Modified ISR hyper-heuristic for tuning automatic genetic clustering chromosome size

M H Adnan\textsuperscript{1}, M F Hassan\textsuperscript{2}, I A Aziz\textsuperscript{2}, O Nurika\textsuperscript{3} and M S Husain\textsuperscript{4}

\textsuperscript{1}Faculty of Art, Computing and Creative Industry, Universiti Pendidikan Sultan Idris, 35900, Tanjung Malim, Perak Darul Ridzuan, Malaysia
\textsuperscript{2}Computer and Information Sciences Department, Universiti Teknologi PETRONAS, 32610 Seri Iskandar, Perak, Malaysia
\textsuperscript{3}DGIT Systems, 3000, Level 4, 313 La Trobe Street, Melbourne, Victoria, Australia
\textsuperscript{4}Ibri College of Applied Sciences, Ministry of Higher Education (MoHE), Sultanate of Oman
E-mail: mhariz@fskik.upsi.edu.my, mfadzil_hassan@utp.edu.my, izzatdin@utp.edu.my, onurika@dgitsystems.com, mshahid.ibr@cas.edu.om

Abstract. Recent works using hyper-heuristics for solving clustering problems have been focusing on Genetic Algorithm. However, to the best of this research knowledge, no work is using hyper-heuristics dedicated for tuning the Genetic algorithm’s chromosome size for automatic clustering problem. The ability to tune the chromosome size is important because it allows the automatic clustering algorithm to be adaptive and dynamic. This paper proposes and evaluates a modified Improvement Selection Rules hyper-heuristic algorithm for tuning automatic genetic clustering chromosome size. The paper reviews related works of Genetic algorithm’s parameters tuning and selective hyper-heuristic algorithms and proposes a modified algorithm. The Iris, Breast Cancer, Wine and E-coli datasets are used for evaluation of the algorithm, based on the fitness, accuracy and robustness. The results indicate that the hyper-heuristic algorithm has produced good performance (fitness) and accuracy but consume considerably higher execution times.

1. Introduction
Hyper-heuristics is defined as the automatic design of dispatching rules using evolutionary computation and machine learning techniques [1]. Hyper-heuristics provide an alternative methodology to meta-heuristics which permit adaptive selection and/or generation of meta-heuristics automatically during the search process [2-5]. Hyper-heuristics permit the integration of two or more meta-heuristic search operators from different meta-heuristics through one defined parent heuristic via non-domain feedback, i.e. meta-heuristic to choose meta-heuristics [2-5]. The selection of a particular search operator to be used in any particular instance can be adaptively and dynamically decided based on the feedback from its previous performance [2]. With meta-heuristics integration/hybridization, each algorithm can exploit the strengths and cover the weaknesses of the collaborating algorithms [2-8].

At present, numerous hyper-heuristics have been developed to investigate Wireless Sensor Network (WSN), multi objective optimization, combinatorial optimization, expensive optimization, agent optimization, scheduling, fog computing, open vehicle routing, prediction, classification, pairwise testing, hybridization of nature inspired algorithms, travelling thief problem (TTP), cyber security and timetabling problem [1, 5, 9-31].
Recent works using hyper-heuristics for solving clustering problems have been focusing on GA (Genetic Algorithm) and Particle Swarm Optimization (PSO) [32-37]. GA parameters, such as population size, its operations and maximum number of generation are tuned using hyper-heuristics by combining the advantages of two or more meta-heuristics [32-35]. However, to the best of this research knowledge, no work is using hyper-heuristics dedicated for tuning the GA’s chromosome size for automatic clustering problem. By not using the hyper-heuristic algorithm, obtaining a dynamic and optimal Genetic Clustering Algorithm for unknown number of clusters (GCUK) chromosome size from a single meta-heuristic would be difficult, as suggested by the No Free Lunch Theorem [2].

Automatic clustering methods using nature-inspired meta-heuristics such as the GA are being progressive but are facing several challenges [38]. For example, the automatic clustering methods need to be enhanced for multi-attributes optimization [38]. In addition, there are limitations such as sub-optimal clustering accuracy and overlapping clusters [39].

