A NEW HYPERHEURISTIC

by

Mustafa MISIR

Yeditepe University

Faculty of Engineering and Architecture

Department of Computer Engineering

2007

A NEW HYPERHEURISTIC

APPROVED BY:

Assist.Prof.Dr. Ender Özcan (Thesis Supervisor)	……………………
Assist.Prof.Dr. Emin Erkan Korkmaz	……………………
Assist.Prof.Dr. Mustafa Türkboyları	……………………

DATE OF APPROVAL: 28/05/2007

ACKNOWLEDGEMENTS

I want to thank Assist. Prof. Dr. Ender Özcan for his helps, valuable ideas and support during my engineering project.

ABSTRACT

A NEW HYPERHEURISTIC

A hyperheuristic is an approach that chooses a heuristic among a set of heuristics and after applying it to a candidate solution, decides whether to accept or reject the new solution. Hyperheuristics are considered to be a higher level of abstraction as compared to metaheuristics. There is a novel metaheuristic called IDWalk (*Intensification /Diversification Walk*) that provides a more intense search, preventing premature convergence to a local optima. In this report, IDWalk is extended using the traditional hyperheuristics framework and a new hyperheuristic mechanism emerged, providing a powerful hyperheuristic. The modified versions of IDWalk, namely; CIDWalk (*Constrained IDWalk*) and EXIDWalk (*Extended IDWalk*) are proposed as two new hyperheuristics. Extensive experiments are performed on fourteen well-known benchmark functions to observe the behaviour of proposed hyperheuristics. Additionally, they are applied to a benchmark set of exam timetabling problems as an instance of a constraint based real-world optimization problem. The results indicate that the proposed approaches are promising.

ÖZET

YENİ BİR ÜST- BULUŞSAL

Bir üst-buluşsal, bir buluşsallar kümesi arasından bir buluşsal seçen ve sonra onu bir aday sonuca uygulayan, yeni çözümün kabul edilmesi ya da rededilmesi kararını veren bir yaklaşımdır. Üst-buluşsallar, meta-buluşsallara nazaran daha yüksek seviyeli oldukları düşünülmektedir. Daha güçlü bir arama sağlayan, bir yerel optimuma erken yakınsamasını engelleyen, IDWalk (*Kuvvetlendirme/Çeşitlendirme Yürüyüşü*)[1] isimli yeni bir meta-buluşsal bulunmaktadır. Bu raporda, IDWalk, geleneksel üst-buluşsallar çerçevesini kullanarak genişletilmektedir ve güçlü bir üst-buluşsal sağlayan yeni bir üst-buluşsal mekanizması ortaya çıkmıştır. IDWalk'ın değiştirilmiş türleri olan CIDWalk (*Kısıtlandırılmış IDWalk*) ve EXIDWalk (*Genişletilmiş IDWalk*) isimli iki yeni üst-buluşsal önerilmiştir. Çok amaçlı deneyler, önerilen üst-buluşsalların davranışlarını gözlemlemek için on dört denetçi test fonksiyonu üzerinde çalıştırılmıştır. Ayrıca, bir kısıta dayalı gerçek-dünya eniyileme problem örneği olarak, bir sınav çizelgeleme problemleri denetçi test kümesine uygulanmıştır. Sonuçlar, önerilen yaklaşımlarının ümit verici olduklarını göstermektedir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS / ABBREVIATIONS

AM	All moves accepted
CF	Choice Function Heuristic Selection
CSP	Constarint Satisfaction Problem
CIDWALK	Constrained IDWalk
CL	Candidate List
DIMM	Dimensional Mutation Operator
EXIDWALK	Extented IDWalk
HYPM	Hyper-Mutation Operator
IDWALK	Intensification/Diversification Walk
IE	Improving and equal moves accepted
MC	Monte Carlo Acceptance Criterion
OI	Only improving moves accepted
SO	Simple Ordered Heuristic Selection
SR	Success Rate
SWPD	Swap Dimension Operator
TFA	Traditional Framework F_A
THH	Traditional Hyperheuristic

1. INTRODUCTION

There exist some applications for Hyperheuristics which is a higher-level abstraction for meta-heuristics that are problem solvers and optimizers for specific problems concept. One of these applications performed as frameworks [2], in these frameworks, the hyperheuristics system was thought in 4 different ways which are FA, FB, FC and FD. These four options have an idea of using some mutational heuristics that provide diversity and hill climbers that generate mostly better solutions as local search methods. However, we only take FA with mutational part, since we wanted to use IDWalk [1] approach which is an optimization metaheuristic that offers advantages for combining simplicity with effectiveness strategy. IDWalk uses only one main parameter called *Max* which denotes maximum number of candidate neighbors for every move. These neighbors come from a candidate list that includes rejected candidates during optimization process. If we reach to *Max* due to not producing any acceptable candidate who has an improved or equal fitness than previous accepted one, then choose one of the rejected candidate from the list according to two different choice mechanisms which are ANY and BEST. ANY selects a candidate randomly as you can understand from its name, BEST chooses the best neighbor by considering their fitnesses. We decided to add a new rejected candidate selection mechanism called TOUR that looks for a candidate according to Tournament Selection [3] method. Also, during experiments we noticed a new strategy called Constrained IDWalk that puts a predefined limit on fitness values for acceptance of a rejected candidate and added it to the system as an another hyperheuristic. Furthermore, another improved version of IDWalk called EXIDWalk was designed to provide neighbor of multiple heuristics during walk.

For the performance analysis phase, experiments were performed on fourteen well-known benchmark functions. Also, a real-world constaint optimization problem which is examination timetabling was used to see what are the behaviours of proprosed hyperheuristics. Timetabling is a very hard NP-complete problem which has some constraints to be handled. There are plenty of groups and researchers who look for some optimized solutions. Due to its NP characteristic, there are no any way of solving it in a polynomial time and it has not an exact solution, so, everybody tries to find the most optimized timetable among rest of simililar studies. In addition, it not easy to find a reasonable result which can be generally-accepteed, that is, one mechanism that performs well on some set of data, can be work extremely bad for another. So that, I did not use only one type of examination timetabling data, there are very different versions of them. However, here, I am not trying to just find the best, I want also to see which hyperheuristic works better on the given data, so I can have a chance to compare them.

This report is organized as follows. Literature survey on hyperheuristics is presented in Chapter 2. Literature survey on IDWalk, CIDWalk and EXIDWalk approaches are presented in Chapter 3. Literature survey on benchmark functions, heuristics for benchmark functions, experimental settings and results on benchmark functions are presented in Chapter 4. Literature survey on timetabling in general and examination timetabling, heuristics for examination timetabling, experimental settings and results on examination timetabling are presented in Chapter 5. Conclusions on the literature study, proposed hyperheuristic frameworks and combinations, experiments and results are presented and possible future works are listed in Chapter 6. The tables and charts which present the experimental results of IDWalk based hyperheuristic frameworks on benchmark functions are presented on Appendix A. The tables and charts which present the experimental results of IDWalk based hyperheuristic combinations on a set of examination timetabling benchmark data with a compherensive approach are presented on Appendix B.

2. HYPERHEURISTICS

Hyperheuristics is an emerging search technology that is motivated, to a large extent, by the goal of raising the level of generality at which optimization systems can operate [4]. It can be defined shortly as heuristics to choose heuristics within an acceptance mechanism. Its aim is to provide more general system to handle a wide range of problem domains rather than current meta-heuristic technology which tends to be customized to a particular problem and the problems that we have been trying to solve are in the type of combinatorial optimization which is in NP-hard category and need to be solved in polynomial time.

Hyperheuristics involve two main parts, first part is selection mechanism. It is an heuristic to choose another one among a heuristic set, there exist many kind of them such as Simple Random, Choice Function [5] etc. In our study, we used Choice Function as heuristic selection mechanism for experiments which have multiple heuristics in their heuristic sets. It was the best choice, since we noticed it after results of the hyperheuristics framework experiments. It chooses heuristics according to its performances by looking back, that is, it decides which heuristic will perform for the next step on the based of previous performance that comes from fitness improvement and execution time of low level heuristics. We proposed a new system called Simple Ordered, as an additional selection mechanism for the heuristic set that has multiple heuristics.

The other part is acceptance. After you choose a heuristic, you apply it onto initial candidate, then you have a new candidate solution with a fitness value. The acceptance mechanism compares fitness values of initial and new candidates according to a defined strategy like IE, IO, Monte Carlo [6], Great Deluge [7]. However, we only used only IE, because of it is success on frameworks.

In Figure 1, there is a flowchart which shows the flow of traditional hyperheuristics framework FA. As you can see, there is an heuristic set that has some number of mutational heuristics and hill climbers. A chosen selection mechanism selects a heuristic from the set and apply it to the current candidate, then send new candidate to the acceptance phase, if it is accepted, then new candidate is assigned instead of current candidate, in the other case, a new heuristic is chosen and the loop performs in the same manner until producing an accepted candidate.

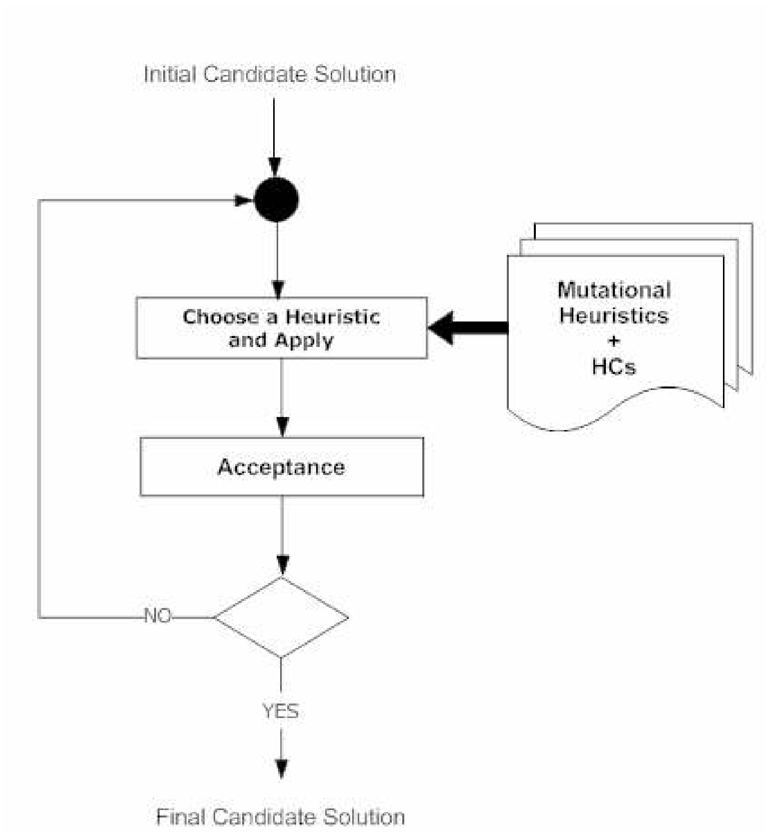
Initial Candidate Solution
Choose a Heuristic and Apply
Mutational Heuristics + HCs
Acceptance
NO
YES
Final Candidate Solution

Figure 2. 1. Traditional Hyperheuristics Framework FA

3. IDWALK

The idea behind IDWalk strategy exactly takes places on the abbreviated form itself, since it stands for *Intensification/Diversification Walk.* Intensification is used for choosing a neighbor by looking its cost on the based of fitness values of candidates, if the cost of a neighbor is less than or equal to the cost of the current solution, then choose it as next move (*"less than or equal to*" is the same as IE acceptance mechanism that is presented and used in hyperheuristics frameworks). Also, IDWalk provides an additional control mechanism that does not permit being stuck at any local optima by Diversification. During walking among neighbors, if the walk does not help to reach an accepted neighbor for number of *Max* move, then, we understand *Intensification* could not be succeeded, after that, it goes to apply *Diversification* by choosing a rejected candidate according to a selection mechanism from a candidate list that includes all rejected candidates for each walk. You can see all the flows from Figure 2.

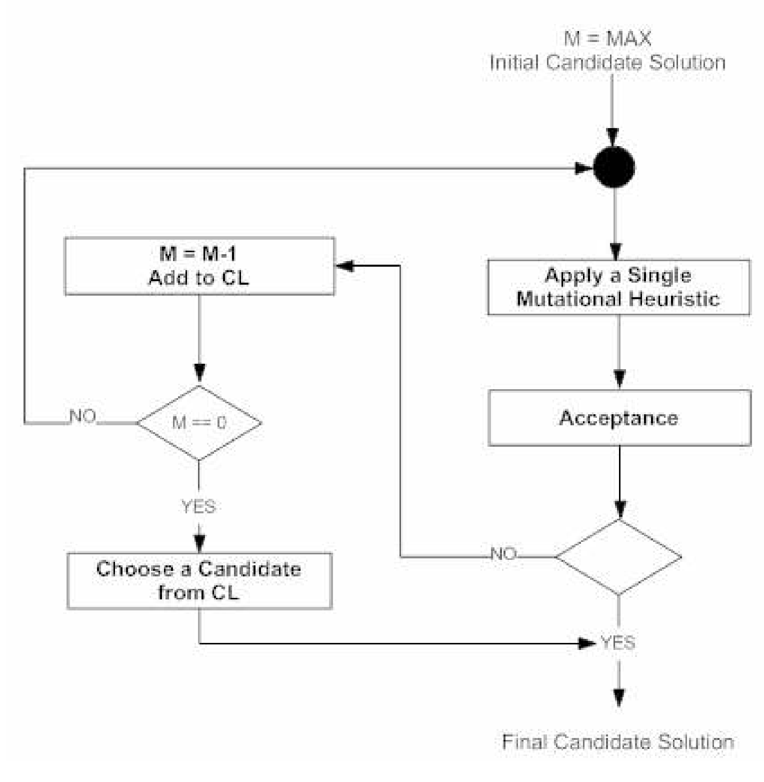
M = MAX
Initial Candidate Solution
M = M-1
Add to CL
Apply a Single Mutational Heuristic
Acceptance
NO
M == 0
YES
Choose a Candidate from CL
NO
YES
Final Candidate Solution

Figure 3. 1. Basically IDWalk (*CL: Candidate List*)

3.1. IDWALK as a Hyperheuristic

IDWalk seems as a good approach for solving combinatorial optimization problems such as Random CSPs, Graph Coloring [6], CELAR Frequency Assignment [1], Spatially-Balanced Square [7] and Car Sequencing Problems [8]. However, there is a restriction about number of heuristics, in this strategy, we can use only one heuristic during walk for neighborhood. According to this situation, we can use this system directly on TFA approach, just by defining heuristic set as only one heuristic with IE acceptance and the heuristic must be mutational, since, IDWalk cannot show its diversification side with hill climbers which does not easily permit to reach the number of rejected candidates to *Max* value.

For instance, take MAX = 10 and a candidate with a fitness value of 20 and *Improving or Equal* acceptance mechanism. In addition, you have a heuristic which is HYPM (*Hyper Mutation*) to flip the bits that reside on the candidate with the probability of 0.5. You apply HYPM on initial candidate solution and you get a new candidate with 19.7 fitness. It is sent to our acceptance mechanism and for a minimization problem, the new candidate will be accepted. Then the loop go to the beginning, again, then apply HYPM onto accepted candidate and a new candidate comes out with a fitness of 19.8. After, it is directed to our acceptance mechanism for the control, it will be rejected and added to our candidate list, then the number of rejected candidates will be checked to see whether it is full or not and it will be seen that there exist just one rejected candidate. The the loop goes to the beginning, again. With the same procedures, many different candidates will be produced and when the size of the rejected candidate list is 10, then the system will choose a rejected candidate list from the list with ANY or BEST or TOUR. We must have chosen one of these selection mechanism, statically. In this example, we have ANY, so, we choose a candidate randomly, and accept it as the new candidate. These routines continue until the reach to the global optimum value or after apprimately 600x50 sec.

3.2. IDWALK with a Heuristic Set

For the second IDWalk based hyperheuristics system, we can use multiple mutational heuristics, but IDWalk does not support multiple heuristics during walk, so we can use one of the selection mechanisms that are defined within the framework to choose the heuristic which will be used during walk. One of the selection mechanisms that I used is Choice Function [5], since, the experiments which were performed on hyperheuristic frameworks [2] showed that for most of the times, this approach is the best one. Another, heuristic selection system is Simple Ordered (*SO*) and it is our proposed idea. We, provided SO, because of providing a neighbor from multiple heuristics which are available in the given heuristic set. That is, SO makes some improvements on IDWalk by extending its neighborhood. Then, a new approach came out, EXIDWalk (*Extended IDWalk*).

3.3. Constrained IDWALK

As an improved version of IDWalk, we propose a new strategy called CIDWalk (*Constrained IDWalk*). It puts a predefined limit for the candidates that are rejected and will be added to the candidate list about their fitness values. We determine this constraint because of the nature of mutational heuristics, since, they produces new candidates that have inconsistent fitness values. That is, mutationally produced candidates have a possibility of having really bad fitness values and these values may cause to go to the situation that is available at very beginning of walk. Maybe, we can give enough time to process all the possible moves for IDWalk, and it can reach a feasible solution, but we must put some limitations about execution time or number of evolutions per run and step, we cannot let the program to run forever.

3.4. Extended IDWALK

In this approach, we extended the neighborhood for each walk by using more than one heuristic. Then, to give equal chance for all the heuristics during a walk, we combined our new selection mechanism SO with IDWalk. It is not just a heuristic set to use one of defined heuristics, it is an ordered usage of the heuristics during walk. By this system, we tries all the heuristics consecutively, if they produce an acceptable candidate, then we accept it according to IE, on the other hand, we put all the rejected candidates to the candidate list without any discrimination among the heuristics to use during the case of being stuck at a local optima.

4. IDWALK BASED HYPERHEURISTICS FOR BENCHMARK OPTIMIZATION

4.1. Benchmark Functions

The best way of experimenting an algorithm, especially for optimization, is using it for different types of problems. However, if you trying to solve one type of problem, then you can use similar problems with different domains. For the time being and the purpose of hyperheuristics concept, we want to provide a problem independent mechanism, so to see its success, we must run the algorithm for different type of problems. We can find many real-world problems, but before applying it to a real world problem such as timetabling, we must try it on some benchmark functions. So, we chosen fourteen well-known benchmark functions to see what the mechanisms do on distinctive problems which are pure mathematical functions with lower and upper bounds.

All the benchmark functions that we used are listed with their mathematical representations in Table 4.1. In addition, in Table 4.2, more detailed view of the functions are provided such as bounds, global optimum values, modality characteristics etc. One of the most important data in Table 4.2 is about modality which shows us number of local optima in the range of function boundaries. If this optimum value is exactly one, then we can say that given benchmark problem has only one local optima and it is the same as global optima that we want to reach and it is called *unimodal* in modality manner. On the other hand, if the function has multiple of local optima, then it is more difficult to find the global optima, since we can get stuck at a local optima in this *Multimodal* structure. As I mentioned before, diversification mechanism of IDWalk provide not to be stuck at such a point. We will not use any hill climber, to encounter with the problem due to local optimum points, but IDWalk to handle this problem. What about their continuity? Three of the benchmark functions are discrete and the rest is continuous. For separability, we have five separable functions that give opportunity to think overall chromosome as dimensions and we can use any algorithm on the dimensions without doing anything to the rest.

Table 4.1. Benchmark functions used during the experiments

Label	*Name*	*Formula*	*Source*
F1	Sphere	$f(\vec{x}) = \sum_{i=1}^{n} x_i^2$	De Jong (1975)[10]
F2	Rosenbrock	$f(\vec{x}) = \sum_{i=1}^{n-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$	De Jong (1975)[10]
F3	Step	$f(\vec{x}) = 6 \cdot n + \sum_{i=1}^{n} \lfloor x_i \rfloor$	De Jong (1975)[10]
F4	Quartic *with noise*	$f(\vec{x}) = \sum_{i=1}^{n} (i \cdot x_i^4 + U(0,1))$	De Jong (1975)[10] Tasoulis (2004)
F5	Foxhole	$f(\vec{x}) = \dfrac{1}{0.002 + \sum_{j=1}^{25} \dfrac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6}}$ $a_{1j} = \begin{cases} -32 \to \mathrm{mod}(j,25) = 1 \\ -16 \to \mathrm{mod}(j,25) = 2 \\ 0 \to \mathrm{mod}(j,25) = 3 \\ 16 \to \mathrm{mod}(j,25) = 4 \\ 32 \to \mathrm{mod}(j,25) = 0 \end{cases}$ $a_{2j} = \begin{cases} -32 \to j > 0 \wedge j \le 5 \\ -16 \to j > 5 \wedge j \le 10 \\ 0 \to j > 10 \wedge j \le 15 \\ 16 \to j > 15 \wedge j \le 20 \\ 32 \to j > 20 \wedge j \le 5 \end{cases}$	De Jong (1975)[10]
F6	Rastrigin	$f(\vec{x}) = 10 \cdot n + \sum_{i=1}^{n} (x_i^2 - 10 \cdot \cos(2\pi x_i))$	Rastrigin (1974)[11]
F7	Schwefel	$f(\vec{x}) = 418.9829 \cdot n + \sum_{i=1}^{n} x_i \cdot \sin(\sqrt{\lvert x_i \rvert})$	Schwefel (1981)[12]
F8	Griewangk	$f(\vec{x}) = \sum_{i=1}^{n} \dfrac{x_i^2}{4000} - \prod \cos(\dfrac{x_i}{\sqrt{i}}) + 1$	Griewangk (1981)[13]
F9	Ackley	$f(\vec{x}) = 20 + e - 20 \cdot e^{-0.2 \cdot \sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n} \sum_{i=1}^{n} \cos(2\pi x_i)}$	Ackley (1987)[14]
F10	Easom	$f(\vec{x}) = -(\prod_{i=1}^{n} \cos(x_i)) \cdot (e^{-\sum_{i=1}^{n} (x_i - \pi)^2})$	Easom (1990)
F11	Schwefel's Double Sum	$f(\vec{x}) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j^2)$	Schwefel (1981)[14]
F12	Royal Road	$f(\vec{x}) = \sum_{s \in S} order(s) \sigma_s(\vec{x}),$ where $\sigma_s(\vec{x}) = \begin{cases} 1 & \text{if } \vec{x} \text{ is an instance of } s \\ 0 & \text{otherwise} \end{cases}$ and s is a schema	Mitchell (1997)[15]
F13	Goldberg	$\begin{array}{\|c\|c\|c\|c\|c\|} \hline \textit{String} & 000 & 001 & 010 & 011 \\ \hline \textit{Value} & 1 & 3 & 3 & 8 \\ \hline \end{array}$	Goldberg (1989a, 1989b)[16]

String	100	101	110	111
Value	5	8	8	0

$$f(\vec{x}) = \sum_{i=1}^{n} Value(x_i)\ ,$$

where x_i is the i^{th} 3-bit string

F14 Whitley

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

$$f(\vec{x}) = \sum_{i=1}^{n} Value(x_i)\ ,$$

where x_i is the i^{th} 4-bit string

Whitley (1991)[17]

Table 4.2. Characteristics of the benchmark functions used during the experiments (*u : Unimodal, m: Multimodal*)

Label	range of x_i	dimension	optimum	modality	isContinuous	isSeparable
F1	[-5.12,5.12]	10	0	u	yes	yes
F2	[-2.048,2.048]	10	0	u	yes	yes
F3	[-5.12,5.12]	10	0	u	yes	yes
F4	[-1.28,1.28]	10	1	m	yes	yes
F5	[-65.536,65.536]	2	1	m	yes	no
F6	[-5.12,5.12]	10	0	m	yes	yes
F7	[-500,500]	10	0	m	yes	yes
F8	[-600,600]	10	0	m	yes	no
F9	[-32.768,32.768]	10	0	m	yes	no
F10	[-100,100]	6	-1	u	yes	no
F11	[-65.536,65.536]	10	0	u	yes	no
F12	n/a	8	0	n/a	no	yes
F13	n/a	30	0	n/a	no	yes
F14	n/a	6	0	n/a	no	yes

4.2. Heuristics for Benchmark Function Optimization

We will use fourteen different benchmark functions, but to solve them or to reach global optima, we must decide on some heuristics which will help to look around for a better candidate who is closer to the target. There are two main choice for the characteristics of heuristics, they are hill climbers and mutational heuristics. We will only use a stochastics approach with mutational heuristics to test IDWalk because of providing exploitation by IDWalk itself not with a hill climber.

We decided on three mutational heuristics which are HYPM, DIMM and SWPD. Swap Dimension Operator (SWPD) chooses two different dimensions on candidate chromosome and swaps them. For Dimensional Mutation Operator (DIMM), it chooses only one dimension and flips all the bits on the based of pre-defined probability which is 0.5. And Hyper-Mutation Operator (HYPM), it works like DIMM, but it does not apply the operation to a randomlay speficied dimension as DIMM does, it flips all the bits with mutation probability of 0.5. They, can also be seen from following figures with their examples.

