Towards Multi-objective Optimization of Automatic Design Space Exploration for Computer Architecture through Hyper-heuristic

Mustafa Latif
Department of Software Engineering,
NED University of Engineering & Technology,
Karachi, Pakistan
mustafalatif@neduet.edu.pk

Muhammad Ali Ismail
Department of Computer and Information Systems Engineering, NED University of Engineering & Technology, Karachi, Pakistan
maismail@neduet.edu.pk

Abstract—**Multi-objective optimization is an NP-hard problem. ADSE (automatic design space exploration) using heuristics has been proved to be an appropriate method in resolving this problem. This paper presents a hyper-heuristic technique to solve the DSE issue in computer architecture. Two algorithms are proposed. A hyper-heuristic layer has been added to the FADSE (framework for automatic design space exploration) and relevant algorithms have been implemented. The benefits of already existing multi-objective algorithms have been joined in order to strengthen the proposed algorithms. The proposed algorithms, namely RRSNS (round-robin scheduling NSGA-II and SPEA2) and RSNS (random scheduling NSGA-II and SPEA2) have been evaluated for the ADSE problem. The results have been compared with NSGA-II and SPEA2 algorithms. Results show that the proposed methodologies give competitive outcomes in comparison with NSGA-II and SPEA2.**

Keywords-hyper-heuristic; multi-objective optimization; design space exploration; x86 processor

I. INTRODUCTION

Computer architectures are getting more complex day by day. As a result, the optimization of multiple objectives becomes harder too. This occurs because usually the objectives are contradictory to each other. It has been recognized that automatic design space exploration (ADSE) is a good solution to the stated issue. In this work, selection hyper-heuristics have been applied in conjunction with All Moves as acceptance criteria. Simple random selection hyper-heuristic method has been used. This hyper-heuristic method is quite dominant in the field because of its execution according to the circumstances. This heuristic type uses existing heuristic methods [1]. Numerous low level heuristics can be combined depending on the problem. Commonly, there are two modules of the traditional selection hyper-heuristic model [2]. These include low level heuristics selection and acceptance. Selection hyper-heuristic has been extensively used in different optimization issues, some of them discussed below. But, to the best of our knowledge, the literature is relatively poor in the application of selection hyper-heuristics in the ADSE field using the FADSE tool.

In [3], a meta optimization function that uses various meta-heuristics simultaneously to obtain better results in appropriate time has been introduced. The difference in our approach and in [3] is the use of round-robin and the random hyper-heuristic introduction in FADSE to optimize the process of DSE. In this work, several meta-heuristic approaches in random and round-robin have been used to acquire the optimized solution in appropriate time. Our scheme has been improving the convergence speed. Additionally, it also provides diversified multi-objective optimizations of computer architecture DSE problem. An additional layer has been added to FADSE. The modified hyper-heuristic algorithms RRSNS (Round Robin Scheduling NSGA-II [16] and SPEA2 [11]) and RSNS (Random Scheduling NSGA-II and SPEA2) are presented in this paper.

II. RELATED WORK

This section contains details about the methods and approaches used in this work. The use of the related variants in previous works is also discussed. The tools and simulators used for implementation and the algorithms used are presented in this section.

A. Selection Hyper-heuristic

Selection hyper-heuristic implementation is one of the focusing areas of this paper. In [12], authors analyzed multiple methodologies of hyper-heuristics for the scheduling problem. They claim that hyper-heuristics yield better results than meta-heuristics in some scenarios. However, the authors have not revealed the details of their introduced hyper-heuristic methods. In [4], a selection hyper-heuristic-based method has been introduced. Authors combined the two components of hyper-heuristic in a fashion that combines late acceptance with random selection criteria. The results have been compared with exiting approaches to evaluate the performance. It has been concluded that this combination provides remarkable results. In [13], a methodology has been introduced that uses selection hyper-heuristic. In addition, acceptance criteria used were stimulated. These acceptance criteria will help in providing feedback for future selections.

Corresponding author: Mustafa Latif

In [14], selection hyper-heuristic has been applied to dynamic optimization problems. Authors reported the sustainability of selection hyper-heuristic in dynamic environments. Classical choice function heuristics selection method and modified choice function heuristics selection method are variants of each other. In [5], it has been argued that modified choice function gives good results when low level heuristics are absent. Authors analyzed the results of modified choice function after the addition of crossover low level heuristics.