The goal of this paper is to propose and evaluate a modified hyper-heuristic algorithm for tuning automatic GA-based clustering chromosome size. The organisation of the paper is as follows. Section 2 includes a related work which provides approaches to tune meta-heuristics parameters and operators related to the GA. Section 3 provides a modified Improvement Selection Rules (ISR) hyper-heuristics for tuning automatic genetic clustering chromosome size. Afterward, Section 4 covers the experimental results and the discussions of the results. Section 5 gives the conclusions and the future works.

2. Related work

Hyper-heuristics have been proposed for GA and designed to tune the parameters and operators of the GAs [32-37]. The related works are summarized in table 1.

| Work  | Descriptions |
|-------|--------------|
| [32]  | Discussed self-adaptation in evolutionary algorithms where the parameters in chromosome co-evolve with the solutions. |
| [33]  | A fuzzy hyper-heuristic framework for GA parameters tuning such as population size, the probability of breeding, mutation probability and maximum number of generations. |
| [34]  | A multi-objective hyper-heuristic genetic algorithm (MHypGA) that incorporates twelve low-level heuristics which are based on different methods of selection, crossover and mutation operations. |
| [35]  | Discussed self-adaptation in the meta-evolutionary algorithm (EA) where EA used to tune GA parameters such as crossover rate, mutation and population size. |

Based on table 1, hyper-heuristics have been used to tune the GA’s population size, selection, crossover, mutation processes, and maximum number of generations. However, according to the researcher’s knowledge, no work is dedicated to tuning the GA’s chromosome size for automatic clustering problem. It has been acknowledged that GA needs to be fine-tuned to specific problems and problem instances, using parameter control [33, 35].

There are two types of hyper-heuristics, i.e. generative and selective [2]. A generative hyper-heuristic combines low-level heuristics (LLH) while a selective hyper-heuristic selects from a set of LLH. The
selection and acceptance mechanisms of selective hyper-heuristics are studied from literature and summarized in table 2.

**Table 2.** Selection and acceptance mechanisms of selective hyper-heuristics [2].

| Algorithm                          | Fitness Improvement /Quality | Iteration /Elapsed Time | Probability to Accept Lower Quality | Diversification | Intensification |
|-----------------------------------|-----------------------------|-------------------------|------------------------------------|-----------------|-----------------|
| Exponential Monte Carlo with counter | Yes                         | Yes                     | Yes                               | No              | No              |
| Choice Function                   | Yes                         | Yes                     | No                                | No              | No              |
| Improvement and Selection Rules   | Yes                         | No                      | No                                | Yes             | Yes             |
| Fuzzy Inference Selection         | Yes                         | No                      | No                                | Yes             | Yes             |

The hyper-heuristics in table 2 have been utilized with GA in the literature [2]. The Choice Function (CF) and Exponential Monte Carlo with counter (EMCQ) were among the earliest hyper-heuristics. The CF used reinforcement learning framework while the EMCQ adopted simulated annealing like a probability density function [2].

Meanwhile, the Improvement Selection Rules (ISR) and the Fuzzy Inference Selection (FIS) were the more recent hyper-heuristics. The ISR utilized Tabu search and three measures (quality, diversify and intensify) while the FIS uses fuzzy rules to accommodate partial truth allowing a smoother transition between the search operators [2, 40]. The FIS and ISR actually perform better than CF and EMCQ but are significantly more time-consuming [2]. Accordingly, the FIS performance was statistically at par with the ISR [2]. However, one of the biggest challenges for FIS is to find the right fuzzy membership estimation where the design choices are often problem dependent and cannot be easily generalized [2].