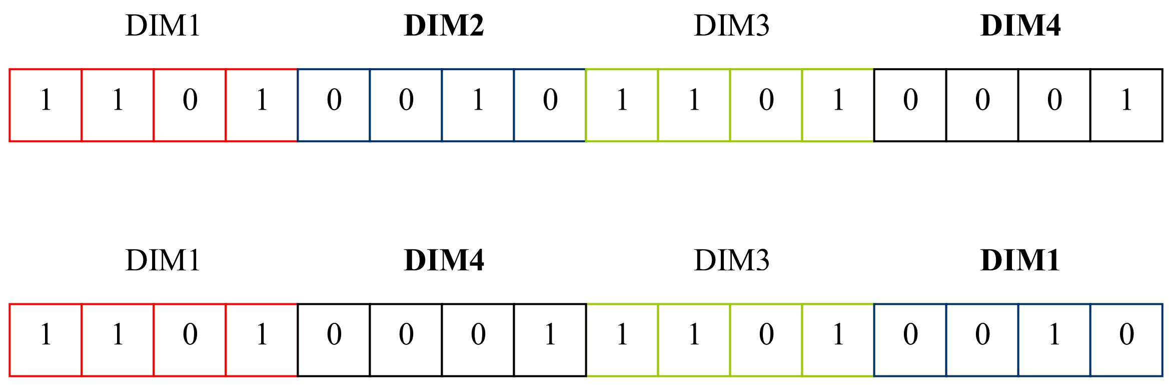

Figure 4.1. SWPD, second dimension and forth one exchanges

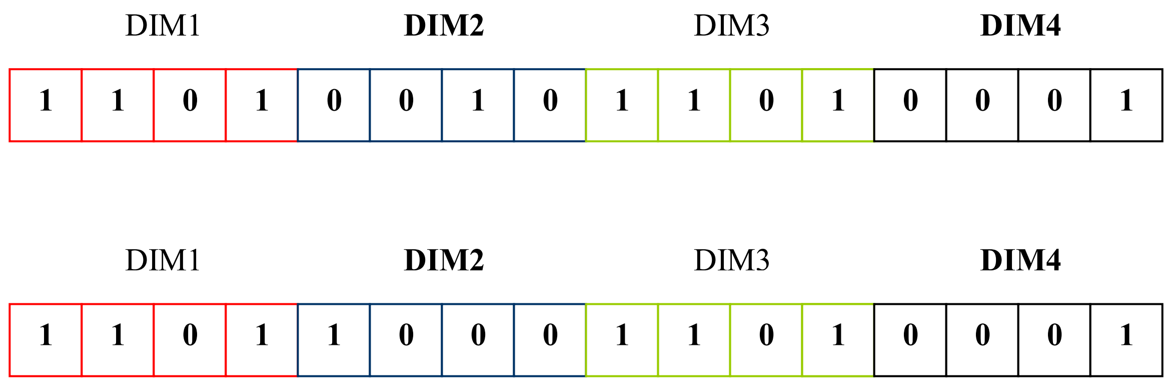

Figure 4.2. DIMM, inverts the bits which are located in dimension 2 by probability of 0.5

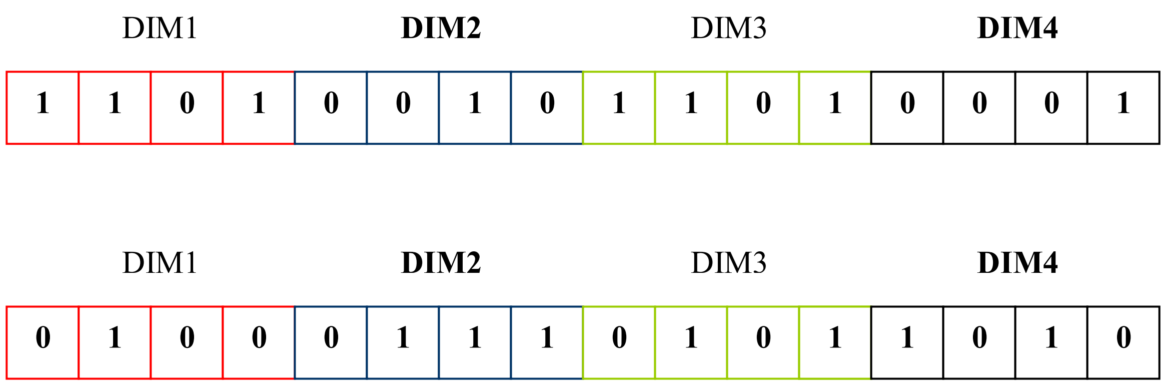

Figure 4.3. HYPM, inverts each bits which are within a candidate solution by the probability of 0.5

4.3. Experimental Settings

We experimented IDWalk, CIDWalk, EXIDWalk and TFA with a single mutational heuristic and a heuristic set which includes three mutational heuristics that are SWPD, DIMM and HYPM. For IDWalk and CIDWalk, we decided on some values of *Max* to find the best *Max* value, the values for max number of walks are 5, 10, 20 ,30, 40 and 50. For IDWalk with SO, EXIDWalk, we used a different *Max* values which are 3, 6, 9, 18, 27, 36, 45 to work with SO system for a fair selection mechanism among three mutational heuristics.

Table 4.3. MN means Max for IDWalk and N = {5,10,20,30,40,50} for parameter tuning. For each Max value, we used 3 different selection mechanism from candidate list. 3M_CF states that all of three mutational heuristics were used in heuristic set under Choice Function selection mechanism

	MAX	M_N	M_N	M_N
	Sets	ANY	BEST	TOUR
IDW	*HYPM*	+	+	+
	DIMM	+	+	+
	SWPD	+	+	+
	3M_CF	+	+	+
CIDW	*HYPM*	+	+	+
	DIMM	+	+	+
	SWPD	+	+	+
	3M_CF	+	+	+

Framework	F_A	F_A	F_A

Table 4.4. Set of experiments for traditional framework F_A

	Sets	
THH	*HYPM*	+
	DIMM	+
	SWPD	+
	3M_CF	+
	Framework	F_A

Table 4.5. MN means Max for IDWalk and N = {3,6,9,18,27,36,45} for parameter tuning. For each Max value, we used 3 different selection mechanism from candidate list. 3M_CF states that all of three mutational heuristics were used in heuristic set under Choice Function selection mechanism.

	MAX	M_N	M_N	M_N
	Sets	ANY	BEST	TOUR
EXIDW	*3M with SO*	+	+	+
	Framework	F_A	F_A	F_A

4.4. Experimental Results

We used *success rate* that is the ratio of successful runs to all runs as the first performance criteria. Experimental results were listed in Table 4.6 whole. IDW really works well with success rate of 0.80 when we look at the rest. The second place is of CIDW, it has a similar performance to IDW and it provides some improvement on F4 from 0.04 to 0.06, but it has a worse performance on F8, F10 and F13. For THH, it performs as the worst hyperheuristics with EXIDW, it only improves F10 from 0.98 of CIDW to 1.00 and EXIDW has only one improvement on F5 from 0.94 (*THH*) to 1.00. In Table 4.7, there are another bunch of data which includes T-Test results of given algorithms, so, we can easily say that there is no a statistically significant difference between the approaches.

.

Table 4.6. The success rate of each hyperheuristic system for each benchmark function with the best results

Label	*IDW*	*CIDW*	*THH*	*EXIDW*
F1	1.00	1.00	1.00	1.00
F2	0.00	0.00	0.00	0.00
F3	1.00	1.00	1.00	1.00
F4	0.04	0.06	0.04	0.04
F5	1.00	1.00	0.94	1.00
F6	1.00	1.00	1.00	1.00
F7	1.00	1.00	1.00	1.00
F8	0.14	0.04	0.00	0.00
F9	1.00	1.00	1.00	1.00
F10	1.00	0.98	1.00	1.00
F11	1.00	1.00	1.00	0.86
F12	1.00	1.00	1.00	1.00
F13	1.00	0.64	0.00	0.00
F14	1.00	1.00	1.00	1.00
Avr.	0.80	0.76	0.71	0.71

Table 4.7. T-Test results that denotes whether there exist statistically significant difference between given hyperheuristics strategies for the best results

HYPERHEURISTICS				
IDW	CIDW	EXIDW	THH	*TTEST*
+	+			2.33E-01
+		+		2.50E-01
+			+	2.21E-01
	+	+		2.66E-01
	+		+	2.24E-01
		+	+	6.18E-01

It was the whole view of the picture, but we must also look at the result for each specific mutational heuristics from Table 4.8 For HYPM, we can easily see that IDW performs better for the mutation. What about DIMM, we have an interesting result for dimensional mutation, since now, CIDW takes the first place. The reason is the same as the reason of using CIDW. That is, for DIMM, we can reach the fact that it can perform really bad moves that cause to go very back and it does not find enough time to reach to the desired global optima. For SWPD, we see the worst result the can be seen, it could not find even one global optima for all the hyperheuristics, its success rate is exactly zero. Actually, the performance of the algorithms on SWPD is not surprising, because, we are aware of that SWPD has a very small search space. For p-values of T-Test that are provided in Table 4.9 show that there is no big difference in statistical manner for 95 confidence level.

Table 4.8. The success rate of each hyperheuristic system for each benchmark function with the highest rate among all the combinations of number of Max neighbors and candidate list selection mechanism on the based of used mutational heuristic

		IDW			*CIDW*			*THH*	
Label	*HYPM*	*DIMM*	*SWPD*	*HYPM*	*DIMM*	*SWPD*	*HYPM*	*DIMM*	*SWPD*
F1	1.00	0.00	0.00	1.00	1.00	0.00	1.00	1.00	0.00
F2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
F3	1.00	1.00	0.00	1.00	1.00	0.00	1.00	1.00	0.00
F4	0.02	0.04	0.00	0.06	0.04	0.00	0.00	0.04	0.00
F5	1.00	1.00	0.00	1.00	1.00	0.00	0.94	0.50	0.00
F6	1.00	0.00	0.00	1.00	1.00	0.00	1.00	1.00	0.00
F7	1.00	0.02	0.00	1.00	0.02	0.00	1.00	1.00	0.00
F8	0.14	0.00	0.00	0.04	0.00	0.00	0.00	0.00	0.00
F9	1.00	0.00	0.00	1.00	1.00	0.00	1.00	1.00	0.00
F10	1.00	0.08	0.00	0.98	0.04	0.00	0.98	0.02	0.00
F11	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
F12	1.00	1.00	0.00	1.00	1.00	0.00	1.00	1.00	0.00
F13	1.00	0.00	0.00	0.64	0.00	0.00	0.00	0.00	0.00
F14	1.00	1.00	0.00	0.92	0.62	0.00	0.86	0.00	0.00
Avr.	0.80	0.30	0.00	0.76	0.48	0.00	0.70	0.47	0.00

Table 4.9. T-Test results that denotes whether there exist statistically significant difference between given hyperheuristics strategies for the best results (For SWPD, we did not look their T-Test results because all the success rates of it is 0 (zero))

HYPERHEURISTICS						
IDW		CIDW		THH		
HYPM	*DIMM*	*HYPM*	*DIMM*	*HYPM*	*DIMM*	*TTEST*
+		+				1.86E-01
+				+		1.86E-01
		+		+		1.96E-01
	+		+			1.52E-01
	+				+	3.06E-01
			+		+	9.05E-01

For 3M_CF case, the result shows that there are no any significant difference between the hyperheuristics frameworks, there are some small improvements according to the rest. For F1, CIDW and THH have full success, but IDW's is 94%. For F4, IDW and CIDW performed at success rate of 0.02, and THH at 0.00. For F4, F7, THH performs worse, too. From T-Test perspective with data in Table 4.11, just for CIDW and THH, there is no statistically significant diffence in 95 percent confidence level.

Table 4.10. The success rate of 3M_CF used hyperheuristic framework for each benchmark function with the highest rate among all the combinations of number of Max neighbors and candidate list selection mechanism on the based of used mutational heuristic

Label	*IDW*	*CIDW*	*THH*
F1	0.94	1.00	1.00
F2	0.00	0.00	0.00
F3	1.00	1.00	1.00
F4	0.02	0.02	0.00
F5	1.00	1.00	0.92
F6	1.00	1.00	1.00
F7	0.30	0.22	0.00
F8	0.00	0.02	0.00
F9	1.00	1.00	1.00
F10	0.98	1.00	1.00
F11	0.00	0.04	0.00
F12	1.00	1.00	1.00
F13	0.00	0.00	0.00
F14	1.00	1.00	1.00
Avr.	0.59	0.59	0.57

Table 4.11. T-Test results that denotes whether there exist statistically significant difference between given hyperheuristics strategies for the best results with 3M_CF

HYPERHEURISTICS			
IDW	CIDW	THH	*TTEST*
+	+		6.09E-01
+		+	3.32E-01
	+	+	1.15E-01

For the experiments which were performed on multiple heuristics, EXIDW approach reachs the best results than the rest. As it is provided in Table 4.12, we have improvements on F1, F4, F5, F7, F10, F11 according to the worst results. It has 71% success rate, the nearest hyperheuristic has a success rate of 59%, this is really nice improvement. By looking T-Test resuls, we can say that EXIDWalk is the best strategy among all, but it is not in 90 percent confidence level.

Table 4.12. Extended results with the additional hyperheuristic of EXIDWalk to the Table 4.10

Label	*IDW*	*CIDW*	*THH*	*EXIDW*
F1	0.94	1.00	1.00	1.00
F2	0.00	0.00	0.00	0.00
F3	1.00	1.00	1.00	1.00
F4	0.02	0.02	0.00	0.04
F5	1.00	1.00	0.92	1.00
F6	1.00	1.00	1.00	1.00
F7	0.30	0.22	0.00	1.00
F8	0.00	0.02	0.00	0.00
F9	1.00	1.00	1.00	1.00
F10	0.98	1.00	1.00	1.00
F11	0.00	0.04	0.00	0.86
F12	1.00	1.00	1.00	1.00
F13	0.00	0.00	0.00	0.00
F14	1.00	1.00	1.00	1.00
Avr.	0.59	0.59	0.57	0.71

Table 4. 13. T-Test results that denotes whether there exist statistically significant difference between given hyperheuristics strategies for the best results

HYPERHEURISTICS				
IDW	CIDW	THH	EXIDW	*TTEST*
+			+	1.40E-01
	+		+	1.65E-01
		+	+	1.39E-01

In Table 4.14, we have *Max* number of walk for each benchmark function on the based of IDW and CIDW. The results give some idea about ideal number of walks for each mutational heuristics as *Max*. According to the table, we understood that some function and mutation combinations have an obvious upper limit for *Max* number of walks. For instance, F1+HYPM can have at most 24 walks, for F2+HYPM, it is 33 walks etc. Actually, we have another important information for characteristics of used mutational heuristics for the functions. For example, if we look at the F3+HYPM combination, we can see that the mutation reach a better candidate at less step, but it does not show it finds the global optima, since there are some experimental results about SWPD, for the same function, its *Max* number of walks is 1 and it means it does not need an approach like IDWalk. The results seems great, but it could not find any global optima, then we can understand that the rate of improvement on SWPD is very small. However, it does not mean, SWPD is rubbish, we can use it with a Hill Climber to reach a desired point.

Table 4.14. *Max* number of steps during walk for IDW and CIDW.

	IDW			*CIDW*		
Label	*HYPM*	*DIMM*	*SWPD*	*HYPM*	*DIMM*	*SWPD*
F1	24	50	50	24	50	50
F2	33	50	50	33	50	50
F3	7	11	1	7	11	1
F4	50	50	50	50	50	50
F5	31	50	50	31	50	50
F6	35	50	50	35	50	50
F7	32	50	50	32	50	50
F8	37	50	50	37	50	50
F9	25	50	50	25	50	50
F10	32	50	1	32	50	1
F11	35	50	50	35	50	50
F12	21	50	1	21	50	1
F13	43	50	1	43	50	1
F14	39	50	1	39	50	1

SR can be thought as the main critearia for analyzing the performance of the used algorithms. However, there are some additional performance measurement elements and their analytical resuslts are provided in Appendix A.

One of the other mechanisms for performance analysis is *average execution time (sec.).* From this perspective, there is no big difference, as it can be seen in Table A1 and Figure A1. However, in general, the execution time of IDWalk is smaller than the rest on

HYPM and CIDWalk follows it. The circumstance for DIMM is not far away them HYPM's, that is, there is no huge distinction for *average execution time (sec.)* on DIMM. But, now, the situation changes, CIDWalk performs in less time and IDWalk as the worst from general view (Table A2, Figure A2). For SWPD, Table A3 and Figure A3 shows that CIDWalk executes faster than the rest, but it is so close to IDWalk. Actually, THH is also close, but it seems the worst one among the all.

As the third criteria, we can use *average hyperheuristics call per execution.* It shows us, called number of hyperheuristics during execution process for optimization on average basis and related table, Table A4, A5, A6, and graph, Figure A4, A5, A6 give a chance to compare the results, respectively for each mutational heuristics that are HYPM, DIMM and SWPD. So, for the first heuristic, number of hyperheuristics call of THH is less than the rest and CIDWalk is the nearest approach to the best. There exist some differences and similarities, the situation changes according to the benchmark function that we tested on. For F1, F2, F3, F4, F6, F8 and F10, there is no huge differences and sometimes they show the same characteristic for average calling. On the other hand, for F5, F7, F9, F11, F12, F13 and F14, there exist big gaps between the approaches. For DIMM, CIDWalk is the best for average case. However, differences between mechanisms are so distinct for some functions such as F1, F5, F6, F7, F9 and F14, the rest is balanced. For SWPD, the algorithms behave similary except some functions which are F2, F4, F5, F8 and F11, the rest is almost the same. For the functions, we have the same characteristic for IDWalk and CIDWalk, but THH does not show the same movements, but its average hyperheuristics calls per execution are the same for all the functions.

For the following criteria, we can use *average fitness evaluations per execution.* In Table A7 and Figure A7, results are provided for HYPM heuristic. There exist huge differences on the benchmark problems except F2, F4 and F8. For average of all functions on the based of given algorithms, the best is IDWalk. For DIMM, from Table A8 and Figure A8, the benchmark functions which are F2, F4, F8, F11 and F13 have simliar characteristics, but the rest is changeable according to the tested approach. As general, THH perfoms less fitness evaluations than the others. As it can be seen easily from Table A9 and Figure A9, F2, F4, F5, F8 and F11 behaves in different ways for the algorithms, but the rest is almost the same on SWPD. For all the functions, THH performs the same number of fitness evaluations. Except F5, IDWalk and CIDWalk has the same characteristics for all the functions as a couple.

Average fitness at the end is another property to compare the results. In the graphs, Figure A10, A11 and A12, I provided another item which is global optimum value of a tested mathematical benchmark function. From Figure A10, we can see the average fitness values at the end of execution, that is, which one goes around the optimum, which one reaches or go to near to a global optima by improving very bad candidate to a suitable one questions can be answered easily just by looking the graph. The ones who are near to the global optimum are the approaches that provide opportunity to walk around the point, but the others can be thought as slowly improving ones or having bad fitness value during the optimization process. For most of the times, the algorithms gave the same average fitness value at the end, but there are some differences. On HYPM, except F2, F13 and F14, tests on the benchmark functions give similar solutions for proposed hyperheuristics, but, for the whole picture, IDWalk seems to the best approach. For DIMM, IDWalk's behaviour is not the same as on HYPM, now, it is the worst one among all as it is given in Table A11, and the best is THH. The situation for IDWalk is easily seen from Figure A11.

The other criteria is *best fitness at the end*. The results are provided in Table A12, A13, A14 and related graphs are Figure A 12, A13, A14 for HYPM, DIMM and SWPD respectively; also, global optimum values for given benchmark functions are listed in the tables and added to the graphs. For HYPM heuristic, now, IDWalk is the best approach and THH is the worst. The biggest different is on F13, IDWalk and CIDWalk is so smaller than THH. With mutational heuristic DIMM, for F11 and F13, IDWalk performs really bad, and for F14 there exit some differences on F7 and F14. With SWPD, results almost the same.

5. AN IDWALK BASED HYPERHEURISTIC FOR EXAMINATION TIMETABLING

5.1. Timetabling

Timetabling is a NP-complete [19], *constraint optimization* problem which cannot be solved in polynomial time and has not an exact optimum solution. Also, it can be defined as "*Timetabling is the allocation, subject to constraints, of given resources to objects being placed in space time, in such a way as to satisfy as nearly as possible a set of desirable objectives* [20]". It can be divided into three main parts that are sets of <V,D,C>.

$$V=\{v_1, v_2, \ldots, v_M\}$$
$$D=\{d_1, d_2, \ldots, d_M\}$$
$$C=\{c_1, c_2, \ldots, c_K\}$$

Among these sets, V represents *variables'* set, and, D set is related to V, it includes *domains of variables*. In timetabling problems, D sets must contain at least one element, if it is exactly one, then we can easily say that it should be *time.* C is another set which depends on V, *constraints* that belongs to the members of V are listed, here. These constraints can be examined in two different groups which are *hard* and *soft* constraints. As it can be understood from their names, hard constrains have an obligation and privilege to be done, the rest, soft constraints, may be handled according to what you want to have.

Many researchers who are from different fields of science and engineering disciplines, have been working on this timetabling subject. They have proposed extremely different ideas and approaches about it. Some of them reached nice solutions on a given specific data, but this is not the aim of the researchers. They want to provide a strategy that can solve all the types of timetabling problems which owns some constraints and priorities on them with a feasible result in a reasonable time. So, they divided this diverse subject into sub-problems such as course timetabling [20], examination timetabling [20], nurse rostering [21], sports timetabling [22] etc. First studies about timetabling are available in the article of Schmidt [23].

5.1.1 Examination Timetabling

Examination timetabling can be defined as the scheduling for the exams of a set of university courses, avoiding overlap of exams of courses having common students, and spreading the exams for the students as much as possible [20]. This sub-problem is one of the most used domain that has been trying to be solved via different kinds of techniques. For instance, Carter et al. [24] applied Different Heuristic Orderings based on Graph Coloring, since, Leighton [25] showed that the timetabling problem can be reduced to the graph coloring problem and Carter provided some widely used benchmark data to analyze any timetabling optimization algorithms. Burke et al. [26, 27] applied a light or a heavy mutation, randomly selecting one mutation operator which is followed by a hill-climbing among them. There exist also some Constraint Satisfaction Strategies with Genetic Algorithms for solving examination timetabling problems were provided in [28]. Paquete et al. [29] applied a Multiobjective Evolutionary Algorithm (MOEA) based on a direct encoding of the mapping between exams and time slots is used to minimize the number of violations of each type of constraints as separate objectives. Wong et al [30] used a Genetic Algorithm utilizing a Non-Elitist Replacement Strategy to solve a single examination timetabling problem at École de Technologie Supérieure. First of all, genetic operators were applied, then, violations were fixed in a hill-climbing approach. Gaspero and Schaerf [31] tested some Tabu Search based algorithms which import several different features from the research about Graph Coloring. Merlot et al. [32] proposed a new hybrid algorithm which involves three phases of programming, simulated annealing and hill-climbing. Petrovic et al. [33] introduced a Case Based Reasoning System to create initial solutions to be used by Great Deluge Algorithm. Burke et al. [34] 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 [35] 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 Ozcan in [36].

We performed our hyperheuristic mechanism just on examination timetabling problem to see the characteristic of EXIDWalk. We used a special approach with the parameters, vectors and the matrices involved in the problem are provided in the Formula (4.1), so, by these representations, we have the terminology to be solve the given examination timetabling problems. From the nature of timetabling problems, they have some constraints and need to be optimized, so, they are called as *Constraint Optimization Problem*. Then, we decided on some constraints for examination timetabling problem, the first constraint is stated in the Formula (4.2) and it says that each exam should be scheduled just for once. Another constraint of the student exam clush remarked in Formula (4.3) and capacity constraint for each slot is formulated in Formula (4.4). In addition, the constraint that between two exams, there must be at least one empty slot is expressed in Formula (4.5) and the number of constraint violations are used to calculate the evaluation function in the Formula (4.6). In the 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).

$$\begin{gathered}
M = \text{number of slots} \\
N = \text{number of exams} \\
C = \text{total capacity for a slot} \\
a_{ij} = \begin{cases} 1 & \text{if } j^{th} \text{ exam is on } i^{th} \text{ slot} \\ 0 & \text{else} \end{cases} \\
b_j = \text{number of students taking exam } j \\
c_{jk} = \text{number of students taking both exams } j \text{ and } k
\end{gathered} \tag{4.1}$$

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

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

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

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

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

5.2. Heuristics for Examination Timetabling

We used four different heuristics, called TOURTARGETCONF1, TOURTARGETCONF2, TOURTARGETCAP and RANDOMASSIGN, which have an aim of preventing some speficic types of conflicts that are provided in above formulas. Three of these heuristics use the mechanism of tournament selection and assignment methods to improve the candidate solution, and the forth one is a mutation based heuristic. For related heuristics to (4.3) and (4.5), they choose a bunch of exams and then take the exam with the highest number of targeted conflict, after that, some number of time slots are selected and assigns the exam to the time slot that has minimum number of conflicts. For capacity conflict, related heuristic chooses some number of time slots and then takes the time slot that has maximum number of conflicts. A predetermined number of exams are chosen among the exams which are available in the selected time slot and the exam that has the highest number of students. Then, it chooses some time slots in a random way and the time slot which have minimum number of attendants is chosen as the assigned slot for selected exam. Mutational heuristic, assings each exam into a randomly chosen time slot with the probability of $(1/\textit{number of courses})$.

5.3. Experimental Data

Our one of proposed hyperheuristic approach which is EXIDWalk is tested on Carter's Benchmarks and the examination timetabling problem data of Yeditepe University, Faculty of Engineering. Properties and parameters for each problem instance are presented in the Table 5.1. Three slots are allocated for each day.

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

Table 5.1. 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	190	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	181	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

5.4. Experimental Results

We used EXIDWalk on each benchmark timetabling data for 50 times. The average best fitness reached is used as the performance criterion for all experiments. The results for each timetabling benchmark data are presented in Table 5.2, Table 5.3 and Figure 5.6. We added another parameter which is standard deviation to see the exact picture, that is, the first column is the average value for the given performance criterion and the second column 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 the comparison of EXIDWalk approach and previous results that are provided on the same benchmark data by Bilgin et al. [33], so, we can see which algorithm performs better than the others on Car-f-92, Car-s-91, Ear-f-83, Hec-s-92, Kfu-s-93, Lse-f-91, Pur-s-93, Rye-s-93, Sta-f-83, Tre-s-92, Uta-s-92, Ute-s-91, Yor-f-83, Yue20011, Yue20012, Yue20013, Yue20021, Yue20022, Yue20023, Yue20031 and Yue20032. By the related tables and graph, we had a t-test result with p-value of 9.62E-02, then we can easily say that, the best algorithm that iz provided in [33] is significantly better than our approach. However, there are some benchmark data such as Car-f-92, Ear-f-83, Rye-s-93 and Uta-s-92 and in these domains, our EXIDWalk performs better than the previously used algorithms. In addition, related results about comparisons of average best fitness values are provided in Appendix B.

Table 5.2. Average best fitness values for the best performing heuristic selection method and acceptance criterion combinations on each problem instance from hyperheuristics frameworks [2]

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 5.3. Average best fitness values for the best performing heuristic selection method and acceptance criterion combinations on each problem instance with EXIDWalk.