B. *FADSE Tool*

In this paper, the tool used for design exploration is FADSE (framework for automatic design space exploration). FADSE is the most commonly used tool for ADSE (automatic design space exploration) in computer architecture [6]. Authors incorporated jMetal library with FADSE to take benefit of a huge number of multi-objective algorithms. Different tools have been found which worked on heuristics before the development of FADSE, e.g. a statistical exploration algorithm based tool ArchExplorer [7] which requires design of the system to perform optimization. Another similar tool is M3Explorer [8] which is relatively more close to FADSE in its working. NASA [9] is also a popular tool for DSE. It has provided a simple interface for simulators to integrate. There are many other tools for DSE [6] but FADSE is more generalized and capable of solving complex computational problems.

C. *SNIPER Simulator*

In this work, SNIPER simulator has been used. It is a high speed simulator for x86 architecture. SNIPER has been developed for calculating the performance of a processor, the energy consumed by the processor and the integration area [15]. SNIPER has been extensively exploited for obtaining results.

D. *Multi-Objective Algorithms*

In this work, two evolutionary multi-objective algorithms of jMetal library have been exploited. The term multi-objective is used because in these problems, multiple contradictory goals need to be achieved. NSGA-II and SPEA2 used in this work are the most widely accepted algorithms to obtain the desired goals in this field [10]. In case of several objectives, the stated two algorithms perform better than other multi-objective algorithms [11].

TABLE I. NOTATIONS USED IN ALGORITHMS

$\mathbf{P_o}$	Initial randomly generated population
$\mathbf{P_{size}}$	Population size
$\mathbf{E_{max}}$	Maximum evaluation count
$\mathbf{E_{current}}$	Current evaluation count
$\mathbf{P_{solnew}}$	Population generated by low level heuristics
$\mathbf{P_{sol}}$	Problem solution

III. RESEARCH METHODOLOGY

This section presents the hyper-heuristic methodology used to accelerate the DSE process of a multi core processor. It has been done by using multiple meta-heuristics in random and round-robin fashion to obtain quasi-optimal solution. The objective is to obtain better results in appropriate time. This strategy improves convergence speed. Moreover, it also provides diversity of multi-objective optimization of DSE problem of computer architecture. FADSE framework [6] has been modified by adding a hyper-heuristic layer as shown in Figure 1. Two selection hyper-heuristic models (RRSNS and RSNS) have been introduced in FADSE. In RRSNS, round-robin fashion scheme [17] has been used to run two state-of-the-art multi-objective optimization algorithms (NSGA-II and SPEA2) as LLH (low level heuristic). There are two variations of RRSNS. RRSNS1 schedules NSGA-II first and then SPEA2 and in RRSNS2, SPEA2 has been scheduled first then NSGA-II. In RSNS, Simple random [12] hyper heuristic selection mechanism has been used to find the quasi-optimal solution. In both RRSNS and RSNS models All Moves [5] move acceptance method has been used.

