5	4.0	25.0	4.0	3.5
RP_GD	31.0	22.5	34.5	34.5	34.5	34.5	34.5

RPD_AM	31.0	22.5	30.0	29.5	31.5	30.0	28.5
RPD_OI	19.5	31.5	19.5	20.0	28.0	16.0	17.5
RPD_IE	11.5	14.0	12.0	12.0	17.5	16.0	17.5
RPD_MC	5.0	4.0	4.5	4.0	25.0	4.0	3.5
RPD_GD	31.0	22.5	30.0	29.5	31.5	30.0	32.5
CF_AM	31.0	22.5	30.0	33.0	9.5	30.0	32.5
CF_OI	19.5	31.5	19.5	20.0	17.5	16.0	17.5
CF_IE	11.5	14.0	12.0	12.0	17.5	16.0	17.5
CF_MC	5.0	4.0	4.5	1.0	1.5	1.0	1.0
CF_GD	19.5	31.5	19.5	20.0	17.5	16.0	17.5
TABU_AM	31.0	22.5	30.0	29.5	31.5	30.0	28.5
TABU_OI	19.5	31.5	19.5	20.0	28.0	16.0	17.5
TABU_IE	11.5	14.0	12.0	12.0	17.5	16.0	17.5
TABU_MC	5.0	4.0	4.5	4.0	25.0	4.0	3.5
TABU_GD	31.0	22.5	30.0	29.5	31.5	30.0	28.5
GR_AM	24.5	9.5	24.5	24.5	5.5	24.5	17.5
GR_OI	19.5	31.5	19.5	20.0	17.5	16.0	17.5
GR_IE	11.5	14.0	12.0	12.0	17.5	16.0	17.5
GR_MC	2.0	4.0	4.5	4.0	3.5	4.0	6.5
GR_GD	24.5	9.5	24.5	24.5	5.5	24.5	17.5

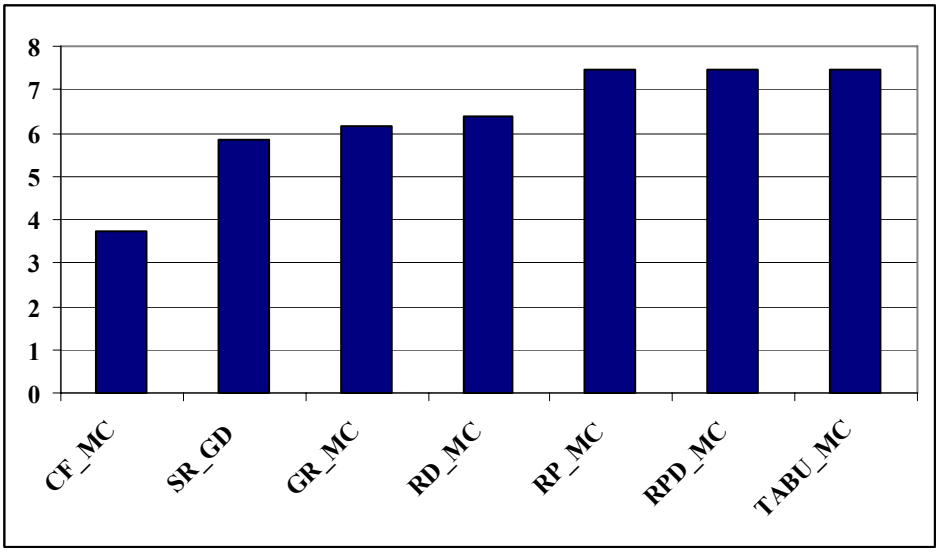

Figure 4.1. Top seven heuristic selection method and acceptance criterion combinations considering the average ranking

These results show that the number of enrollment parameter is the key parameter when applying hyperheuristics to the examination timetabling problem. According to this parameter the examination timetabling problem instances can be divided into two groups. There are seventeen instances in the first group and three instances in the second group. The timetabling problem instance Car-f-92 is on the border between these two groups. First group consists of instances with total number of enrollments less or equal to 45051. Second group consists of instances with total number of enrollments greater or equal to 58981. The timetabling instance Car-f-92 has 54062 enrollments.

The heuristic selection method Choice Function combined with the acceptance criterion Monte Carlo performed significantly better than the rest of the hyperheuristic combinations on eight instances of the first group. The heuristic selection method Simple Random combined with the acceptance criterion Great Deluge performed significantly better than the rest of the hyperheuristic combinations on five instances of the first group. Various subsets of the heuristic selection methods combined with either acceptance criteria Monte Carlo or Great Deluge performed significantly better than the rest of the hyperheuristic combinations on four instances of the first group.

The heuristic selection method Simple Random combined with the acceptance criterion IE performed significantly better than the rest of the hyperheuristic combinations on Pur-s-93, which belongs to the second group and has the largest number of students, exams and enrollments in the whole set. Various subsets of the heuristic selection methods combined with the acceptance criterion IE performed significantly better than the rest of the hyperheuristic combinations on two instances of the second group. Various subsets of the heuristic selection methods combined with the acceptance criteria IE or Monte Carlo and the heuristic selection method Simple Random combined with the acceptance criterion Great Deluge performed significantly better than the rest of the hyperheuristic combinations on the instance Car-f-92.

Since the same CPU time is given to all of the experiments it can be concluded that if enough CPU time is given to the heuristic selection method Choice Function with the acceptance criterion Monte Carlo or the heuristic selection method Simple Random combined with the acceptance criterion Great Deluge than they will perform significantly better then the rest of the hyperheuristic combinations on examination timetabling problem.

5. CONCLUSIONS

Although hyperheuristics are called "*heuristics to choose heuristics*" there is a second decisive step involved in them which is the acceptance of the candidate solution modified by the chosen heuristic. At the early stage of the research on hyperheuristics the focus was on the first step: the heuristic selection. Various heuristic selection methods have been developed and successfully applied to the real world problems. At these stage simple acceptance criteria AM and OI was used. However the focus of the research shifted from heuristic selection methods to acceptance criteria at recent work. Researchers deployed complex acceptance criteria in hyperheuristics combining them with the heuristic selection method Simple Random. The results showed that the choice of the acceptance criterion used in the hyperheuristics resulted in significant performance variances.

The idea behind the work presented in this thesis was the choice of the heuristic selection method and acceptance criterion combination and the choice of heuristic set will produce performance variances on different optimization problems. The soundness of this statement is empirically evaluated. A hyperheuristic framework with seven heuristic selection methods and five acceptance criteria is implemented. As a result 35 hyperheuristic combinations are ready to be applied to any optimization problem deploying any kind of heuristic set. All of the hyperheuristic combinations are applied to 14 benchmark functions with different properties and 21 real world examination timetabling problem instances with various sizes from various institutions.

The results of the experiments on benchmark functions yielded that various subsets of the heuristic selection methods combined with the acceptance criteria OI and IE performed well on nine benchmark functions. These functions consist of all of the discrete and deceptive functions, and various multimodal functions and unimodal functions with flat surfaces in the search space. The acceptance criteria OI and IE performed significantly better than the rest of the acceptance criteria on nine benchmark functions but there were two examples, Foxhole and Griewangk Functions, where they were significantly outperformed by the acceptance criteria AM, Monte Carlo and Great Deluge. Although the choice of the acceptance criterion was critical in the performance on eleven benchmark

functions the results on two examples, which are Griewangk and Sphere Functions, pointed out the importance of the heuristic selection methods.

During the research three further hyperheuristic frameworks are proposed which utilize hill-climbers in their own specific ways. The contributions of mutational heuristics are also evaluated within two frameworks. As a result it is found out that hill-climbers alone do not perform well on benchmark functions. Different types of mutational heuristics are needed to escape the local optima of the benchmark functions. Different hyperheuristic frameworks performed better than the rest on different benchmark functions. This fact shows that each of the proposed frameworks is useful on specific types of problems.

It is observed from the experimental results on the examination timetabling problem that the heuristic selection method Choice Function combined with the acceptance criterion Monte Carlo and the heuristic selection method Simple Random combined with the acceptance criterion Great Deluge performs better then the rest of the hyperheuristic combinations on most of the examination timetabling problems. However the number of enrollment parameter is very important when applying the hyperheuristics to the examination timetabling problem. The hyperheuristic combinations mentioned in this paragraph were not among the top performers on the problem instances with higher number of enrollments. This fact leads to the conclusion that the problem instances with higher number of enrollments might require more execution time.

The experimental results on both the benchmark functions and examination timetabling problem instances supported the statement that there will be significant performance variances between different heuristic selection methods combined with different acceptance criteria applied on different problems and even on different instances of the same type of problems. Therefore the knowledge of hyperheuristic combination performances on specific types of optimization problems and even on specific types of instances is critical for successful applications. In this thesis the performance of 35 such hyperheuristic combinations are evaluated on 14 benchmark functions and 21 examination timetabling problem instances. The results are presented and generalized. Future work may involve development and implementation of further heuristic selection methods and

acceptance criteria and the performance evaluations of the resulting hyperheuristic combinations on different optimization problems.

APPENDIX A: EXPERIMENTAL RESULTS TABLES OF HYPERHEURISTICS PATTERNS ON BENCHMARK FUNCTIONS

Results of performance evaluations of hyperheuristic patterns H1 – H11 are given in Table A.1 - Table A.14. Two columns are given for each: “Average Best Fitness Obtained” and “Average Number of Evaluations to Converge”. The mean value is given in the first column and the standard deviation in the second column. The runs where the global optimum of the benchmark function is found by the optimization method is considered to be successful. “S. R.” stands for the ratio of successful runs to all runs. The performance criterion is the “Average Number of Evaluations to Convergence” where at least one hyperheuristic pattern converges to the global optimum. If non of the hyperheuristic patterns converged to the global optimum the performance criterion is the “Average Best Fitness”. The results of the experiments are statistically evaluated using t-test with 95 per cent confidence level. The hyperheuristic patterns which perform significantly better then the rest of the hyperheuristic patterns are highlighted using boldface characters.

Table A.1. Results of performance evaluations of hyperheuristic patterns on Sphere Function

	Average Best Fitness		**Average Num. of Eval.**		**S. R.**
H1	**0.00E+00**	**0.00E+00**	**1.46E+03**	**4.37E+02**	**100.00%**
H2	**0.00E+00**	**0.00E+00**	**1.48E+03**	**5.61E+02**	**100.00%**
H3	**0.00E+00**	**0.00E+00**	**1.55E+03**	**5.38E+02**	**100.00%**
H4	**0.00E+00**	**0.00E+00**	**1.53E+03**	**5.55E+02**	**100.00%**
H5	**0.00E+00**	**0.00E+00**	**1.49E+03**	**5.07E+02**	**100.00%**
H6	**0.00E+00**	**0.00E+00**	**1.53E+03**	**4.60E+02**	**100.00%**
H7	**0.00E+00**	**0.00E+00**	**1.62E+03**	**4.99E+02**	**100.00%**
H8	**0.00E+00**	**0.00E+00**	**1.73E+03**	**5.80E+02**	**100.00%**
H9	**0.00E+00**	**0.00E+00**	**1.44E+03**	**5.67E+02**	**100.00%**
H10	0.00E+00	0.00E+00	1.08E+04	1.15E+03	100.00%
H11	0.00E+00	0.00E+00	4.06E+03	3.18E+03	100.00%

Table A.2. Results of performance evaluations of hyperheuristic patterns on Rosenbrock Function

	Average Best Fitness		**Average Num. of Eval.**		**S. R.**
H1	1.70E-12	4.61E-14	-	-	0.00%
H2	1.70E-12	5.90E-14	-	-	0.00%
H3	1.70E-12	4.70E-14	-	-	0.00%
H4	1.70E-12	4.96E-14	-	-	0.00%
H5	**4.03E-28**	**0.00E+00**	**1.43E+07**	**7.18E+06**	**100.00%**
H6	1.70E-12	6.23E-14	-	-	0.00%
H7	1.70E-12	4.13E-14	-	-	0.00%
H8	2.03E-15	1.27E-14	-	-	90.00%
H9	1.71E-12	4.30E-14	-	-	0.00%
H10	4.03E-28	0.00E+00	2.20E+07	5.40E+06	100.00%
H11	3.16E-03	2.60E-03	-	-	0.00%

Table A.3. Results of performance evaluations of hyperheuristic patterns on Step Function

	Average Best Fitness		**Average Num. of Eval.**		**S. R.**
H1	1.05E+01	2.89E+00	-	-	0.00%
H2	1.11E+01	3.02E+00	-	-	0.00%
H3	0.00E+00	0.00E+00	6.83E+05	3.58E+05	100.00%
H4	1.06E+01	2.95E+00	-	-	0.00%
H5	1.05E+01	3.23E+00	-	-	0.00%
H6	0.00E+00	0.00E+00	2.51E+05	1.33E+05	100.00%
H7	3.76E+00	9.71E-01	-	-	0.00%
H8	0.00E+00	0.00E+00	3.24E+05	1.24E+05	100.00%
H9	0.00E+00	0.00E+00	7.88E+05	5.46E+05	100.00%
H10	**0.00E+00**	**0.00E+00**	**1.36E+05**	**5.92E+04**	**100.00%**
H11	5.20E-01	5.00E-01	-	-	48.00%

Table A.4. Results of performance evaluations of hyperheuristic patterns on Quartic Function

	Average Best Fitness		**Average Num. of Eval.**		**S. R.**
H1	1.78E+00	7.76E-01	-	-	8.00%
H2	1.89E+00	7.78E-01	-	-	0.00%
H3	1.53E+00	4.16E-01	-	-	8.00%
H4	1.65E+00	5.93E-01	-	-	12.00%
H5	**1.33E+00**	**3.04E-01**	**-**	**-**	**12.00%**
H6	1.55E+00	4.27E-01	-	-	4.00%
H7	**1.24E+00**	**2.57E-01**	**-**	**-**	**18.00%**
H8	**1.20E+00**	**2.70E-01**	**-**	**-**	**26.00%**
H9	1.52E+00	4.71E-01	-	-	6.00%
H10	**1.18E+00**	**1.40E-01**	**-**	**-**	**8.00%**
H11	1.47E+00	1.48E-01	-	-	4.00%

Table A.5. Results of performance evaluations of hyperheuristic patterns on Foxhole Function

	Average Best Fitness		**Average Num. of Eval.**		**S. R.**
H1	3.66E+00	2.53E+00	-	-	0.00%
H2	2.32E+00	1.90E+00	-	-	6.00%
H3	9.98E-01	6.75E-16	-	-	12.00%
H4	9.98E-01	6.97E-16	-	-	4.00%
H5	2.07E+00	1.68E+00	-	-	0.00%
H6	9.98E-01	3.12E-16	-	-	34.00%
H7	9.98E-01	7.54E-15	-	-	4.00%
H8	9.98E-01	2.85E-16	-	-	46.00%
H9	9.98E-01	8.01E-16	-	-	14.00%
H10	9.98E-01	2.75E-16	-	-	40.00%
H11	**9.98E-01**	**1.11E-16**	**1.39E+05**	**1.07E+05**	**100.00%**

Table A.6. Results of performance evaluations of hyperheuristic patterns on Rastrigin Function

	Average Best Fitness		**Average Num. of Eval.**		**S. R.**
H1	2.36E+01	5.97E+00	-	-	0.00%
H2	2.31E+01	4.56E+00	-	-	0.00%
H3	0.00E+00	0.00E+00	2.28E+06	9.62E+05	100.00%
H4	2.24E+01	4.79E+00	-	-	0.00%
H5	2.30E+01	5.81E+00	-	-	0.00%
H6	0.00E+00	0.00E+00	3.88E+05	1.69E+05	100.00%
H7	6.25E+00	1.44E+00	-	-	0.00%
H8	0.00E+00	0.00E+00	5.33E+05	2.11E+05	100.00%
H9	0.00E+00	0.00E+00	2.33E+06	1.40E+06	100.00%
H10	**0.00E+00**	**0.00E+00**	**2.52E+05**	**1.17E+05**	**100.00%**
H11	2.51E+00	5.59E-01	-	-	0.00%

Table A.7. Results of performance evaluations of hyperheuristic patterns on Schwefel Function

	Average Best Fitness		**Average Num. of Eval.**		**S. R.**
H1	1.48E+03	4.12E+02	-	-	0.00%
H2	1.39E+03	3.79E+02	-	-	0.00%
H3	1.27E-04	1.27E-13	-	-	98.00%
H4	1.19E+03	2.75E+02	-	-	0.00%
H5	1.48E+03	3.42E+02	-	-	0.00%
H6	1.27E-04	0.00E+00	1.57E+05	6.75E+04	100.00%
H7	2.09E+02	2.09E+02	-	-	2.00%
H8	1.27E-04	0.00E+00	1.98E+05	1.09E+05	100.00%
H9	1.27E-04	0.00E+00	6.08E+05	2.88E+05	100.00%
H10	**1.27E-04**	**0.00E+00**	**9.81E+04**	**4.34E+04**	**100.00%**
H11	1.66E+01	4.11E+01	-	-	82.00%

Table A.8. Results of performance evaluations of hyperheuristic patterns on Griewangk Function

	Average Best Fitness		**Average Num. of Eval.**		**S. R.**
H1	5.87E-01	1.11E+00	-	-	4.00%
H2	9.32E-01	1.63E+00	-	-	4.00%
H3	5.19E-02	2.38E-02	-	-	4.00%
H4	5.42E-01	5.44E-01	-	-	6.00%
H5	0.00E+00	0.00E+00	4.34E+06	2.79E+06	100.00%
H6	1.03E-02	5.42E-03	-	-	12.00%
H7	4.54E-01	3.92E-01	-	-	4.00%
H8	0.00E+00	0.00E+00	4.88E+06	3.91E+06	100.00%
H9	5.04E-02	2.72E-02	-	-	4.00%
H10	0.00E+00	0.00E+00	2.37E+06	1.52E+06	100.00%
H11	**0.00E+00**	**0.00E+00**	**2.95E+05**	**2.87E+05**	**100.00%**

Table A.9. Results of performance evaluations of hyperheuristic patterns on Ackley Function

	Average Best Fitness		**Average Num. of Eval.**		**S. R.**
H1	4.32E+00	7.74E+00	-	-	76.00%
H2	5.44E+00	8.33E+00	-	-	70.00%
H3	**2.89E-14**	**0.00E+00**	**9.35E+03**	**1.88E+04**	**100.00%**
H4	2.51E-01	1.75E+00	-	-	98.00%
H5	4.00E+00	7.58E+00	-	-	78.00%
H6	**2.89E-14**	**0.00E+00**	**4.00E+03**	**4.16E+03**	**100.00%**
H7	**2.89E-14**	**0.00E+00**	**1.08E+04**	**3.44E+04**	**100.00%**
H8	**2.89E-14**	**0.00E+00**	**3.02E+03**	**2.45E+03**	**100.00%**
H9	**2.89E-14**	**0.00E+00**	**5.79E+03**	**1.02E+04**	**100.00%**
H10	**2.89E-14**	**0.00E+00**	**1.10E+04**	**1.69E+03**	**100.00%**
H11	2.89E-14	0.00E+00	2.20E+04	2.00E+04	100.00%