Instance	Average Best Fitness	Standard Deviation
Carf92	-1.29E-02	1.09E-03
Cars91	-1.01E-01	3.30E-02
Earf83	-7.51E-03	6.44E-04
Hecs92	-2.19E-02	3.86E-03
Kfus93	-2.82E-02	4.93E-03
Lsef91	-1.26E-02	1.24E-03
Purs93	-1.38E-03	6.08E-05

Ryes93	-1.09E-02	1.66E-03
Staf83	-2.65E-03	5.27E-05
Tres92	-6.72E-02	1.39E-02
Utas92	-2.03E-02	1.72E-03
Utes92	-1.99E-03	1.30E-04
Yorf83	-8.15E-03	4.07E-04
Yue20011	-7.67E-02	1.31E-02
Yue20012	-6.50E-02	9.14E-03
Yue20013	-2.13E-01	2.62E-02
Yue20021	-2.82E-02	5.35E-03
Yue2022	-1.08E-02	1.62E-03
Yue20023	-1.44E-02	5.06E-04
Yue20031	-1.44E-02	2.24E-03
Yue20032	-4.29E-03	5.01E-04

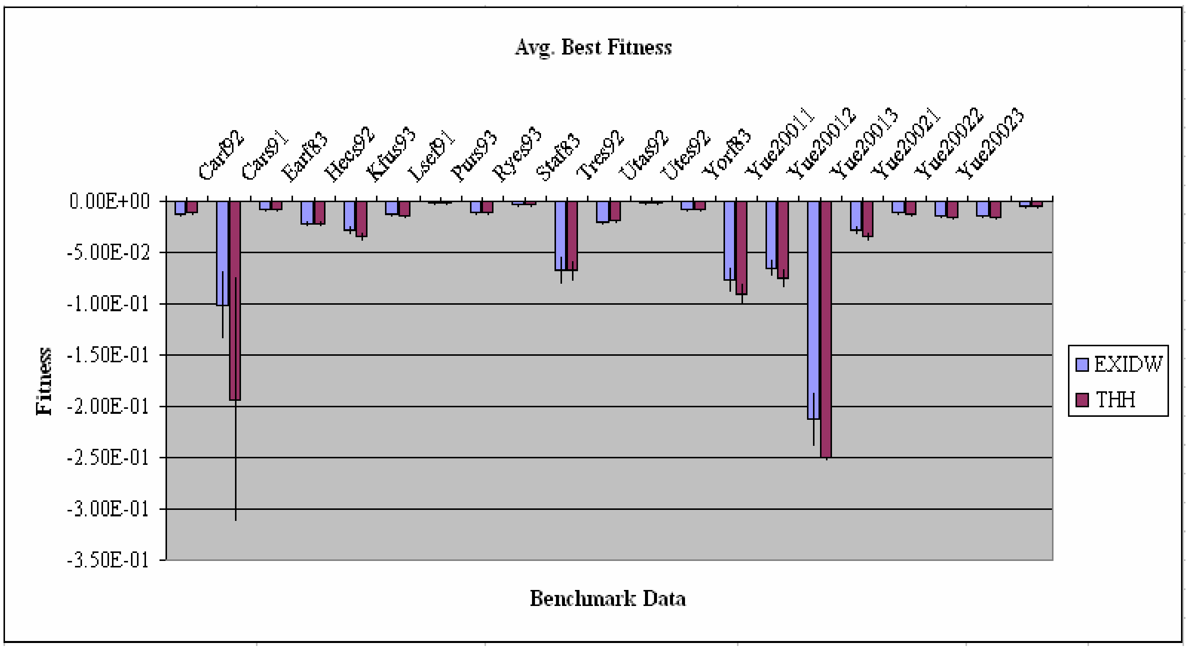

Figure 5.1. Graphical view of average best fitness at the end with starndart deviations for tested algorithms (IDW, CIDW, EXIDW and THH) on a heuristic set which has HYPM, DIMM and SWPD

There exist another interesting results about EXIDWalk on the based of RANK test of MS Excel. The related table and graph, Table 5.4 and Figure 5.2, show the ranking of each hyperheuristic approach on a tested benchmark data, so, we can easily see the order of success for a given timetabling data such as for Carf92 the order involves EXIDW, TABU_IE, CF_MC, SR_IE, SR_GD and SR_MC, consecutively. This is not the interesting one of the results, interesting one comes from average rankings as whole. So, we can see that EXIDW and CF_MC performs similar, but the rest is worse than both. This conclusion expresses that EXIDWalk is a promising approach.

Table 5.4. Examination timetabling results of the hyperheuristics which provide at least one the best solution on testted set of data for examination timetabling

Instance	EXIDW	TABU_IE	CF_MC	SR_GD	SR_IE	SR_MC
Carf92	1.0	2.5	2.5	5.0	4.0	6.0
Cars91	3.0	1.0	5.0	2.0	4.0	6.0
Earf83	1.0	6.0	2.0	4.0	5.0	3.0
Hecs92	1.0	5.0	2.0	4.0	6.0	3.0
Kfus93	2.0	5.0	3.0	1.0	4.0	6.0
Lsef91	2.0	6.0	1.0	3.0	5.0	4.0
Purs93	2.0	3.0	5.0	4.0	1.0	6.0
Ryes93	1.0	5.0	2.0	3.0	4.0	6.0
Staf83	4.0	6.0	2.0	2.0	5.0	2.0
Tres92	2.0	4.0	3.0	1.0	5.0	6.0
Utas92	1.0	2.0	4.0	5.0	3.0	6.0
Utes92	2.0	6.0	1.0	4.0	5.0	3.0
Yorf83	2.0	5.0	1.0	4.0	6.0	3.0
Yue20011	2.0	6.0	3.0	1.0	5.0	4.0
Yue20012	2.0	6.0	3.0	1.0	5.0	4.0
Yue20013	4.0	6.0	1.5	3.0	5.0	1.5
Yue20021	3.0	6.0	2.0	1.0	5.0	4.0
Yue20022	4.0	6.0	1.0	2.5	5.0	2.5
Yue20023	3.0	6.0	1.0	4.0	5.0	2.0

Yue20031	2.0	6.0	1.0	3.0	5.0	4.0
Yue20032	3.0	6.0	1.0	4.0	5.0	2.0
AVG.	**2.2**	**5.0**	**2.2**	**2.9**	**4.6**	**4.0**

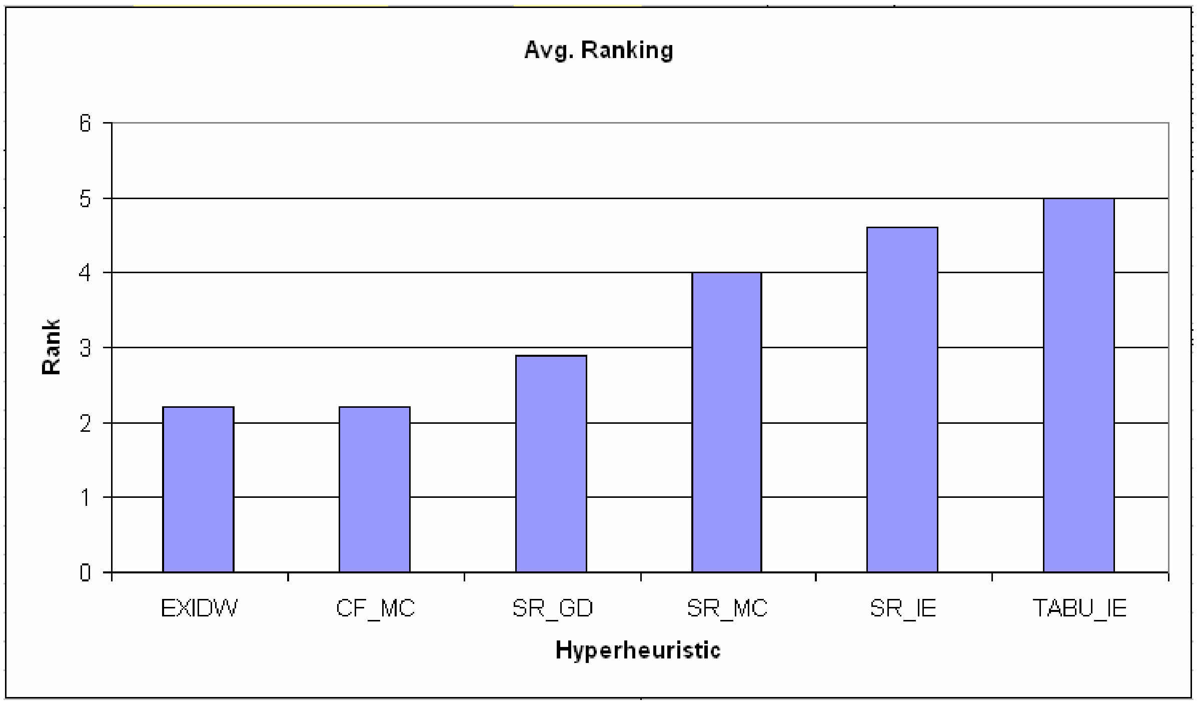

Figure 5.2. Average rankings of the hyperheuristics which provide at least one the best solution on testted set of data for examination timetabling

6. CONCLUSIONS AND FUTURE WORKS

Nowadays, many researchers have been working on hyperheuristics concept and they propose new approaches to solve optimization problems. There were two main mechanisms which are heuristic selection and acceptance to reach a powerful hyperheuristic system, so, very different combinations of the mechanisms have been tested. This is nice, but there must be some other systems to improve the performance of presented algorithms. Then, we faced to face with IDWalk which involves *Intensiication* and *Diversification* devices with a candidate list strategy and embedded it to the traditional hyperheuristic structure. So, we got a new hyperheuristic called IDWalk. Actually, *Intensification* is already alive in the current hyperheuristics with hill climbers and acceptance mechanisms, and Diversification is also available with mutational heuristics. We did not changed the idea of acceptance, but we fixed it into IE which is the acceptance mechanism of IDWalk during a walk. However, we removed out hill climbers, since, it decreases the possibility of having rejected candidates. If we used it, we could not see the power of candidate list strategy for rejected candidates. For *Diversification*, we used three different selection mechanisms that are ANY, BEST and TOUR to choose a rejected candidate not to be stuck at a local optima from the list which have *Max* number of members.

We used IDWalk as a new hyperheuristic, but this was not the whole idea that we proposed. We tried to improve it and then reached two new IDWalk based approaches which are CIDWalk and EXIDWalk. CIDWalk has a aim to prevent adding a very bad candidate who has bad fitness value on the based of given limit and EXIDWalk provides an opportunity of extended neighborhood during a walk.

We tested IDWalk, CIDWalk and EXIDWalk with different *Max* numbers and rejected candidate selection mechanisms to see the effect of the *Max,* the selection and their different combinations during optimization on 14 well-known benchmark functions and 21 examination timetabling data as real-world problems.

For the whole picture with benchmark problems, we reached to a conclusion that IDWalk is the best approach and CIDWalk follows it. On the other hand, THH and EXIDWalk has similar performances, but worse than the others. However, this situation comes from the performance of HYPM. It surprisingly works better than FA and FC frameworks [2] on its own, it performs almost the same with the best approach that is presented in [2]. If we compare IDWalk with THH, we see a dramatically increasing performace from 47% to 80%. For a heuristics set that includes just one heuristic which is DIMM, we encountered that CIDWalk has improved result by 18% of increasing performance. These are nice results, but what about just for tests that were done with multiple heuristics? We reached another interesting result that says that EXIDWalk is the best approach among all. As a statistical data, we saw that IDWalk and CIDWalk performs almost the same with success rate of 59%, THH is similar with 59% success, on the other hand, EXIDWalk has a success rate of 71%. So, we reached to a fact that EXIDWalk is the best mechanism with heuristic sets. Then, we tested it on some benchmark data of examination timetabling.

For experimental results on the examination timetabling problem, we tested just EXIDWalk with four heuristics and compared the results to the traditional hyperheuristics. We saw that EXIDWalk performs better for some problems which are Car-f-92, Ear-f-83, Rye-s-93 and Uta-s-92, but for the rest, the traditional approach is better. However, this is not the whole story, if we look at the ranking analysis of tested strategies, the similarity between EXIDWalk and the best structure which is among traditional hyperheuristics can be seen, their average ranking values are almost the same.

According to the experimental results on both the benchmark functions and examination timetabling problem instances, we can say that IDWalk has a future with its improved version called EXIDWalk. For future works, we may provide a smart and dynamic system that decides on *Max* and rejected candidate selection mechanisms and we can test it with more mutational heuristics. Also, CEXIDWalk (*Constrained and Extended IDWalk*) can be tested on the same problems as a new mechanism. To see the performace of the proposed approaches on the other problems, we may experiment our algorithms on an antother widely used problem called *Stock Cutting* in industry as a new problem domain.

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

Results for a detailed analysis on given benchmark functions are provided in Table A. 1 - Table A. 18. All the characteristics which are Average Execution Time, Average Hyperheuristics call per Execution, Average Fitness Evaluation per Execution, Average Fitness at the End, Average Best Fitness at the End can be seen from the presented tables. Also, related bar charts of tested hyperheuristics mechanisms are settled just after the table on the based of their behaviours. All of the properties are shown for four times, since I used four different mutational heuristic sets. First three sets are sets of just one heuristic, HYPM, DIMM and SWPD, the rest is the combination of these alone heuristics, that is, heuristic sets inlcude multiple heuristics. For heuristic sets which have only one mutational heuristic, I tested with IDWalk, CIDWalk and THH hyperheuristics mechanisms. For the rest, I used EXIDWalk as an extra strategy with IDWalk, CIDWalk and THH, together.

Table A.1. Data table of average execution time (sec.) for tested algorithms (IDW, CIDW and THH) with HYPM

	HYPERHEURISTICS		
Label	*IDW*	*CIDW*	*THH*
F1	3.44E-01	3.49E-01	2.82E-01
F2	5.31E+01	5.59E+01	5.36E+01
F3	9.46E-02	9.90E-02	9.70E-02
F4	5.47E+01	5.45E+01	5.55E+01
F5	7.02E-02	1.88E+00	2.06E+00
F6	5.87E+00	6.60E+00	5.83E+00
F7	6.10E+00	8.40E+00	5.25E+00
F8	5.83E+01	5.95E+01	6.01E+01
F9	3.93E-01	5.45E-01	3.76E-01
F10	9.28E+00	9.62E+00	9.56E+00
F11	3.00E+01	1.97E+01	1.98E+01
F12	1.81E-01	2.25E-01	5.38E-02
F13	7.54E+00	1.38E+01	2.05E+01
F14	1.27E+00	4.17E+00	4.73E+00

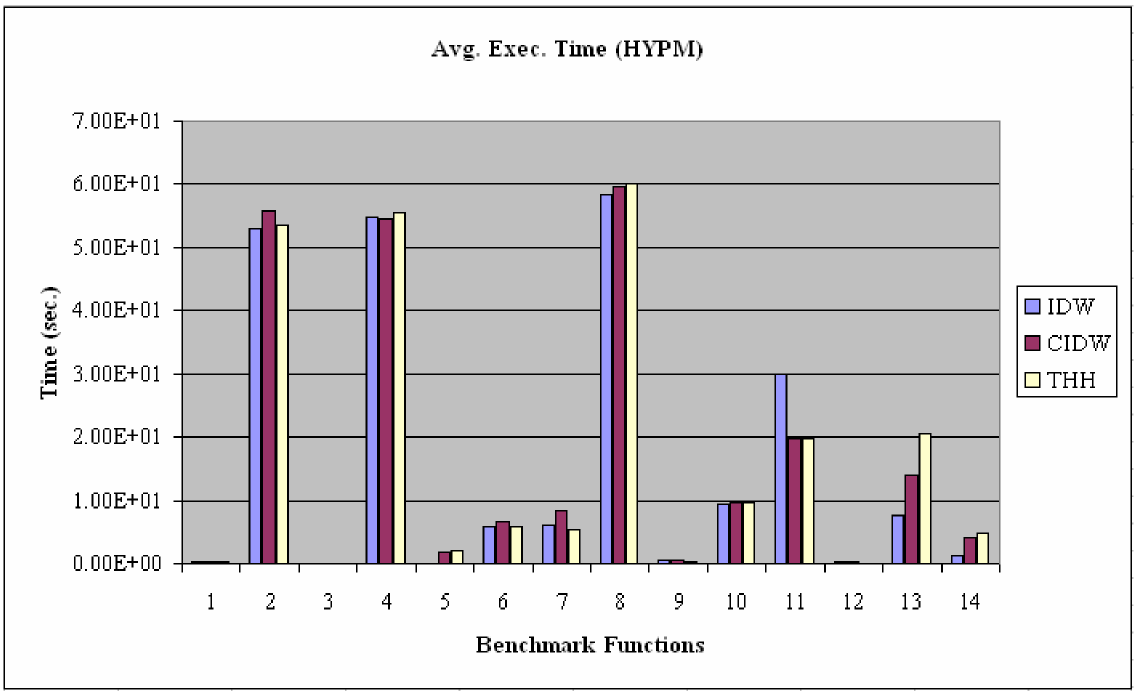

Figure A.1. Graphical view of average execution time (sec.) for tested algorithms (IDW, CIDW and THH) with HYPM

Table A. 2. Data table of average execution time (sec.) for tested algorithms (IDW, CIDW and THH) with DIMM

	HYPERHEURISTICS		
Label	*IDW*	*CIDW*	*THH*
F1	6.42E+00	1.00E+00	4.03E-01
F2	6.73E+00	7.08E+00	8.04E+00
F3	2.42E-02	2.32E-02	2.44E-02
F4	9.39E+00	8.96E+00	1.01E+01
F5	1.15E-01	6.19E-01	7.40E+00
F6	8.80E+00	1.27E+00	5.50E-01
F7	9.52E+00	9.69E+00	6.35E-01
F8	1.34E+01	1.35E+01	1.44E+01
F9	9.85E+00	2.92E+00	6.33E-01
F10	9.99E+00	6.23E+00	7.57E+00
F11	1.59E+01	1.44E+01	1.57E+01
F12	1.60E-02	1.04E-02	1.00E-02
F13	1.31E+01	1.20E+01	1.38E+01
F14	1.23E-01	3.81E+00	5.19E+00

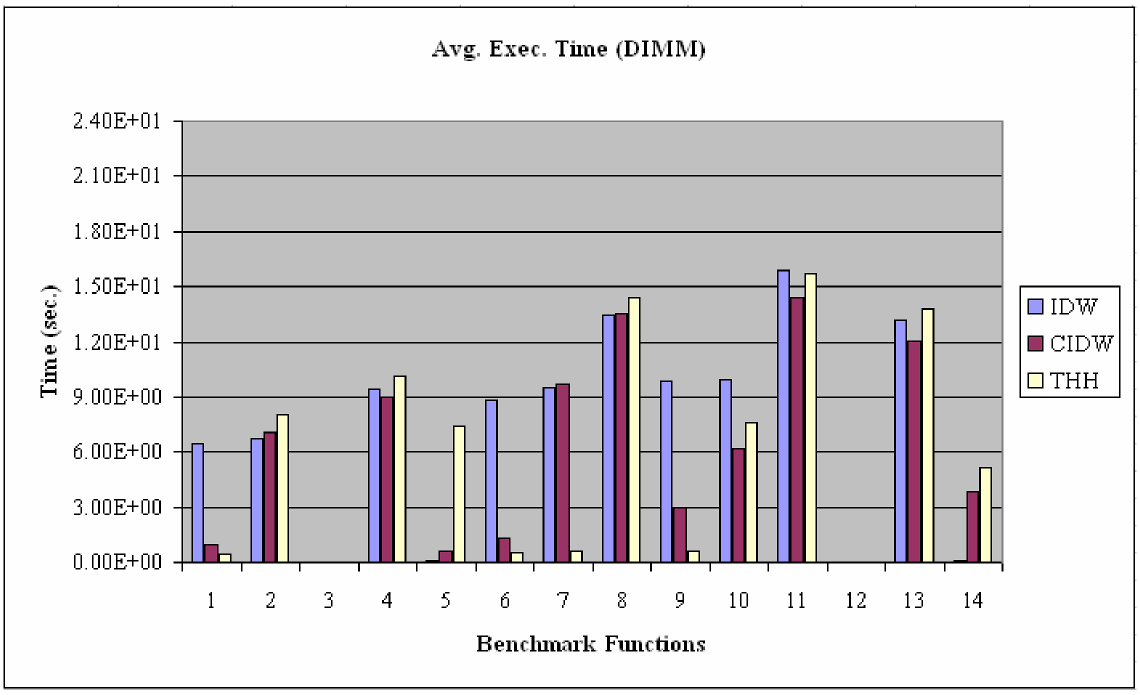

Figure A.2. Graphical view of average execution time (sec.) for tested algorithms (IDW, CIDW and THH) with DIMM

Table A.3. Data table of average execution time (sec.) for tested algorithms (IDW, CIDW and THH) with SWPD

	HYPERHEURISTICS		
Label	*IDW*	*CIDW*	*THH*
F1	4.96E+00	5.05E+00	5.58E+00
F2	4.25E+00	4.42E+00	5.62E+00
F3	8.01E+00	7.85E+00	7.64E+00
F4	6.57E+00	6.97E+00	8.37E+00
F5	1.35E+01	1.38E+01	1.51E+01
F6	7.74E+00	7.63E+00	7.97E+00
F7	8.76E+00	8.65E+00	8.89E+00
F8	1.09E+01	1.11E+01	1.32E+01
F9	9.46E+00	9.32E+00	9.22E+00
F10	8.70E+00	8.42E+00	7.71E+00
F11	1.23E+01	1.23E+01	1.36E+01
F12	4.50E+00	4.30E+00	4.25E+00
F13	7.94E+00	7.76E+00	8.06E+00
F14	4.80E+00	4.67E+00	4.42E+00

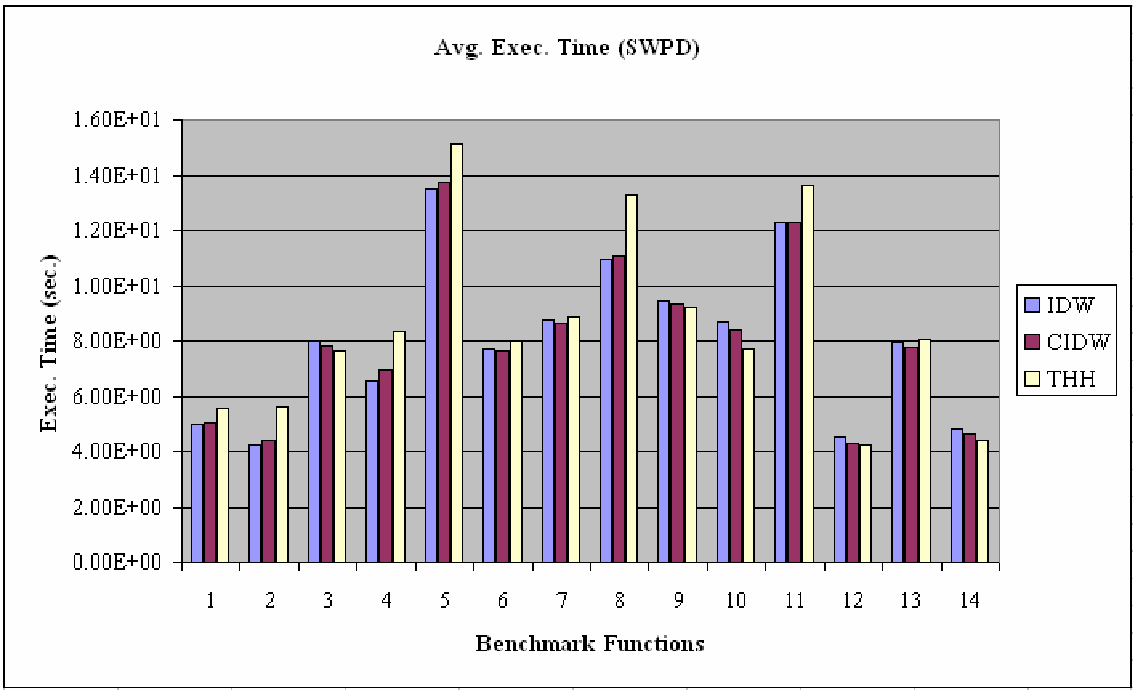

Figure A.3. Graphical view of average execution time (sec.) for tested algorithms (IDW, CIDW and THH) with SWPD

Table A.4. Data table of average hyperheuristics call per execution for tested algorithms (IDW, CIDW and THH) with HYPM

	HYPERHEURISTICS		
Label	*IDW*	*CIDW*	*THH*
F1	1.90E+04	1.99E+04	1.59E+04
F2	3.00E+06	3.00E+06	3.00E+06
F3	5.15E+03	5.43E+03	5.15E+03
F4	3.00E+06	3.00E+06	3.00E+06
F5	8.96E+03	2.40E+05	2.59E+05
F6	3.21E+05	3.35E+05	3.15E+05
F7	3.18E+05	4.49E+05	2.78E+05
F8	2.94E+06	3.00E+06	3.00E+06
F9	2.09E+04	2.90E+04	1.98E+04
F10	7.49E+05	7.92E+05	7.92E+05
F11	1.44E+06	9.77E+05	9.67E+05
F12	4.04E+04	4.95E+04	1.15E+04
F13	1.05E+06	2.15E+06	3.00E+06
F14	5.63E+05	1.83E+06	1.75E+06

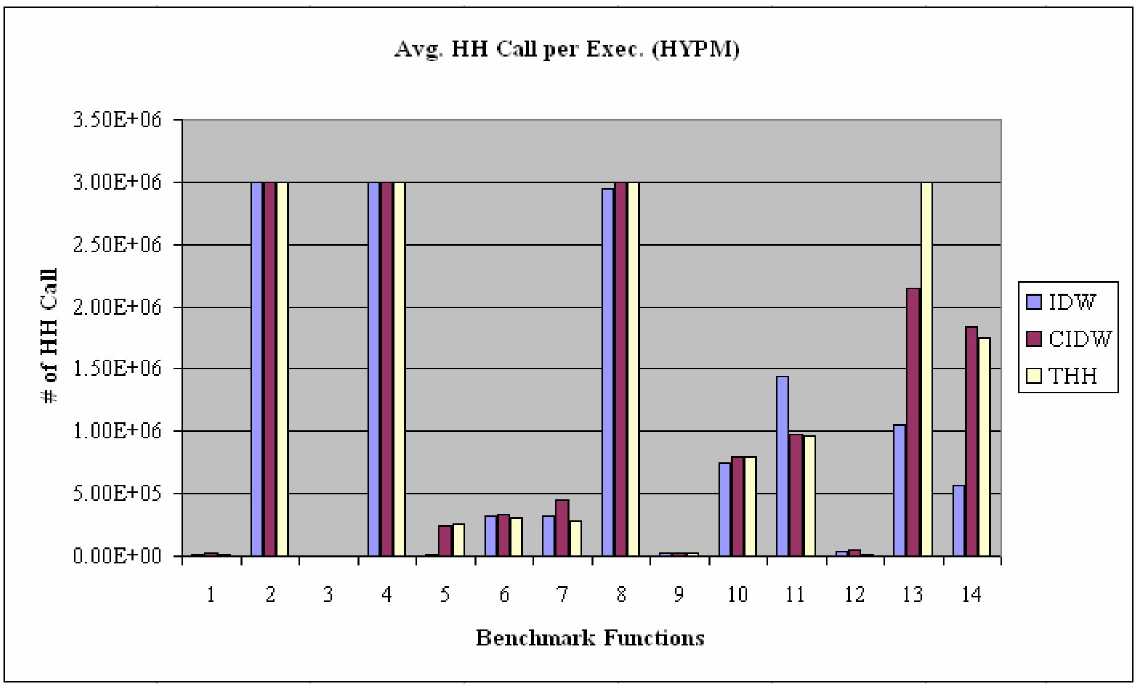

Figure A. 4. Graphical view of average hyperheuristics call per execution for tested algorithms (IDW, CIDW and THH) with HYPM