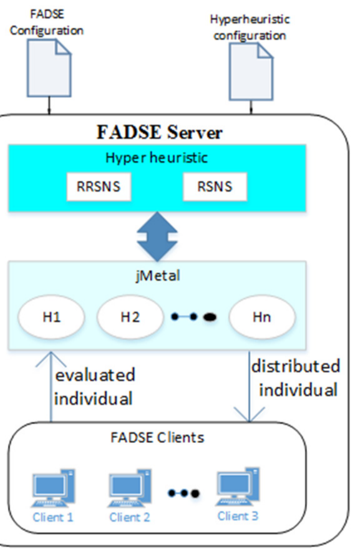

Fig. 1. FADSE with hyper-heuristic layer

Table I displays the notations used in this study. The pseudo codes of RRSNS and RSNS are shown in Figures 2 and 3 respectively. In RRSNS, FADSE is configured by using its configuration file in which the number of objectives, parameters of processor architecture to be tuned, and the database configuration to reused already simulated results have been defined. Also, the number of FADSE clients and configuration parameters of NSGA-II and SPEA2 meta-heuristic algorithms and their order of scheduling have been defined. After initialization, a random solution of the problem is produced and passes as a parent population to the first one in the round-robin order of LLH. LLH then applies the evolutionary operation (crossover, mutation) and generate offspring population. The generated individuals are evaluated using sniper simulator. Sniper simulator is used as the objective function of the ADSE process. When the first one produces the solution of the problem, it is passed to the next order LLH and so on. The last one gives its solution to the first on the LLH list. The iterative process continues until the termination condition is reached i.e. when the evaluation count reached its maximum limit defined in the initialization phase. In RSNS, the initialization stage is similar to the RRSNS except that the order of LLH is not defined. Instead, the order is generated randomly. All the generated solutions are accepted as parent population of the next iteration. The iteration is finished when the current evaluation count equals to the maximum evaluation count.

```
1  Procedure RRSNS()
   Begin
2  Initilization() /* initilize NSGA-II and SPEA2 algorithm, initial random
   population $P_o$, $P_{size}$, $E_{max}$   $E_{current}$=0*/
3  Selected_order_list_of_LLH=List_perturbative_LLH()
4  $P_{sol}$=$P_o$/* assign randomly generated solution as a problem solution*/
5  While $E_{current}$ < $E_{max}$ do
6  For i=1 to Selected_order_list_of_LLH.length
7  $P_{solnew}$=Generate_population_through_LLH($P_{sol}$)
8  $P_{sol}$=$P_{solnew}$ /*update the problem solution with newly generated solution*/
9  End loop
10 End while
11 Return $P_{sol}$
12 End
```

Fig. 2. RRSNS algorithm

```
1  Procedure RSNS()
   Begin
2  Initilization() /* initilize NSGA-II and SPEA2 algorithm, initial random
   population $P_o$, $P_{size}$, $E_{max}$   $E_{current}$=0*/
3  $P_{sol}$=$P_o$/* assign randomly generated solution as a problem solution*/
4  While $E_{current}$ < $E_{max}$ do
5  Selected_LLH=Randomly_select_LLH(List_of_perturbative_LLH()
6  $P_{solnew}$=Generate_population_through_LLH($P_{sol}$)
7  $P_{sol}$=$P_{solnew}$ /*update the problem solution with newly generated solution*/
8  End while
9  Return $P_{sol}$
10 End
```

Fig. 3. RSNS Algorithm

TABLE II. HARDWARE PARAMETERS TO TUNE

Microarchitecture parameter	Range of values		
	Min	**Max**	**Individual**
Number of cores	1	4	3
DRAM interleaving controllers	1	64	7
L1 data cache associativity	8	16	2
L1 data cache size [KB]	32	256	4
L2 cache associativity	8	16	2
L2 cache size [KB]	32	2048	7
L3 cache associativity	8	16	2
L3 cache size [KB]	128	32768	9
L3 cache shared cores	1	16	5

The DSE process of 211,680 configurations of the x86 multi-core processor has been used. The list of parameters and their possible values are shown in Table II. The possible values of each parameter are exponents of 2 i.e. in case of the number of cores, the possible values are {1, 2, 4}. The DSE process was done 5 times with 2 objectives and 6 x86 architectural parameters. Seven SPLASH-2 [18] benchmarks (fmm, lu.cont, ocean.cont, radix, water.sp, volrend, fft) were used to apply on each individual in the DSE process. Intel Quad core i7, 3.2GHz processor with 4GB RAM has been used to implement the DSE process. Two FADSE clients were used, running in distributed environment with one FADSE server. Fifty individuals per generation and a total of 50 generations were used. The configuration parameters of the multi-objective optimization algorithms used in the study are shown in Table III.

TABLE III. CONFIGURATION PARAMETERS

Parameters	Value
Crossover operator	Single point
Crossover probability	0.90
Mutation operator	Bit-flip
Mutation probability	0.16
Selection operator	Binary tournament
Population size	50
Archive size	50
Teremination condition	After 50 generations

IV. RESULTS

Comparison of the proposed algorithms (RRSNS and RSNS) has been done with already presented state-of-the-art multi-objective evolutionary algorithms (NSGA-II and SPEA2). Initially the two variations of RRSNS (RRSNS1 and RRSNS2 were compared. The main difference between these variations is that in RRSNS1, NSGA-II is scheduled first and in RRSNS2, SPEA2 is scheduled first using the coverage quality metric [19]. The coverage metric can be used to compare two different multi-objective algorithms. It gives quantitative results regarding the percent of an algorithms' individuals are non-dominated by the individuals of another algorithm. Figure 4 shows that the coverage of RRSNS1 is better than RRSNS2's. The results are mostly equal in the first 15 generations and then RRSNS1 gives better results. Figure 5 illustrates that both RRSNS1 and RSNS are competing each other after the 27th generation but overall the RSNS gives better results.