The ISR Technique exploits three rules via its improvement, diversification and intensification operators [40]. The improvement operator checks for improvements in the objective function (chromosome fitness). The diversification operator determines how diverse the current and the previously generated solutions are against the population of potential candidate solutions (robustness). Finally, the intensification operator measures how close the current and the previously generated solution are against the population of solutions (accuracy). The improvement, diversification and intensification operators can be mapped to the GA chromosome fitness, robustness and accuracy respectively as the selection and acceptance rules. The mapping is shown in the subsequent section.

3. **Proposed Modified ISR Hyper-heuristic**

The hyper-heuristic algorithm is proposed for a GA-based automatic clustering technique known as the Automatic Genetic Clustering Algorithm for unknown number of clusters (GCUK) [38]. The optimal chromosome size for the automatic clustering has not been known. Therefore, the hyper-heuristic is used to adaptively and automatically select the instance of the automatic clustering algorithms with different chromosome size during the clustering process, as shown in figure 1.
According to [33], there are three important factors for hyper-heuristic algorithm design, namely problem, quality and time factors. The three important factors for hyper-heuristic algorithm have been proposed as follows:

3.1. Problem Factor
The problem is to identify the optimized size (or length) of the automatic genetic clustering chromosomes. The hyper-heuristic algorithm is designed to adaptively select the instances of the automatic clustering with different chromosome sizes. As illustrated in figure 1, the proposed hyper-heuristic algorithm used several automatic clustering instances with different chromosome sizes, $l$ as follows:

1. The hyper-heuristic algorithm initiates automatic clustering instances with chromosome size = $a$.
2. Automatic clustering runs population initialization, fitness computation and genetic operations.
3. The hyper-heuristic selection and acceptance operator determines if the current instance performed better than current threshold.
4. If yes, the automatic clustering continues to run until meet termination condition. If no, the hyper-heuristic algorithm initiates automatic clustering instance with chromosome size = $b$, $c$ or $d$.
5. Repeat step 2, step 3 and step 4.

The hyper-heuristic selects the automatic clustering instances to be used at any iterations adaptively and dynamically through one defined parent heuristic via non-domain feedback. The design was inspired by [33] work that used hyper-heuristic to tune the GA parameters. The hyper-heuristic algorithm has used the ISR that dynamically and self-adaptively improves the solutions at runtime [2]. As a result, the strengths of each automatic clustering instance running on the different chromosome sizes can be utilized and the weaknesses of the collaborating algorithms can be balanced. The proposed hyper-heuristic measured the solution quality by optimizing the chromosome sizes.

3.2. Quality Factor
A suitable hyper-heuristic strategy, i.e. the improvement selection rules (ISR) has been selected. The ISR is based on Tabu search and measures the quality, diversification and intensification operators [2] as shown in table 3.
Table 3. Mapping of the Hyper-heuristic ISR operators to automatic genetic clustering parameters.

| ISR operators | Clustering parameters |
|---------------|-----------------------|
| Quality       | Fitness               |
| Diversification | Robustness           |
| Intensification | Accuracy             |

The quality operator shown in table 3 determines improvement in the optimization objective function. The diversification operator checks how diverse the current and previously generated solutions are against the population of potential candidate solutions. Meanwhile, the intensification operator evaluates how close the current and previously generated solutions are against the population solutions. These three ISR operators have been mapped to the automatic clustering quality validation parameters, namely fitness, robustness and accuracy as shown in table 3. The accuracy is the difference between the suggested number of clusters and the real number of clusters [41]. Meanwhile, the robustness was measured by the difference between the most optimal and the second most optimal number of clusters. The mapping was made according to the similarity of the ISR operators to the automatic clustering quality validation parameters.