Table A.5. Data table of average hyperheuristics call per execution for tested algorithms (IDW, CIDW and THH) with DIMM

	HYPERHEURISTICS		
Label	*IDW*	*CIDW*	*THH*
F1	3.00E+06	4.95E+05	1.59E+05
F2	3.00E+06	3.00E+06	3.00E+06
F3	7.51E+03	7.51E+03	7.51E+03
F4	2.95E+06	2.99E+06	2.99E+06
F5	2.48E+04	1.26E+05	1.50E+06
F6	3.00E+06	4.59E+05	1.68E+05
F7	3.00E+06	3.00E+06	1.73E+05
F8	3.00E+06	3.00E+06	3.00E+06
F9	3.00E+06	9.17E+05	1.72E+05
F10	3.00E+06	2.94E+06	2.95E+06
F11	3.00E+06	3.00E+06	3.00E+06
F12	9.80E+03	6.25E+03	5.24E+03
F13	3.00E+06	3.00E+06	3.00E+06
F14	9.46E+04	2.94E+06	1.75E+06

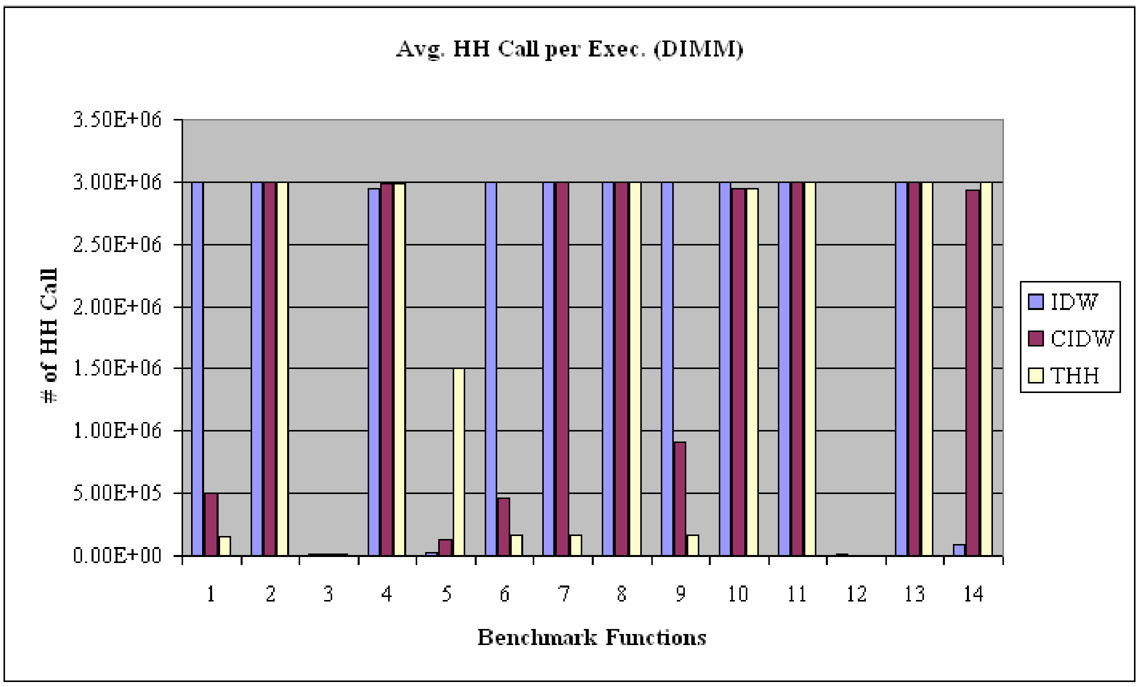

Figure A.5. Graphical view of average hyperheuristics call per execution for tested algorithms (IDW, CIDW and THH) with DIMM

Table A.6. Data table of average hyperheuristics call per execution for tested algorithms (IDW, CIDW and THH) with SWPD

	HYPERHEURISTICS		
Label	*IDW*	*CIDW*	*THH*
F1	3.00E+06	3.00E+06	3.00E+06
F2	3.00E+06	3.00E+06	3.00E+06
F3	3.00E+06	3.00E+06	3.00E+06
F4	3.00E+06	3.00E+06	3.00E+06
F5	3.00E+06	3.00E+06	3.00E+06
F6	3.00E+06	3.00E+06	3.00E+06
F7	3.00E+06	3.00E+06	3.00E+06
F8	3.00E+06	3.00E+06	3.00E+06
F9	3.00E+06	3.00E+06	3.00E+06
F10	3.00E+06	3.00E+06	3.00E+06
F11	3.00E+06	3.00E+06	3.00E+06
F12	3.00E+06	3.00E+06	3.00E+06
F13	3.00E+06	3.00E+06	3.00E+06
F14	3.00E+06	3.00E+06	3.00E+06

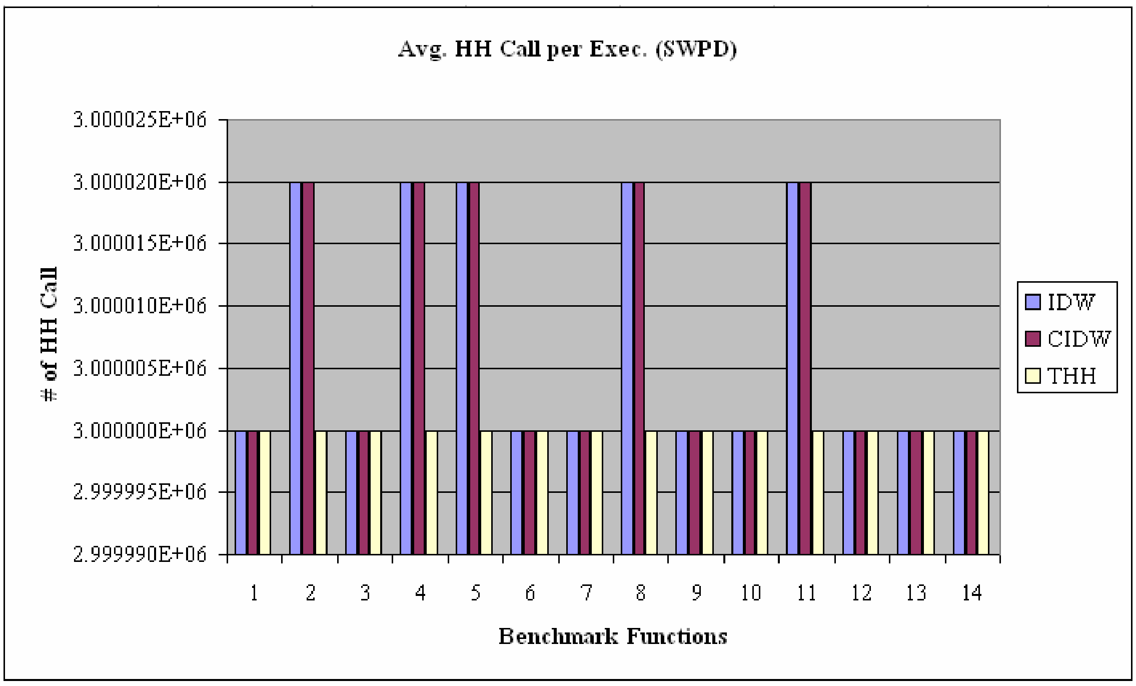

Figure A.6. Graphical view of average hyperheuristics call per execution for tested algorithms (IDW, CIDW and THH) with SWPD

Table A.7. Data table of average fitness evaluations per execution for tested algorithms (IDW, CIDW and THH) with HYPM

	HYPERHEURISTICS		
Label	*IDW*	*CIDW*	*THH*
F1	1.90E+04	1.99E+04	1.59E+04
F2	3.00E+06	3.00E+06	3.00E+06
F3	5.15E+03	5.43E+03	5.15E+03
F4	3.00E+06	3.00E+06	3.00E+06
F5	8.97E+03	2.40E+05	2.59E+05
F6	3.21E+05	3.35E+05	3.15E+05
F7	3.18E+05	4.49E+05	2.78E+05
F8	2.94E+06	3.00E+06	3.00E+06
F9	2.09E+04	2.90E+04	1.98E+04
F10	7.49E+05	7.92E+05	7.92E+05
F11	1.44E+06	9.77E+05	9.67E+05
F12	4.04E+04	4.95E+04	1.15E+04
F13	1.05E+06	2.15E+06	3.00E+06
F14	5.63E+05	1.83E+06	1.75E+06

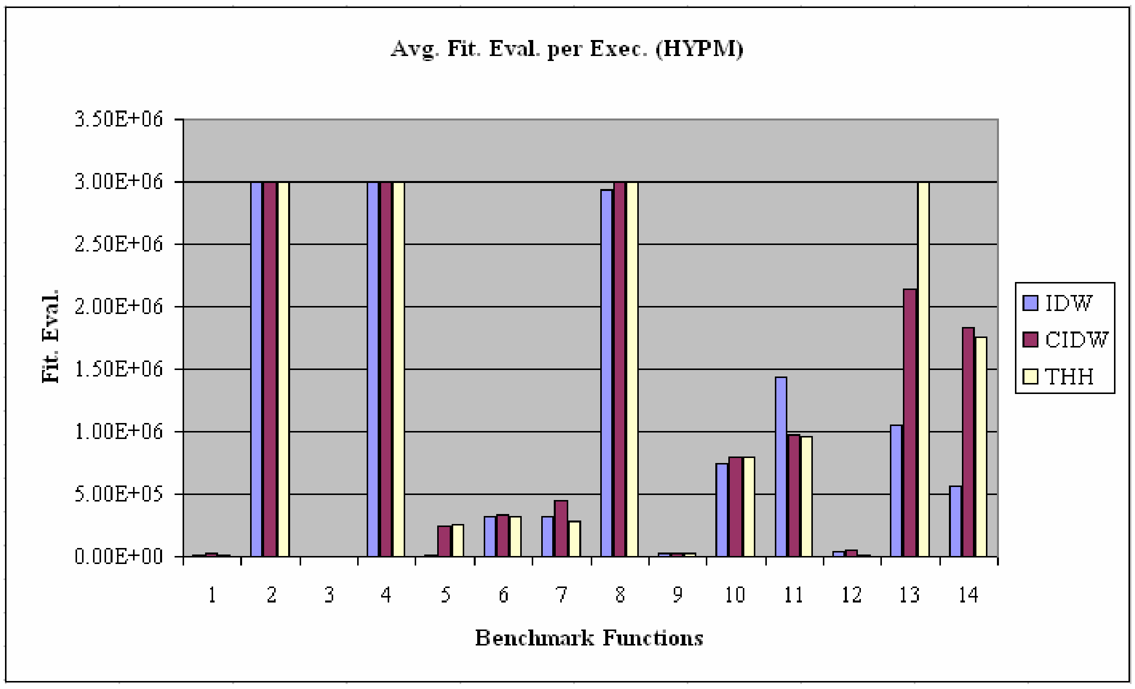

Figure A.7. Graphical view of average fitness evaluations per execution for tested algorithms (IDW, CIDW and THH) with HYPM

Table A.8. Data table of average fitness evaluations per execution for tested algorithms (IDW, CIDW and THH) with DIMM

	HYPERHEURISTICS		
Label	*IDW*	*CIDW*	*THH*
F1	3.00E+06	4.96E+05	1.59E+05
F2	3.00E+06	3.00E+06	3.00E+06
F3	7.51E+03	7.51E+03	7.51E+03
F4	2.95E+06	2.99E+06	2.99E+06
F5	2.48E+04	1.26E+05	1.50E+06
F6	3.00E+06	4.59E+05	1.68E+05
F7	3.00E+06	3.00E+06	1.73E+05
F8	3.00E+06	3.00E+06	3.00E+06
F9	3.00E+06	9.17E+05	1.72E+05
F10	3.00E+06	2.94E+06	2.95E+06
F11	3.00E+06	3.00E+06	3.00E+06
F12	9.81E+03	6.25E+03	5.24E+03
F13	3.00E+06	3.00E+06	3.00E+06
F14	9.46E+04	2.94E+06	3.00E+06

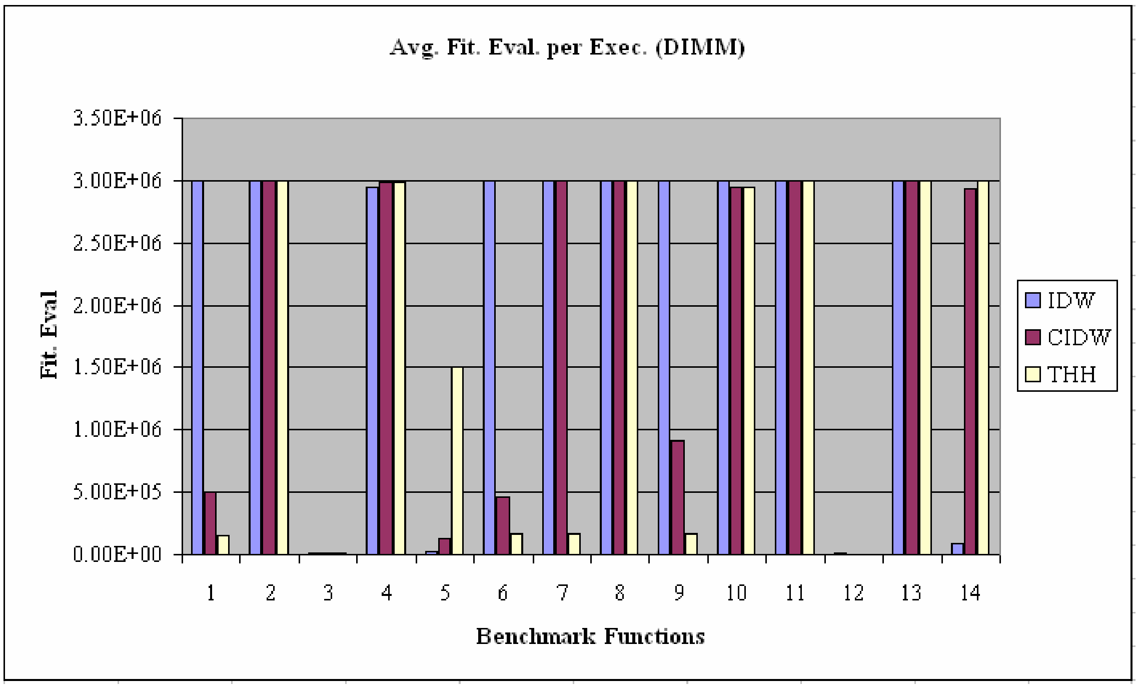

Figure A.8. Graphical view of average fitness evaluations per execution for tested algorithms (IDW, CIDW and THH) with DIMM

Table A.9. Data table of average fitness evaluations per execution for tested algorithms (IDW, CIDW and THH) with SWPD

	HYPERHEURISTICS		
Label	*IDW*	*CIDW*	*THH*
F1	3.00E+06	3.00E+06	3.00E+06
F2	3.00E+06	3.00E+06	3.00E+06
F3	3.00E+06	3.00E+06	3.00E+06
F4	3.00E+06	3.00E+06	3.00E+06
F5	3.00E+06	3.00E+06	3.00E+06
F6	3.00E+06	3.00E+06	3.00E+06
F7	3.00E+06	3.00E+06	3.00E+06
F8	3.00E+06	3.00E+06	3.00E+06
F9	3.00E+06	3.00E+06	3.00E+06
F10	3.00E+06	3.00E+06	3.00E+06
F11	3.00E+06	3.00E+06	3.00E+06
F12	3.00E+06	3.00E+06	3.00E+06
F13	3.00E+06	3.00E+06	3.00E+06
F14	3.00E+06	3.00E+06	3.00E+06

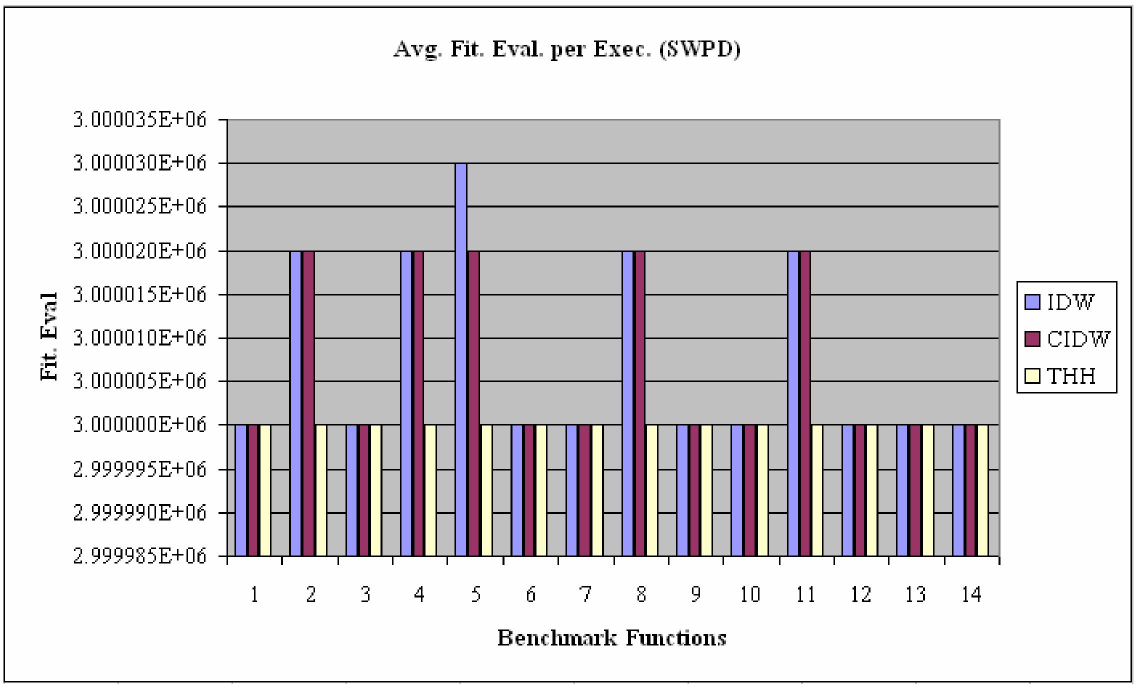

Figure A.9. Graphical view of average fitness evaluations per execution for tested algorithms (IDW, CIDW and THH) with SWPD

Table A.10. Data table of average fitness at the end for tested algorithms (IDW, CIDW and THH) with HYPM

		HYPERHEURISTICS		
Label	*IDW*	*CIDW*	*THH*	*GLOBAL*
F1	0.00E+00	0.00E+00	0.00E+00	1.00E-50
F2	1.98E-08	4.89E-08	1.77E-08	1.79E-28
F3	0.00E+00	0.00E+00	0.00E+00	5.00E-01
F4	2.93E+00	2.54E+00	2.52E+00	1.00E+00
F5	9.98E-01	9.98E-01	9.98E-01	9.98E-01
F6	0.00E+00	0.00E+00	0.00E+00	1.00E-50
F7	1.27E-04	1.27E-04	1.27E-04	1.27E-04
F8	1.19E-01	1.06E-01	1.10E-01	1.00E-50
F9	2.84E-14	2.84E-14	2.84E-14	2.89E-14
F10	-1.00E+00	-9.80E-01	-9.80E-01	-1.00E+00
F11	7.78E-26	7.78E-26	7.78E-26	7.78E-26
F12	0.00E+00	0.00E+00	0.00E+00	5.00E-01
F13	2.00E-01	7.20E-01	8.60E+00	5.00E-01
F14	0.00E+00	1.52E+00	2.80E-01	5.00E-01

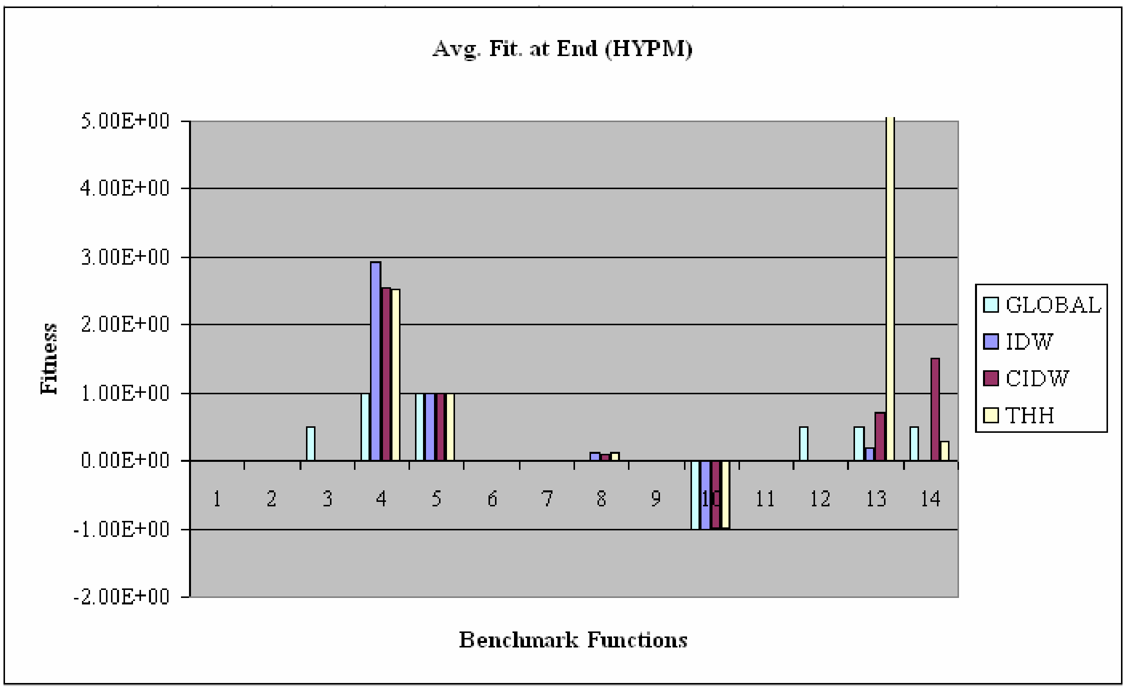

Figure A.10. Graphical view of average fitness at the end for tested algorithms (IDW, CIDW and THH) with HYPM

Table A.11. Data table of average fitness at the end for tested algorithms (IDW, CIDW and THH) with DIMM

HYPERHEURISTICS				
Label	*IDW*	*CIDW*	*THH*	*GLOBAL*
F1	1.64E+00	0.00E+00	0.00E+00	1.00E-50
F2	9.04E+01	7.02E-01	6.48E-01	1.79E-28
F3	0.00E+00	0.00E+00	0.00E+00	5.00E-01
F4	5.88E+00	1.91E+00	1.55E+00	1.00E+00
F5	9.98E-01	9.98E-01	9.98E-01	9.98E-01
F6	1.51E+01	0.00E+00	0.00E+00	1.00E-50
F7	3.48E+02	1.27E-04	1.27E-04	1.27E-04
F8	8.25E+00	1.12E-01	1.16E-01	1.00E-50
F9	6.49E+00	2.84E-14	2.84E-14	2.89E-14
F10	0.00E+00	-2.00E-02	-2.00E-02	-1.00E+00
F11	1.71E+03	1.37E-07	2.22E-12	7.78E-26
F12	0.00E+00	0.00E+00	0.00E+00	5.00E-01
F13	2.49E+02	4.37E+01	2.65E+01	5.00E-01
F14	0.00E+00	5.04E+00	3.52E+00	5.00E-01

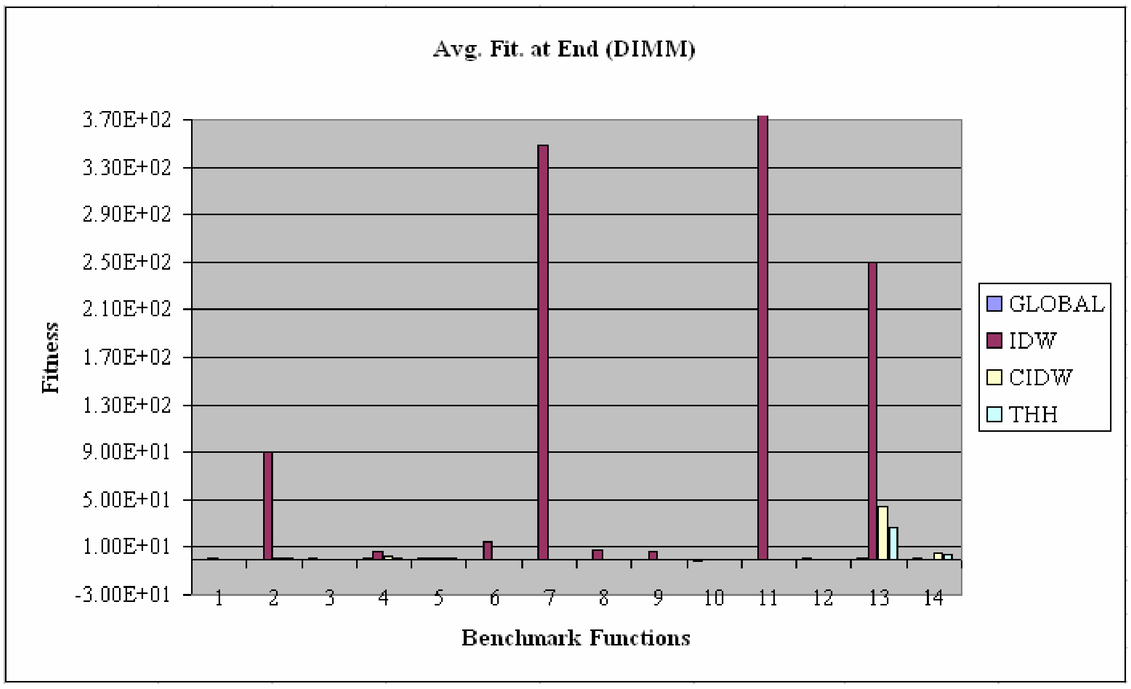

Figure A.11. Graphical view of average fitness at the end for tested algorithms (IDW, CIDW and THH) with DIMM