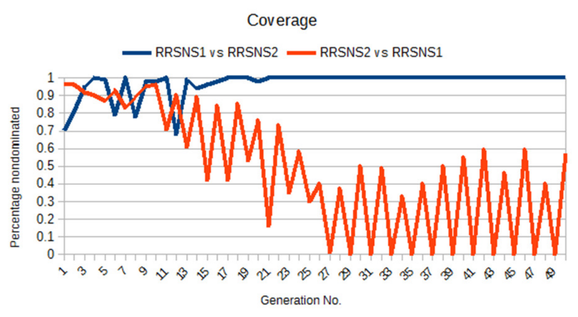

Fig. 4. Coverage comparison between RRSNS1 and RRSNS2

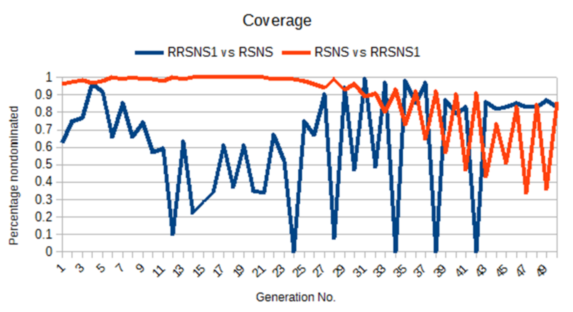

Fig. 5. Coverage comparison between RRSNS1 and RSNS

Figures 6 and 7 show that RRSNS and RSNS give better results than NSGA-II. Figures 8 and 9 show that RRSNS and RSNS give better coverage than SPEA2. In Figure 8 it can be seen that in the initial generation SPEA2 gives good readings but after the 17th generation RRSNS1 gives better results.