Table A.10. Results of performance evaluations of hyperheuristic patterns on Easom Function

	Average Best Fitness		**Average Num. of Eval.**		**S. R.**
H1	0.00E+00	0.00E+00	-	-	0.00%
H2	0.00E+00	0.00E+00	-	-	0.00%
H3	-6.00E-02	2.37E-01	-	-	6.00%
H4	-2.00E-02	1.40E-01	-	-	2.00%
H5	0.00E+00	0.00E+00	-	-	0.00%
H6	-1.00E+00	9.99E-16	6.66E+06	5.94E+06	100.00%
H7	-2.01E-02	1.40E-01	-	-	2.00%
H8	-1.00E+00	9.99E-16	4.16E+06	3.13E+06	100.00%
H9	-3.08E-05	3.94E-05	-	-	0.00%
H10	**-1.00E+00**	**9.99E-16**	**1.71E+06**	**1.16E+06**	**100.00%**
H11	-4.07E-01	4.64E-01	-	-	34.00%

Table A.11. Results of performance evaluations of hyperheuristic patterns on Rotated Hyper-Ellipsoid Function

	Average Best Fitness		**Average Num. of Eval.**		**S. R.**
H1	1.48E-12	5.23E-13	-	-	0.00%
H2	1.36E-12	6.02E-13	-	-	0.00%
H3	1.67E-12	4.22E-13	-	-	0.00%
H4	1.58E-12	5.80E-13	-	-	0.00%
H5	**7.78E-26**	**0.00E+00**	**2.48E+05**	**7.52E+04**	**100.00%**
H6	1.70E-12	6.22E-13	-	-	0.00%
H7	1.58E-12	5.55E-13	-	-	0.00%
H8	7.78E-26	0.00E+00	3.58E+05	1.23E+05	100.00%
H9	1.31E-12	5.32E-13	-	-	0.00%
H10	**7.78E-26**	**0.00E+00**	**2.41E+05**	**5.04E+04**	**100.00%**
H11	4.26E+01	1.88E+01	-	-	0.00%

Table A.12. Results of performance evaluations of hyperheuristic patterns on Royal Road Function

	Average Best Fitness		**Average Num. of Eval.**		**S. R.**
H1	7.76E+00	4.72E-01	-	-	0.00%
H2	7.76E+00	4.72E-01	-	-	0.00%
H3	0.00E+00	0.00E+00	1.61E+05	6.89E+04	100.00%
H4	4.24E+00	5.12E-01	-	-	0.00%
H5	7.76E+00	4.72E-01	-	-	0.00%
H6	0.00E+00	0.00E+00	2.06E+05	9.74E+04	100.00%
H7	2.96E+00	1.96E-01	-	-	0.00%
H8	0.00E+00	0.00E+00	2.46E+05	9.46E+04	100.00%
H9	0.00E+00	0.00E+00	1.62E+05	6.63E+04	100.00%
H10	**0.00E+00**	**0.00E+00**	**1.14E+05**	**5.53E+04**	**100.00%**
H11	2.82E+00	3.84E-01	-	-	0.00%

Table A.13. Results of performance evaluations of hyperheuristic patterns on Goldberg Function

	Average Best Fitness		**Average Num. of Eval.**		**S. R.**
H1	4.20E+01	5.62E+00	-	-	0.00%
H2	4.14E+01	6.47E+00	-	-	0.00%
H3	0.00E+00	0.00E+00	3.19E+05	8.87E+04	100.00%
H4	4.07E+01	7.28E+00	-	-	0.00%
H5	4.22E+01	5.05E+00	-	-	0.00%
H6	0.00E+00	0.00E+00	2.05E+05	5.57E+04	100.00%
H7	2.04E+01	1.61E+00	-	-	0.00%
H8	0.00E+00	0.00E+00	2.73E+05	9.73E+04	100.00%
H9	0.00E+00	0.00E+00	3.84E+05	1.35E+05	100.00%
H10	**0.00E+00**	**0.00E+00**	**1.02E+05**	**3.91E+04**	**100.00%**
H11	1.96E+01	1.42E+00	-	-	0.00%

Table A.14. Results of performance evaluations of hyperheuristic patterns on Whitley Function

	Average Best Fitness		**Average Num. of Eval.**		**S. R.**
H1	9.96E+00	1.30E+00	-	-	0.00%
H2	1.02E+01	1.74E+00	-	-	0.00%
H3	0.00E+00	0.00E+00	3.44E+04	1.15E+04	100.00%
H4	1.56E+00	9.20E-01	-	-	24.00%
H5	1.01E+01	1.72E+00	-	-	0.00%
H6	0.00E+00	0.00E+00	1.73E+04	9.05E+03	100.00%
H7	0.00E+00	0.00E+00	2.97E+07	3.02E+07	100.00%
H8	0.00E+00	0.00E+00	1.49E+04	9.34E+03	100.00%
H9	0.00E+00	0.00E+00	3.33E+04	1.28E+04	100.00%
H10	**0.00E+00**	**0.00E+00**	**7.90E+03**	**5.18E+03**	**100.00%**
H11	0.00E+00	0.00E+00	6.73E+06	5.96E+06	100.00%

APPENDIX B: EXPERIMENTAL RESULTS TABLES AND GRAPHICS OF HYPERHEURISTICS ON BENCHMARK FUNCTIONS

Results of performance evaluations of hyperheuristic algorithms composed of various heuristic selection methods and acceptance criteria are given in Table B.1 - Table B.14. Two rows are given for each algorithm. The mean value is given in the first row and the standard deviation in the second row. The runs where the global optimum of the benchmark function is found by the optimization method is considered to be successful. The performance criterion is the "Average Number of Evaluations to Convergence" where at least one hyperheuristic pattern converges to the global optimum. If non of the hyperheuristic patterns converged to the global optimum the performance criterion is the "Average Best Fitness". The results of the experiments are statistically evaluated using t-test with 95 per cent confidence level. The hyperheuristic algorithms which perform significantly better then the rest of the hyperheuristic algorithms are highlighted using boldface characters. *NC* stands for "Not Converged" throughout the tables.

Table B.1. Average number of fitness evaluations to convergence for Sphere Function

	AM	*OI*	*IE*	*MC*	*GD*
SR	**2.86E+03**	**2.15E+03**	**2.15E+03**	**2.69E+03**	**2.86E+03**
	2.65E+03	**1.38E+03**	**1.38E+03**	**2.04E+03**	**2.65E+03**
RD	5.61E+03	4.74E+03	4.74E+03	5.31E+03	5.61E+03
	2.94E+03	2.54E+03	2.54E+03	3.72E+03	2.94E+03
RP	6.95E+03	**1.55E+03**	**1.55E+03**	**2.62E+03**	6.95E+03
	2.83E+04	**3.67E+02**	**3.67E+02**	**2.06E+03**	2.83E+04
RPD	5.18E+03	4.64E+03	4.64E+03	5.75E+03	5.18E+03
	2.37E+03	2.42E+03	2.42E+03	2.41E+03	2.37E+03
CF	**2.19E+03**	**1.76E+03**	**1.58E+03**	**1.75E+03**	**2.02E+03**
	9.10E+02	**5.77E+02**	**5.09E+02**	**5.63E+02**	**9.30E+02**
TABU	5.73E+03	5.11E+03	5.11E+03	4.84E+03	5.73E+03
	2.58E+03	2.36E+03	2.36E+03	2.50E+03	2.58E+03
GR	6.80E+03	6.80E+03	6.80E+03	6.80E+03	6.80E+03
	3.01E+03	3.01E+03	3.01E+03	3.01E+03	3.01E+03

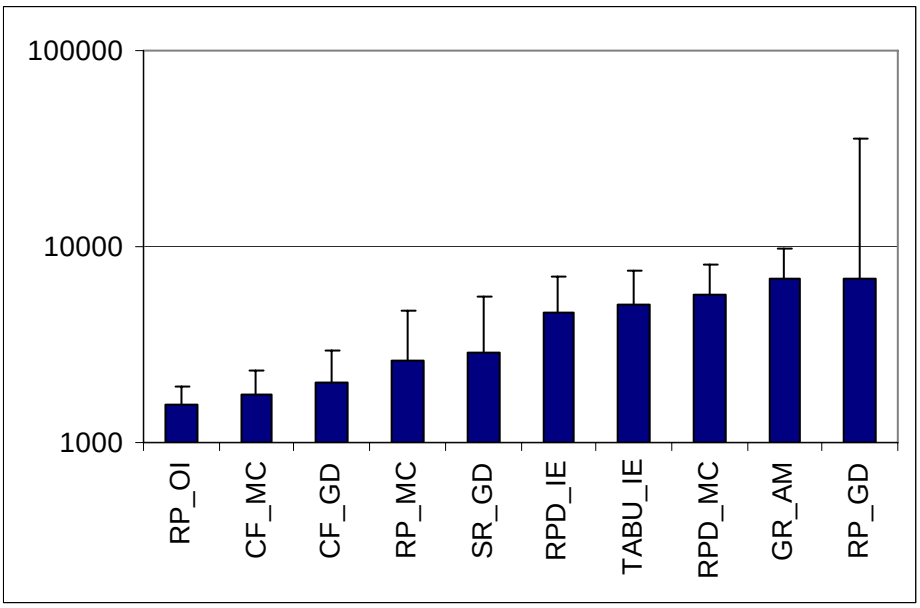

Figure B.1. Average number of fitness evaluations to convergence for Sphere Function

Table B.2. Average best fitness values for Rosenbrock Function

	AM	*OI*	*IE*	*MC*	*GD*
SR	3.52E-05	**5.38E-17**	**5.88E-16**	1.04E-05	6.28E-06
	4.40E-05	**3.77E-16**	**3.03E-15**	1.64E-05	1.05E-05
RD	7.95E-13	**1.79E-15**	**1.79E-15**	5.96E-13	5.83E-13
	5.57E-13	**1.18E-14**	**1.18E-14**	4.60E-13	5.04E-13
RP	8.77E-03	**3.44E-16**	**1.85E-15**	1.11E-04	2.13E-04
	6.48E-03	**1.13E-15**	**1.09E-14**	2.09E-04	4.92E-04
RPD	1.25E-12	**4.74E-16**	**2.66E-16**	1.07E-12	1.02E-12
	5.72E-13	**1.57E-15**	**1.53E-15**	5.53E-13	6.85E-13
CF	5.50E-07	**1.28E-16**	**1.82E-16**	3.28E-07	3.81E-07
	6.68E-07	**8.98E-16**	**9.67E-16**	4.73E-07	5.59E-07
TABU	5.22E-13	**2.57E-16**	**2.57E-16**	3.47E-13	3.60E-13
	4.21E-13	**1.26E-15**	**1.26E-15**	2.95E-13	3.97E-13
GR	**2.04E-16**	1.50E-12	**5.38E-17**	**1.50E-16**	**7.26E-16**
	8.42E-16	4.39E-13	**3.77E-16**	**7.64E-16**	**3.13E-15**

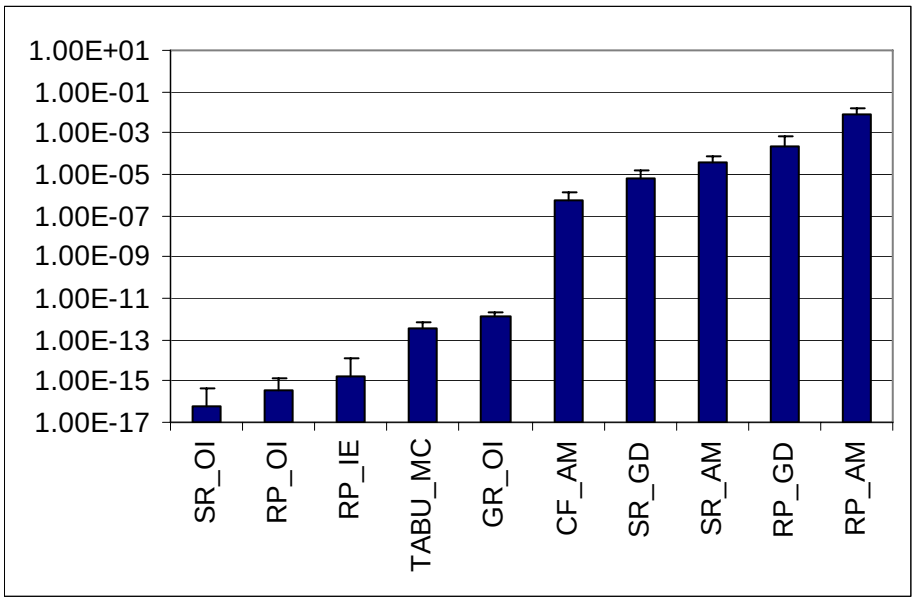

Figure B.2. Average best fitness values for Rosenbrock Function

Table B.3. Average number of fitness evaluations to convergence for Step Function

	AM	*OI*	*IE*	*MC*	*GD*
SR	*NC*	**4.34E+05**	**3.27E+05**	*NC*	4.12E+07
	NC	**2.22E+05**	**1.94E+05**	*NC*	5.01E+06
RD	*NC*	**3.81E+05**	**2.81E+05**	*NC*	4.16E+07
	NC	**1.75E+05**	**1.54E+05**	*NC*	9.01E+05
RP	*NC*	**4.34E+05**	**2.98E+05**	*NC*	4.21E+07
	NC	**2.16E+05**	**1.38E+05**	*NC*	1.06E+06
RPD	*NC*	**4.17E+05**	**3.27E+05**	*NC*	4.12E+07
	NC	**2.68E+05**	**1.52E+05**	*NC*	3.62E+06
CF	*NC*	**3.88E+05**	**2.90E+05**	*NC*	4.15E+07
	NC	**2.12E+05**	**1.25E+05**	*NC*	1.43E+06
TABU	*NC*	**3.93E+05**	**3.00E+05**	*NC*	4.14E+07
	NC	**1.88E+05**	**1.68E+05**	*NC*	1.00E+06
GR	6.75E+05	*NC*	6.75E+05	6.04E+05	6.75E+05
	2.70E+05	*NC*	2.70E+05	2.56E+05	2.70E+05

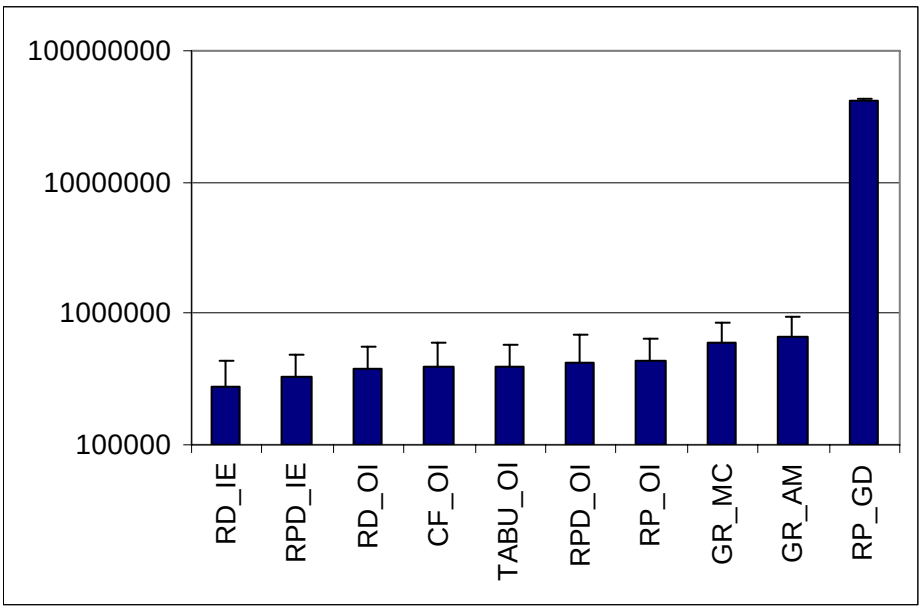

Figure B.3. Average number of fitness evaluations to convergence for Step Function

Table B.4. Average best fitness values for Quatic Function

	AM	*OI*	*IE*	*MC*	*GD*
SR	1.50E+00	**1.25E+00**	**1.20E+00**	**1.31E+00**	**1.20E+00**
	1.32E-01	**2.34E-01**	**2.63E-01**	**1.33E-01**	**1.57E-01**
RD	1.41E+00	**1.25E+00**	**1.21E+00**	**1.30E+00**	**1.18E+00**
	1.44E-01	**2.60E-01**	**2.10E-01**	**1.08E-01**	**1.56E-01**
RP	1.55E+00	**1.22E+00**	**1.25E+00**	**1.30E+00**	**1.20E+00**
	1.60E-01	**2.58E-01**	**1.65E-01**	**1.58E-01**	**1.64E-01**
RPD	1.49E+00	**1.24E+00**	**1.21E+00**	**1.28E+00**	**1.19E+00**
	1.40E-01	**2.03E-01**	**2.17E-01**	**1.08E-01**	**1.32E-01**
CF	1.45E+00	**1.22E+00**	**1.21E+00**	**1.30E+00**	**1.17E+00**
	1.32E-01	**2.15E-01**	**2.23E-01**	**1.22E-01**	**1.74E-01**
TABU	1.39E+00	**1.19E+00**	**1.20E+00**	**1.26E+00**	**1.18E+00**
	1.37E-01	**1.88E-01**	**2.16E-01**	**1.35E-01**	**1.40E-01**
GR	**1.26E+00**	1.87E+00	**1.30E+00**	**1.23E+00**	**1.26E+00**
	2.49E-01	6.20E-01	**2.54E-01**	**2.23E-01**	**2.16E-01**

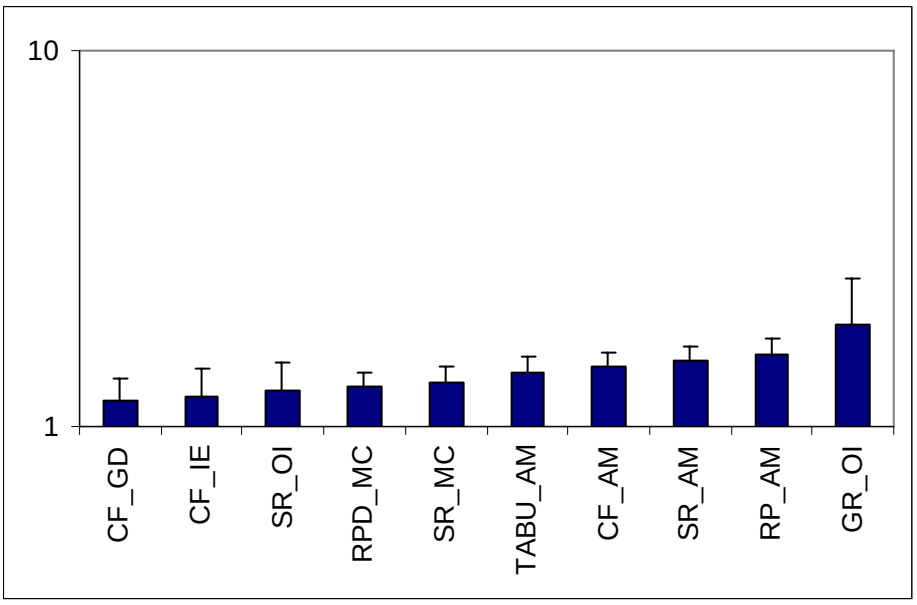

Figure B.4. Average best fitness values for Quartic Function

Table B.5. Average number of fitness evaluations to convergence for Foxhole Function