Table A.12. Data table of average fitness at the end for tested algorithms (IDW, CIDW and THH) with SWPD

	HYPERHEURISTICS			
Label	*IDW*	*CIDW*	*THH*	*GLOBAL*
F1	8.31E+01	8.31E+01	8.31E+01	1.00E-50
F2	2.01E+03	1.76E+03	1.84E+03	1.79E-28
F3	5.28E+01	5.28E+01	5.28E+01	5.00E-01
F4	1.78E+01	1.73E+01	1.34E+01	1.00E+00
F5	4.82E+02	4.82E+02	4.82E+02	9.98E-01
F6	1.82E+02	1.82E+02	1.82E+02	1.00E-50
F7	4.18E+03	4.18E+03	4.18E+03	1.27E-04
F8	2.86E+02	2.86E+02	2.86E+02	1.00E-50
F9	2.11E+01	2.11E+01	2.11E+01	2.89E-14
F10	0.00E+00	0.00E+00	0.00E+00	-1.00E+00
F11	3.30E+04	3.08E+04	3.01E+04	7.78E-26
F12	8.00E+00	8.00E+00	8.00E+00	5.00E-01
F13	5.04E+02	5.04E+02	5.04E+02	5.00E-01
F14	9.33E+01	9.33E+01	9.33E+01	5.00E-01

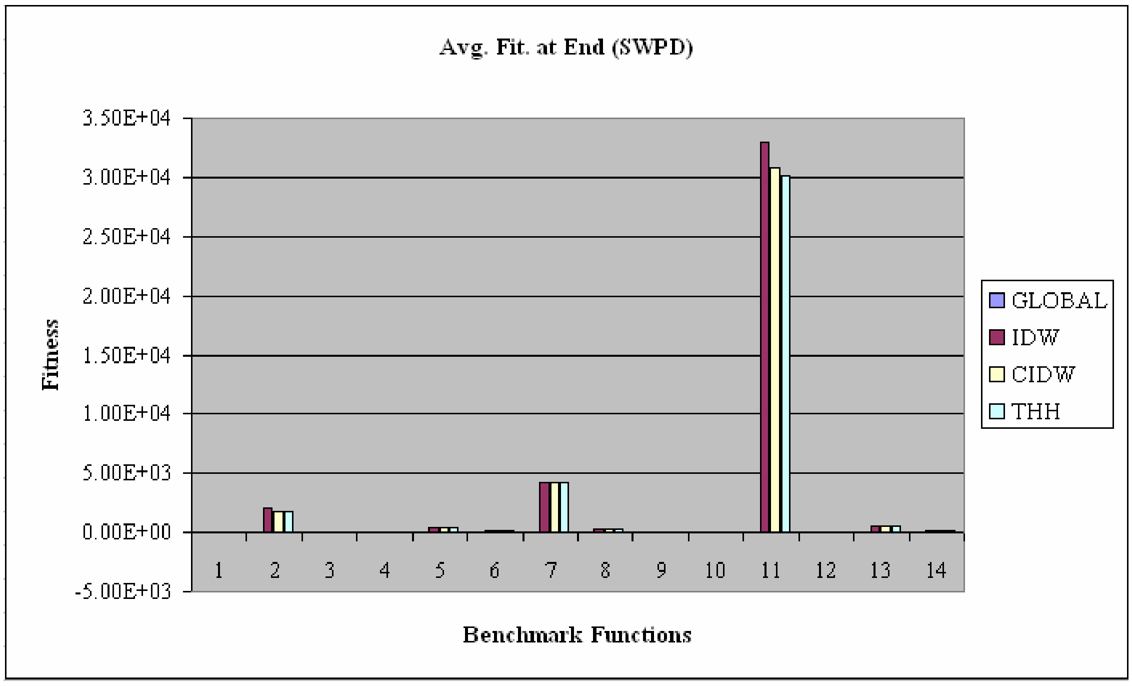

Figure A.12. Graphical view of average fitness at the end for tested algorithms (IDW, CIDW and THH) with SWPD

Table A. 13. Data table of average best fitness at the end for tested algorithms (IDW, CIDW and THH) with HYPM

	HYPERHEURISTICS			
Label	*IDW*	*CIDW*	*THH*	*GLOBAL*
F1	0.00E+00	0.00E+00	0.00E+00	1.00E-50
F2	1.98E-08	4.89E-08	1.77E-08	1.79E-28
F3	0.00E+00	0.00E+00	0.00E+00	5.00E-01
F4	1.96E+00	1.97E+00	2.52E+00	1.00E+00
F5	9.98E-01	9.98E-01	9.98E-01	9.98E-01
F6	0.00E+00	0.00E+00	3.38E-01	1.00E-50
F7	1.27E-04	1.27E-04	1.27E-04	1.27E-04
F8	1.19E-01	1.06E-01	1.10E-01	1.00E-50
F9	2.84E-14	2.84E-14	2.84E-14	2.89E-14
F10	-1.00E+00	-9.80E-01	1.41E-01	-1.00E+00
F11	7.78E-26	7.78E-26	7.78E-26	7.78E-26
F12	0.00E+00	0.00E+00	0.00E+00	5.00E-01
F13	1.20E-01	7.20E-01	8.60E+00	5.00E-01
F14	0.00E+00	5.60E-01	2.80E-01	5.00E-01

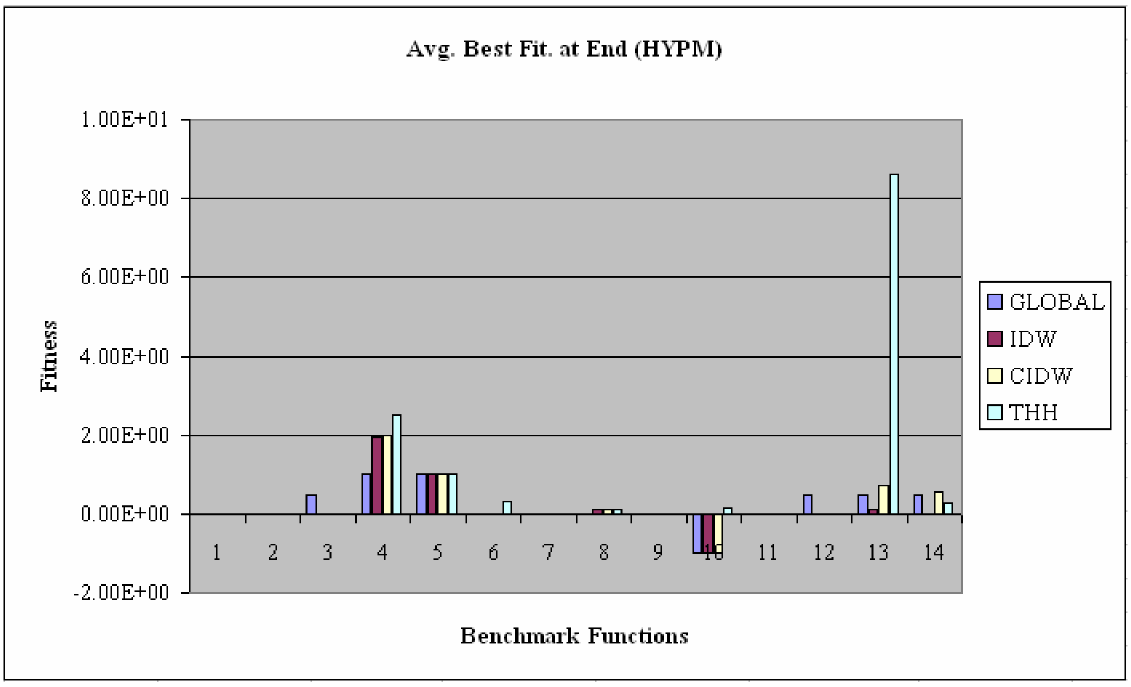

Figure A.13. Graphical view of average best fitness at the end for tested algorithms (IDW, CIDW and THH) with HYPM

Table A.14. Data table of average best fitness at the end for tested algorithms (IDW, CIDW and THH) with DIMM

HYPERHEURISTICS				
Label	*IDW*	*CIDW*	*THH*	*GLOBAL*
F1	1.65E-03	0.00E+00	0.00E+00	1.00E-50
F2	9.15E-01	7.02E-01	6.48E-01	1.79E-28
F3	0.00E+00	0.00E+00	0.00E+00	5.00E-01
F4	1.46E+00	1.49E+00	1.55E+00	1.00E+00
F5	9.98E-01	9.98E-01	9.98E-01	9.98E-01
F6	3.56E-01	0.00E+00	0.00E+00	1.00E-50
F7	2.18E+00	1.27E-04	1.27E-04	1.27E-04
F8	2.40E-01	1.12E-01	4.06E-02	1.00E-50
F9	5.90E-01	2.84E-14	2.84E-14	2.89E-14
F10	-4.61E-06	-2.00E-02	-2.00E-02	-1.00E+00
F11	6.86E+01	1.37E-07	2.22E-12	7.78E-26
F12	0.00E+00	0.00E+00	0.00E+00	5.00E-01
F13	7.69E+01	3.40E+01	2.65E+01	5.00E-01
F14	0.00E+00	3.64E+00	3.52E+00	5.00E-01

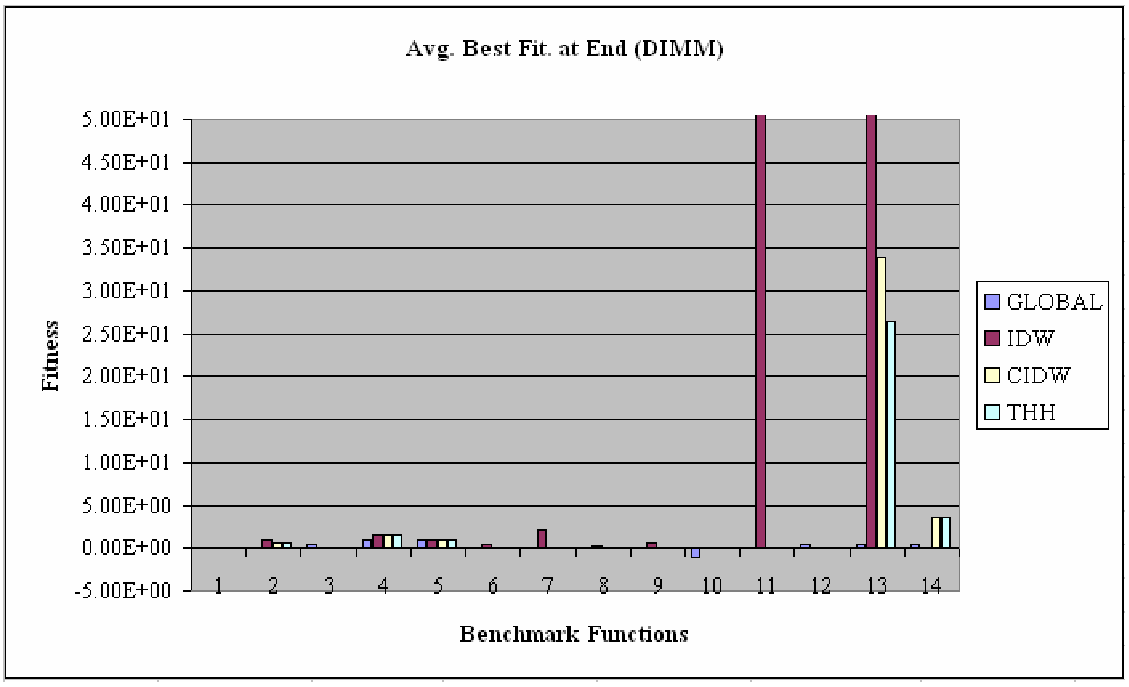

Figure A.14. Graphical view of average best fitness at the end for tested algorithms (IDW, CIDW and THH) with DIMM

Table A.15. Data table of average best fitness at the end for tested algorithms (IDW, CIDW and THH) with SWPD

	HYPERHEURISTICS			
Label	*IDW*	*CIDW*	*THH*	*GLOBAL*
F1	8.31E+01	8.31E+01	8.31E+01	1.00E-50
F2	1.57E+03	1.57E+03	1.84E+03	1.79E-28
F3	5.28E+01	5.28E+01	5.28E+01	5.00E-01
F4	1.36E+01	1.36E+01	1.34E+01	1.00E+00
F5	4.82E+02	4.82E+02	4.82E+02	9.98E-01
F6	1.82E+02	1.82E+02	1.82E+02	1.00E-50
F7	4.18E+03	4.18E+03	4.18E+03	1.27E-04
F8	2.86E+02	2.86E+02	2.86E+02	1.00E-50
F9	2.11E+01	2.11E+01	2.11E+01	2.89E-14
F10	0.00E+00	0.00E+00	0.00E+00	-1.00E+00
F11	2.98E+04	2.98E+04	3.01E+04	7.78E-26
F12	8.00E+00	8.00E+00	8.00E+00	5.00E-01
F13	5.04E+02	5.04E+02	5.04E+02	5.00E-01
F14	9.33E+01	9.33E+01	9.33E+01	5.00E-01

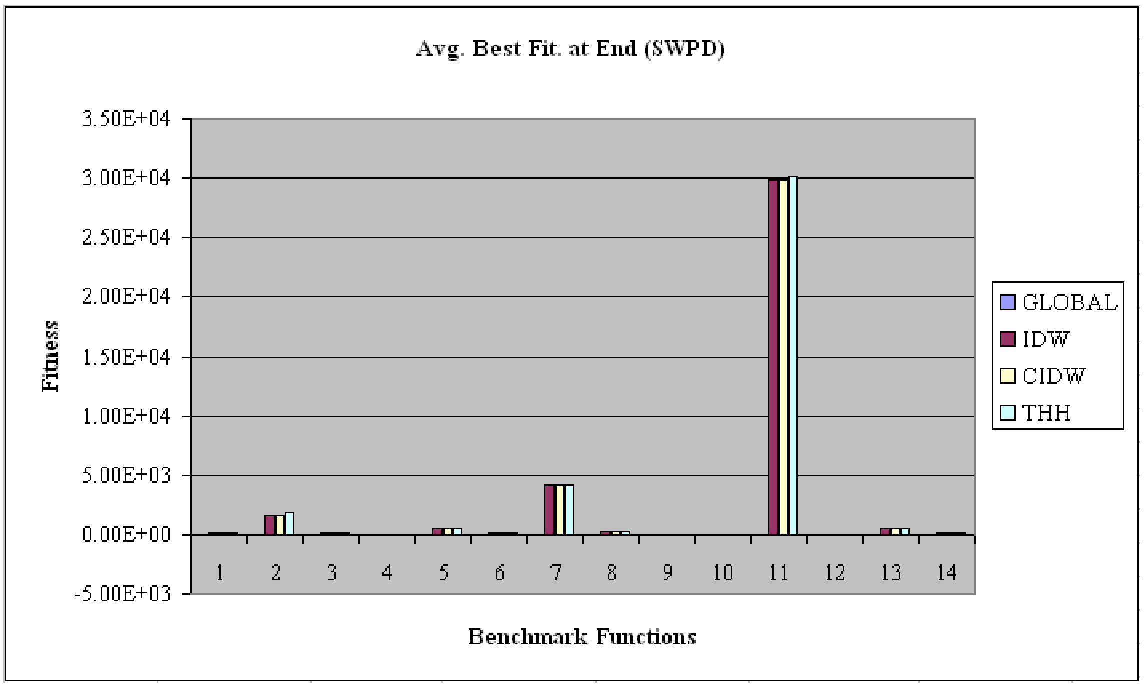

Figure A.15. Graphical view of average best fitness at the end for tested algorithms (IDW, CIDW and THH) with SWPD

Table A.16. Data table of average execution time (sec.) for tested algorithms (IDW, CIDW, EXIDW and THH) with a heuristic set which has HYPM, DIMM and SWPD

	HYPERHEURISTICS			
Label	*IDW*	*CIDW*	*EXIDW*	*THH*
F1	9.61E+00	3.57E-01	2.35E-01	5.04E-01
F2	1.36E+01	1.30E+01	2.27E+01	9.81E+00
F3	7.90E-02	8.04E-02	6.56E-02	1.07E-01
F4	2.70E+01	2.74E+01	2.31E+01	2.36E+01
F5	2.54E+00	3.48E+00	1.32E+00	5.02E+00
F6	8.72E-01	4.10E-01	3.43E-01	1.24E+00
F7	1.26E+01	1.24E+01	2.12E-01	1.30E+01
F8	1.96E+01	1.89E+01	2.75E+01	1.67E+01
F9	3.80E+00	5.15E-01	2.54E-01	4.80E-01
F10	6.58E+00	7.61E+00	5.05E+00	5.29E+00
F11	1.82E+01	1.73E+01	2.30E+01	1.75E+01
F12	1.80E-01	3.44E-02	2.74E-02	5.12E-02
F13	1.26E+01	1.33E+01	1.33E+01	1.13E+01
F14	9.16E-02	1.20E-02	4.20E-03	1.94E+00

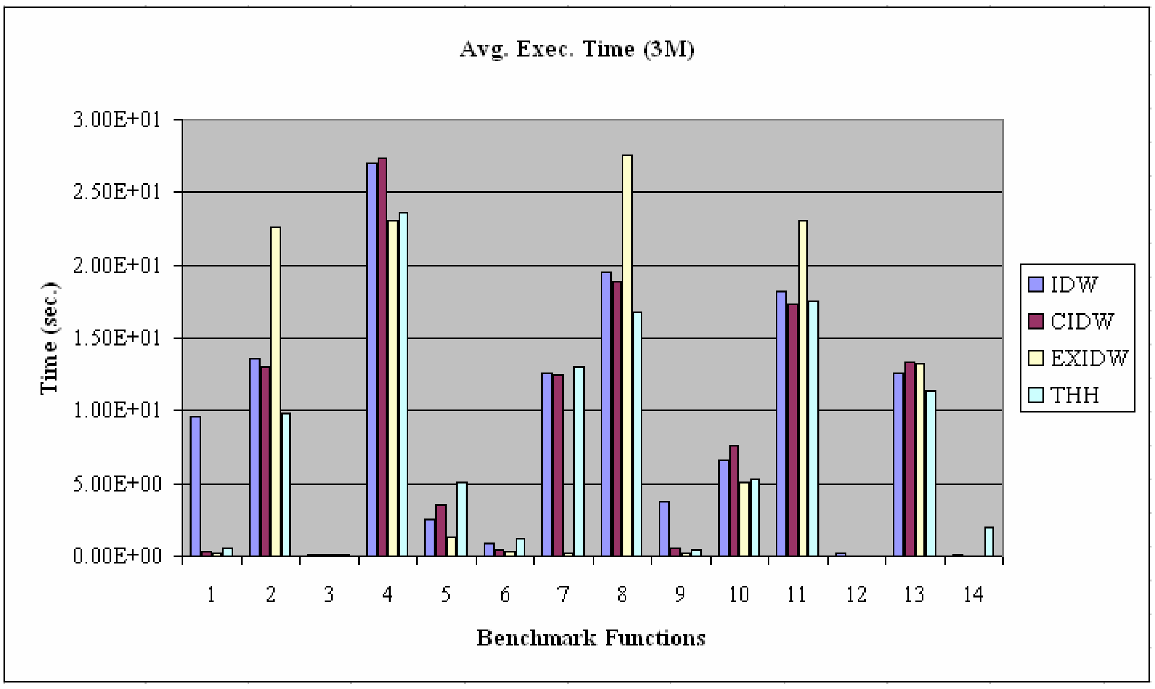

Figure A.16. Graphical view of average execution time (sec.) for tested algorithms (IDW, CIDW, EXIDW and THH) with a heuristic set which has HYPM, DIMM and SWPD

Table A.17. Data table of average hyperheuristic call per execution for tested algorithms (IDW, CIDW, EXIDW and THH) with a heuristic set which has HYPM, DIMM and SWPD

	HYPERHEURISTICS			
Label	*IDW*	*CIDW*	*EXIDW*	*THH*
F1	1.44E+06	6.91E+04	3.21E+04	1.01E+05
F2	3.00E+06	3.00E+06	3.00E+06	3.00E+06
F3	9.49E+03	9.64E+03	8.19E+03	1.41E+04
F4	3.00E+06	3.00E+06	3.00E+06	3.00E+06
F5	3.93E+05	5.18E+05	2.22E+05	7.83E+05
F6	1.26E+05	6.82E+04	4.25E+04	2.41E+05
F7	2.98E+06	2.91E+06	2.51E+04	3.00E+06
F8	3.00E+06	3.00E+06	3.00E+06	3.00E+06
F9	4.75E+05	7.51E+04	3.01E+04	6.60E+04
F10	1.04E+06	1.24E+06	8.59E+05	7.42E+05
F11	3.00E+06	3.00E+06	2.37E+06	3.00E+06
F12	4.84E+04	9.24E+03	1.09E+04	1.38E+04
F13	3.00E+06	3.00E+06	3.00E+06	3.00E+06
F14	3.18E+04	4.35E+03	2.28E+03	7.14E+05

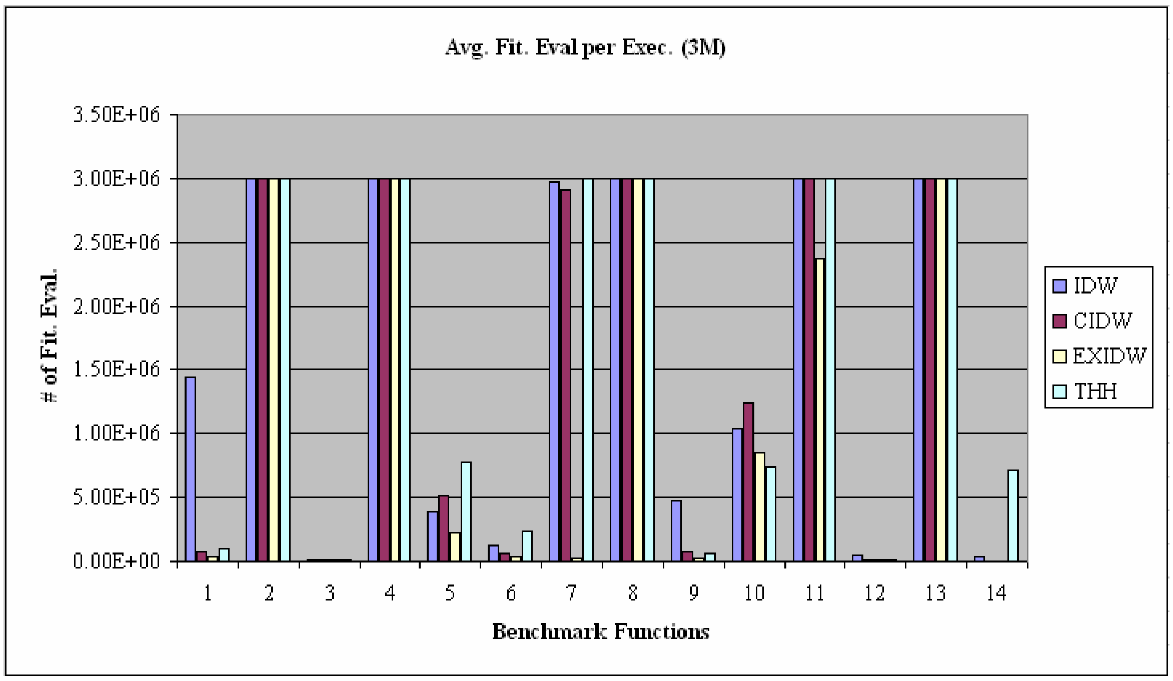

Figure A.17. Graphical view of average hyperheuristic call per execution for tested algorithms (IDW, CIDW, EXIDW and THH) with a heuristic set which has HYPM, DIMM and SWPD

Table A.18. Data table of average fitness evaluations per execution for tested algorithms (IDW, CIDW, EXIDW and THH) with a heuristic set which has HYPM, DIMM and SWPD

	HYPERHEURISTICS			
Label	*IDW*	*CIDW*	*EXIDW*	*THH*
F1	1.44E+06	6.91E+04	3.21E+04	1.01E+05
F2	3.00E+06	3.00E+06	3.00E+06	3.00E+06
F3	9.49E+03	9.64E+03	8.19E+03	1.41E+04
F4	3.00E+06	3.00E+06	3.00E+06	3.00E+06
F5	3.93E+05	5.18E+05	2.22E+05	7.83E+05
F6	1.26E+05	6.82E+04	4.26E+04	2.41E+05
F7	2.98E+06	2.91E+06	2.51E+04	3.00E+06
F8	3.00E+06	3.00E+06	3.00E+06	3.00E+06
F9	4.75E+05	7.51E+04	3.01E+04	6.60E+04
F10	1.04E+06	1.24E+06	8.59E+05	7.42E+05
F11	3.00E+06	3.00E+06	2.37E+06	3.00E+06
F12	4.84E+04	9.24E+03	1.09E+04	1.38E+04
F13	3.00E+06	3.00E+06	3.00E+06	3.00E+06
F14	3.18E+04	4.35E+03	2.28E+03	7.14E+05

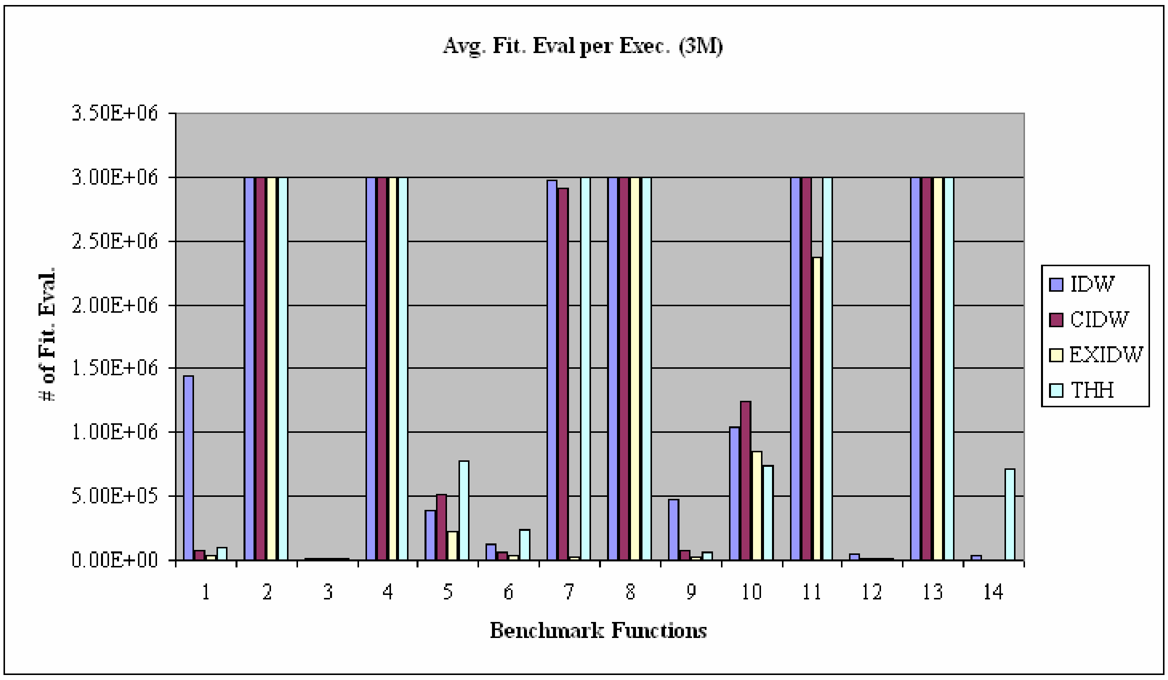

Figure A.18. Graphical view of average fitness evaluations per execution for tested algorithms (IDW, CIDW, EXIDW and THH) with a heuristic set which has HYPM, DIMM and SWPD

Table A.19. Data table of average fitness at the end for tested algorithms (IDW, CIDW, EXIDW and THH) with a heuristic set which has HYPM, DIMM and SWPD.