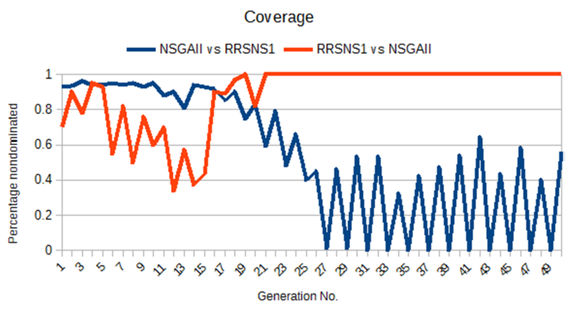

Fig. 6. Coverage comparison between NSGA-II and RRSNS1

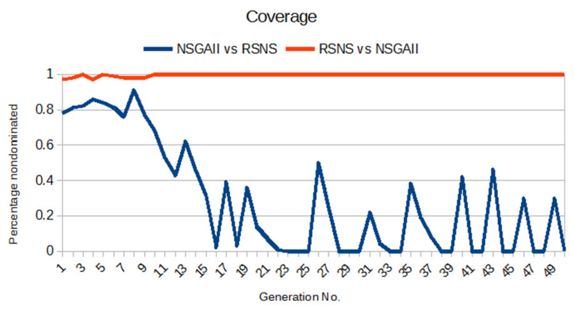

Fig. 7. Coverage comparison between NSGA-II and RSNS

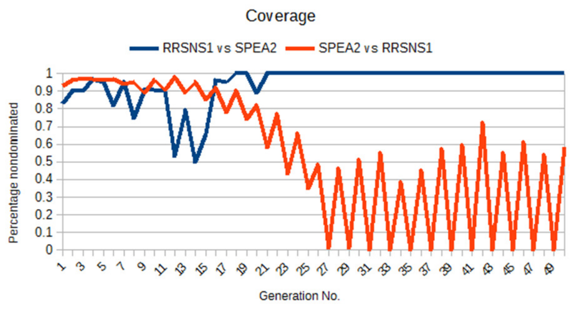

Fig. 8. Coverage comparison between RRSNS1 and SPEA2

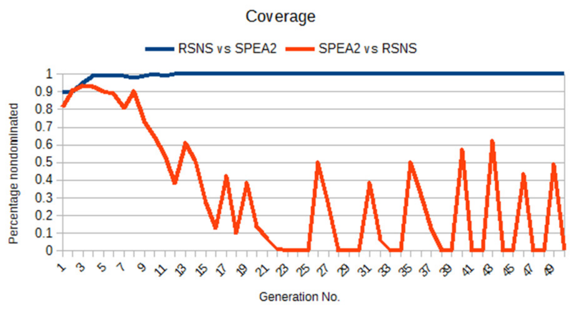

Fig. 9. Coverage comparison between RSNS and SPEA2

In Figure 10, the mentioned algorithms using hyper-volume metric [19] are compared. It can be seen that RRSNS1 convergences faster than all the other algorithms. SPEA2 reaches its maximum at the very beginning but stucks in local minima due to the archiving mechanism and density computation method (duplicate solution) [11]. RSNS is fast and reaches its maximum hyper-volume at the 16th generation as compared to the RRSNS1, who reached its maximum at 37th the generation. Both RRSNS1 and RSNS give better hyper-volume than NSGA-II and SPEA2 because they both use the strengths of both NSGA-II and SPEA2. The two set hyper-volume (TSHD) metric [20] shown in Figure 11 shows that ~94% of the hyper-volume is common to both the RRSNS1 and RSNS at the initial stage. After that the common hyper-volume percentage increases up to ~99.5%. Initially RSNS contribution in the non-dominated hyper-volume is good up to the 21th generation, then RRSNS1 contribution is greater than RSNS up to the final generation.

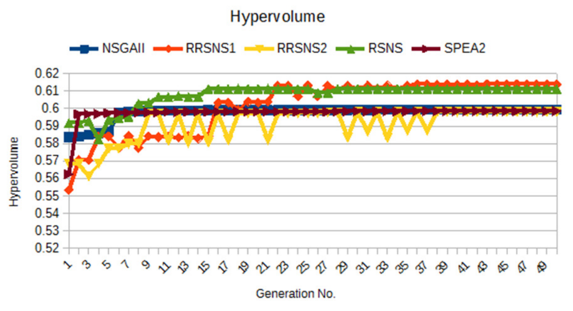

Fig. 10. Hypervolume comparison

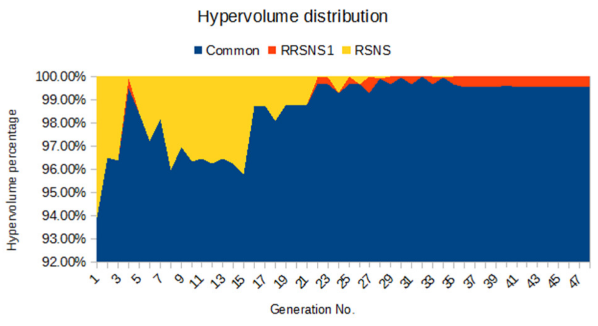

Fig. 11. Two set hypervolume difference (RRSNS1 vs RSNS)

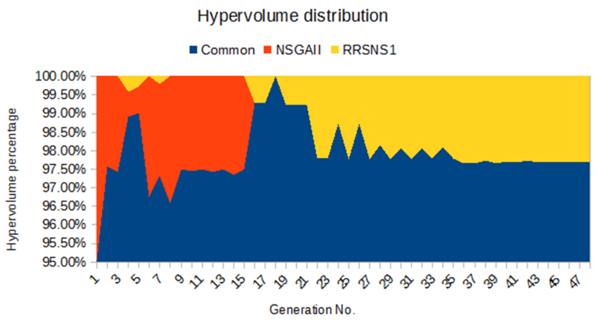

Fig. 12. Two set hypervolume difference (NSGA-II vs RRSNS1)

Figures 12 and 13 show the TSHD between NSGA-II vs RRSNS1 and NSGA-II vs RSNS. Figure 12 illustrates that NSGA-II non-dominated hyper-volume is good before the 17th generation. After that, RRSNS1 produces greater non-dominated hyper-volume, although all the time ~97% of the total hyper-volume is common among these two. It can be interpreted form Figure 13, that most of the time RSNS gives all non-dominated hyper-volume, but it can also be seen that the difference is only 2% and 98% non-dominated hyper-volume is common. Similarly, in Figures 14 and 15 the comparison of TSHD between RRSNS1 vs SPEA2 and RSNS vs SPEA2 is shown. It can be observed in both Figures that initially SPEA2 gives better solution, but later on RRSNS1 and RSNS produce superior hyper-volume than SPEA2. In Figure 16, the Pareto front approximation of all compared algorithms is shown.