	AM	*OI*	*IE*	*MC*	*GD*
SR	**4.72E+04**	*NC*	*NC*	**4.09E+04**	**4.61E+04**
	4.42E+04	*NC*	*NC*	**4.04E+04**	**4.31E+04**
RD	**4.44E+04**	*NC*	*NC*	**3.18E+04**	**3.94E+04**
	3.86E+04	*NC*	*NC*	**2.95E+04**	**3.28E+04**
RP	1.04E+05	*NC*	*NC*	**5.55E+04**	1.09E+05
	1.40E+05	*NC*	*NC*	**6.49E+04**	1.69E+05
RPD	**3.40E+04**	*NC*	*NC*	**5.36E+04**	**3.31E+04**
	3.54E+04	*NC*	*NC*	**7.07E+04**	**3.40E+04**
CF	**2.63E+04**	*NC*	*NC*	**3.82E+04**	**3.37E+04**
	2.58E+04	*NC*	*NC*	**3.82E+04**	**3.25E+04**
TABU	**4.94E+04**	*NC*	*NC*	**4.63E+04**	**4.52E+04**
	4.04E+04	*NC*	*NC*	**4.05E+04**	**4.31E+04**
GR	*NC*	*NC*	*NC*	*NC*	*NC*
	NC	*NC*	*NC*	*NC*	*NC*

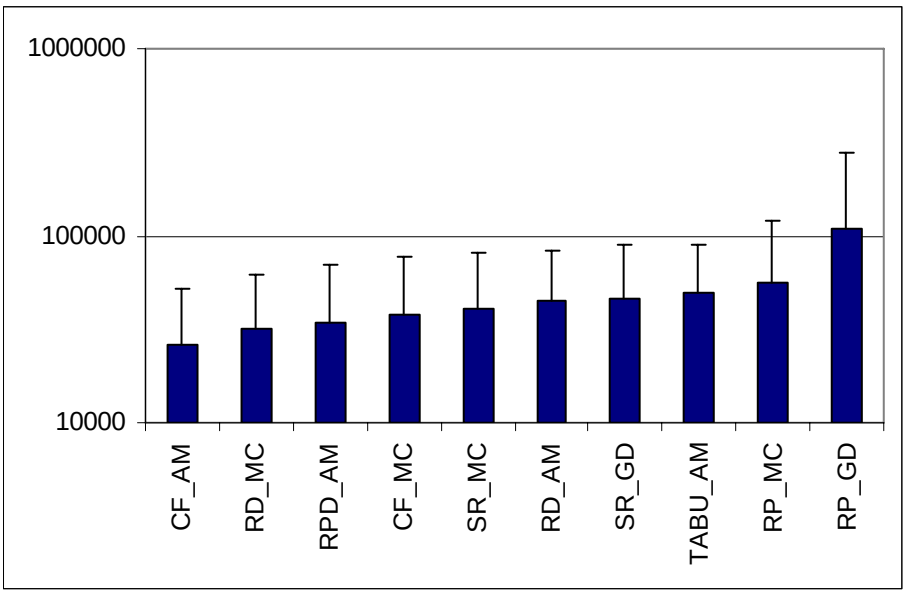

Figure B.5. Average number of fitness evaluations to convergence for Foxhole Function

Table B.6. Average number of fitness evaluations to convergence for Rastrigin Function

	AM	*OI*	*IE*	*MC*	*GD*
SR	*NC*	**4.69E+05**	**5.11E+05**	*NC*	*NC*
	NC	**2.07E+05**	**2.25E+05**	*NC*	*NC*
RD	*NC*	**4.56E+05**	**4.49E+05**	*NC*	*NC*
	NC	**2.39E+05**	**1.76E+05**	*NC*	*NC*
RP	*NC*	**4.79E+05**	**4.66E+05**	*NC*	*NC*
	NC	**2.05E+05**	**2.09E+05**	*NC*	*NC*
RPD	*NC*	**4.82E+05**	**4.76E+05**	*NC*	*NC*
	NC	**1.84E+05**	**2.07E+05**	*NC*	*NC*
CF	*NC*	**4.53E+05**	**4.82E+05**	*NC*	*NC*
	NC	**2.03E+05**	**2.06E+05**	*NC*	*NC*
TABU	*NC*	**4.59E+05**	**4.98E+05**	*NC*	*NC*
	NC	**1.46E+05**	**2.30E+05**	*NC*	*NC*
GR	8.37E+05	*NC*	8.37E+05	8.72E+05	8.37E+05
	2.88E+05	*NC*	2.88E+05	3.39E+05	2.88E+05

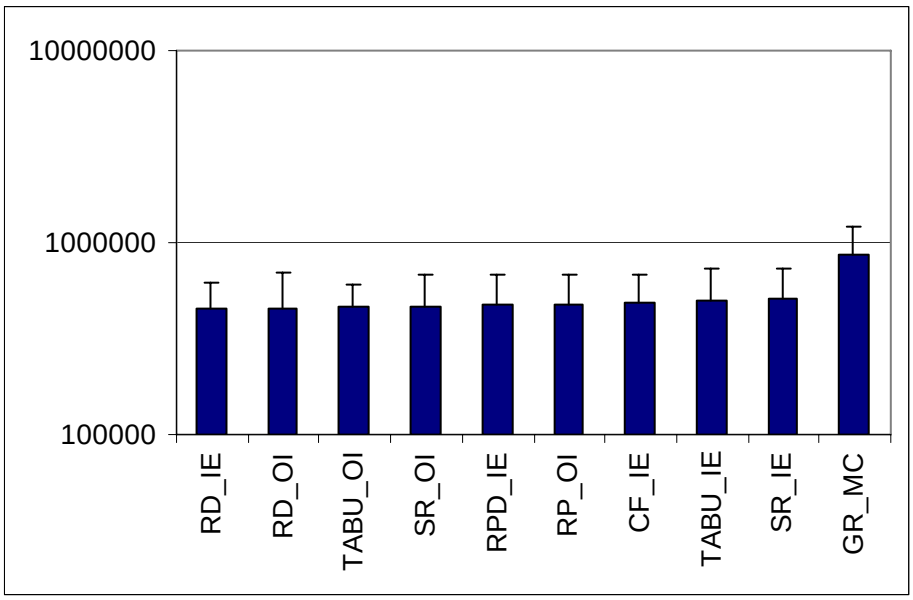

Figure B.6. Average number of fitness evaluations to convergence for Rastrigin Function

Table B.7. Average number of fitness evaluations to convergence for Schwefel Function

	AM	*OI*	*IE*	*MC*	*GD*
SR	*NC*	**1.60E+05**	**1.68E+05**	3.78E+07	3.56E+07
	NC	**1.04E+05**	**1.07E+05**	3.41E+06	2.23E+06
RD	*NC*	**1.47E+05**	**1.72E+05**	4.02E+07	3.75E+07
	NC	**7.67E+04**	**9.38E+04**	3.35E+06	2.17E+06
RP	*NC*	**1.51E+05**	**1.68E+05**	3.67E+07	3.34E+07
	NC	**7.17E+04**	**7.84E+04**	6.75E+06	4.97E+06
RPD	*NC*	**1.76E+05**	**1.78E+05**	3.95E+07	3.70E+07
	NC	**8.07E+04**	**9.35E+04**	7.28E+06	5.20E+06
CF	*NC*	**1.73E+05**	**1.57E+05**	3.74E+07	3.45E+07
	NC	**8.29E+04**	**7.60E+04**	8.12E+06	6.57E+06
TABU	*NC*	**1.35E+05**	**1.95E+05**	3.91E+07	3.51E+07
	NC	**5.59E+04**	**1.00E+05**	2.50E+06	3.01E+06
GR	2.64E+05	*NC*	2.64E+05	3.11E+05	2.64E+05
	1.48E+05	*NC*	1.48E+05	1.55E+05	1.48E+05

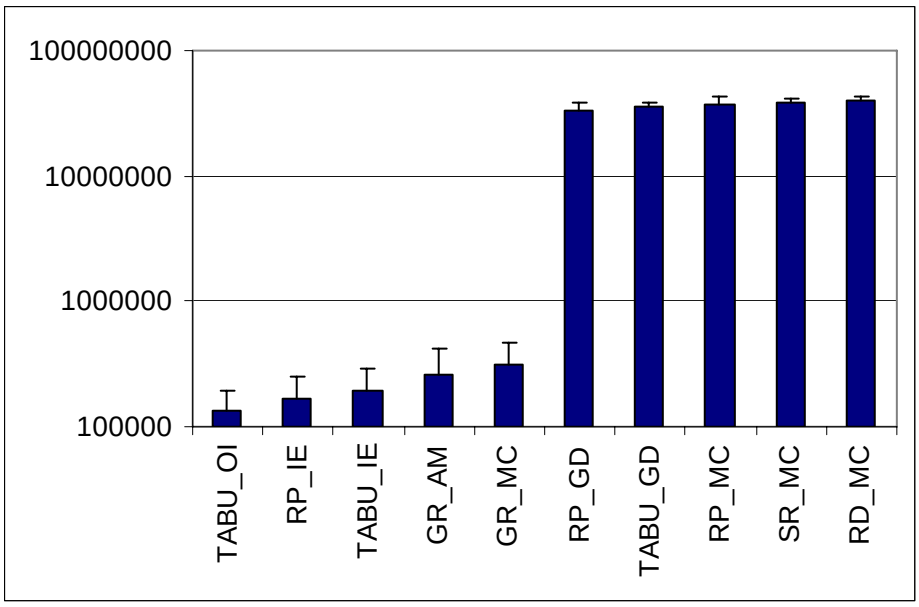

Figure B.7. Average number of fitness evaluations to convergence for Schwefel Function

Table B.8. Average number of fitness evaluations to convergence for Griewangk Function

	AM	*OI*	*IE*	*MC*	*GD*
SR	3.64E+05	5.72E+06	5.72E+06	4.40E+05	3.64E+05
	2.88E+05	5.16E+06	5.16E+06	4.61E+05	2.88E+05
RD	8.36E+05	6.03E+06	6.03E+06	6.92E+05	8.36E+05
	6.98E+05	5.03E+06	5.03E+06	7.06E+05	6.98E+05
RP	6.30E+05	5.15E+06	5.15E+06	5.10E+05	6.30E+05
	1.83E+06	4.02E+06	4.02E+06	7.05E+05	1.83E+06
RPD	7.60E+05	4.23E+06	4.23E+06	1.05E+06	7.60E+05
	9.03E+05	3.20E+06	3.20E+06	1.36E+06	9.03E+05
CF	**2.21E+05**	5.29E+06	4.65E+06	**1.71E+05**	**1.86E+05**
	2.00E+05	3.90E+06	3.33E+06	**1.09E+05**	**1.40E+05**
TABU	9.85E+05	5.27E+06	5.27E+06	1.08E+06	9.85E+05
	8.69E+05	5.10E+06	5.10E+06	9.06E+05	8.69E+05
GR	4.36E+06	*NC*	4.36E+06	4.83E+06	4.36E+06
	3.58E+06	*NC*	3.58E+06	3.45E+06	3.58E+06

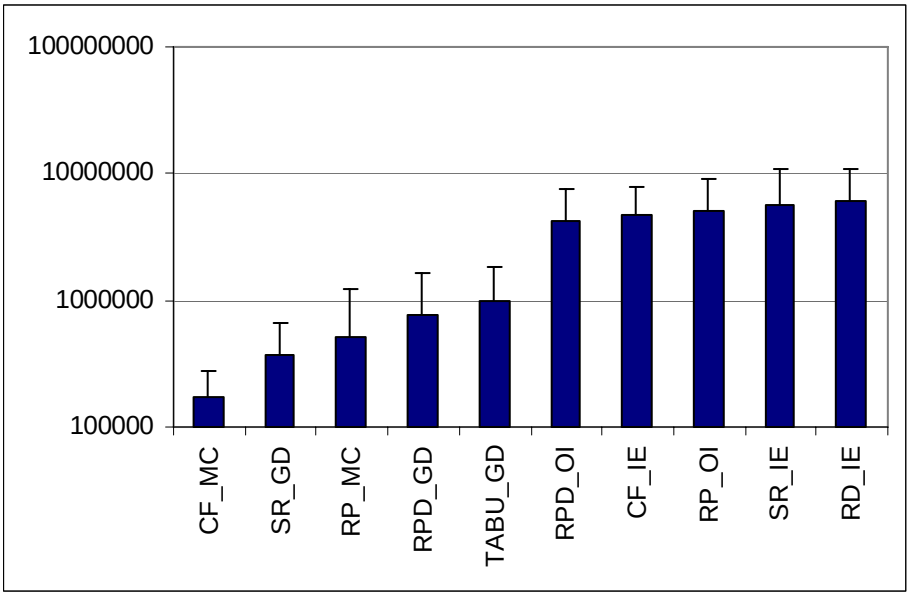

Figure B.8. Average number of fitness evaluations to convergence for Griewangk Function

Table B.9. Average number of fitness evaluations to convergence for Ackley Function

	AM	*OI*	*IE*	*MC*	*GD*
SR	**7.25E+03**	**3.17E+03**	**3.17E+03**	**4.69E+03**	**7.25E+03**
	6.39E+03	**1.60E+03**	**1.60E+03**	**3.59E+03**	**6.39E+03**
RD	**7.37E+03**	**5.45E+03**	**6.10E+03**	**6.55E+03**	**7.37E+03**
	4.86E+03	**3.69E+03**	**4.39E+03**	**4.40E+03**	**4.86E+03**
RP	1.03E+06	**3.43E+03**	**3.01E+03**	**4.26E+03**	3.01E+05
	4.31E+06	**2.74E+03**	**1.28E+03**	**2.25E+03**	8.02E+05
RPD	**7.96E+03**	**5.54E+03**	**6.08E+03**	**7.27E+03**	**7.96E+03**
	6.88E+03	**3.55E+03**	**3.59E+03**	**5.48E+03**	**6.88E+03**
CF	**4.82E+03**	**3.15E+03**	**2.80E+03**	**3.80E+03**	**5.56E+03**
	3.25E+03	**2.09E+03**	**1.20E+03**	**2.72E+03**	**3.86E+03**
TABU	**7.68E+03**	**6.75E+03**	**6.44E+03**	**8.58E+03**	**7.68E+03**
	6.83E+03	**4.10E+03**	**4.79E+03**	**6.47E+03**	**6.83E+03**
GR	**9.18E+03**	*NC*	**9.18E+03**	**9.18E+03**	**9.18E+03**
	3.69E+03	*NC*	**3.69E+03**	**3.54E+03**	**3.69E+03**

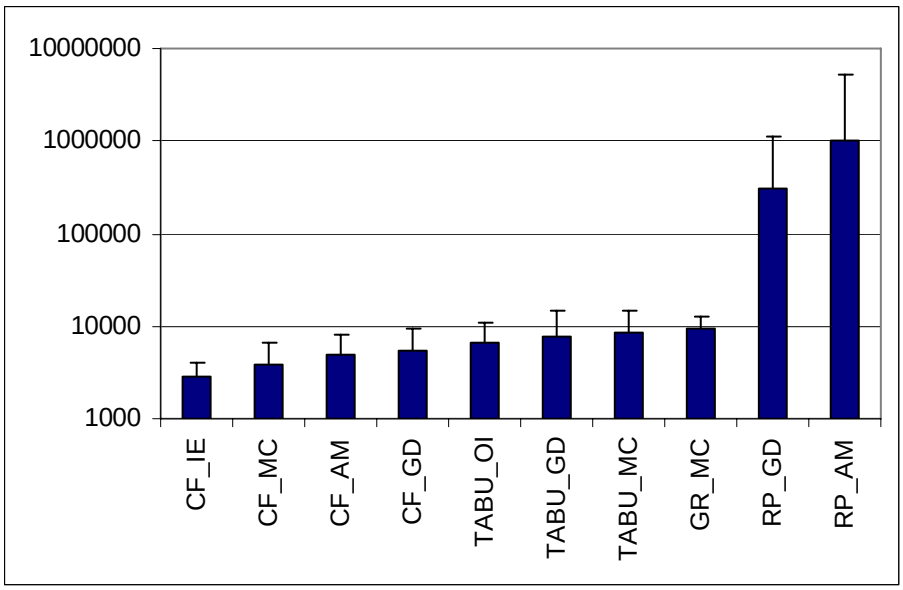

Figure B.9. Average number of fitness evaluations to convergence for Ackley Function

Table B.10. Average number of fitness evaluations to convergence for Easom Function

	AM	*OI*	*IE*	*MC*	*GD*
SR	*NC*	**5.12E+06**	**3.63E+06**	*NC*	*NC*
	NC	**3.55E+06**	**2.41E+06**	*NC*	*NC*
RD	*NC*	**5.12E+06**	**3.51E+06**	*NC*	*NC*
	NC	**3.70E+06**	**2.40E+06**	*NC*	*NC*
RP	*NC*	**5.25E+06**	**4.37E+06**	*NC*	*NC*
	NC	**3.76E+06**	**3.45E+06**	*NC*	*NC*
RPD	*NC*	**5.41E+06**	**3.52E+06**	*NC*	*NC*
	NC	**3.90E+06**	**2.82E+06**	*NC*	*NC*
CF	*NC*	**4.60E+06**	**2.94E+06**	*NC*	*NC*
	NC	**4.14E+06**	**2.51E+06**	*NC*	*NC*
TABU	*NC*	**5.94E+06**	**3.78E+06**	*NC*	*NC*
	NC	**4.02E+06**	**3.10E+06**	*NC*	*NC*
GR	**8.17E+06**	*NC*	**8.17E+06**	**7.27E+06**	**8.17E+06**
	5.43E+06	*NC*	**5.43E+06**	**5.42E+06**	**5.43E+06**