	HYPERHEURISTICS				
Label	*IDW*	*CIDW*	*EXIDW*	*THH*	*GLOBAL*
F1	2.91E-17	0.00E+00	0.00E+00	0.00E+00	1.00E-50
F2	1.23E-02	5.13E+00	1.68E-05	1.60E+00	1.79E-28
F3	0.00E+00	0.00E+00	0.00E+00	0.00E+00	5.00E-01
F4	2.49E+00	2.05E+00	2.40E+00	1.90E+00	1.00E+00
F5	9.98E-01	9.98E-01	9.98E-01	9.98E-01	9.98E-01
F6	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E-50
F7	1.40E-04	2.66E+00	1.27E-04	1.14E+01	1.27E-04
F8	1.29E-01	1.18E-01	1.19E-01	4.61E-01	1.00E-50
F9	2.84E-14	2.84E-14	2.84E-14	2.84E-14	2.89E-14
F10	-1.00E+00	-1.00E+00	-9.80E-01	-1.00E+00	-1.00E+00
F11	2.28E-01	2.43E-01	2.09E-15	7.49E+02	7.78E-26
F12	0.00E+00	0.00E+00	0.00E+00	0.00E+00	5.00E-01
F13	2.96E+01	3.31E+01	1.55E+01	6.09E+01	5.00E-01
F14	0.00E+00	0.00E+00	0.00E+00	0.00E+00	5.00E-01

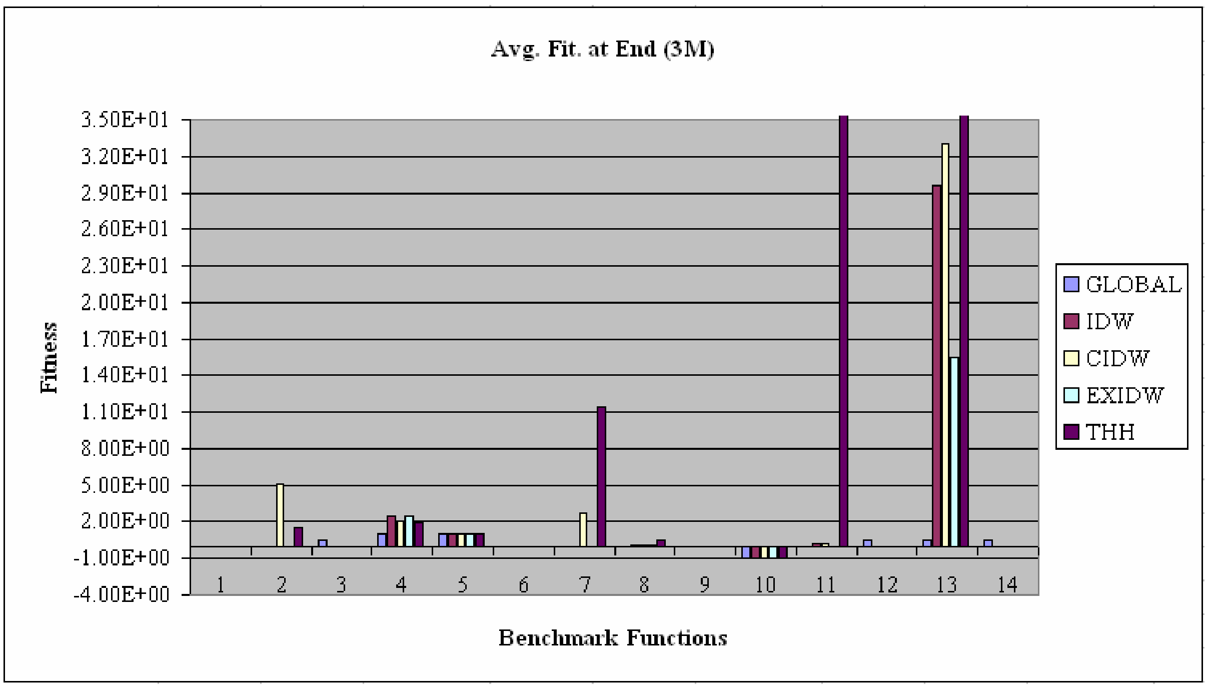

Figure A.19. Graphical view of average fitness at the end for tested algorithms (IDW, CIDW, EXIDW and THH) with a heuristic set which has HYPM, DIMM and SWPD

Table A.20. Data table of average best fitness at the end for tested algorithms (IDW, CIDW, EXIDW and THH) with a heuristic set which has HYPM, DIMM and SWPD

	HYPERHEURISTICS				
Label	*IDW*	*CIDW*	*EXIDW*	*THH*	*GLOBAL*
F1	1.82E-17	0.00E+00	0.00E+00	0.00E+00	1.00E-50
F2	1.23E-02	1.28E-02	1.68E-05	1.60E+00	1.79E-28
F3	0.00E+00	0.00E+00	0.00E+00	0.00E+00	5.00E-01
F4	1.61E+00	1.57E+00	1.57E+00	1.90E+00	1.00E+00
F5	9.98E-01	9.98E-01	9.98E-01	9.98E-01	9.98E-01
F6	0.00E+00	0.00E+00	0.00E+00	0.00E+00	1.00E-50
F7	1.40E-04	1.40E-04	1.27E-04	1.14E+01	1.27E-04
F8	1.29E-01	1.18E-01	1.19E-01	2.31E-01	1.00E-50
F9	2.84E-14	2.84E-14	2.84E-14	2.84E-14	2.89E-14
F10	-1.00E+00	-1.00E+00	-9.80E-01	-1.00E+00	-1.00E+00
F11	2.28E-01	2.43E-01	2.09E-15	7.38E+02	7.78E-26
F12	0.00E+00	0.00E+00	0.00E+00	0.00E+00	5.00E-01
F13	2.84E+01	3.31E+01	1.55E+01	4.15E+01	5.00E-01
F14	0.00E+00	0.00E+00	0.00E+00	0.00E+00	5.00E-01

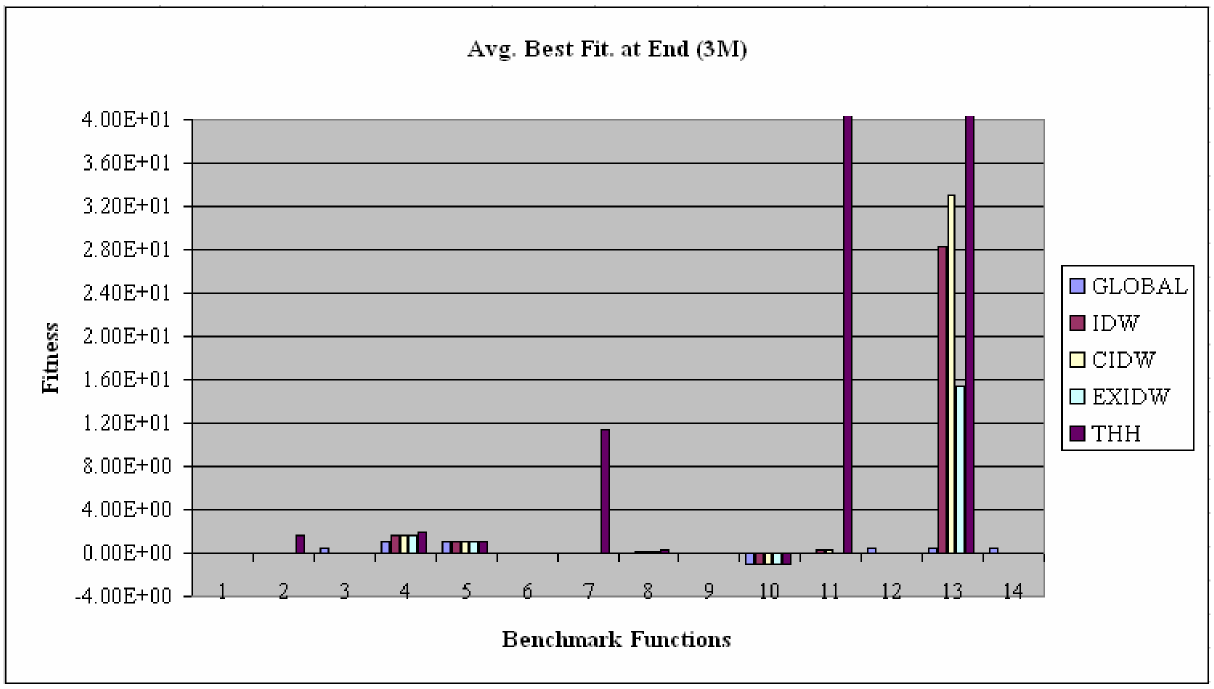

Figure A.20. Graphical view of average best fitness at the end for tested algorithms (IDW, CIDW, EXIDW and THH) with a heuristic set which has HYPM, DIMM and SWPD

APPENDIX B: EXPERIMENTAL RESULTS, TABLES AND CHARTS OF HYPERHEURISTICS PATTERNS ON A SET OF DATA FOR EXAMINATION TIMETABLING

In this Appendix, average best fitness values of the best combinations of selection and acceptan mechanisms are provided in Table B.1 and visualized in Figure B.1 – Figure B.21.

Table B.1. Examination timetabling results of the hyperheuristics which provide at least one the best solution on testted set of data for examination timetabling

	EXIDW	TABU_IE	CF_MC	SR_GD	SR_IE	SR_MC
Car-f-92	-1.29E-02	-1.02E-02	-1.02E-02	-9.42E-03	-9.91E-03	-7.08E-03
	1.09E-03	1.18E-03	5.85E-04	1.01E-03	1.09E-03	4.97E-04
Car-s-91	-1.01E-01	-1.93E-01	-3.90E-02	-1.06E-01	-8.71E-02	-1.58E-02
	3.30E-02	1.20E-01	6.46E-03	3.84E-02	2.13E-02	1.53E-03
Ear-f-83	-7.51E-03	-4.52E-03	-7.27E-03	-5.35E-03	-4.69E-03	-5.71E-03
	6.44E-04	3.88E-04	4.94E-04	4.38E-04	3.88E-04	3.55E-04
Hec-s-92	-2.19E-02	-8.12E-03	-2.19E-02	-1.67E-02	-7.97E-03	-1.68E-02
	3.86E-03	1.43E-03	2.43E-03	3.99E-03	1.61E-03	2.26E-03
Kfu-s-93	-2.82E-02	-2.43E-02	-2.79E-02	-3.40E-02	-2.47E-02	-2.20E-02
	4.93E-03	4.74E-03	4.02E-03	4.30E-03	4.88E-03	2.57E-03
Lse-f-91	-1.26E-02	-1.02E-02	-1.42E-02	-1.19E-02	-1.06E-02	-1.09E-02
	1.24E-03	1.33E-03	1.38E-03	1.42E-03	1.69E-03	1.04E-03
Pur-s-93	-1.38E-03	-1.37E-03	-9.39E-04	-1.06E-03	-1.41E-03	-6.37E-04
	6.08E-05	6.77E-05	2.92E-05	4.15E-05	6.98E-05	1.64E-05
Rye-s-93	-1.09E-02	-8.88E-03	-1.08E-02	-9.39E-03	-9.35E-03	-7.28E-03
	1.66E-03	1.71E-03	1.37E-03	1.30E-03	1.89E-03	8.88E-04
Sta-f-83	-2.65E-03	-2.63E-03	-2.68E-03	-2.68E-03	-2.64E-03	-2.68E-03
	5.27E-05	-2.63E-03	9.08E-06	1.26E-05	1.26E-05	1.04E-05
Tre-s-92	-6.72E-02	-4.25E-02	-4.53E-02	-6.79E-02	-3.95E-02	-3.43E-02
	1.39E-02	7.83E-03	5.90E-03	1.08E-02	7.55E-03	4.02E-03
Uta-s-92	-2.03E-02	-1.87E-02	-1.41E-02	-1.25E-02	-1.81E-02	-8.35E-03
	1.72E-03	1.79E-03	8.51E-04	9.83E-04	1.65E-03	4.96E-04
Ute-s-91	-1.99E-03	-1.55E-03	-2.27E-03	-1.69E-03	-1.58E-03	-1.95E-03
	1.30E-04	8.85E-05	8.64E-05	1.31E-04	1.12E-04	1.09E-04
Yor-f-83	-8.15E-03	-5.32E-03	-8.32E-03	-6.24E-03	-5.26E-03	-6.50E-03
	4.07E-04	3.99E-04	4.57E-04	4.71E-04	3.38E-04	4.05E-04
Yue20011	-7.67E-02	-5.76E-02	-7.49E-02	-9.02E-02	-5.89E-02	-6.55E-02

	1.31E-02	1.22E-02	9.06E-03	1.07E-02	1.02E-02	8.03E-03
Yue20012	-6.50E-02	-4.79E-02	-5.80E-02	-7.54E-02	-4.82E-02	-5.05E-02
	9.14E-03	6.38E-03	6.59E-03	9.38E-03	7.40E-03	6.04E-03
Yue20013	-2.13E-01	-1.60E-01	-2.50E-01	-2.37E-01	-1.66E-01	-2.50E-01
	2.62E-02	3.20E-02	0.00E+00	1.87E-02	3.91E-02	0.00E+00
Yue20021	-2.82E-02	-1.83E-02	-3.20E-02	-3.45E-02	-2.04E-02	-2.65E-02
	5.35E-03	3.78E-03	3.76E-03	4.55E-03	4.30E-03	2.64E-03
Yue20022	-1.08E-02	-8.30E-03	-1.26E-02	-1.09E-02	-8.48E-03	-1.09E-02
	1.62E-03	9.17E-04	9.08E-04	9.83E-04	1.05E-03	8.94E-04
Yue20031	-1.44E-02	-1.24E-02	-1.52E-02	-1.42E-02	-1.26E-02	-1.49E-02
	5.06E-04	6.94E-04	2.69E-04	4.89E-04	6.88E-04	3.02E-04
Yue20032	-1.44E-02	-8.07E-03	-1.59E-02	-1.31E-02	-8.69E-03	-1.26E-02
	2.24E-03	1.79E-03	1.65E-03	1.89E-03	1.39E-03	9.62E-04
Yue20033	-4.29E-03	-3.19E-03	-5.42E-03	-4.14E-03	-3.63E-03	-4.61E-03
	5.01E-04	3.14E-04	3.68E-04	4.04E-04	3.59E-04	3.64E-04