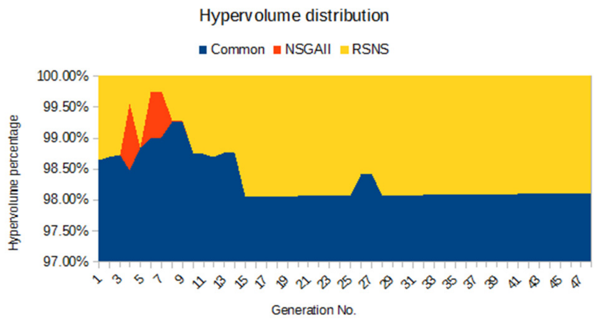

Fig. 13. Two set hypervolume difference (NSGA-II vs RSNS)

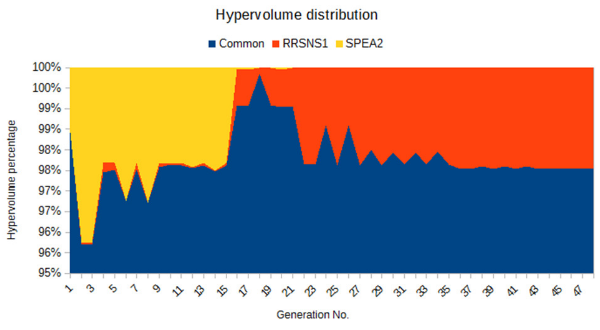

Fig. 14. Two set hypervolume difference (RRSNS1 vs SPEA2)

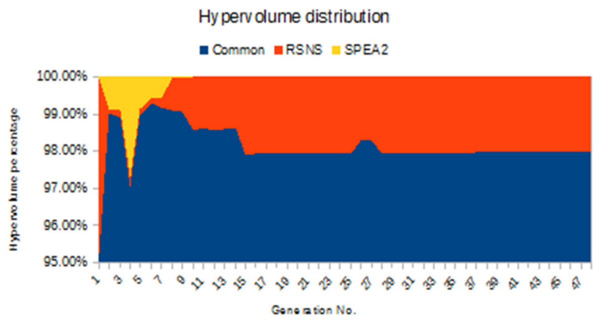

Fig. 15. Two set hypervolume difference (RSNS vs SPEA2)

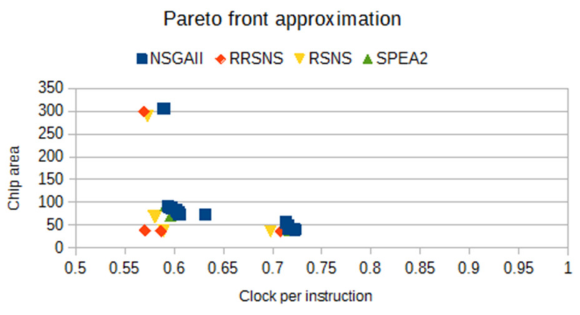

Fig. 16. Pareto front approximation RRSNS, RSNS, NSGA-II ans SPEA2

From Figure 16, it can be seen that RRSNS gives a unique solution which no other algorithm discovered during the ADSE process. Also, RSNS gives a more dominated solution as compared to NSGA-II and SPEA2.

V. Conclusion

Hyper-heuristic is a heuristic that runs on top of the existing or new low level heuristics to get the advantages of a wide range of existing or new heuristics to solve complex multi-objective optimization problems. In this paper, two hyper-heuristic selection algorithms have been proposed to solve the DSE process of an x86 multi core processor. A new hyper-heuristic layer has been added in FADSE and two algorithms were proposed are implemented. The biggest advantage of our proposed algorithms (RRSNS, RSNS) is that they use the combined advantages of existing multi-objective algorithms to solve the DSE problem. The proposed RRSNS and RSNS algorithms have been evaluated for ADSE problem and compared with two state-of-the-art multi-objective algorithms, NSGA-II and SPEA2. Results indicate that the proposed approaches give competitive results compared to NSGA-II and SPEA2.

In the future, more existing multi-objective algorithms can be added in RRSNS and RSNS so that they produce more diversified and prominent solutions for the DSE process of computer architecture.