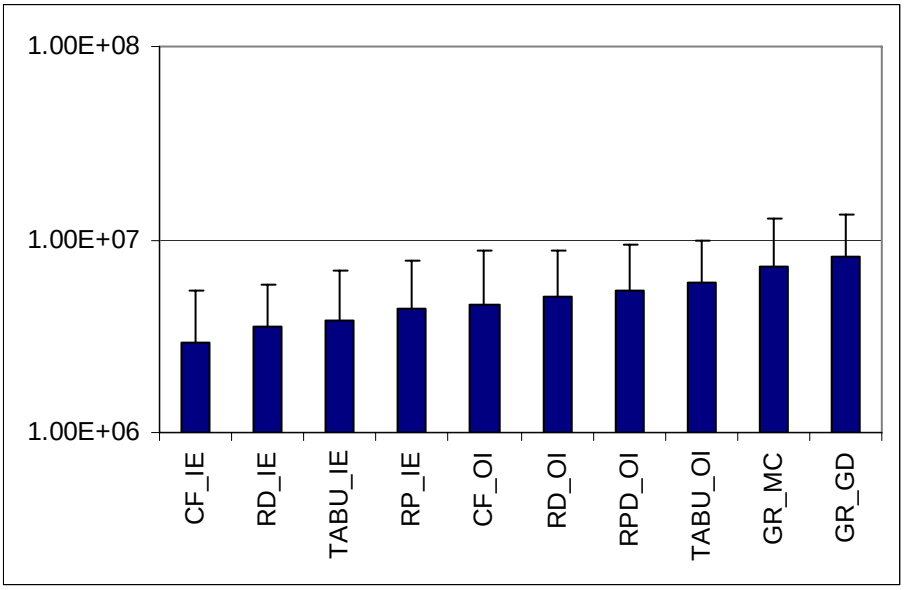

Figure B.10. Average number of fitness evaluations to convergence for Easom Function

Table B.11. Average number of fitness evaluations to convergence for Rotated Hyper-Ellipsoid Function

	AM	*OI*	*IE*	*MC*	*GD*
SR	*NC*	**3.58E+05**	**3.23E+05**	*NC*	*NC*
	NC	**1.21E+05**	**9.26E+04**	*NC*	*NC*
RD	*NC*	**3.10E+05**	**2.97E+05**	*NC*	*NC*
	NC	**1.26E+05**	**1.32E+05**	*NC*	*NC*
RP	*NC*	**3.51E+05**	**3.43E+05**	*NC*	*NC*
	NC	**1.29E+05**	**1.44E+05**	*NC*	*NC*
RPD	*NC*	**2.94E+05**	**2.80E+05**	*NC*	*NC*
	NC	**1.02E+05**	**1.04E+05**	*NC*	*NC*
CF	*NC*	**3.24E+05**	**3.23E+05**	*NC*	*NC*
	NC	**1.16E+05**	**1.55E+05**	*NC*	*NC*
TABU	*NC*	*NC*	*NC*	*NC*	*NC*
	NC	*NC*	*NC*	*NC*	*NC*
GR	5.04E+05	*NC*	5.04E+05	4.77E+05	5.04E+05
	1.07E+05	*NC*	1.07E+05	9.33E+04	1.07E+05

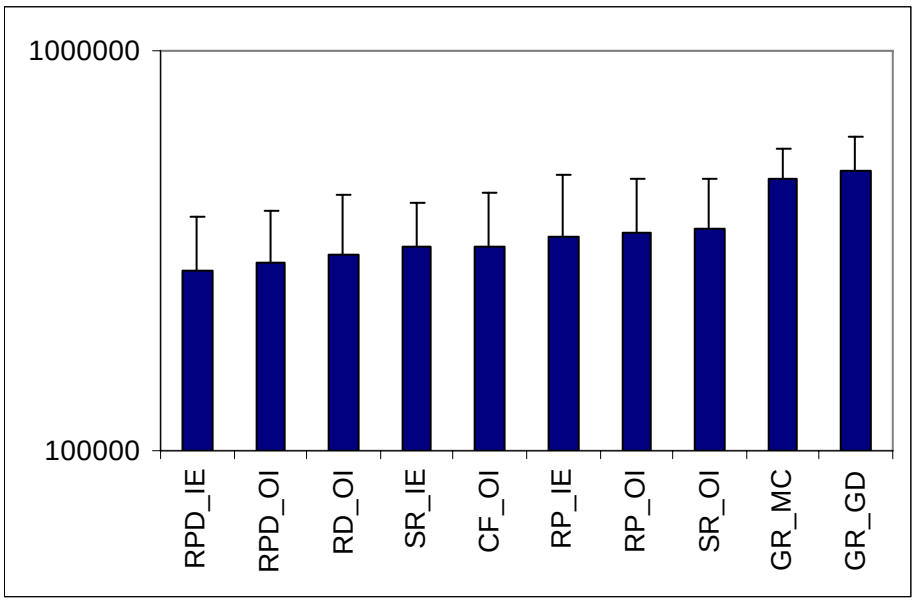

Figure B.11. Average number of fitness evaluations to convergence for Rotated Hyper-Ellipsoid Function

Table B.12. Average number of fitness evaluations to convergence for Royal Road Function

	AM	*OI*	*IE*	*MC*	*GD*
SR	*NC*	**2.21E+05**	**2.06E+05**	3.47E+08	2.08E+08
	NC	**1.06E+05**	**9.28E+04**	1.36E+06	1.50E+07
RD	*NC*	**2.23E+05**	**2.25E+05**	3.44E+08	2.20E+08
	NC	**1.15E+05**	**1.17E+05**	2.22E+06	8.97E+06
RP	*NC*	**2.39E+05**	**2.19E+05**	3.46E+08	2.15E+08
	NC	**1.23E+05**	**1.09E+05**	1.37E+06	1.14E+07
RPD	*NC*	**2.21E+05**	**2.19E+05**	3.43E+08	2.15E+08
	NC	**1.36E+05**	**1.27E+05**	2.01E+06	1.44E+07
CF	*NC*	**2.18E+05**	**2.44E+05**	3.09E+08	2.12E+08
	NC	**1.21E+05**	**1.20E+05**	2.38E+06	1.30E+07
TABU	*NC*	**2.22E+05**	**2.28E+05**	3.49E+08	2.19E+08
	NC	**9.99E+04**	**9.81E+04**	2.38E+06	1.27E+07
GR	4.57E+05	*NC*	4.57E+05	4.71E+05	*NC*
	1.84E+05	*NC*	1.84E+05	2.16E+05	*NC*

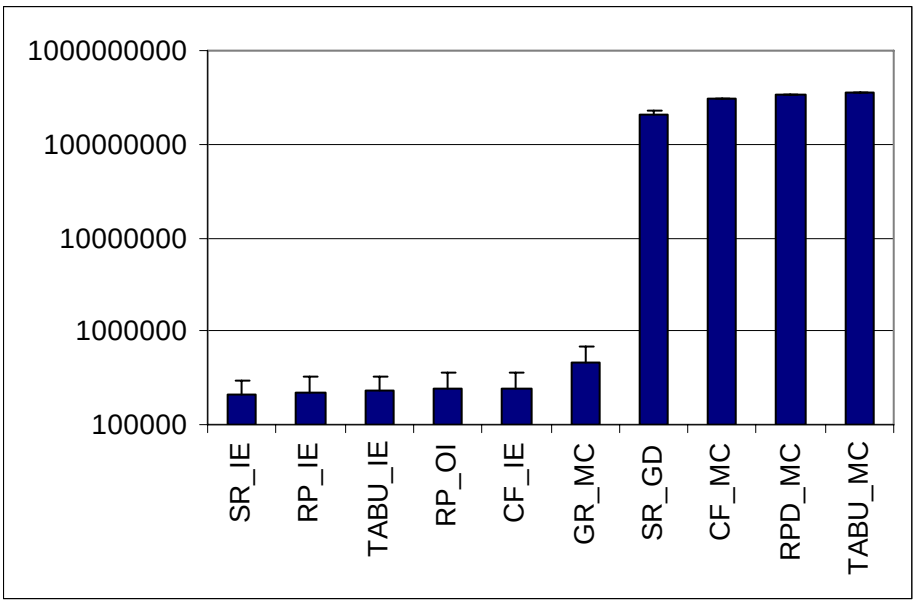

Figure B.12. Average number of fitness evaluations to convergence for Royal Road Function

Table B.13. Average number of fitness evaluations to convergence for Goldberg Function

	AM	*OI*	*IE*	*MC*	*GD*
SR	*NC*	**2.36E+05**	**2.13E+05**	*NC*	1.96E+08
	NC	**7.91E+04**	**7.64E+04**	*NC*	5.82E+05
RD	*NC*	**2.45E+05**	**2.34E+05**	*NC*	1.89E+08
	NC	**9.52E+04**	**8.90E+04**	*NC*	5.32E+05
RP	*NC*	**2.48E+05**	**2.10E+05**	1.99E+08	1.98E+08
	NC	**1.04E+05**	**7.40E+04**	2.71E+05	1.09E+06
RPD	*NC*	**2.13E+05**	**2.25E+05**	*NC*	1.92E+08
	NC	**8.85E+04**	**9.69E+04**	*NC*	7.04E+05
CF	*NC*	**2.36E+05**	**2.08E+05**	1.98E+08	1.95E+08
	NC	**7.22E+04**	**5.82E+04**	1.45E+05	7.08E+05
TABU	*NC*	**2.10E+05**	**2.12E+05**	1.90E+08	1.87E+08
	NC	**8.73E+04**	**7.98E+04**	1.25E+05	4.57E+05
GR	4.34E+05	*NC*	4.34E+05	4.62E+05	4.34E+05
	1.50E+05	*NC*	1.50E+05	1.45E+05	1.50E+05

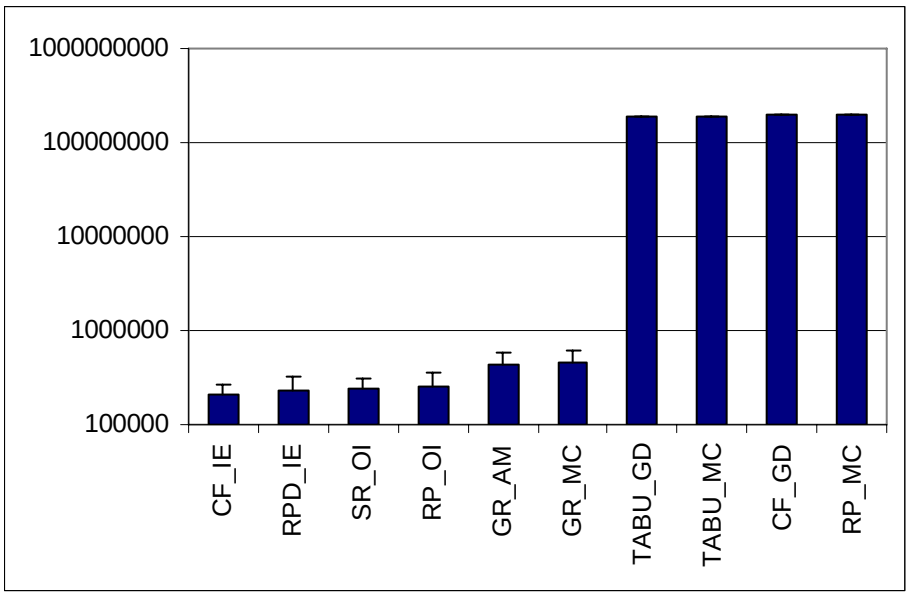

Figure B.13. Average number of fitness evaluations to convergence for Goldberg Function

Table B.14. Average number of fitness evaluations to convergence for Whitley Function

	AM	*OI*	*IE*	*MC*	*GD*
SR	2.12E+07	**1.47E+04**	**1.46E+04**	1.97E+07	2.12E+07
	2.90E+07	**7.02E+03**	**8.02E+03**	2.10E+07	2.90E+07
RD	2.75E+07	**1.61E+04**	**1.45E+04**	2.10E+07	2.75E+07
	2.73E+07	**7.75E+03**	**7.71E+03**	2.05E+07	2.73E+07
RP	2.08E+07	**1.28E+04**	**1.43E+04**	2.35E+07	2.08E+07
	2.31E+07	**5.74E+03**	**7.19E+03**	2.19E+07	2.31E+07
RPD	3.30E+07	**1.36E+04**	**1.50E+04**	2.18E+07	3.30E+07
	3.97E+07	**6.28E+03**	**8.79E+03**	2.36E+07	3.97E+07
CF	2.41E+07	**1.39E+04**	**1.33E+04**	1.97E+07	1.86E+07
	2.49E+07	**8.18E+03**	**8.35E+03**	1.68E+07	1.66E+07
TABU	2.97E+07	**1.42E+04**	**1.30E+04**	2.40E+07	2.97E+07
	2.78E+07	**9.53E+03**	**7.30E+03**	2.25E+07	2.78E+07
GR	2.73E+04	*NC*	2.73E+04	2.72E+04	2.73E+04
	1.34E+04	*NC*	1.34E+04	1.25E+04	1.34E+04

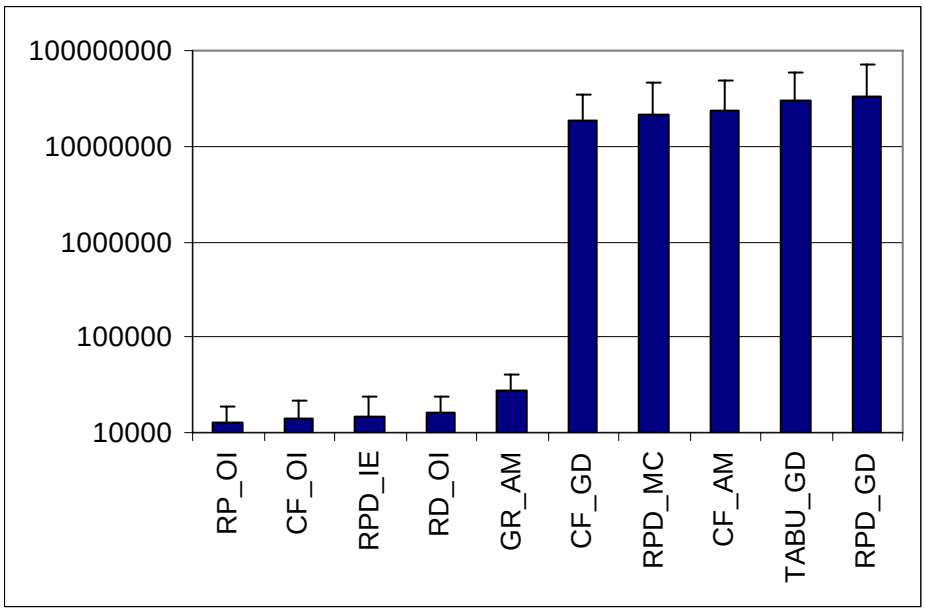

Figure B.14. Average number of fitness evaluations to convergence for Whitley Function

APPENDIX C: EXPERIMENTAL RESULTS TABLES AND GRAPHICS OF HYPERHEURISTICS ON EXAMINATION TIMETABLING

Results of the performance evaluations of the hyperheuristic algorithms composed of various heuristic selection methods and acceptance criteria are given in Table C.1 - Table C.21. Two rows are given for each algorithm. The mean value is given in the first row and the standard deviation in the second row. The performance criterion is the “Average Best Fitness Value”. The results of the experiments are statistically evaluated using t-test with 95 per cent confidence level. The hyperheuristic algorithm which perform significantly better then the rest of the hyperheuristic algorithms are highlighted using boldface characters.

Table C.1. Average best fitness values for Car-f-92

	AM	*OI*	*IE*	*MC*	*GD*
SR	-1.11E-03	-6.03E-03	**-9.91E-03**	-7.08E-03	**-9.42E-03**
	2.56E-05	5.49E-04	**1.09E-03**	4.97E-04	**1.01E-03**
RD	-1.13E-03	-5.77E-03	**-9.87E-03**	**-8.56E-03**	-1.12E-03
	2.30E-05	5.07E-04	**1.12E-03**	**5.64E-04**	1.95E-05
RP	-1.12E-03	-5.68E-03	**-1.01E-02**	**-8.72E-03**	-1.12E-03
	1.89E-05	4.32E-04	**1.02E-03**	**6.68E-04**	1.98E-05
RPD	-1.12E-03	-5.69E-03	**-1.01E-02**	**-8.56E-03**	-1.12E-03
	1.63E-05	4.94E-04	**1.08E-03**	**7.22E-04**	1.70E-05
CF	-1.11E-03	-5.75E-03	**-9.68E-03**	**-1.02E-02**	-5.85E-03
	1.60E-05	4.89E-04	**1.30E-03**	**5.85E-04**	5.59E-04
TABU	-1.13E-03	-5.73E-03	**-1.02E-02**	**-8.49E-03**	-1.13E-03
	2.41E-05	4.68E-04	**1.18E-03**	**6.88E-04**	2.06E-05
GR	-2.31E-03	-5.57E-03	**-9.06E-03**	**-8.82E-03**	-2.32E-03
	5.76E-05	4.82E-04	**9.19E-04**	**5.82E-04**	6.66E-05

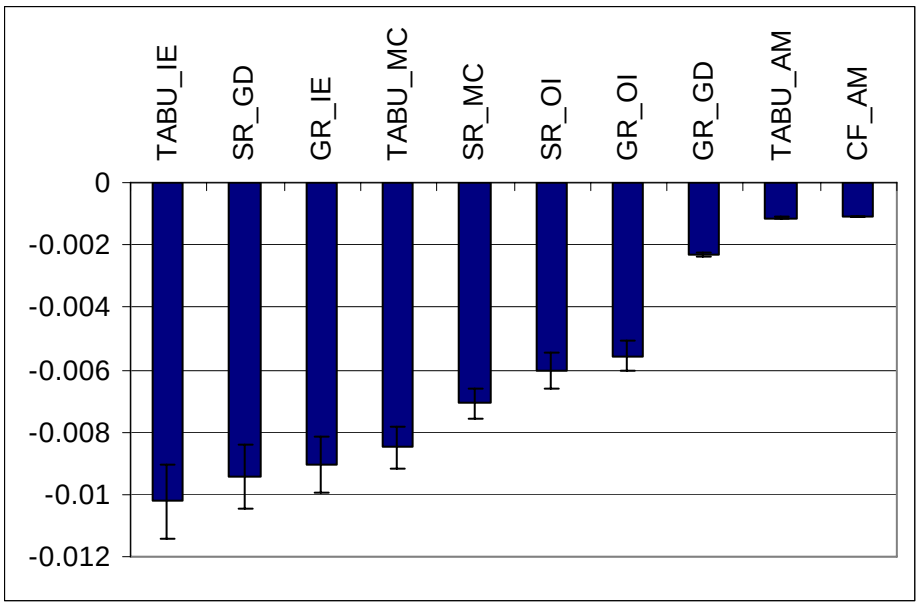

Figure C.1. Average best fitness values for Car-f-92

Table C.2. Average best fitness values for Car-s-91

	AM	*OI*	*IE*	*MC*	*GD*
SR	-1.37E-03	-1.48E-02	-8.71E-02	-1.58E-02	-1.06E-01
	3.18E-05	1.75E-03	2.13E-02	1.53E-03	3.84E-02
RD	-1.33E-03	-1.41E-02	**-1.62E-01**	-2.46E-02	-1.34E-03
	2.57E-05	1.70E-03	**6.12E-02**	2.71E-03	2.11E-05
RP	-1.34E-03	-1.41E-02	**-1.64E-01**	-2.53E-02	-1.33E-03
	2.01E-05	1.47E-03	**6.70E-02**	3.52E-03	2.63E-05
RPD	-1.34E-03	-1.42E-02	**-1.88E-01**	-2.48E-02	-1.34E-03
	1.87E-05	1.37E-03	**1.03E-01**	2.68E-03	1.98E-05
CF	-1.37E-03	-1.40E-02	**-1.52E-01**	-3.90E-02	-1.42E-02
	2.37E-05	1.37E-03	**5.31E-02**	6.46E-03	1.41E-03
TABU	-1.34E-03	-1.40E-02	**-1.93E-01**	-2.48E-02	-1.34E-03
	2.01E-05	1.76E-03	**1.20E-01**	2.82E-03	2.75E-05
GR	-3.01E-03	-1.27E-02	-8.23E-02	-2.17E-02	-3.02E-03
	7.36E-05	1.36E-03	2.36E-02	2.19E-03	7.54E-05