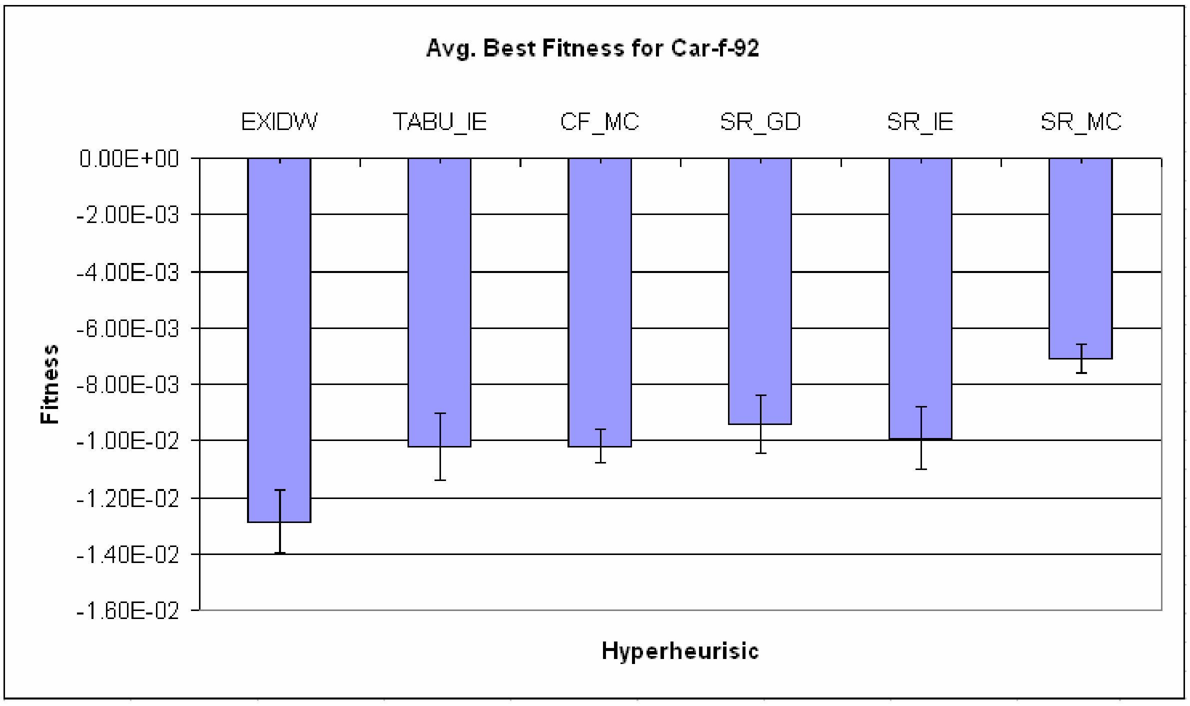

Figure B.1. Average Best Fitness values for Car-f-92

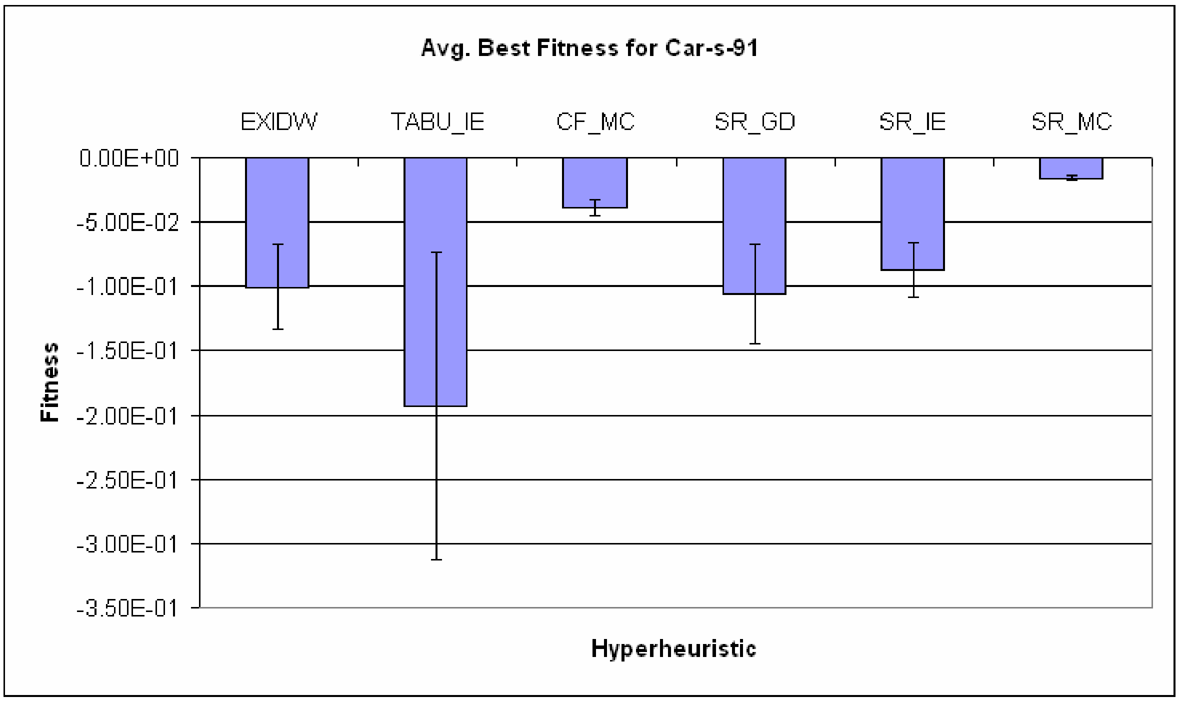

Figure B.2. Average Best Fitness values for Car-s-91

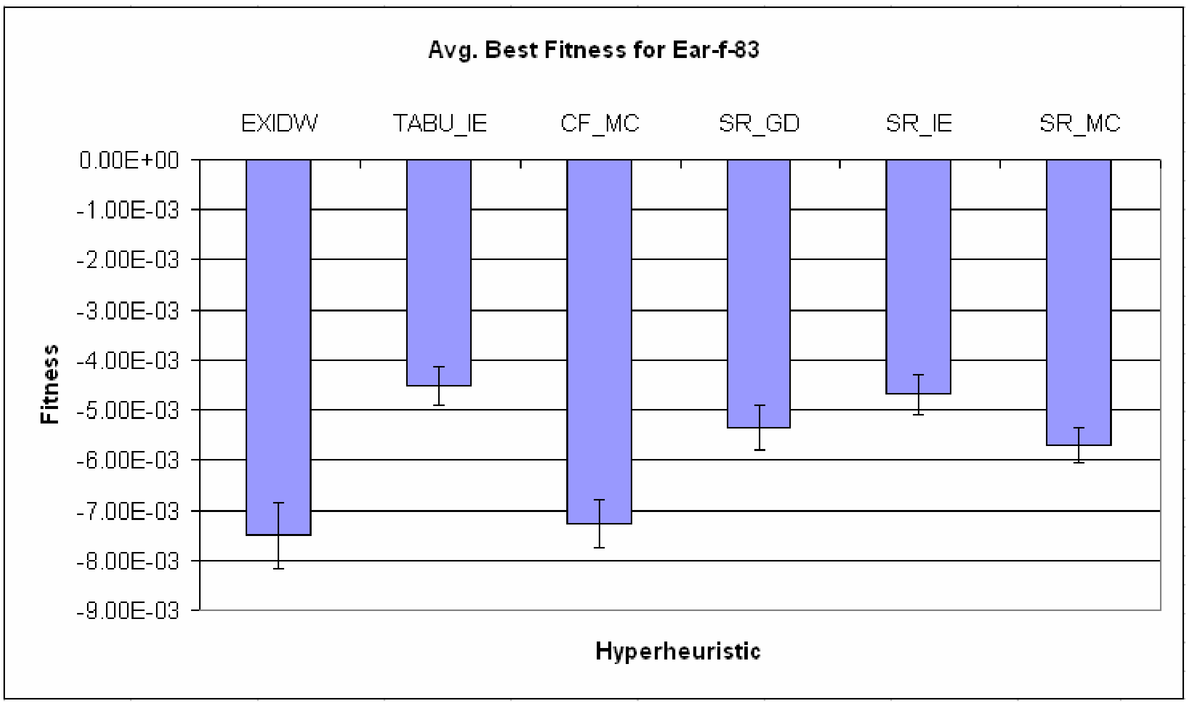

Figure B.3. Average Best Fitness values for Ear-f-83

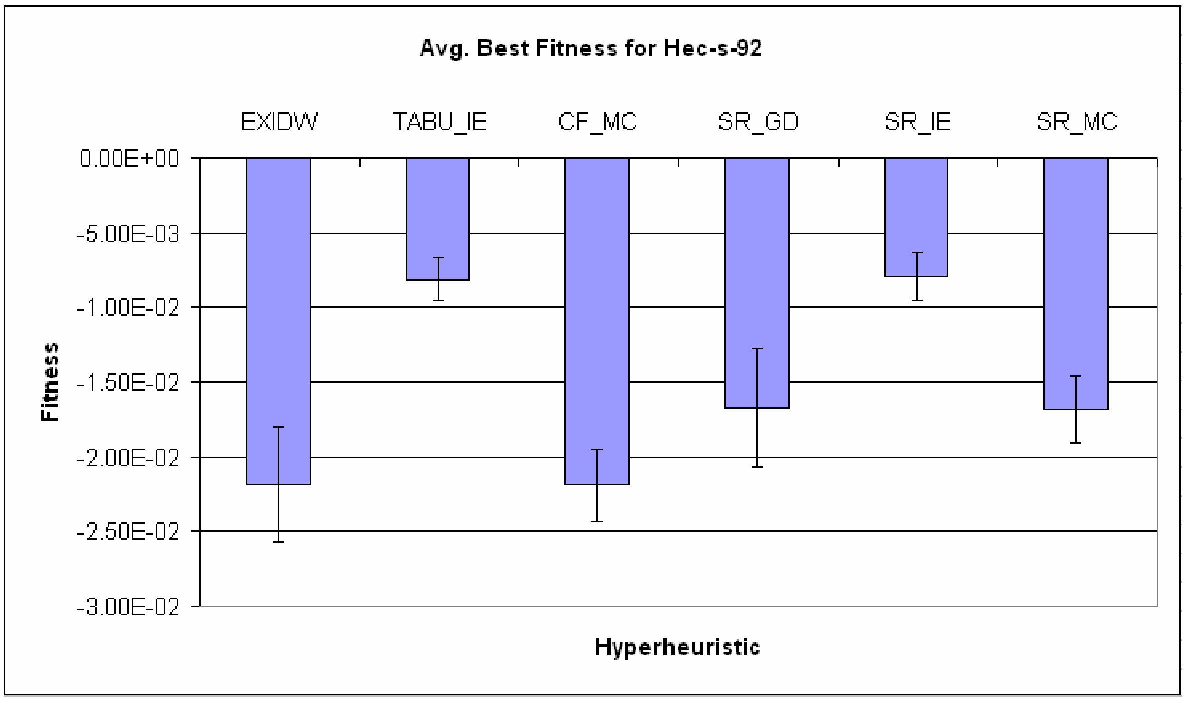

Figure B.4. Average Best Fitness values for Hec-s-92

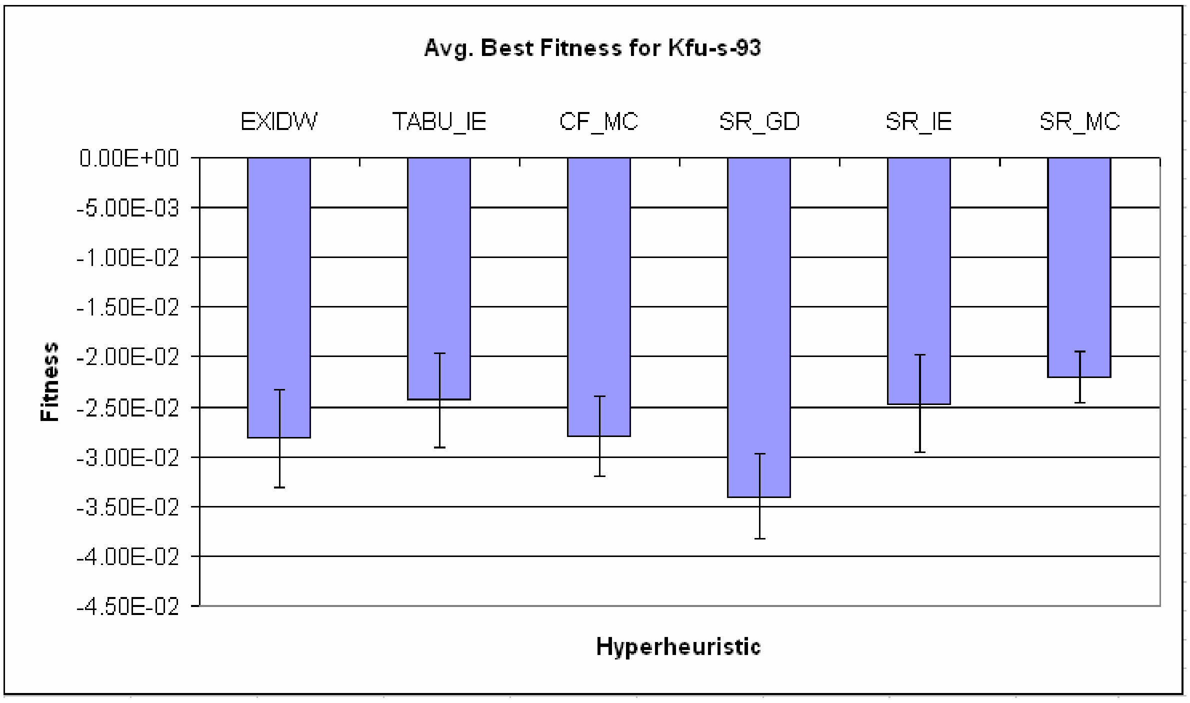

Figure B.5. Average Best Fitness values for Kfu-s-93

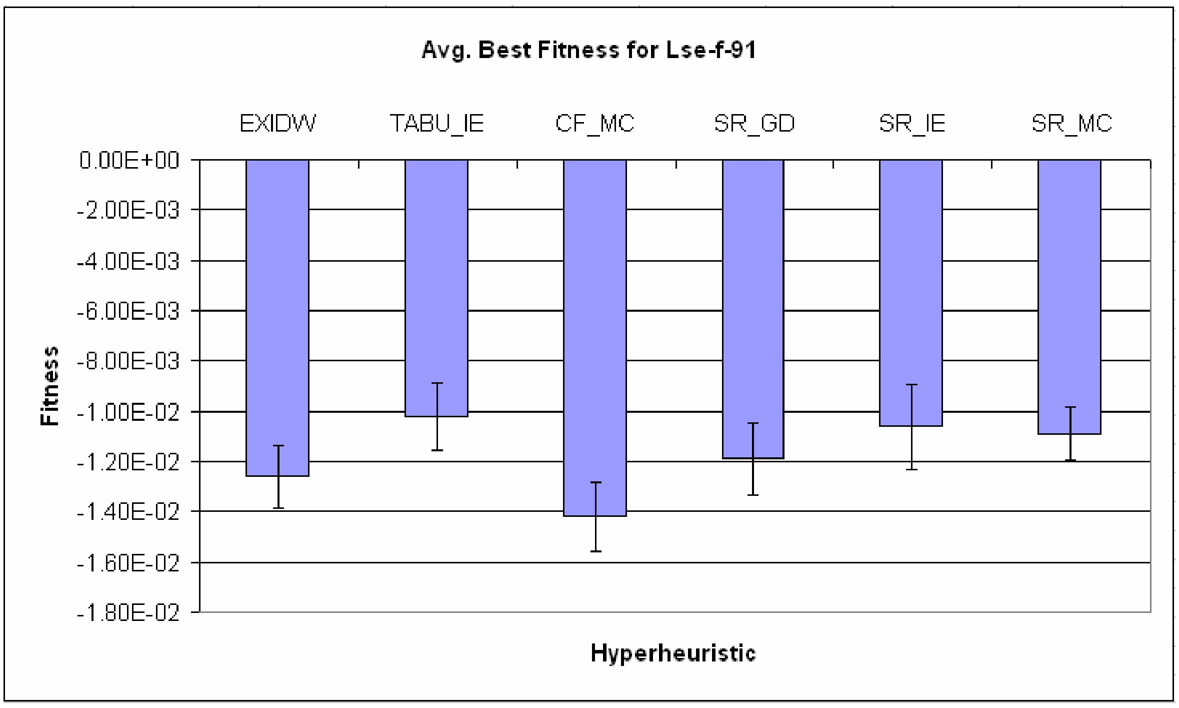

Figure B.6. Average Best Fitness values for Lse-f-91

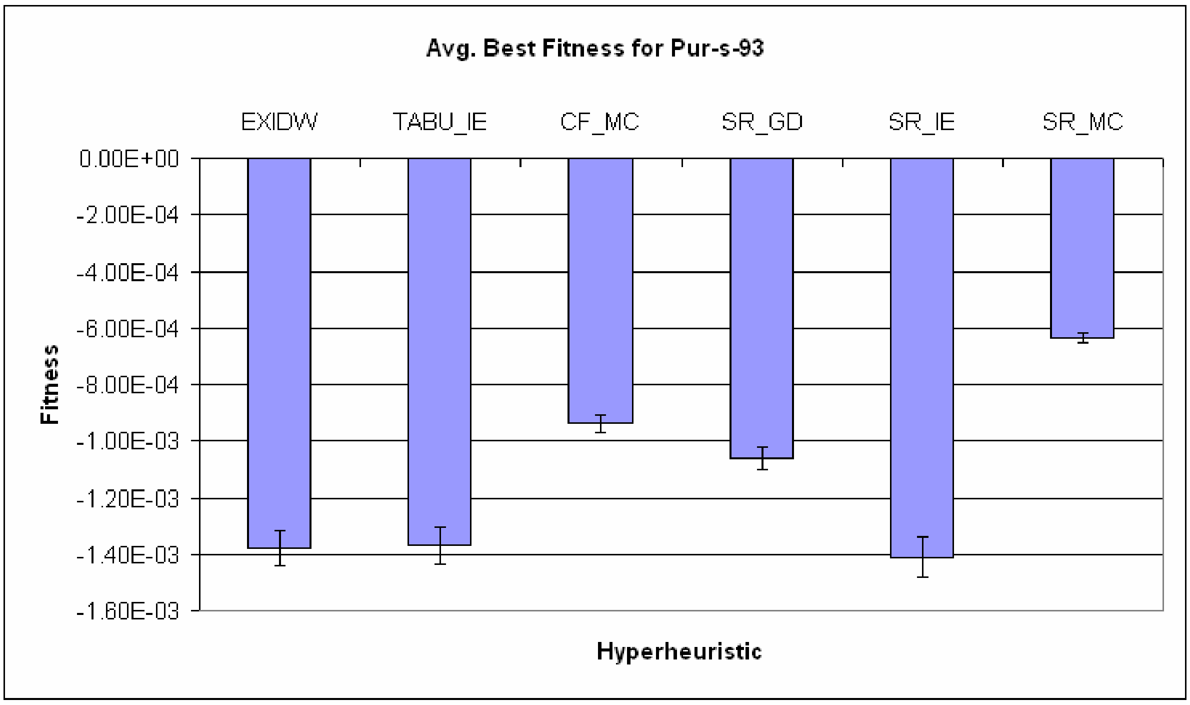

Figure B.7. Average Best Fitness values for Pur-s-93

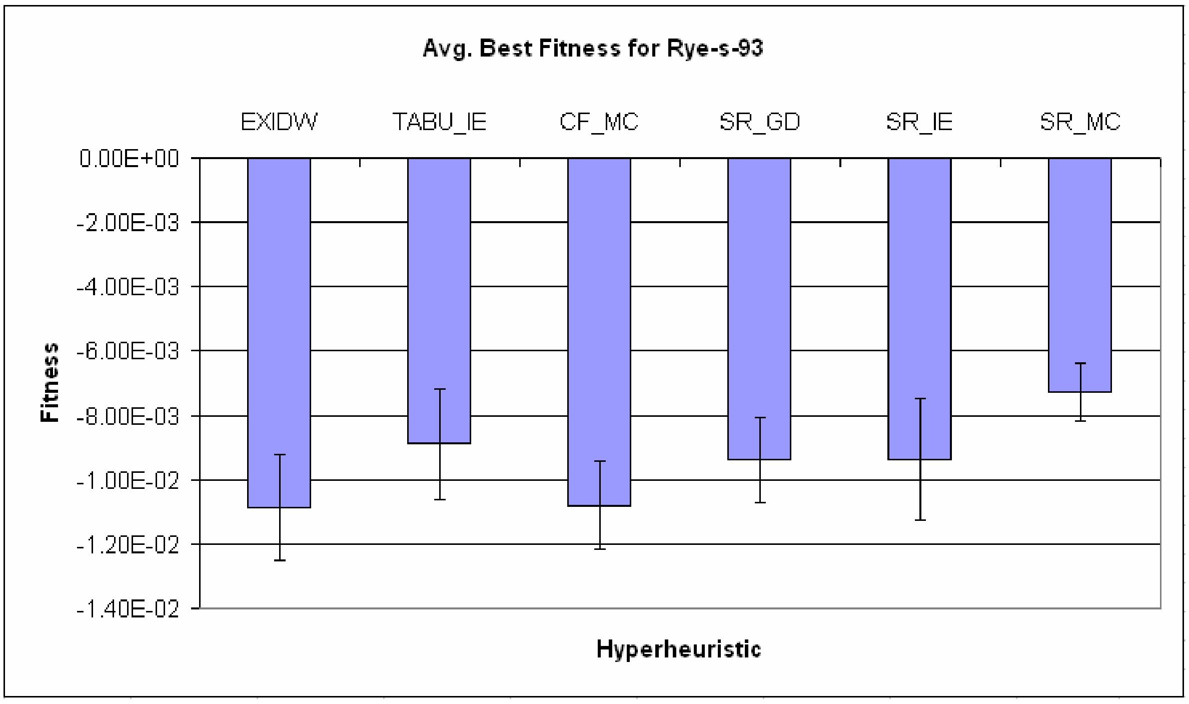

Figure B.8. Average Best Fitness values for Rye-s-93

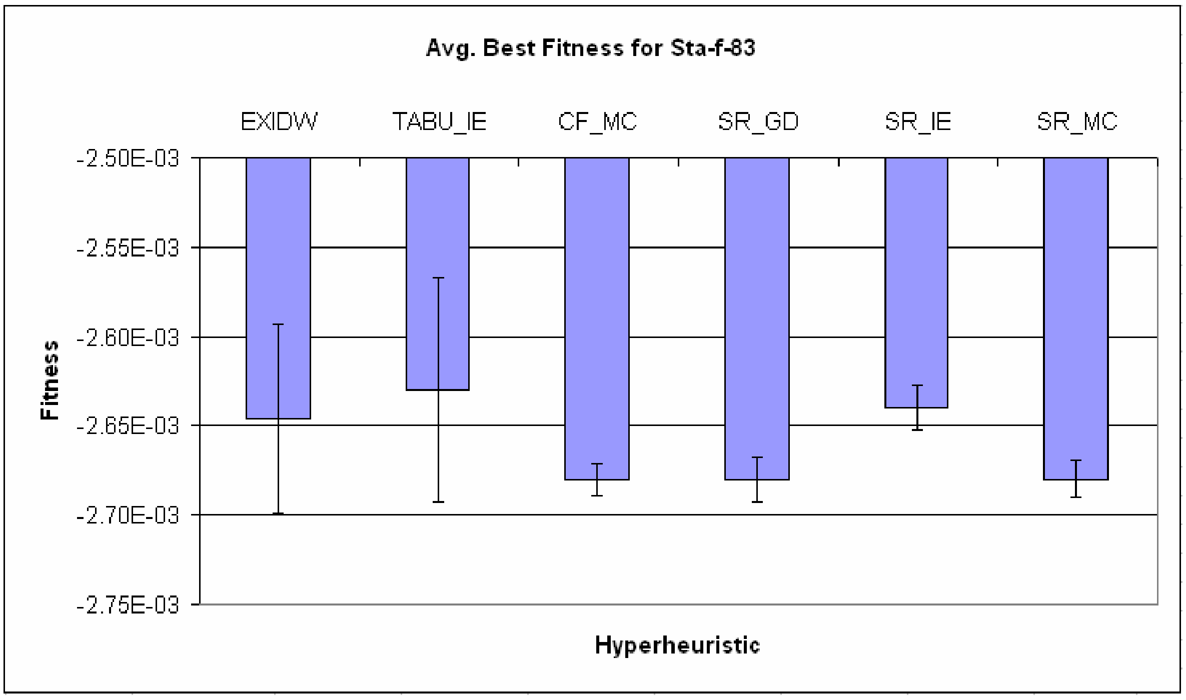

Figure B.9. Average Best Fitness values for Sta-f-83

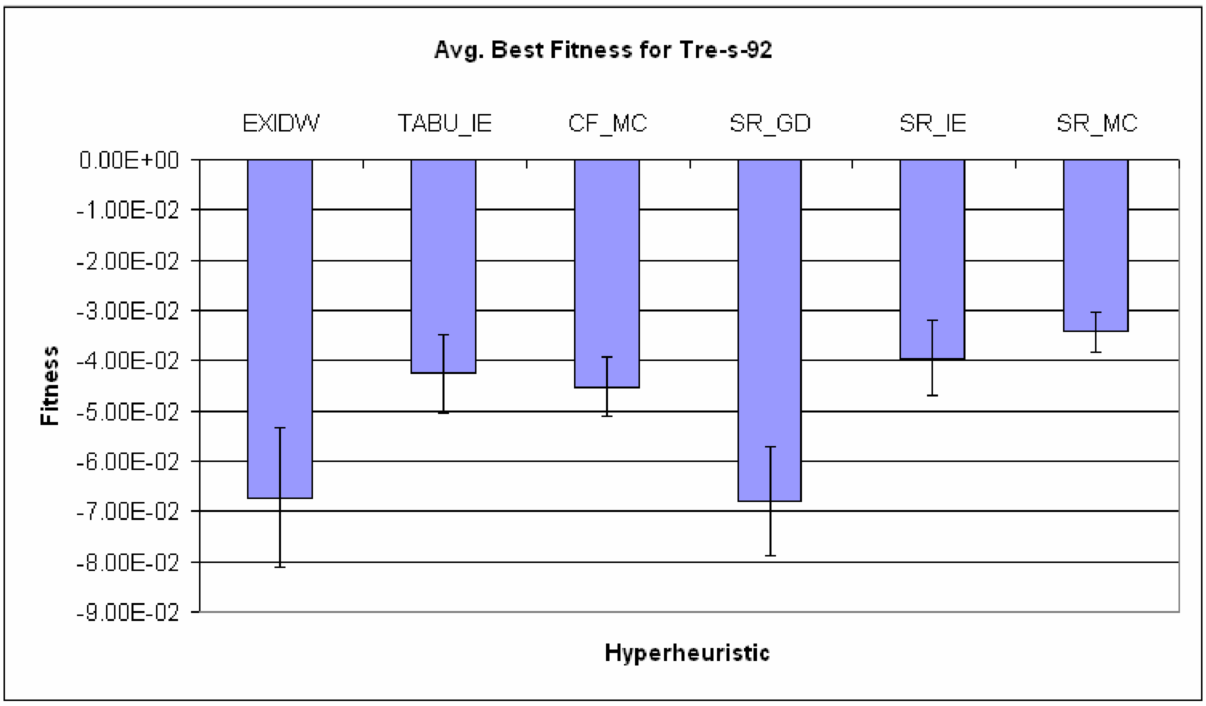

Figure B. 10. Average Best Fitness values for Tre-s-92

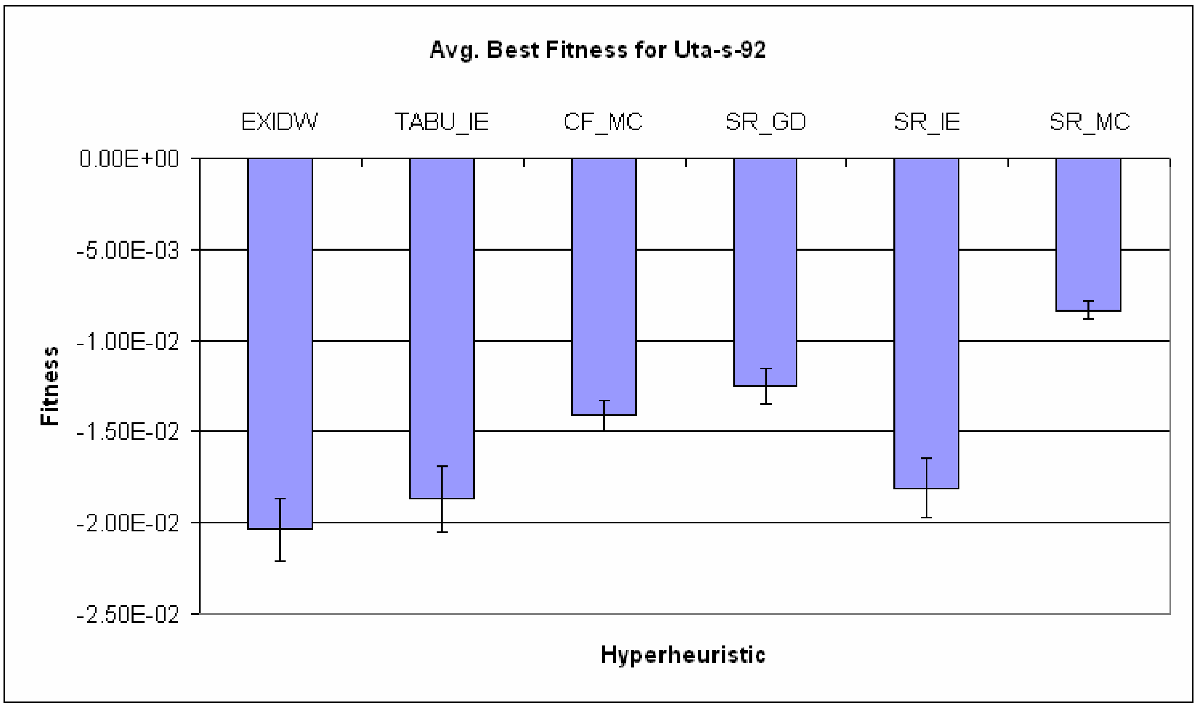

Figure B.11. Average Best Fitness values for Uta-s-92

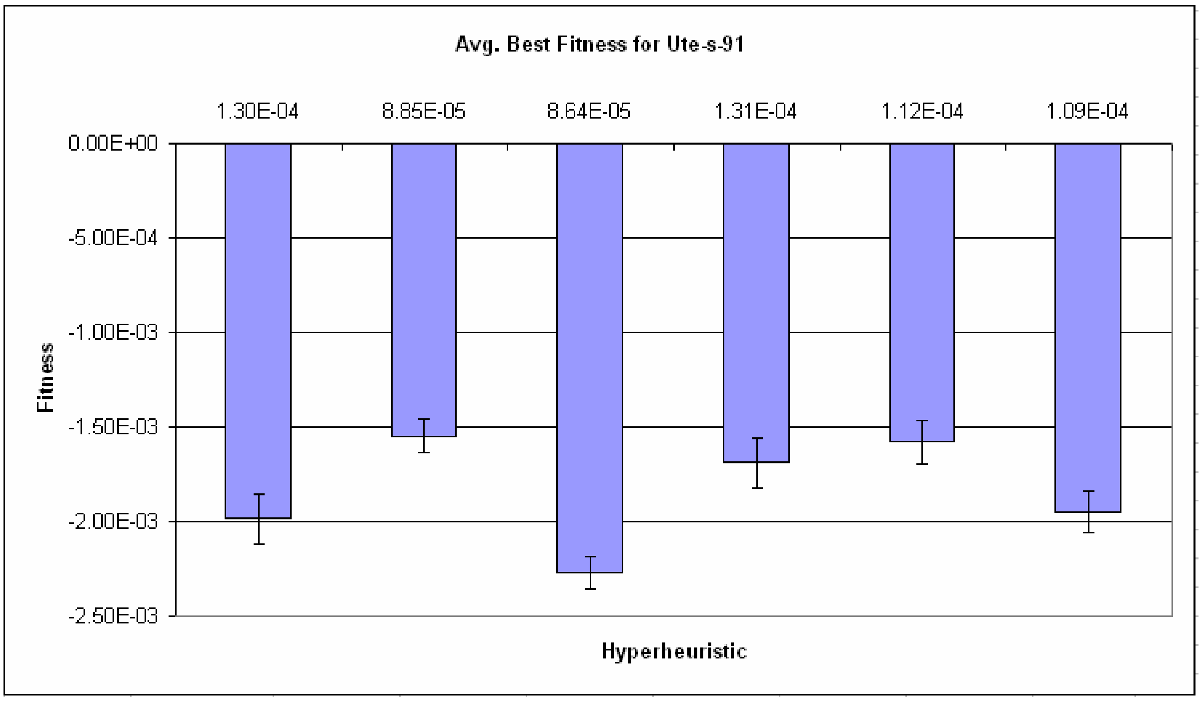

Figure B.12. Average Best Fitness values for Ute-s-91

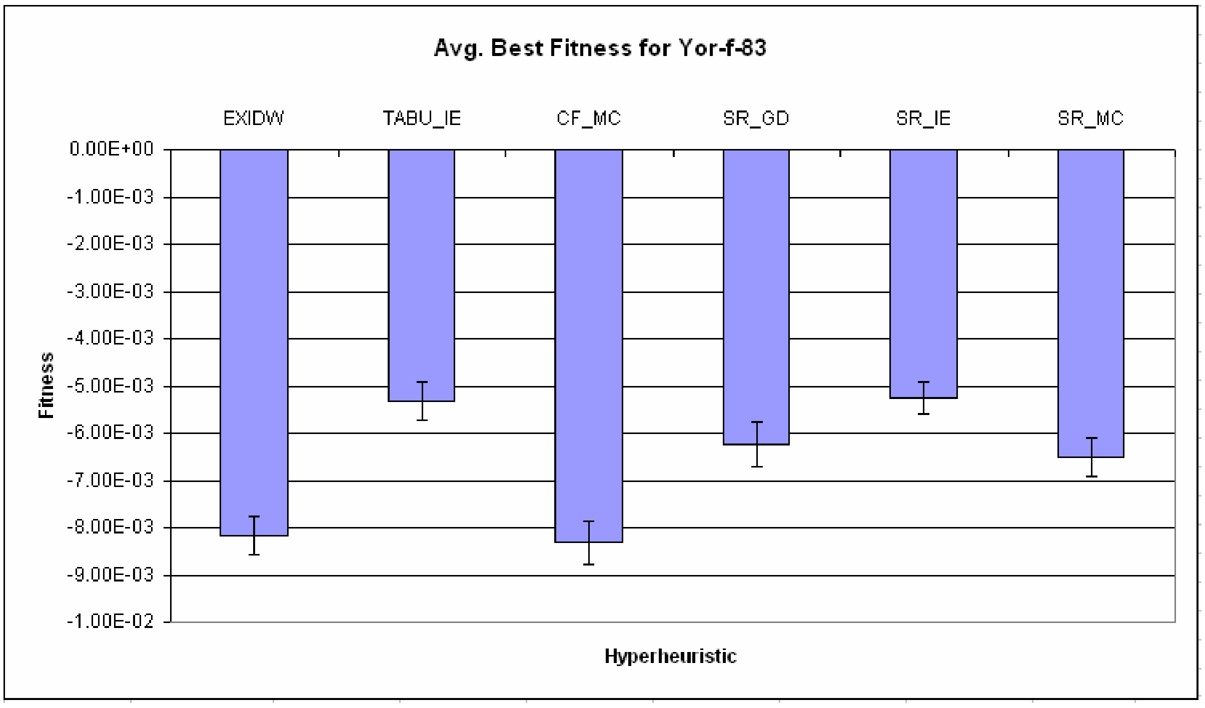

Figure B.13. Average Best Fitness values for Yor-f-83

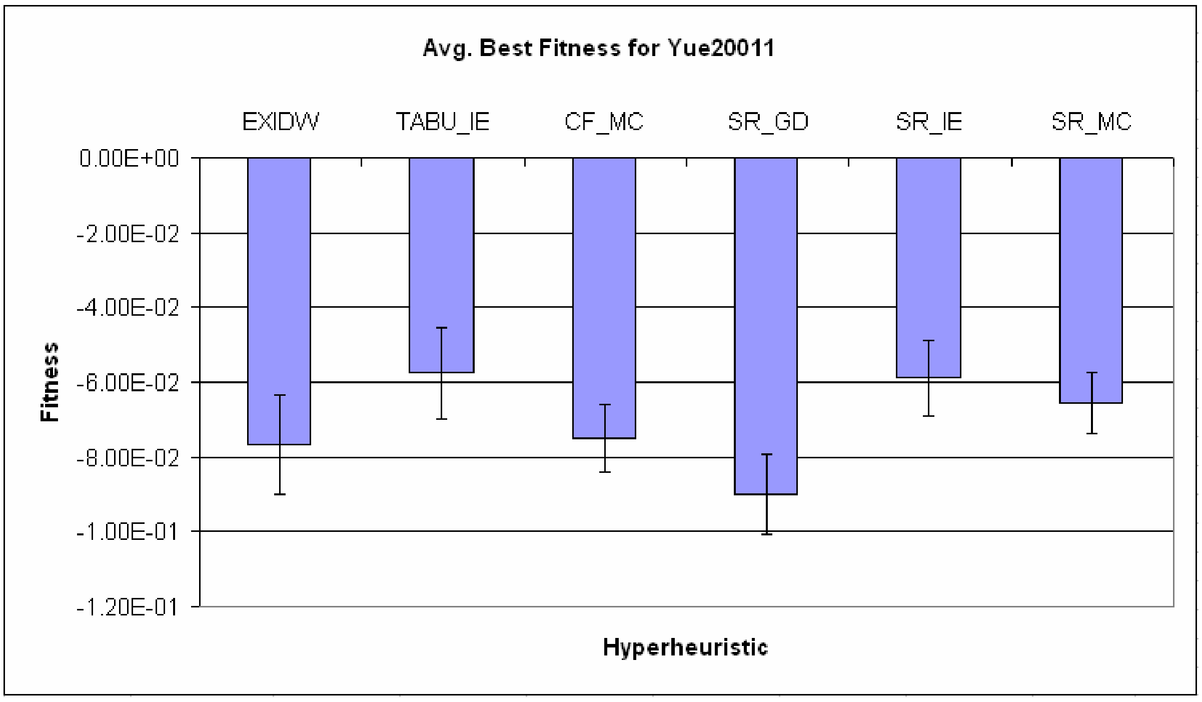

Figure B.14. Average Best Fitness values for Yue20011

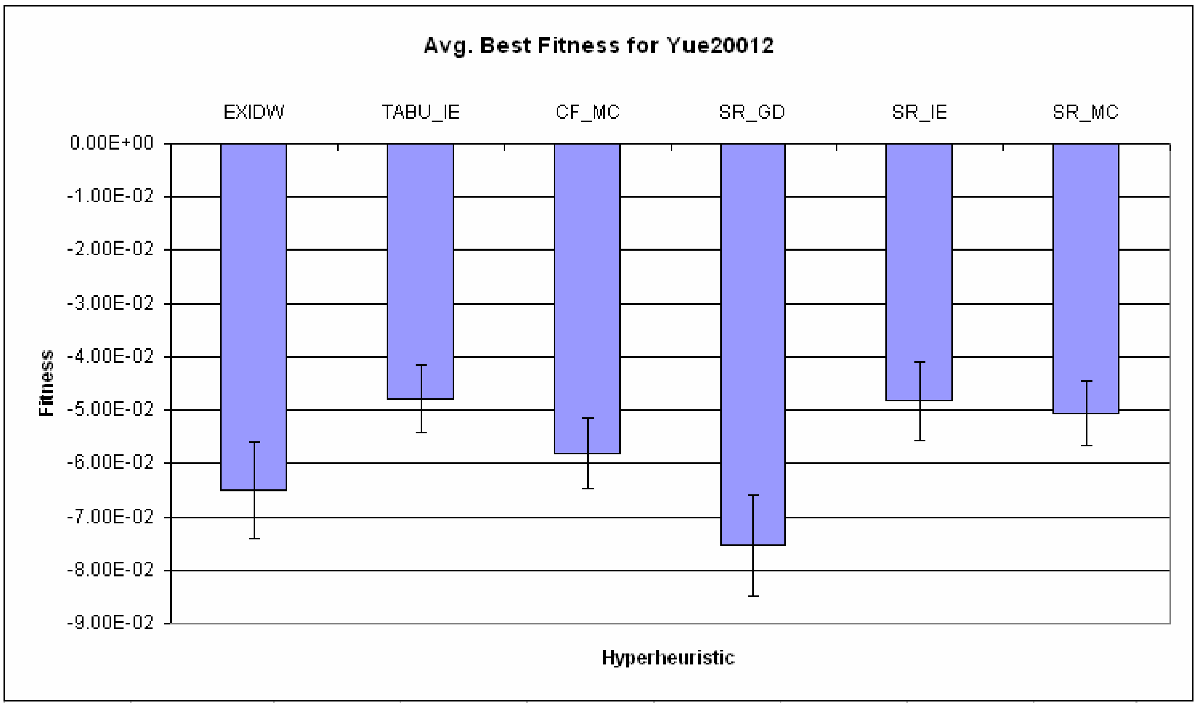

Figure B.15. Average Best Fitness values for Yue20012

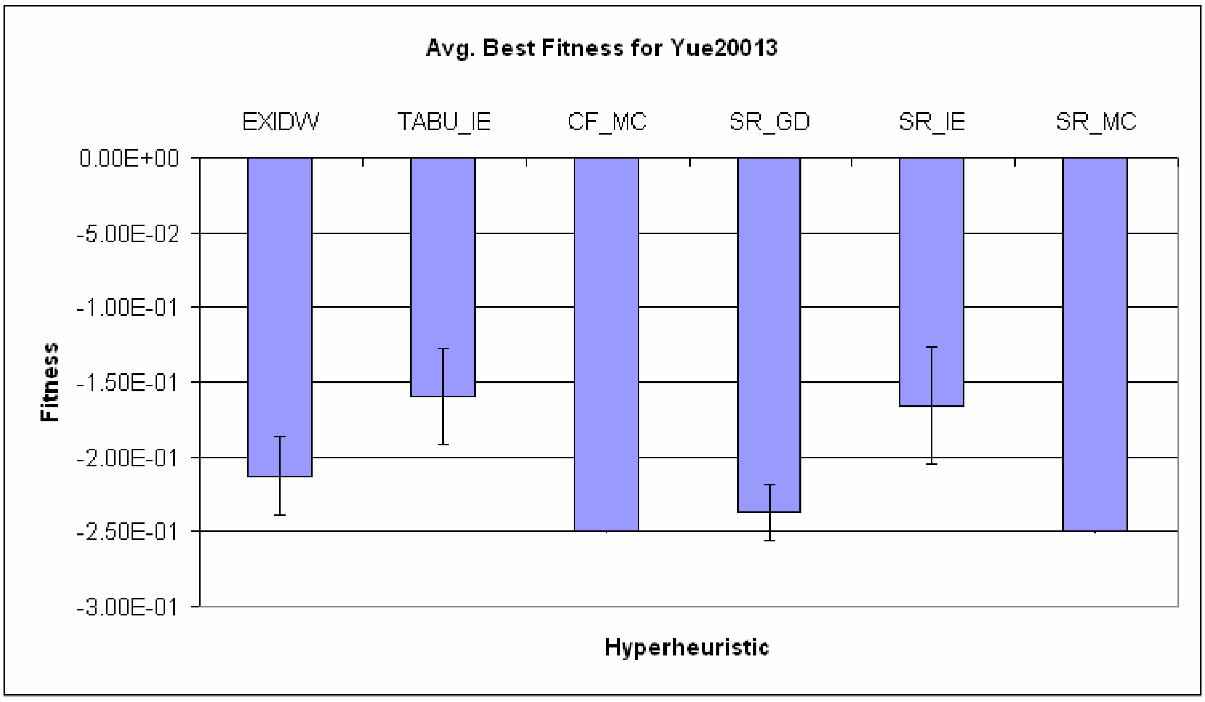

Figure B.16. Average Best Fitness values for Yue20013

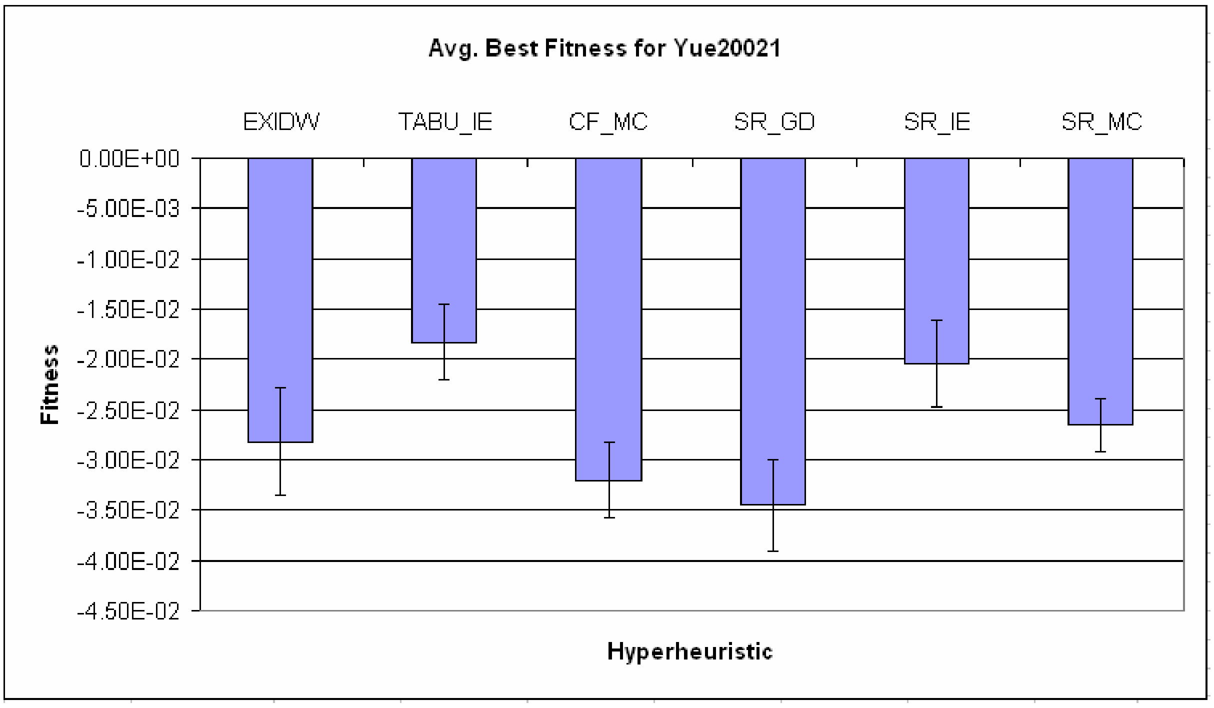

Figure B.17. Average Best Fitness values for Yue20021

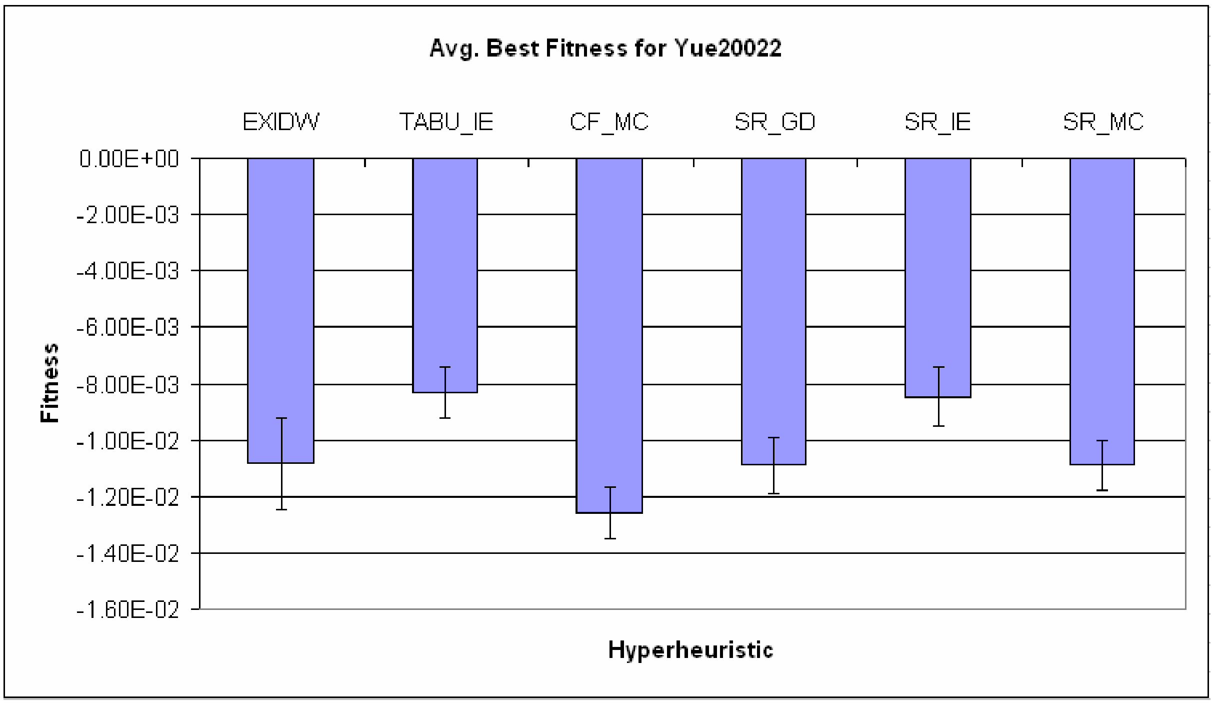

Figure B.18. Average Best Fitness values for Yue20022

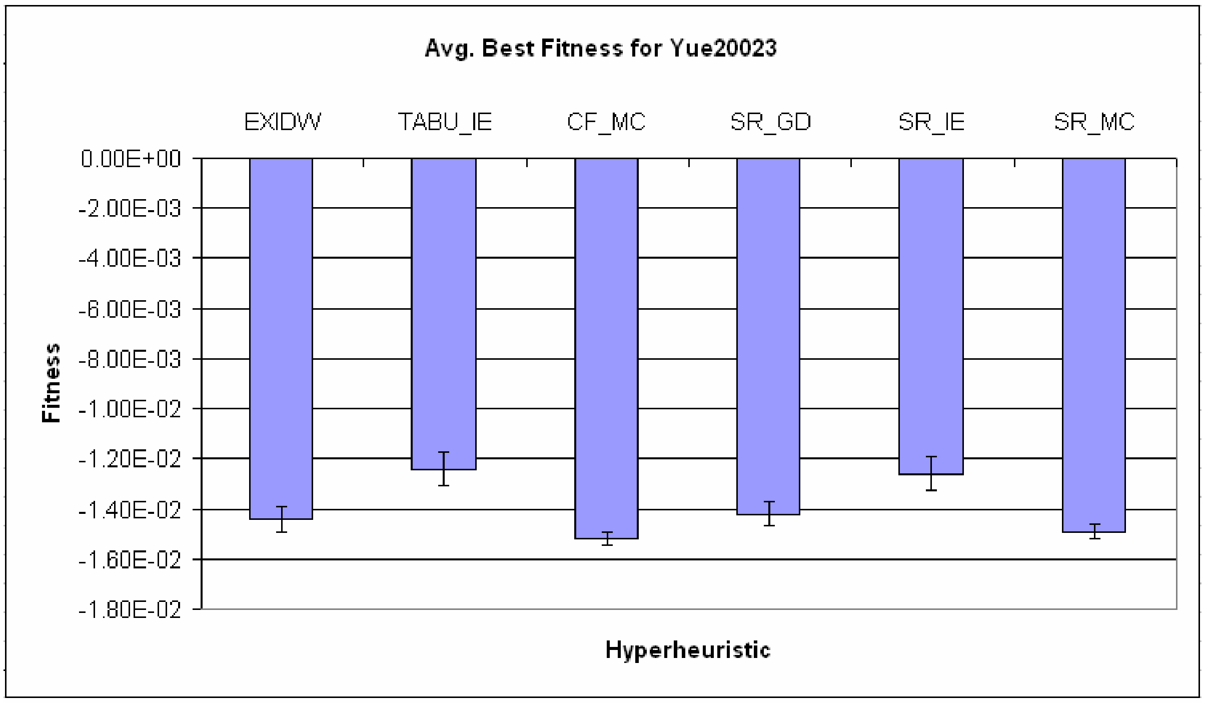

Figure B.19. Average Best Fitness values for Yue20023

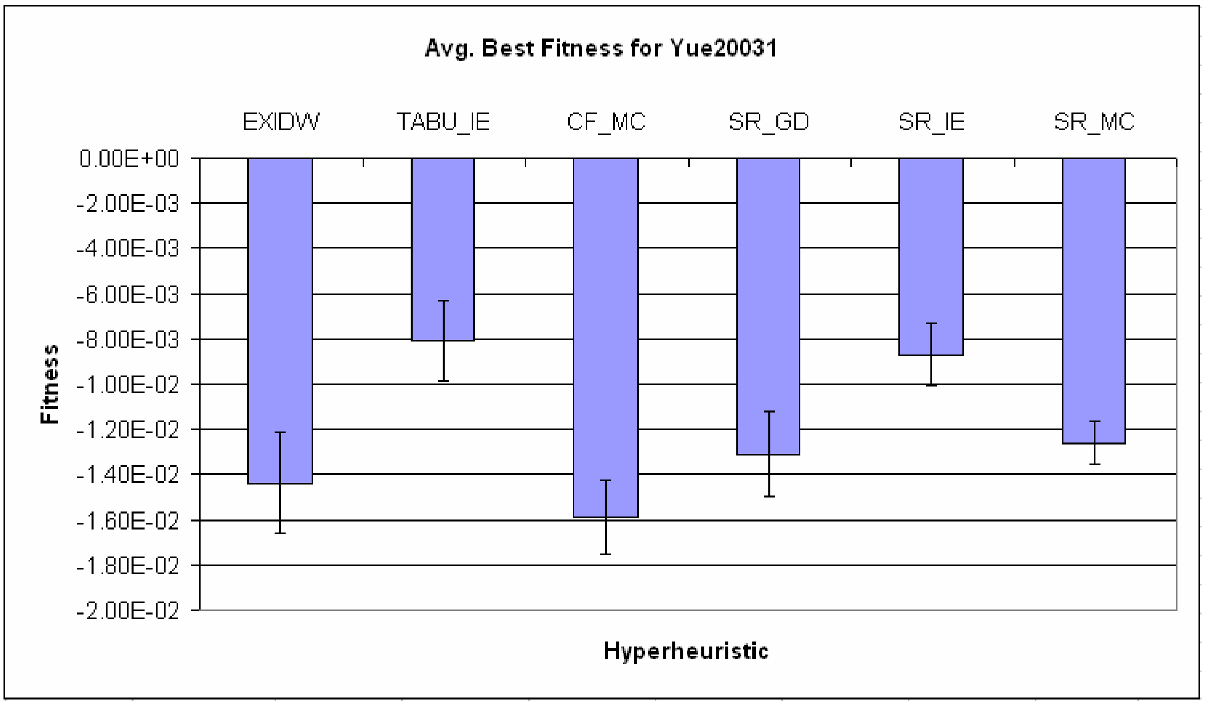

Figure B.20. Average Best Fitness values for Yue20031

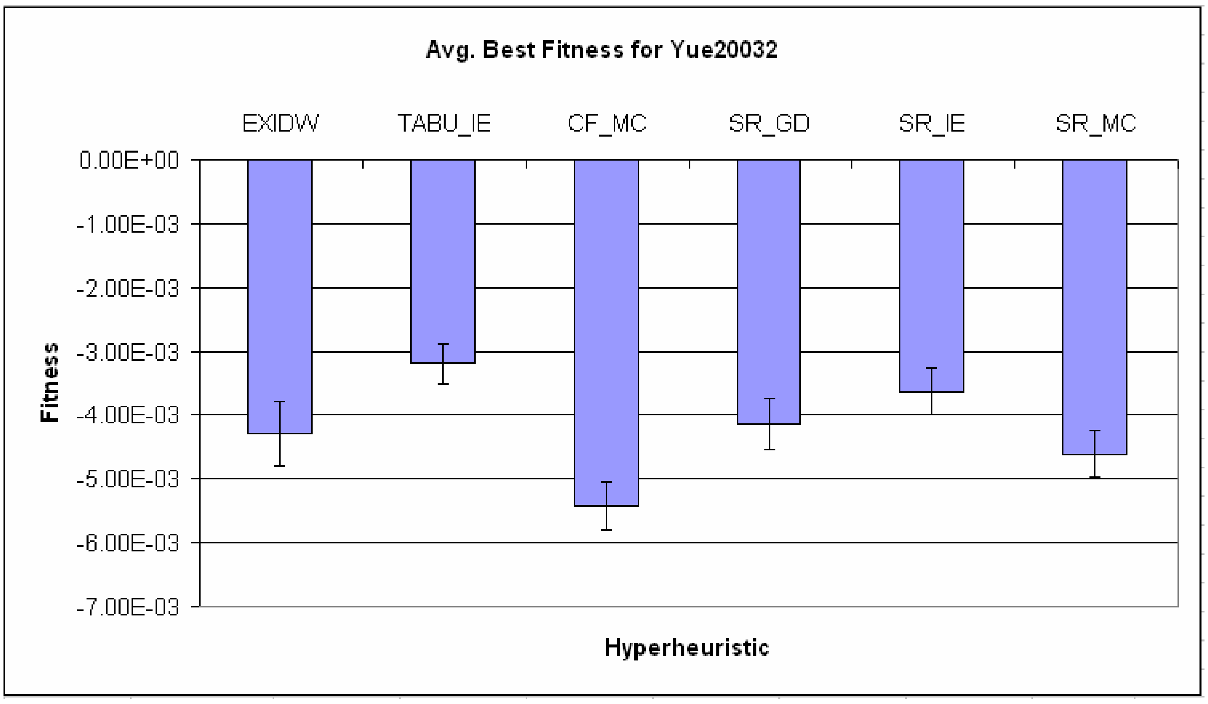

Figure B.21. Average Best Fitness values for Yue20032

REFERENCES

[1]Neveu B., Trombettoni G. and Glover F., "*IDWalk: A Candidate List Strategy with a Simple Diversification Device*", Lecture Notes in Computer Science 2004 ISSU 3258, page(s) 423-437 Springer-Verlag.

[2]Ozcan E., Bilgin B. and Korkmaz E.E., "*Hill Clibers and Mutational Heurstics in Hyperheuristics*".

[3]Poli R. and Langdon W.B., "*Backward-chaining evolutionary algorithms*", Artificial Intelligence - Elsevier 2006 vol 170; number 11, page(s) 953-982 Elsevier Science B.V., Amsterdam.

[4]Burke E., Hart E., Kendall G., Newall J., Ross P. and Schulenburg S., "*Hyper-heuristics: An Emerging Direction in Modern Search Technology*", International Series in Operations Research and Management Science 2003 ISSU 57, page(s) 457-474 Kluwer Academic Publishers.

[5]Kendall G., Soueiga E. and Cowling P., "*Choice Function and Random Hyperheuristics*", Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning, pp. 667-671.

[6]Ulker O., "*Graph Coloring Problems: A Short Survey*"

[7]Gomes C., Sellman M., van Es C., van Es H., "The Challenge of Generating Spatially Balanced Scientific Experiment Designs", In CP-AI-OR'04, Nice, 2004.