References

[1] E. K. Burke, M. R. Hyde, G. Kendall, G. Ochoa, E. Ozcan, J. R. Woodward, “A Classification of Hyper-Heuristic Approaches: Revisited”, in: Handbook of Metaheuristics, pp. 453-477, Springer, 2019

[2] S. S. Choong, L. P. Wong, C. P. Lim, “Automatic design of hyper-heuristic based on reinforcement learning”, Information Sciences, Vol. 436-437, pp. 89-107, 2018

[3] L. Vintan, R. Chis, M. A. Ismail, C. Cotofana, “Improving computing systems automatic multiobjective optimization through meta-optimization”, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 35, No. 7, pp. 1125-1129, 2016

[4] W. G. Jackson, E. Ozcan, J. H. Drake, “Late Acceptance-Based Selection Hyper-Heuristics for Cross-Domain Heuristic Search”, 13th UK Workshop on Computational Intelligence, Guildford, UK, September 9-11, 2013

[5] J. H. Drake, E. Ozcan, E. K. Burke, “A Modified Choice Function Hyper-Heuristic Controlling Unary and Binary Operators”, IEEE Congress on Evolutionary Computation, Sendai, Japan, May 25-28, 2015

[6] H. A. Calborean, Multi-Objective Optimization of Advanced Computer Architectures Using Domain-Knowledge, PhD Thesis, University of Sibiu, 2011

[7] V. Zaccaria, G. Palermo, F. Castro, C. Silvano, G. Mariani, "Multicube Explorer: An Open Source Framework for Design Space Exploration of Chip Multi-Processors", 23th International Conference on Architecture of Computing Systems 2010, Hannover, Germany, February 22-23, 2010

[8] C. Silvano, W. Fornaciari, G. Palermo, V. Zaccaria, F. Castro, M. Martinez, S. Bocchio, R. Zafalon, P. Avasare, G. Vanmeerbeeck, C. Ykman-Couvreur, M. Wouters, C. Kavka, L. Onesti, A. Turco, U. Bondik, G. Mariani, H. Posadas, E. Villar, C. Wu, F. Dongrui, Z. Hao, T. Shibin, "Multicube: Multi-Objective Design Space Exploration of Multi-Core Architectures", IEEE Computer Society Annual Symposium on VLSI, Kefalonia, Greece, July 5-7, 2010

[9] Z. J. Jia, A. D. Pimentel, M. Thompson, T. Bautista, A. Nunez, "NASA: A Generic Infrastructure for System-Level MP-SoC Design Space Exploration", 8th IEEE Workshop on Embedded Systems for Real-Time Multimedia, Scottsdale, USA, October 28-29, 2010

[10] J. J. Durillo, A. J. Nebro, "jMetal: A java framework for multi-objective optimization", Advances in Engineering Software, Vol. 42, No. 10, pp. 760-771, 2011

[11] E. Zitzler, M. Laumanns, L. Thiele, SPEA2: Improving the Strength Pareto Evolutionary Algorithm, TIK-Report, Vol. 103, Swiss Federal Institute of Technology, 2001

[12] P. Cowling, G. Kendall, E. Soubeiga, "A Hyperheuristic Approach to Scheduling a Sales Summit", in: Lecture Notes in Computer Science, Vol. 2079, Springer, 2001

[13] R. Bai, J. Blazewicz, E. K. Burke, G. Kendall, B. McCollum, "A simulated annealing hyper-heuristic methodology for flexible decision support", 4OR, Vol. 10, No. 1, pp. 43-66, 2012

[14] B. Kiraz, A. S. Etaner-Uyar, E. Ozcan, "Selection hyper-heuristics in dynamic environments", Journal of the Operational Research Society, Vol. 64, No. 12, pp. 1753-1769, 2013

[15] R. Chis, A. Florea, C. Buduleci, L. Vintan, "Multi-objective optimization for an enhanced multi-core SNIPER simulator", Proceedings of the Romanian Academy-Series A, Vol. 19, No. 1, pp. 85-93, 2018

[16] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II", IEEE Transactions on Evolutionary Computation, Vol. 6, No. 2, pp. 182-197, 2002

[17] A. Kheiri, E. Ozcan, "A Hyper-Heuristic with a Round Robin Neighbourhood Selection", in: Lecture Notes in Computer Science, Vol. 7832, Springer, 2013

[18] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, A. Gupta, "The SPLASH-2 Programs: Characterization and Methodological Considerations", 22nd Annual International Symposium on Computer Architecture, Santa Margherita Ligure, Italy, June 22-24, 1995

[19] C. A. C. Coello, G. B. Lamont, D. A. Van Veldhuizen, Evolutionary Algorithms for Solving Multi-Objective Problems, Springer, 2007

[20] E. Zitzler, L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach", IEEE Transactions on Evolutionary Computation, Vol. 3, No. 4, pp. 257-271, 1999