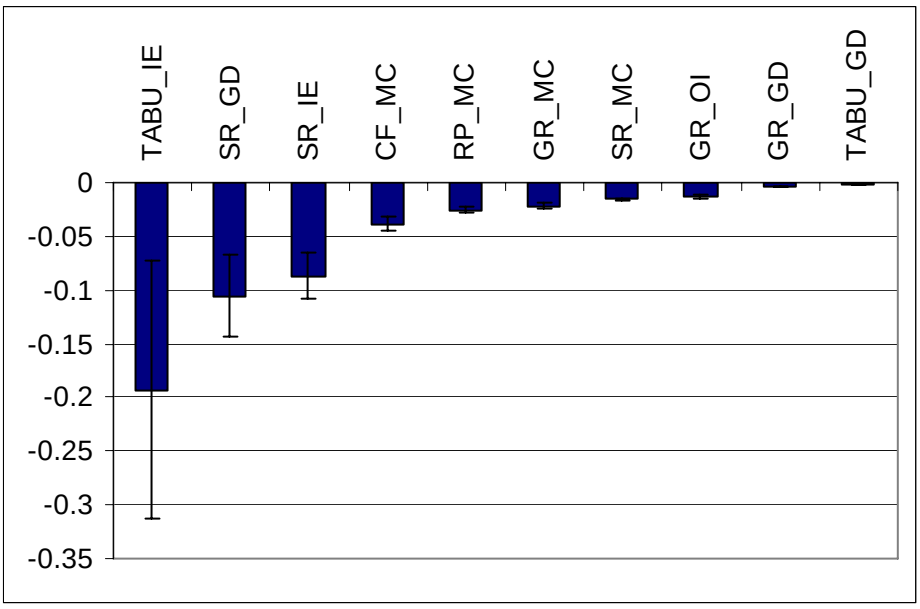

Figure C.2. Average best fitness values for Car-s-91

Table C.3. Average best fitness values for Ear-f-83

	AM	*OI*	*IE*	*MC*	*GD*
SR	-2.03E-03	-4.43E-03	-4.69E-03	-5.71E-03	-5.35E-03
	3.66E-05	3.03E-04	3.88E-04	3.55E-04	4.38E-04
RD	-1.91E-03	-4.11E-03	-4.54E-03	-6.66E-03	-1.90E-03
	4.23E-05	3.31E-04	3.85E-04	4.03E-04	4.69E-05
RP	-1.85E-03	-4.22E-03	-4.56E-03	-6.58E-03	-1.84E-03
	4.97E-05	3.34E-04	4.02E-04	5.14E-04	4.16E-05
RPD	-1.89E-03	-4.15E-03	-4.53E-03	-6.55E-03	-1.88E-03
	4.66E-05	3.37E-04	3.78E-04	3.44E-04	3.90E-05
CF	-1.88E-03	-4.22E-03	-4.54E-03	**-7.27E-03**	-4.10E-03
	4.27E-05	3.32E-04	3.40E-04	**4.94E-04**	3.31E-04
TABU	-1.91E-03	-4.16E-03	-4.52E-03	-6.48E-03	-1.90E-03
	3.84E-05	3.05E-04	3.88E-04	3.63E-04	3.20E-05
GR	-2.84E-03	-4.16E-03	-4.66E-03	-6.22E-03	-2.81E-03
	6.90E-05	3.76E-04	4.04E-04	4.21E-04	6.02E-05

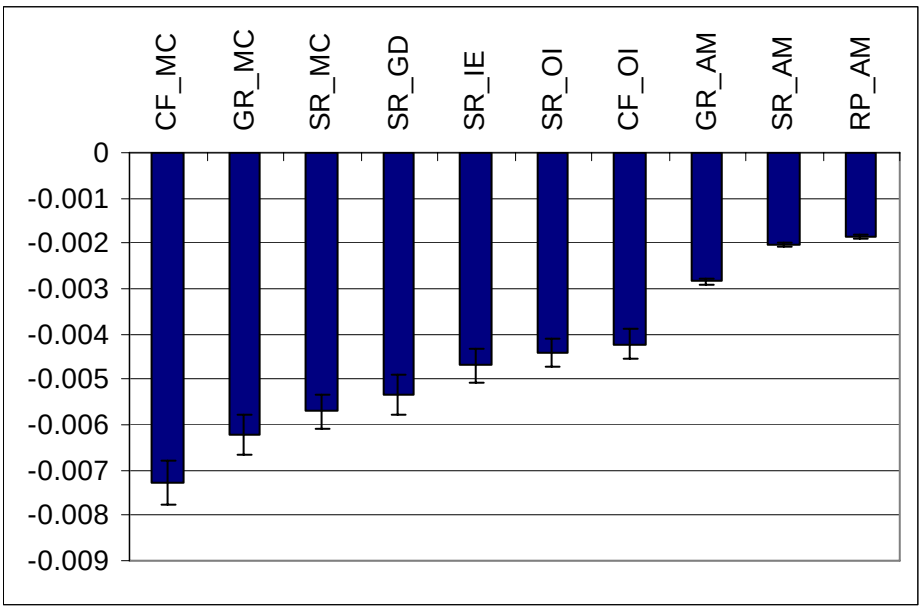

Figure C.3. Average best fitness values for Ear-f-83

Table C.4. Average best fitness values for Hec-s-92

	AM	*OI*	*IE*	*MC*	*GD*
SR	-3.21E-03	-7.84E-03	-7.97E-03	-1.68E-02	-1.67E-02
	1.49E-04	1.56E-03	1.61E-03	2.26E-03	3.99E-03
RD	-2.72E-03	-7.81E-03	-8.42E-03	-1.93E-02	-2.71E-03
	1.22E-04	2.01E-03	1.79E-03	2.44E-03	1.31E-04
RP	-2.70E-03	-7.62E-03	-8.29E-03	-1.90E-02	-2.66E-03
	1.24E-04	1.87E-03	2.02E-03	1.84E-03	1.24E-04
RPD	-2.77E-03	-7.92E-03	-8.32E-03	-1.90E-02	-2.72E-03
	2.10E-04	1.54E-03	2.05E-03	2.10E-03	1.33E-04
CF	-2.74E-03	-7.87E-03	-8.65E-03	**-2.19E-02**	-7.59E-03
	2.29E-04	1.57E-03	2.06E-03	**2.43E-03**	1.70E-03
TABU	-2.74E-03	-8.42E-03	-8.12E-03	-1.86E-02	-2.70E-03
	1.70E-04	2.46E-03	1.43E-03	2.03E-03	1.29E-04
GR	-5.86E-03	-8.25E-03	-8.42E-03	**-2.13E-02**	-5.86E-03
	3.99E-04	1.69E-03	1.81E-03	**3.54E-03**	3.90E-04

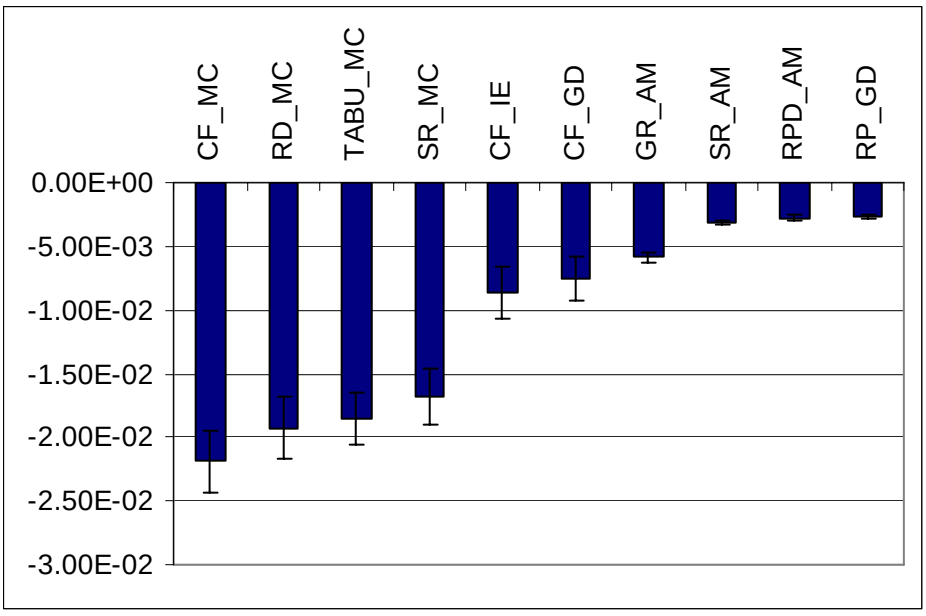

Figure C.4. Average best fitness values for Hec-s-92

Table C.5. Average best fitness values for Kfu-s-93

	AM	*OI*	*IE*	*MC*	*GD*
SR	-1.69E-03	-1.31E-02	-2.47E-02	-2.20E-02	**-3.40E-02**
	5.87E-05	2.23E-03	4.88E-03	2.57E-03	**4.30E-03**
RD	-1.44E-03	-1.20E-02	-2.47E-02	-2.50E-02	-1.43E-03
	3.64E-05	2.38E-03	4.08E-03	3.34E-03	3.34E-05
RP	-1.43E-03	-1.29E-02	-2.50E-02	-2.55E-02	-1.42E-03
	3.67E-05	2.48E-03	4.60E-03	3.24E-03	3.26E-05
RPD	-1.44E-03	-1.21E-02	-2.47E-02	-2.52E-02	-1.43E-03
	3.79E-05	2.14E-03	4.21E-03	3.61E-03	3.38E-05
CF	-1.42E-03	-1.29E-02	-2.50E-02	-2.79E-02	-1.21E-02
	4.13E-05	2.86E-03	3.45E-03	4.02E-03	1.99E-03
TABU	-1.43E-03	-1.26E-02	-2.43E-02	-2.55E-02	-1.42E-03
	3.29E-05	2.21E-03	4.74E-03	3.64E-03	3.39E-05
GR	-4.93E-03	-9.13E-03	-2.41E-02	-3.06E-02	-5.03E-03
	2.67E-04	1.93E-03	4.57E-03	3.02E-03	2.62E-04

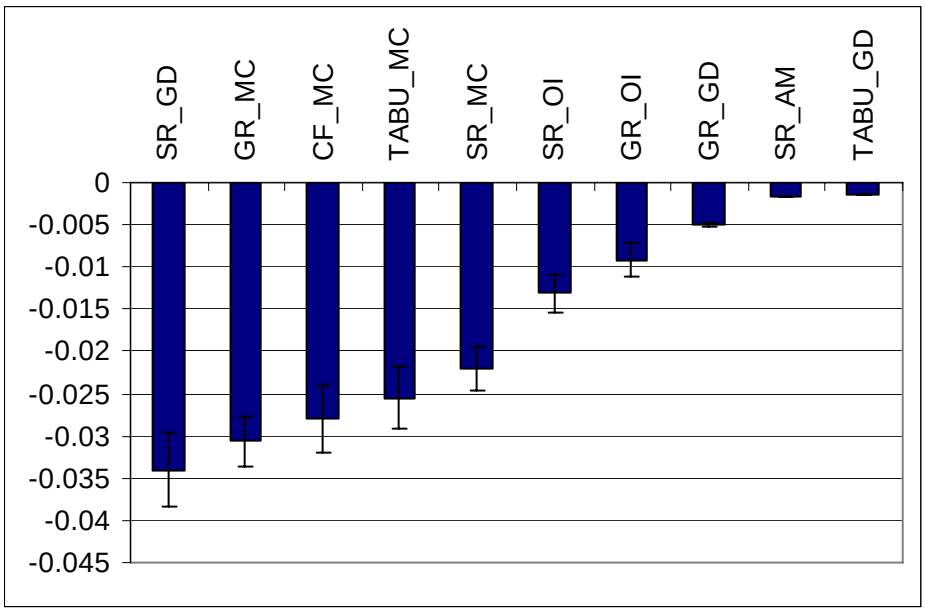

Figure C.5. Average best fitness values for Kfu-s-93

Table C.6. Average best fitness values for Lse-f-91

	AM	*OI*	*IE*	*MC*	*GD*
SR	-2.65E-03	-7.20E-03	-1.06E-02	-1.09E-02	-1.19E-02
	5.75E-05	8.84E-04	1.69E-03	1.04E-03	1.42E-03
RD	-2.50E-03	-6.82E-03	-9.94E-03	-1.24E-02	-2.50E-03
	5.10E-05	8.25E-04	1.53E-03	1.12E-03	5.91E-05
RP	-2.39E-03	-6.57E-03	-1.04E-02	-1.23E-02	-2.40E-03
	4.66E-05	9.07E-04	1.74E-03	1.44E-03	6.74E-05
RPD	-2.51E-03	-6.81E-03	-1.02E-02	-1.20E-02	-2.49E-03
	6.51E-05	6.25E-04	1.30E-03	1.16E-03	6.34E-05
CF	-2.47E-03	-6.64E-03	-1.00E-02	**-1.42E-02**	-6.75E-03
	5.66E-05	6.90E-04	1.15E-03	**1.38E-03**	8.38E-04
TABU	-2.51E-03	-6.64E-03	-1.02E-02	-1.22E-02	-2.50E-03
	6.08E-05	8.32E-04	1.33E-03	1.21E-03	5.44E-05
GR	-4.41E-03	-6.64E-03	-1.02E-02	-1.25E-02	-4.47E-03
	1.74E-04	7.76E-04	1.45E-03	1.41E-03	1.34E-04

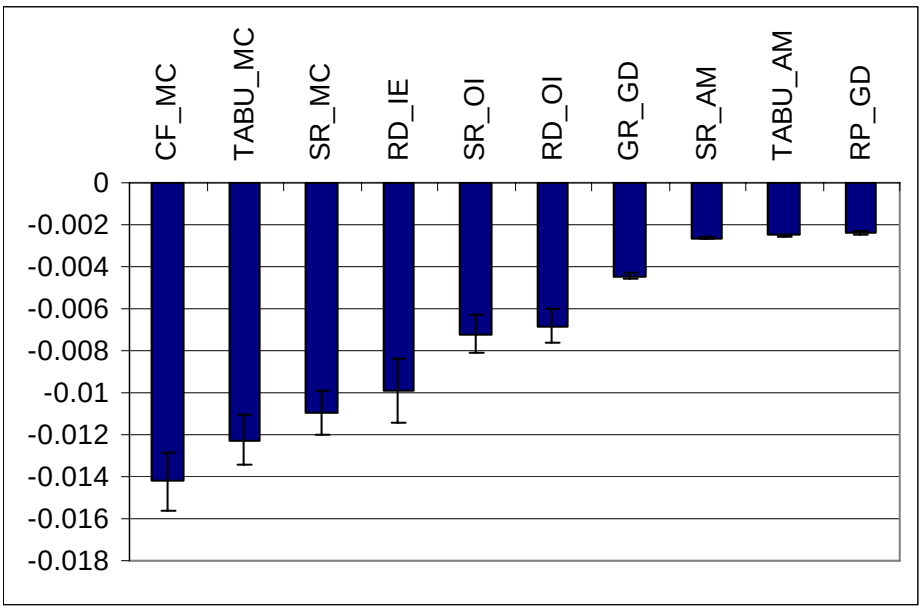

Figure C.6. Average best fitness values for Lse-f-91

Table C.7. Average best fitness values for Pur-s-93

	AM	*OI*	*IE*	*MC*	*GD*
SR	-3.11E-04	-1.08E-03	**-1.41E-03**	-6.37E-04	-1.06E-03
	4.52E-06	3.97E-05	**6.98E-05**	1.64E-05	4.15E-05
RD	-2.86E-04	-9.89E-04	-1.37E-03	-7.77E-04	-2.88E-04
	4.00E-06	4.56E-05	6.18E-05	2.12E-05	4.25E-06
RP	-2.81E-04	-1.00E-03	-1.38E-03	-7.83E-04	-2.81E-04
	3.75E-06	3.92E-05	6.60E-05	2.32E-05	4.11E-06
RPD	-2.87E-04	-9.97E-04	-1.38E-03	-7.74E-04	-2.86E-04
	4.11E-06	4.28E-05	6.24E-05	1.84E-05	4.89E-06
CF	-2.97E-04	-1.00E-03	-1.36E-03	-9.39E-04	-1.00E-03
	5.98E-06	5.97E-05	6.14E-05	2.92E-05	4.75E-05
TABU	-2.92E-04	-1.01E-03	-1.37E-03	-7.70E-04	-2.92E-04
	5.53E-06	4.22E-05	6.77E-05	2.36E-05	4.46E-06
GR	-5.90E-04	-9.44E-04	-1.29E-03	-8.57E-04	-5.89E-04
	8.83E-06	3.66E-05	6.60E-05	2.68E-05	8.72E-06