[8]Jaszkiewicz A., Kominek P., Kubiak M., "*Adaptation of the Genetic Local Search Algorithm to a Car Sequencing Problem*", 7th National Conference on Evolutionary Algorithms and Global Optimization, Kazimierz Dolny, Poland, pages 67-74, 2004.

[9]Ozcan E., Bilgin B. and Korkmaz E.E., "*A Comprehensive Analysis of Hyper-heuristics*".

[10]K. A. De Jong, "*An analysis of the behavior of a class of genetic adaptive systems*", PhD thesis, University of Michigan, 1975.

[11]Rastrigin, L.A., "Extremel Control System. In Theoritical Foundations of Engineering Cyberneics Series, Moscow, Nauka, Russian (1974)"

[12]Schwefel, H.P., "*Numerical Optimization of Computer Models, John Wiley & Sons (1981), English Translation of Numerische Optimierung von Computer Modellen mittels der Evoulution sstrategie (1977)*"

[13]Griewangk, A.O, "*Generalized Descent of Global Optimization, Journal of Optimization Theory and Applications, 34: 11.39 (1981)*"

[14]Ackley, D., "*An Empirical Study of Bit Vector Function Optimization. Genetic Algorithms and Simulated Annealing*, (1987) 170-215"

[15]Mitchell, M., and Forrest, S., "*Fitness Landscapes: Royal road Functions. Handbook of Evolutionary Computation, Baeck, T., Fogel, D., Michalewiz, Z., (Ed.), Institue of Physics Publishing and Oxford University* (1997)"

[16]Goldberg, D.E., "*Genetic Algorithms and Walsh Functions: Part I-II, A Gentle Introduction. Complex Systems* (1989) 129-152"

[17]Whitley, D., "*Fundemental Principles of Deception in Genetic Search*. In G.J.E. Rawlings (Ed.), Foundations of Genetic Algorithms, Morgan Kaufmann, San Matco, CA (1991)"

[18]Burke E.K. and Petrovic S., "*Recent Research Directions in Automated Timetabling*",

the European Journal of Operational Research, 140(2) 266-280, 2002.

[19]Even S., Itai A. and Shamir A., "*On the Complexity of Timetable and Multicommodity Flow Problems*", SIAM J. Comput., 5 (4) (1976) 691-703.

[20]Schaerf A., "*A Survey of Automated Timetabling*", Artificial Intelligence Review 13(2) 87-127.

[21]Burke E.K., De Causmaecker P., G. Vanden Berghe and H. Van Landeghem, "*The State of Art of Nurse Rostering*", Journal of Scheduling (2004): 441-499.

[22]M.A. Trick, "*A Schedule-then-Break Approach to Sports Timetabling*", Lecture Notes in Computer Science, 2001, ISSU 2079, pages 242-253.

[23]G. Schmidt, and T. Strohlein, "*Time Table Construction-An Annotated Bibliography*", The Computer Journal, 23(4):307-316, 1979.

[24] F.T. Leighton, "*A Graph Coloring Algorithm for Large Scheduling Problems*", Journal of Reasearch of the National Bureau of Standards, 84:489-506, 1979.

[25] M.W. Carter, G. Laporte, and S.T. Lee (1996) "*Examination timetabling: algorithmic strategies and applications.",* Journal of the Operational Research Society , 47:373-383.

[26] 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.

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

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33]Bilgin B., Ozcan E. and Korkmaz E.E., "*An Experimental Study oh Hyper-heuristics and Exam Timetabling*".