The modified ISR operators are as follows:

i. Fitness: this is the quality improvement operator that compares the current best fitness solution \( F_{Currentbest} \) with the previous best fitness solution \( F_{Lastbest} \) [2]. For example, if the \( F_{Currentbest} \) is 5 and \( F_{Lastbest} \) is 3, the value of fitness improvement is 2. It indicates that the current solution improves from the last solutions. If the improvement value is less than zero, the hyper-heuristic algorithm will initiate another instance of automatic genetic clustering.

\[
\text{Fitness Improvement} = F_{Currentbest} - F_{Lastbest}
\]

ii. Robustness (exploration): this operator calculates the difference between the number of cluster of current best solution \( F_{Currentbest} \) with the second-best solution \( F_{Secondbest} \) [2]. For example, the \( F_{Currentbest} \) is 5 and the \( F_{Secondbest} \) is 2, the robustness is 3. It shows a difference between the number of clusters of current best solution and second-best solution. The larger robustness indicates larger exploration. Then, the operator compares the robustness with previous iteration best solution \( F_{Lastbest} \) as follows:

\[
\text{Robustness} = K_{F_{Currentbest}} - K_{F_{Lastbest}}
\]

If the value is less than zero, the hyper-heuristic algorithm will initiate another enhanced GCUK algorithm.

iii. Accuracy (exploitation): the accuracy is equal to 1 divided by the difference between the number of clusters of the best solution \( K_{F_{Currentbest}} \) with the previous best solutions’ average number of clusters \( K_{F_{listAvg}} \). It is formulated as follows:

\[
\text{Acc} = \frac{1}{K_{F_{Currentbest}} - K_{F_{listAvg}}}
\]
For example, if the $K_{F_{Current\ best}}$ is 5 and the $K_{F_{List\ Avg}}$ is 3, the accuracy is 0.5. An accuracy of 1 is the highest accuracy that indicates the number of cluster for each best solution is accurate. Then, this operator calculates the difference between the current best solution accuracy and the previous best solution accuracy [2]. The exploitation improves given the followings:

$$Exploitation = Acc_{Current\ best} - Acc_{Last\ best} \quad (4)$$

$Acc_{Current\ best}$ is the current best solution accuracy while $Acc_{Last\ best}$ is the previous best solution accuracy. For example, if the $Acc_{Current\ best}$ is 1 and the $Acc_{Last\ best}$ is 0.5, it shows that there is an improvement in the exploitation. Meanwhile, if the value is less than zero, the hyper-heuristic algorithm will initiate another enhanced instance of automatic genetic clustering.

According to [2], the selection of the search operators needs to take into account the balance between exploration and exploitation. The proposed Tabu search ISR-based hyper-heuristic algorithm is described by the pseudocode shown in figure 2 (right) and compared with the existing algorithm (left) [40].

| Initialize the population | Initialize the population |
|---------------------------|---------------------------|
| Initialize $Tabu_{max}$, $F_{max}$ fitness evaluation limit, $\Theta_{max}$ iteration, and population size $S$ | Initialize Tabulist and best solution list, $F_{list}$ |
| Initialize the population of solution $z$, $Z = \{Z_1, Z_2, Z_3\}$ | Initialize average number of cluster for $F_{list}$ |
| Select a meta-heuristic algorithm $H_i$ from the pool of meta-heuristics | Initialize $Tabu_{max}$ = 11 |
| While all interaction tuples ($I$) are not covered and $F_{max}$fitness evaluation limit is not reached | Select random GCUK algorithm, GCUK, from the pool of GCUK algorithms |
| Obtain $z_{best}$ from the population $Z$ and add to the final suite $T_{S}$ | While $T < Tabu_{max}$ do |
| **F1 = Improvement Operator** | Apply GCUK, to produce the best Solution from the population and add to $F_{list}$ |
| **F2 = Diversify Operator** | **F3 = Accuracy operator** |
| **F3 = Intensify Operator** | If the meta-heuristic selection and acceptance mechanism evaluation, $\psi(H_i, F_1, F_2, F_3)$, improves |
| If the meta-heuristic selection and acceptance mechanism evaluation, $\psi(H_i, F_1, F_2, F_3)$, improves | Then Keep $H_i$ |
| Then Keep $H_i$ | End |
| Else | Else |
| Add $H_i$ to Tabu list | Add GCUK, to Tabu list |
| If (Tabu list is full) // $Tabu_{max}$ | If Tabu list is full // $Tabu_{max}$ |
| Then randomly select a new and different meta-heuristic algorithm from Tabu List | Then randomly select new and different GCUK from Tabulist |
| Else | Else |
| Select a new meta-heuristic algorithm $H_i$ from the pool of meta-heuristics | Select new GCUK from the pool of GCUK algorithms |
| End | $T++$ |