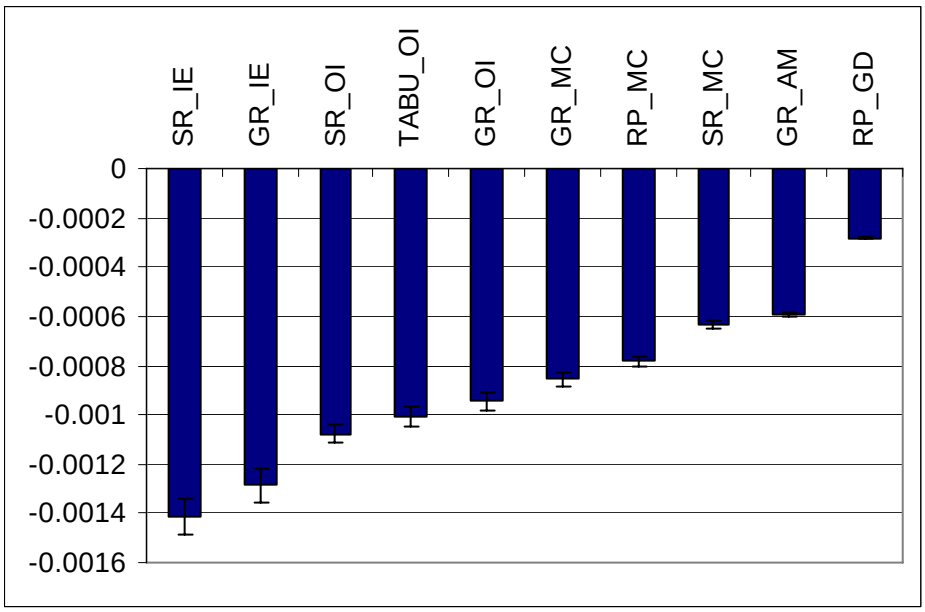

Figure C.7. Average best fitness Values for Pur-s-93

Table C.8. Average best fitness values for Rye-s-93

	AM	*OI*	*IE*	*MC*	*GD*
SR	-7.86E-04	-5.47E-03	-9.35E-03	-7.28E-03	-9.39E-03
	3.06E-05	8.58E-04	1.89E-03	8.88E-04	1.30E-03
RD	-7.37E-04	-5.33E-03	-9.08E-03	-8.70E-03	-7.37E-04
	1.65E-05	8.87E-04	2.00E-03	1.16E-03	2.33E-05
RP	-7.20E-04	-5.16E-03	-9.19E-03	-8.84E-03	-7.20E-04
	2.13E-05	8.85E-04	1.48E-03	1.34E-03	1.92E-05
RPD	-7.32E-04	-4.82E-03	-9.46E-03	-8.69E-03	-7.33E-04
	1.87E-05	6.52E-04	1.94E-03	8.83E-04	1.77E-05
CF	-7.26E-04	-5.37E-03	-9.45E-03	**-1.08E-02**	-5.28E-03
	1.50E-05	1.04E-03	1.79E-03	**1.37E-03**	7.78E-04
TABU	-7.41E-04	-4.88E-03	-8.88E-03	-9.39E-03	-7.43E-04
	2.27E-05	8.17E-04	1.71E-03	1.44E-03	2.26E-05
GR	-1.99E-03	-4.76E-03	-8.33E-03	-9.51E-03	-2.00E-03
	7.73E-05	5.82E-04	1.66E-03	1.07E-03	8.68E-05

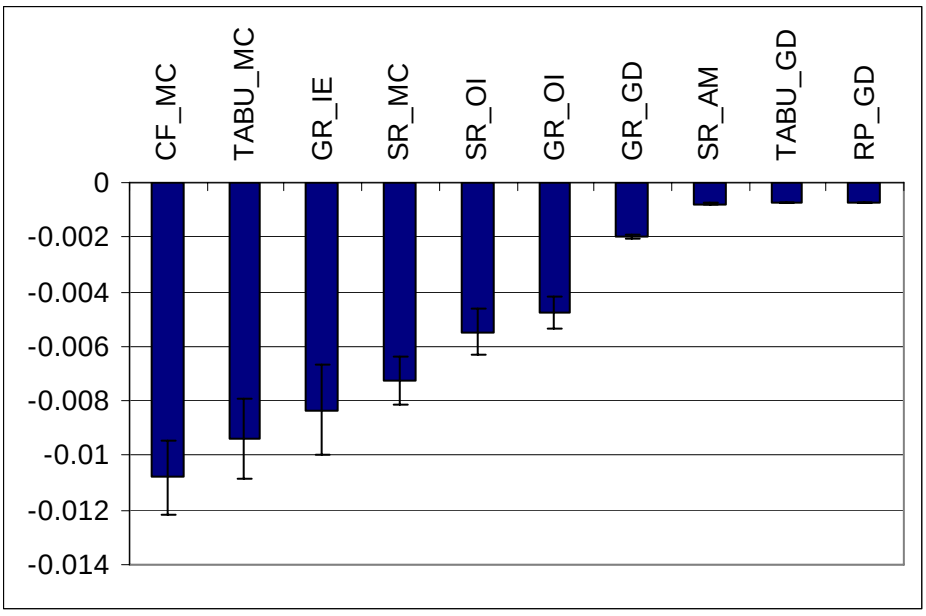

Figure C.8. Average best fitness values for Rye-s-93

Table C.9. Average best fitness values for Sta-f-83

	AM	*OI*	*IE*	*MC*	*GD*
SR	-1.25E-03	-2.60E-03	-2.64E-03	**-2.68E-03**	**-2.68E-03**
	2.17E-05	7.58E-05	4.49E-05	**1.04E-05**	**1.26E-05**
RD	-1.26E-03	-2.61E-03	-2.64E-03	**-2.67E-03**	-1.26E-03
	2.11E-05	7.14E-05	5.11E-05	**1.18E-05**	1.76E-05
RP	-1.25E-03	-2.62E-03	-2.63E-03	**-2.68E-03**	-1.25E-03
	2.39E-05	6.57E-05	5.34E-05	**1.46E-05**	2.32E-05
RPD	-1.25E-03	-2.62E-03	-2.63E-03	**-2.68E-03**	-1.25E-03
	2.09E-05	6.75E-05	5.75E-05	**1.14E-05**	1.99E-05
CF	-1.29E-03	-2.61E-03	-2.62E-03	**-2.68E-03**	-2.63E-03
	2.21E-05	6.50E-05	6.53E-05	**9.08E-06**	5.43E-05
TABU	-1.27E-03	-2.64E-03	-2.63E-03	**-2.68E-03**	-1.26E-03
	2.21E-05	4.57E-05	6.28E-05	**1.52E-05**	2.17E-05
GR	-1.93E-03	-2.59E-03	-2.63E-03	**-2.68E-03**	-1.92E-03
	4.23E-05	8.21E-05	5.28E-05	**1.19E-05**	4.24E-05

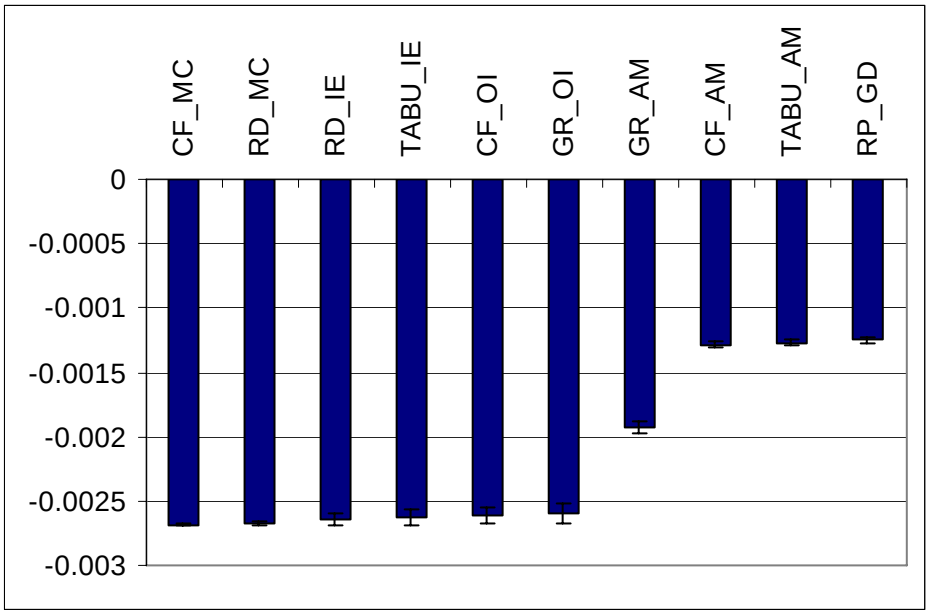

Figure C.9. Average best fitness values for Sta-f-83

Table C.10. Average best fitness values for Tre-s-92

	AM	*OI*	*IE*	*MC*	*GD*
SR	-4.15E-03	-1.91E-02	-3.95E-02	-3.43E-02	**-6.79E-02**
	8.23E-05	2.04E-03	7.55E-03	4.02E-03	**1.08E-02**
RD	-3.78E-03	-1.83E-02	-4.16E-02	-4.00E-02	-3.78E-03
	8.82E-05	2.12E-03	8.70E-03	5.19E-03	7.48E-05
RP	-3.76E-03	-1.90E-02	-4.22E-02	-3.99E-02	-3.73E-03
	1.06E-04	1.83E-03	7.18E-03	4.20E-03	9.18E-05
RPD	-3.77E-03	-1.88E-02	-4.15E-02	-3.90E-02	-3.81E-03
	9.34E-05	1.67E-03	5.46E-03	4.20E-03	1.15E-04
CF	-3.73E-03	-1.88E-02	-4.03E-02	-4.53E-02	-1.82E-02
	8.46E-05	2.01E-03	7.57E-03	5.90E-03	1.99E-03
TABU	-3.81E-03	-1.89E-02	-4.25E-02	-3.96E-02	-3.79E-03
	8.28E-05	2.35E-03	7.83E-03	4.29E-03	8.31E-05
GR	-7.45E-03	-1.82E-02	-3.83E-02	-4.28E-02	-7.44E-03
	2.51E-04	2.00E-03	6.62E-03	4.52E-03	2.15E-04

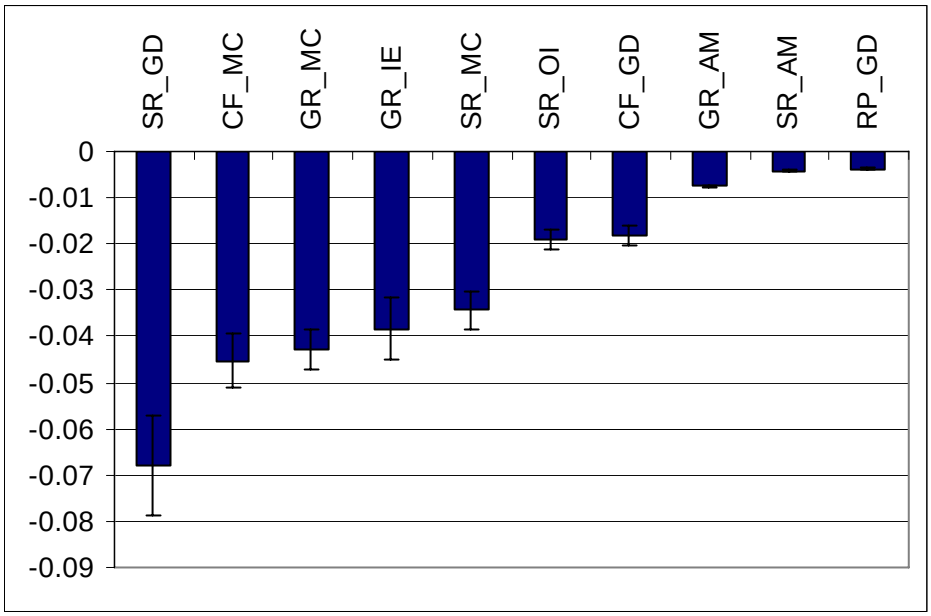

Figure C.10. Average best fitness values for Tre-s-92

Table C.11. Average best fitness values for Uta-s-92

	AM	*OI*	*IE*	*MC*	*GD*
SR	-1.34E-03	-8.66E-03	**-1.81E-02**	-8.35E-03	-1.25E-02
	2.90E-05	5.37E-04	**1.65E-03**	4.96E-04	9.83E-04
RD	-1.20E-03	-8.24E-03	**-1.81E-02**	-1.15E-02	-1.20E-03
	2.37E-05	4.21E-04	**1.56E-03**	7.01E-04	1.44E-05
RP	-1.19E-03	-8.37E-03	**-1.81E-02**	-1.13E-02	-1.19E-03
	1.53E-05	6.01E-04	**1.74E-03**	7.32E-04	1.92E-05
RPD	-1.19E-03	-8.32E-03	**-1.84E-02**	-1.13E-02	-1.19E-03
	1.89E-05	5.60E-04	**1.57E-03**	7.09E-04	1.35E-05
CF	-1.23E-03	-8.34E-03	**-1.78E-02**	-1.41E-02	-8.27E-03
	2.91E-05	5.76E-04	**1.57E-03**	8.51E-04	5.39E-04
TABU	-1.21E-03	-8.21E-03	**-1.87E-02**	-1.12E-02	-1.21E-03
	1.92E-05	4.86E-04	**1.79E-03**	5.68E-04	2.32E-05
GR	-2.47E-03	-7.55E-03	-1.66E-02	-9.85E-03	-2.46E-03
	7.78E-05	5.13E-04	1.49E-03	5.47E-04	5.18E-05

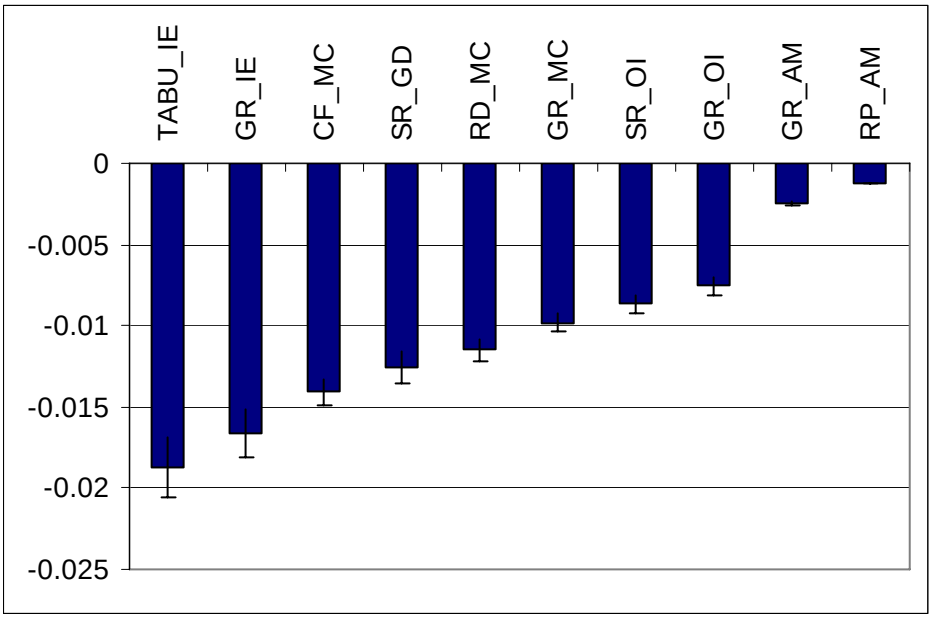

Figure C.11. Average best fitness values for Uta-s-92

Table C.12. Average best fitness values for Ute-s-91

	AM	*OI*	*IE*	*MC*	*GD*
SR	-1.00E-03	-1.52E-03	-1.58E-03	-1.95E-03	-1.69E-03
	1.91E-05	9.45E-05	1.12E-04	1.09E-04	1.31E-04
RD	-9.76E-04	-1.48E-03	-1.54E-03	-2.15E-03	-9.72E-04
	1.79E-05	9.40E-05	9.78E-05	9.83E-05	1.72E-05
RP	-9.68E-04	-1.49E-03	-1.55E-03	-2.15E-03	-9.68E-04
	1.66E-05	1.00E-04	1.02E-04	8.61E-05	2.14E-05
RPD	-9.78E-04	-1.48E-03	-1.54E-03	-2.16E-03	-9.73E-04
	2.05E-05	9.79E-05	7.41E-05	9.49E-05	1.72E-05
CF	-9.66E-04	-1.47E-03	-1.57E-03	**-2.27E-03**	-1.50E-03
	1.89E-05	1.13E-04	1.03E-04	**8.64E-05**	8.21E-05
TABU	-9.73E-04	-1.47E-03	-1.55E-03	-2.17E-03	-9.77E-04
	1.49E-05	9.27E-05	8.85E-05	9.50E-05	1.84E-05
GR	-1.43E-03	-1.48E-03	-1.57E-03	-2.13E-03	-1.43E-03
	3.24E-05	1.06E-04	8.66E-05	8.25E-05	3.49E-05

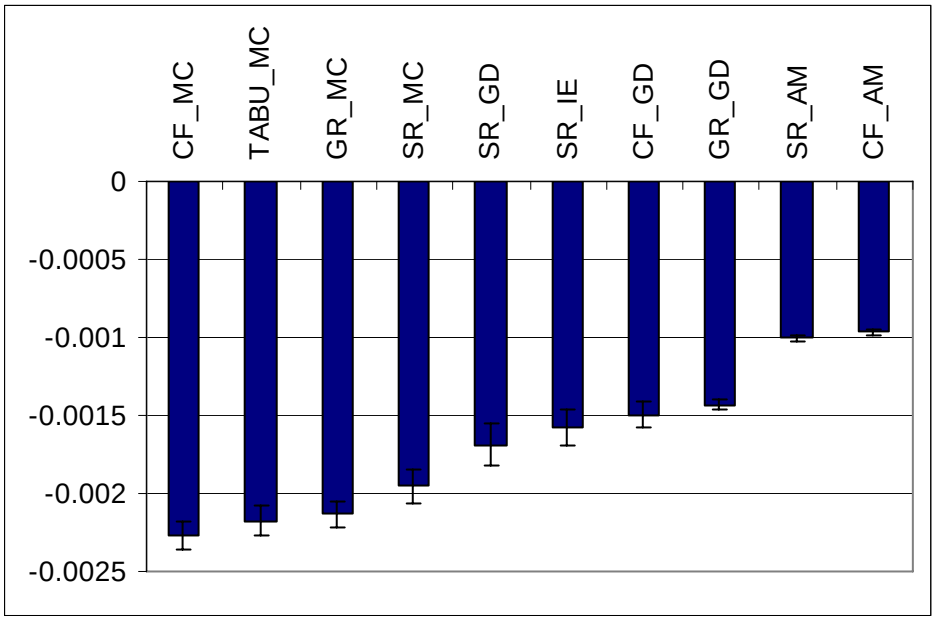

Figure C.12. Average best fitness values for Ute-s-91

Table C.13. Average best fitness values for Yor-f-83

	AM	*OI*	*IE*	*MC*	*GD*
SR	-2.55E-03	-4.92E-03	-5.26E-03	-6.50E-03	-6.24E-03
	4.55E-05	2.94E-04	3.38E-04	4.05E-04	4.71E-04
RD	-2.40E-03	-4.84E-03	-5.24E-03	-7.47E-03	-2.38E-03
	3.54E-05	3.07E-04	4.04E-04	4.04E-04	3.20E-05
RP	-2.34E-03	-4.77E-03	-5.27E-03	-7.48E-03	-2.33E-03
	3.98E-05	3.87E-04	3.47E-04	5.42E-04	3.83E-05
RPD	-2.39E-03	-4.78E-03	-5.28E-03	-7.60E-03	-2.38E-03
	4.43E-05	3.98E-04	3.87E-04	5.03E-04	3.73E-05
CF	-2.36E-03	-4.78E-03	-5.28E-03	**-8.32E-03**	-4.76E-03
	2.81E-05	2.42E-04	3.55E-04	**4.57E-04**	3.61E-04
TABU	-2.41E-03	-4.83E-03	-5.32E-03	-7.45E-03	-2.41E-03
	4.44E-05	3.45E-04	3.99E-04	4.02E-04	4.47E-05
GR	-3.13E-03	-4.73E-03	-5.22E-03	-6.80E-03	-3.13E-03
	6.92E-05	2.84E-04	3.73E-04	4.54E-04	6.84E-05

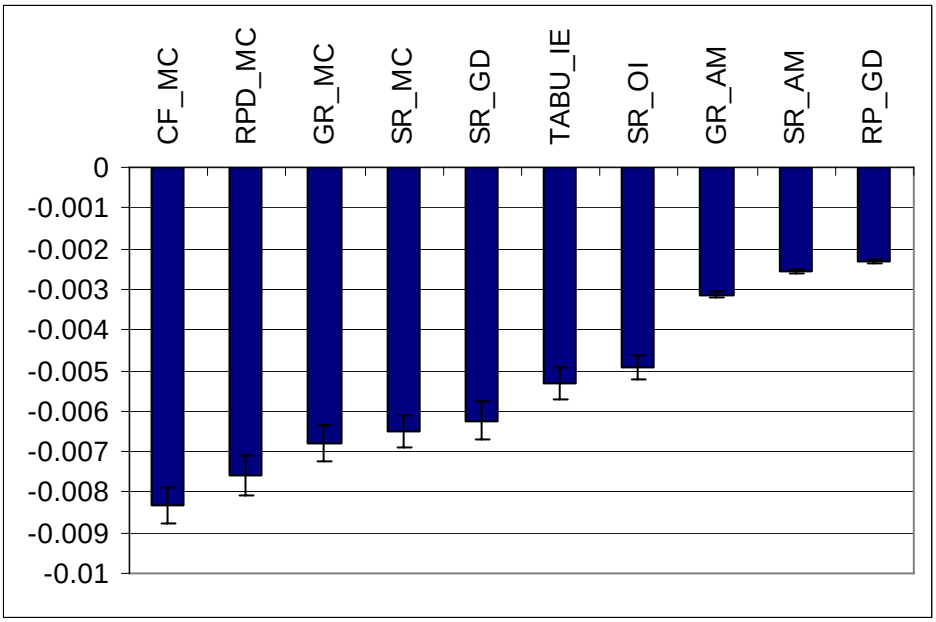

Figure C.13. Average best fitness values for Yor-f-83

Table C.14. Average best fitness values for Yue20011

	AM	*OI*	*IE*	*MC*	*GD*
SR	-9.76E-03	-3.09E-02	-5.89E-02	-6.55E-02	**-9.02E-02**
	2.50E-04	4.18E-03	1.02E-02	8.03E-03	**1.07E-02**
RD	-8.65E-03	-3.01E-02	-5.93E-02	-7.06E-02	-8.60E-03
	2.70E-04	3.48E-03	1.01E-02	8.13E-03	2.47E-04
RP	-8.61E-03	-2.94E-02	-5.66E-02	-7.05E-02	-8.59E-03
	2.26E-04	4.42E-03	1.06E-02	9.73E-03	3.12E-04
RPD	-8.65E-03	-2.94E-02	-5.93E-02	-7.09E-02	-8.69E-03
	3.11E-04	4.47E-03	1.22E-02	9.08E-03	3.08E-04
CF	-8.49E-03	-3.02E-02	-5.85E-02	-7.49E-02	-3.03E-02
	2.74E-04	5.10E-03	1.20E-02	9.06E-03	4.59E-03
TABU	-8.69E-03	-2.94E-02	-5.76E-02	-6.81E-02	-8.68E-03
	3.30E-04	3.99E-03	1.22E-02	9.57E-03	3.47E-04
GR	-1.92E-02	-2.81E-02	-5.98E-02	-8.55E-02	-1.92E-02
	1.10E-03	4.54E-03	1.32E-02	9.19E-03	1.29E-03

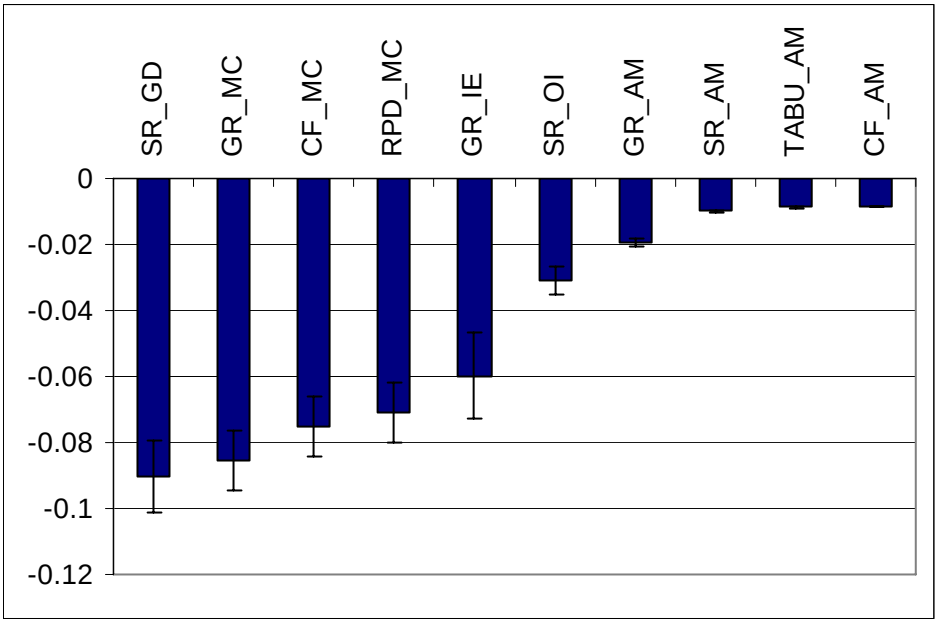

Figure C.14. Average best fitness values for Yue20011

Table C.15. Average best fitness values for Yue20012

	AM	*OI*	*IE*	*MC*	*GD*
SR	-6.61E-03	-2.46E-02	-4.82E-02	-5.05E-02	**-7.54E-02**
	1.83E-04	3.76E-03	7.40E-03	6.04E-03	**9.38E-03**
RD	-6.15E-03	-2.29E-02	-4.84E-02	-5.42E-02	-6.06E-03
	1.87E-04	2.98E-03	8.35E-03	7.56E-03	1.56E-04
RP	-6.10E-03	-2.32E-02	-4.81E-02	-5.43E-02	-6.06E-03
	1.81E-04	3.84E-03	7.23E-03	6.25E-03	1.32E-04
RPD	-6.12E-03	-2.33E-02	-4.73E-02	-5.52E-02	-6.11E-03
	1.81E-04	3.09E-03	8.82E-03	5.74E-03	1.87E-04
CF	-6.08E-03	-2.34E-02	-4.80E-02	-5.80E-02	-2.35E-02
	1.79E-04	2.94E-03	7.51E-03	6.59E-03	3.40E-03
TABU	-6.10E-03	-2.28E-02	-4.79E-02	-5.55E-02	-6.05E-03
	1.83E-04	3.33E-03	6.38E-03	6.19E-03	1.32E-04
GR	-1.31E-02	-2.27E-02	-4.89E-02	-6.32E-02	-1.31E-02
	5.40E-04	3.66E-03	7.07E-03	5.90E-03	5.50E-04

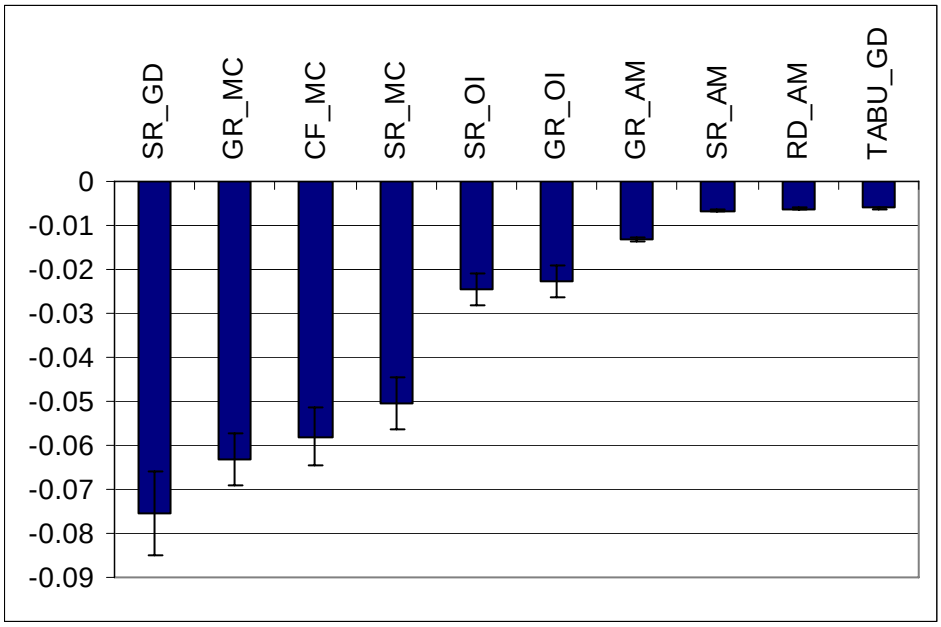

Figure C.15. Average best fitness values for Yue20012

Table C.16. Average best fitness values for Yue20013

	AM	*OI*	*IE*	*MC*	*GD*
SR	-1.40E-01	-9.64E-02	-1.66E-01	**-2.50E-01**	-2.37E-01
	9.03E-03	2.69E-02	3.91E-02	**0.00E+00**	1.87E-02
RD	-1.40E-01	-9.53E-02	-1.67E-01	**-2.50E-01**	-1.38E-01
	8.67E-03	2.68E-02	3.70E-02	**0.00E+00**	8.80E-03
RP	-1.43E-01	-9.82E-02	-1.68E-01	**-2.50E-01**	-1.41E-01
	1.07E-02	2.64E-02	3.06E-02	**0.00E+00**	9.82E-03
RPD	-1.38E-01	-1.03E-01	-1.61E-01	**-2.50E-01**	-1.40E-01
	7.26E-03	2.48E-02	3.54E-02	**0.00E+00**	1.22E-02
CF	-1.37E-01	-1.00E-01	-1.59E-01	**-2.50E-01**	-9.82E-02
	9.78E-03	2.81E-02	3.37E-02	**0.00E+00**	2.22E-02
TABU	-1.38E-01	-9.70E-02	-1.60E-01	**-2.50E-01**	-1.40E-01
	9.10E-03	2.70E-02	3.20E-02	**0.00E+00**	8.81E-03
GR	-1.96E-01	-9.02E-02	-1.70E-01	**-2.50E-01**	-1.99E-01
	1.26E-02	1.90E-02	3.82E-02	**0.00E+00**	1.09E-02

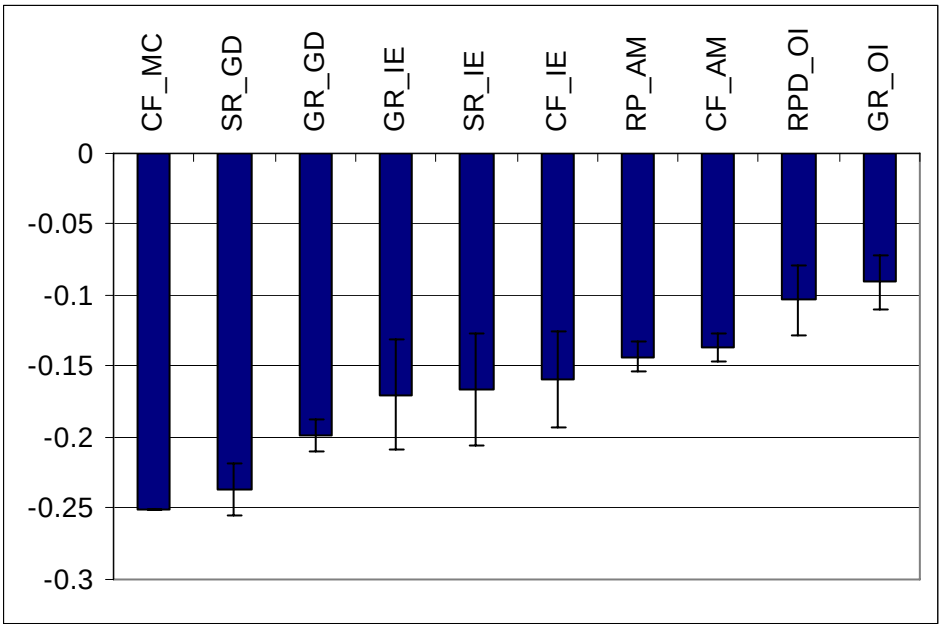

Figure C.16. Average best fitness values for Yue20013

Table C.17. Average best fitness values for Yue20021

	AM	*OI*	*IE*	*MC*	*GD*
SR	-3.73E-03	-1.38E-02	-2.04E-02	-2.65E-02	**-3.45E-02**
	1.38E-04	2.80E-03	4.30E-03	2.64E-03	**4.55E-03**
RD	-3.36E-03	-1.27E-02	-1.83E-02	-2.93E-02	-3.35E-03
	1.25E-04	2.26E-03	4.11E-03	3.18E-03	1.17E-04
RP	-3.25E-03	-1.26E-02	-1.74E-02	-2.89E-02	-3.23E-03
	1.15E-04	3.39E-03	3.65E-03	3.32E-03	1.41E-04
RPD	-3.35E-03	-1.31E-02	-1.89E-02	-2.87E-02	-3.38E-03
	1.42E-04	2.02E-03	4.19E-03	3.56E-03	1.24E-04
CF	-3.31E-03	-1.24E-02	-1.93E-02	-3.20E-02	-1.23E-02
	1.21E-04	2.78E-03	4.08E-03	3.76E-03	2.51E-03
TABU	-3.35E-03	-1.31E-02	-1.83E-02	-2.95E-02	-3.38E-03
	1.03E-04	2.25E-03	3.78E-03	3.62E-03	1.38E-04
GR	-7.69E-03	-1.24E-02	-1.87E-02	-3.11E-02	-7.82E-03
	4.89E-04	2.74E-03	5.11E-03	5.20E-03	4.97E-04

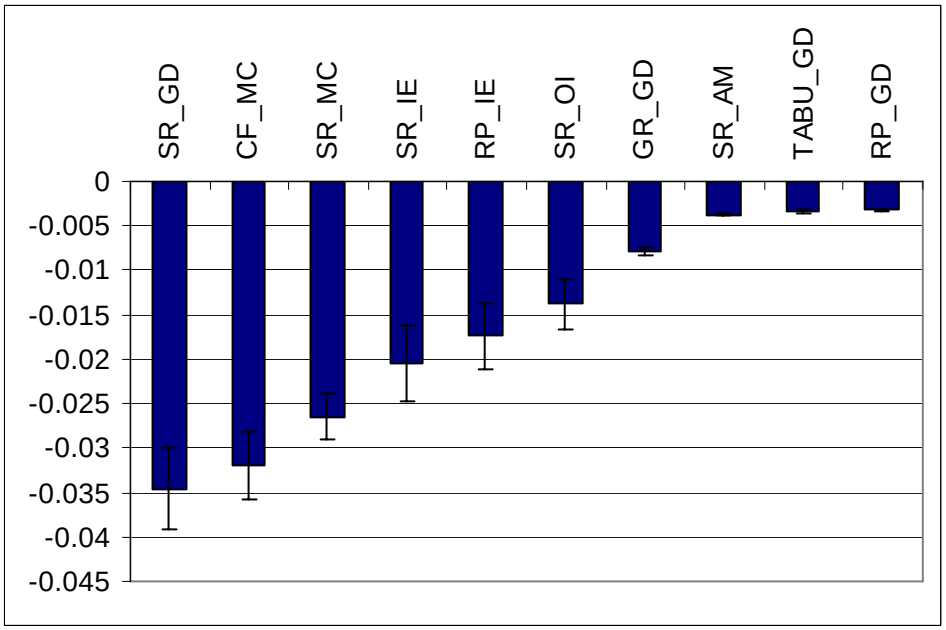

Figure C.17. Average best fitness values for Yue20021

Table C.18. Average best fitness values for Yue20022

	AM	*OI*	*IE*	*MC*	*GD*
SR	-2.68E-03	-7.37E-03	-8.48E-03	-1.09E-02	-1.09E-02
	7.14E-05	6.37E-04	1.05E-03	8.94E-04	9.83E-04
RD	-2.51E-03	-7.05E-03	-8.31E-03	-1.16E-02	-2.50E-03
	8.98E-05	6.69E-04	9.36E-04	8.07E-04	8.03E-05
RP	-2.38E-03	-6.98E-03	-8.26E-03	-1.17E-02	-2.39E-03
	6.61E-05	9.46E-04	8.74E-04	8.95E-04	8.32E-05
RPD	-2.49E-03	-6.86E-03	-8.23E-03	-1.16E-02	-2.49E-03
	7.92E-05	6.80E-04	1.04E-03	7.06E-04	7.54E-05
CF	-2.44E-03	-7.01E-03	-8.02E-03	**-1.26E-02**	-6.98E-03
	6.34E-05	7.06E-04	9.67E-04	**9.08E-04**	7.23E-04
TABU	-2.52E-03	-6.98E-03	-8.30E-03	-1.16E-02	-2.52E-03
	9.85E-05	7.71E-04	9.17E-04	7.28E-04	8.36E-05
GR	-5.00E-03	-6.77E-03	-8.26E-03	-1.15E-02	-4.94E-03
	2.18E-04	7.73E-04	9.48E-04	8.97E-04	2.22E-04

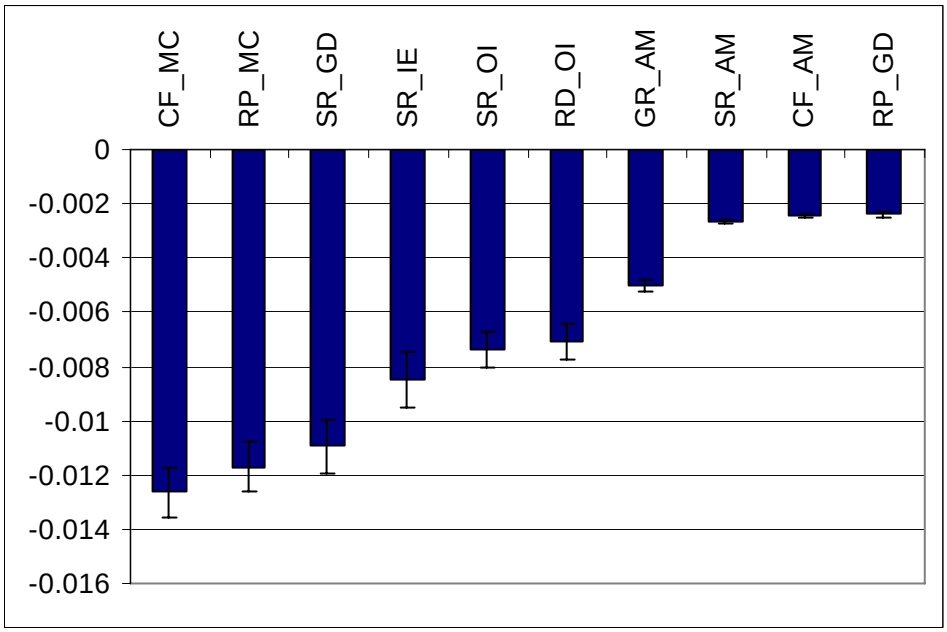

Figure C.18. Average best fitness values for Yue20022

Table C.19. Average best fitness values for Yue20023

	AM	*OI*	*IE*	*MC*	*GD*
SR	-1.35E-02	-1.22E-02	-1.26E-02	-1.49E-02	-1.42E-02
	2.47E-04	7.65E-04	6.88E-04	3.02E-04	4.89E-04
RD	-1.34E-02	-1.20E-02	-1.26E-02	**-1.51E-02**	-1.34E-02
	1.84E-04	6.79E-04	8.57E-04	**3.37E-04**	1.94E-04
RP	-2.38E-03	-6.98E-03	-1.24E-02	-1.17E-02	-2.39E-03
	6.61E-05	9.46E-04	6.77E-04	8.95E-04	8.32E-05
RPD	-2.49E-03	-6.86E-03	-1.23E-02	-1.16E-02	-2.49E-03
	7.92E-05	6.80E-04	8.60E-04	7.06E-04	7.54E-05
CF	-1.34E-02	-1.21E-02	-1.26E-02	**-1.52E-02**	-1.20E-02
	1.93E-04	6.61E-04	8.27E-04	**2.69E-04**	7.02E-04
TABU	-2.52E-03	-6.98E-03	-1.24E-02	-1.16E-02	-2.52E-03
	9.85E-05	7.71E-04	6.94E-04	7.28E-04	8.36E-05
GR	-1.45E-02	-1.22E-02	-1.24E-02	-1.50E-02	-1.45E-02
	1.64E-04	7.23E-04	7.75E-04	2.71E-04	1.84E-04

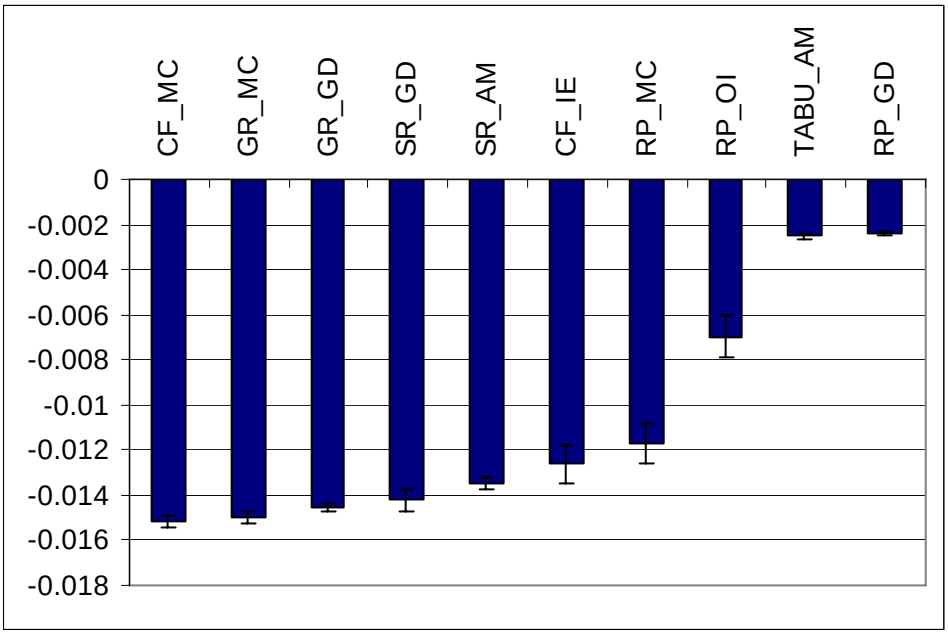

Figure C.19. Average best fitness values for Yue20023

Table C.20. Average best fitness values for Yue20031

	AM	*OI*	*IE*	*MC*	*GD*
SR	-2.92E-03	-7.54E-03	-8.69E-03	-1.26E-02	-1.31E-02
	8.20E-05	1.15E-03	1.39E-03	9.62E-04	1.89E-03
RD	-2.75E-03	-7.23E-03	-8.06E-03	-1.42E-02	-2.72E-03
	1.11E-04	1.21E-03	1.48E-03	1.42E-03	8.47E-05
RP	-2.63E-03	-6.92E-03	-8.64E-03	-1.42E-02	-2.60E-03
	8.81E-05	1.11E-03	1.73E-03	1.40E-03	6.77E-05
RPD	-2.72E-03	-7.32E-03	-8.16E-03	-1.42E-02	-2.67E-03
	1.16E-04	1.10E-03	1.44E-03	1.56E-03	7.73E-05
CF	-2.68E-03	-6.99E-03	-8.44E-03	**-1.59E-02**	-7.19E-03
	1.07E-04	1.09E-03	1.62E-03	**1.65E-03**	1.10E-03
TABU	-2.73E-03	-7.11E-03	-8.07E-03	-1.41E-02	-2.71E-03
	9.07E-05	1.21E-03	1.79E-03	1.51E-03	8.47E-05
GR	-5.14E-03	-7.45E-03	-8.29E-03	-1.42E-02	-5.14E-03
	2.23E-04	1.28E-03	1.87E-03	1.75E-03	2.15E-04

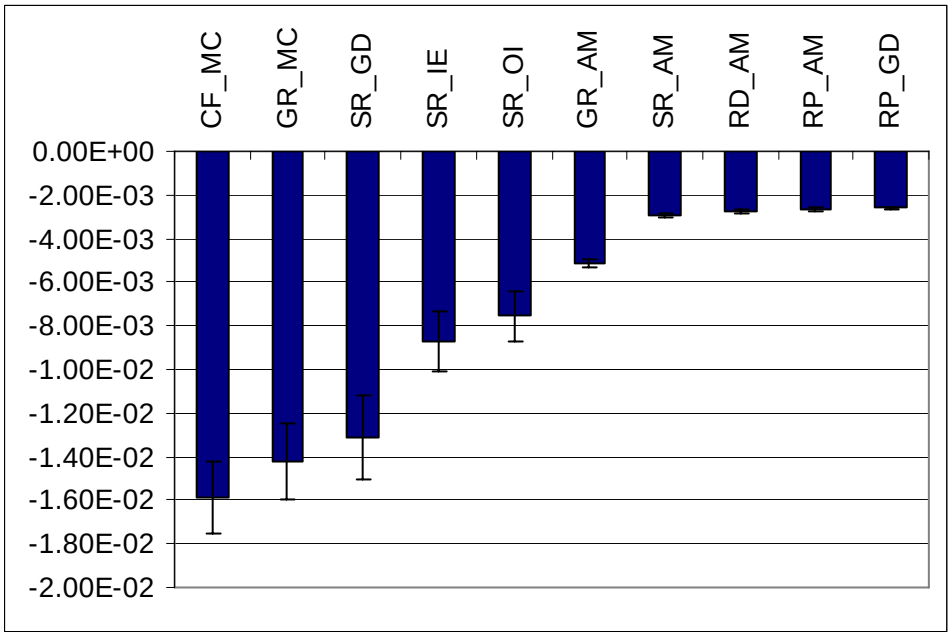

Figure C.20. Average best fitness values for Yue20031

Table C.21. Average best fitness values for Yue20032

	AM	*OI*	*IE*	*MC*	*GD*
SR	-1.83E-03	-3.42E-03	-3.63E-03	-4.61E-03	-4.14E-03
	5.64E-05	2.70E-04	3.59E-04	3.64E-04	4.04E-04
RD	-1.81E-03	-3.06E-03	-3.29E-03	-4.92E-03	-1.81E-03
	5.02E-05	2.42E-04	2.89E-04	4.25E-04	5.46E-05
RP	-1.72E-03	-3.12E-03	-3.33E-03	-4.90E-03	-1.72E-03
	5.38E-05	2.79E-04	3.17E-04	3.68E-04	5.89E-05
RPD	-1.80E-03	-3.11E-03	-3.34E-03	-5.02E-03	-1.78E-03
	5.70E-05	2.91E-04	3.62E-04	3.65E-04	4.02E-05
CF	-1.77E-03	-3.09E-03	-3.32E-03	**-5.42E-03**	-3.10E-03
	5.70E-05	2.41E-04	3.66E-04	**3.68E-04**	2.73E-04
TABU	-1.82E-03	-3.20E-03	-3.19E-03	-4.99E-03	-1.82E-03
	5.58E-05	2.72E-04	3.14E-04	3.83E-04	3.98E-05
GR	-3.07E-03	-3.18E-03	-3.43E-03	-4.73E-03	-3.07E-03
	9.31E-05	3.01E-04	3.80E-04	2.91E-04	9.36E-05

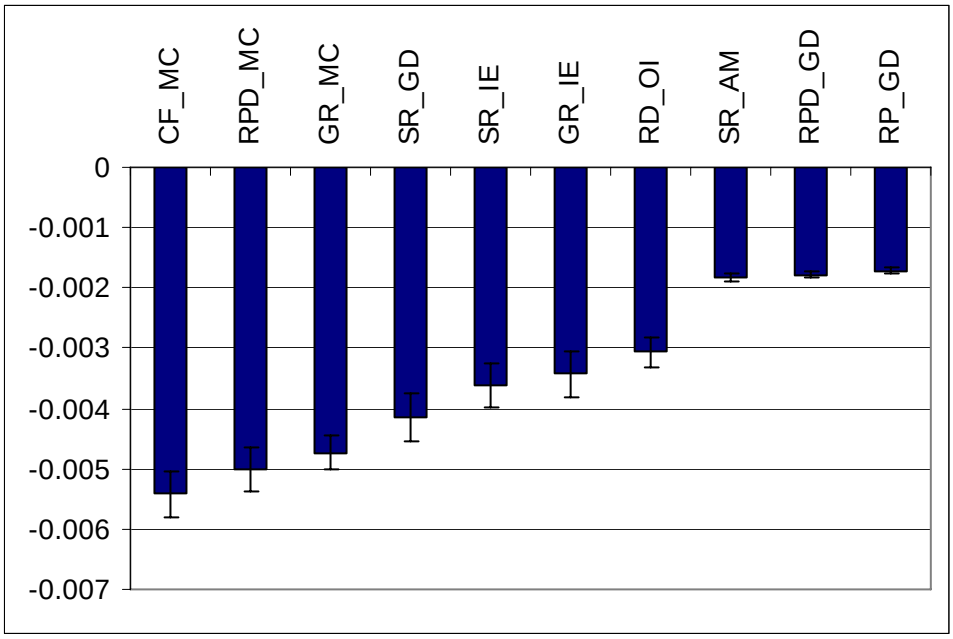

Figure C.21. Average best fitness values for Yue20032

REFERENCES

[1] Cowling, P., G. Kendall, and E. Soubeiga, "A Hyperheuristic Approach to Scheduling a Sales Summit", Selected Papers of the Third International Conference on the Practice and Theory of Automated Timetabling (PATAT2000), Konstanz, 16-18 August 2000, pp.176-190, 2000.

[2] Burke, E., G. Kendall, J. Newall, E. Hart, P. Ross and S. Schulenburg, "Hyper-heuristics: An Emerging Direction in Modern Search Technology", in F. Glover and G. A. Kochenberger (eds.), *Handbook of Metaheuristics*, pp.457-474, Kluwer Academic Publishers, Boston, Dordrecht, London, 2003.

[3] Burke, E. and E. Soubeiga, *"Scheduling Nurses Using a Tabu-Search Hyperheuristic", 1st Multidisciplinary Intl. Conf. on Scheduling: Theory and Applications (MISTA 2003),* Nottingham, 13-16 August 2003, Vol. 1, pp. 197-218, 2003.

[4] Burke, E. K., G. Kendall, and E. Soubeiga, *"A Tabu-Search Hyper-heuristic for Timetabling and Rostering", Journal of Heuristics,* Vol 9, No. 6, pp. 451-470, 2003.

[5] Ayob, M. and G. Kendall, *"A Monte Carlo Hyper-Heuristic to Optimise Component Placement Sequencing For Multi Head Placement Machine"*, *Proceedings of the International Conference on Intelligent Technologies, InTech'03,* Chiang Mai, 17-19 December 2003, pp. 132-141, 2003.

[6] Kendall, G. and M. Mohamad, *"Channel Assignment in Cellular Communication Using a Great Deluge Hyper-heuristic", Proceedings of the 2004 IEEE International Conference on Network (ICON2004),* Singapore, 16-19 November 2004, pp. 769-773, 2004.

[7] Ross, P., "Hyper-heuristics", in E. K. Burke and G. Kendall (eds.), *Search Methodologies, Introductory Tutorials in Optimization and Decision Support Techniques*, pp. 529-556, Springer Verlag, New York, 2005.

[8] Burke, E. K., A. Meisels, S. Petrovic and R. Qu, *"A Graph-Based Hyper Heuristic for Timetabling Problems", Accepted for publication in the European Journal of Operational Research*, 2005.

[9] Even, S., A. Itai and A. Shamir, *"On the Complexity of Timetable and Multicommodity Flow Problems", SIAM Journal of Computation*, Vol. 5(4), pp. 691-703, 1976.

[10] Carter, M. W., G. Laporte, and S. T. Lee, *"Examination Timetabling: Algorithmic Strategies and Applications". Journal of the Operational Research Society,* 47, pp. 373-383, 1996.

[11] Soubeiga, E., *"Development and Application of Hyperheuristics to Personnel Scheduling"*, Ph.D. Thesis, School of Computer Science and Information Technology, The University of Nottingham, 2003.

[12] Nareyek, A., Choosing Search Heuristics by Non-Stationary Reinforcement Learning, 2001.

[13] Burke E. K., S. Petrovic, and R. Qu, *"Case Based Heuristic Selection for Timetabling Problems", Accepted for publication in the Journal of Scheduling,* Vol.9 No2, 2006.

[14] De Jong, K., "*An Analysis of the Behaviour of a Class of Genetic Adaptive Systems"*, PhD thesis, University of Michigan, 1975.

[15] Rastrigin, L. A., "Extremal Control Systems", in *Theoretical Foundations of Engineering Cybernetics Series*, Moscow, Nauka, Russian, 1974.

[16] Schwefel, H. P., *Numerical Optimization of Computer Models*, John Wiley & Sons, New York, 1981.

[17] Griewangk, A. O., *"Generalized Descent of Global Optimization", Journal of Optimization Theory and Applications*, Vol. 34, pp. 11-39, 1981.

[18] Ackley, D. H., "An Empirical Study of Bit Vector Function Optimization", in L. Davis (ed.), *Genetic Algorithms and Simulated Annealing,* pp. 170-204, Pitman Publishing, London, 1987.

[19] Easom, E. E., *A Survey of Global Optimization Techniques,* M. Eng. thesis, University of Louisville, 1990.

[20] Mitchell, M. and S. Forrest, "Fitness Landscapes: Royal Road Functions", in T. Baeck, D. Fogel and Z. Michalewiz (Eds.), *Handbook of Evolutionary Computation*, pp. 1-25, Institute of Physics Publishing and Oxford University, Bristol, 1997.

[21] Goldberg, D. E., "Genetic Algorithms and Walsh Functions: Part I, A Gentle Introduction", in *Complex Systems,* pp. 129-152, 1989.

[22] Goldberg, D. E., "Genetic Algorithms and Walsh Functions: Part II, Deception and Its Analysis", in *Complex Systems*, pp. 153-171, 1989.

[23] Whitley, D., "Fundamental Principles of Deception in Genetic Search", in G. J. E. Rawlings (ed.), *Foundations of Genetic Algorithms*, pp. 221-241, Morgan Kaufmann, San Mateo, 1991.

[24] Davis, L., *"Bit Climbing, Representational Bias, and Test Suite Design", Proceedings of the 4th International Conference on Genetic Algorithms,* San Diego, July 1991, pp. 18-23, 1991.

[25] Özcan, E., *"An Empirical Investigation on Memes, Self-generation and Nurse Rostering", The Sixth International Conference on the Practice and Theory of Automated Timetabling (PATAT2006)*, Brno, 30 August – 1 September 2006, in review, 2006.

[26] Özcan, E., *"Memetic Algorithms for Nurse Rostering", Lecture Notes in Computer Science, Springer-Verlag, The 20th International Symposium on Computer and Information Sciences*, İstanbul, 26-28 October 2005, pp. 482-492, 2005.

[27] Burke, E. K., J. P. Newall, and R. F. Weare, *"A Memetic Algorithm for University Exam Timetabling", 1st International Conference on the Practice and Theory of Automated Timetabling (ICPTAT'95)*, Edinburgh, 30 August - 1 September 1995, pp. 241-250, 1996.

[28] Burke, E. K., D. Elliman, P. Ford, and B. Weare, *"Examination Timetabling in British Universities- A Survey", Lecture Notes in Computer Science*, Springer-Verlag, vol. 1153, pp. 76–90, 1996.

[29] Marin, H. T., *"Combinations of GAs and CSP Strategies for Solving Examination Timetabling Problems",* Ph. D. Thesis, Instituto Tecnologico y de Estudios Superiores de Monterrey, 1998.

[30] Paquete, L. F. and C. M. Fonseca, *"A Study of Examination Timetabling with Multiobjective Evolutionary Algorithms", Proceedings of the 4th Metaheuristics International Conference (MIC 2001)*, Porto, 16-20 July 2001, pp. 149-154, 2001.

[31] Wong, T., P. Côté and P. Gely, *"Final Exam Timetabling: A Practical Approach", Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering*, Winnipeg, 12-15 May 2002, vol. 2, pp. 726-731, 2002.

[32] Di Gaspero, L. and A. Schaerf, *"Tabu Search Techniques for Examination Timetabling", Lecture Notes in Computer Science, selected papers from the Third International Conference on Practice and Theory of Automated Timetabling (PATAT2000),* Konstanz, 16-18 August 2000, pp. 104-117, 2000.

[33] Merlot, L.T.G., N. Boland, B. D. Hughes, and P. J. Stuckey, *"A Hybrid Algorithm for the Examination Timetabling Problem", Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling (PATAT2002)*, Gent, 21-23 August 2002, pp. 348-371, 2002.

[34] Petrovic, S., Y. Yang, and M. Dror, *"Case-based Initialisation for Examination Timetabling", Proceedings of 1st Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2003)*, Nottingham, 13-16 August 2003, pp. 137-154, 2003.

[35] Burke, E. K. and J. P. Newall, *"Solving Examination Timetabling Problems through Adaptation of Heuristic Orderings", Models and Algorithms for Planning and Scheduling Problems. Annals of Operations Research,* vol. 129, pp. 107-134, 2004.

[36] Ozcan, E. and E. Ersoy, *"Final Exam Scheduler – FES", Proceedings of 2005 IEEE Congress on Evolutionary Computation*, Edinburgh, 2-5 September 2005, vol. 2, pp. 1356-1363, 2005.

[37] Ozcan, E., "Towards an XML based standard for Timetabling Problems: TTML", in G. Kendall (ed.), *Multidisciplinary Scheduling: Theory and Applications*, pp. 163-186, Springer Verlag, New York, 2005.