**Figure 2.** The original hyper-heuristic algorithm [40] (left) and modified hyper-heuristic algorithm (right).
In the pseudocode shown in figure 2, the Tabu list refers to a list that keeps the GCUK algorithms that fail to improve the current threshold (F1, F2, F3). The Tabu$_{max}$ is the size of the list that can be deduced based on the number of heuristics [40]. The proposed Tabu$_{max}$ in the modified hyper-heuristic algorithm (right) is 11 because there are 11 instances of the automatic genetic clustering algorithm with different chromosome size being run in turn by the hyper-heuristic algorithms to find the optimal solutions.

3.3. Time Factor

The time factor has been measured by the real time it takes for the proposed hyper-heuristic algorithm to deliver results [33]. The time should be feasible for the intended application [33].

4. Results and Discussion

The Iris, WDBC, Wine and E.coli datasets were used as benchmarking datasets. These datasets were tested on the proposed modified ISR hyper-heuristic algorithm. Table 4 shows the results.

Table 4. The results of the automatic genetic clustering with modified hyper-heuristic algorithm tested with benchmarking datasets (number of runs = 10).

| Dataset | Average fitness | Average execution time (ms) | Average fitness | Average execution time (ms) |
|---------|----------------|----------------------------|----------------|-----------------------------|
| Iris    | 458.0359       | 246.4                      | 1199.475       | 29239.7                    |
| WCBC    | 702.5747       | 4133.2                     | 1175.603       | 232876.2                   |
| Wine    | 337.6981       | 335.5                      | 751.2566       | 6039                        |
| E.coli  | 548.3666       | 1763.5                     | 823.6711       | 3762.2                     |
| Average | 511.6688       | 1619.65                    | 987.5014       | 67979.28                   |

The results in table 4 clearly show that an increase in the fitness for the benchmarking datasets can be gained by using the hyper-heuristic algorithm. The increases in fitness were 475.8326 higher (987.5014-511.6688) for automatic genetic clustering with hyper-heuristic compared to the automatic genetic clustering for all benchmarking datasets. However, it is identified that the execution time had highly increased by 67467.6112ms (1.12 minutes). The longest execution time took 232876.2ms (3.9 minutes). The main reason why the hyper-heuristic algorithm has higher execution time is because it was running multiple automatic genetic clustering instances alternately and the selection and acceptance operator as shown in figure 2.

Then, the results of number of clusters were measured based on accuracy and robustness. Table 5 shows the comparisons.
Table 5. The results and comparison of the number of clusters validation of the automatic genetic clustering with the hyper-heuristic algorithm (accuracy/robustness).

| Benchmarking dataset | Best solution of enhanced GCUK with a hyper-heuristic algorithm |
|----------------------|---------------------------------------------------------------|
| Iris                 | 0/1 or +1/3                                                  |
| WCBC                 | +16/1                                                        |
| Wine                 | 0/1                                                          |
| E. coli              | 0/1                                                          |

From table 5, the number of clusters is similar to the original dataset when the accuracy values are zero. A negative accuracy value will show the produced number of clusters is lesser than the real number of clusters and for positive accuracy, it is vice versa. A robustness higher than zero shows that the algorithm is robust. It is identified that the accuracy and robustness of the hyper-heuristic algorithm are similar to the enhanced GCUK for the iris, wine and E coli datasets. Meanwhile, the accuracy is better for WCBC dataset.

The results have proven that the ISR-based hyper-heuristic algorithm has produced good performance (fitness) but consume considerably higher execution times [2, 42]. For a larger dataset, the execution time of the hyper-heuristic has to be given careful consideration.

5. Conclusion
The automatic genetic algorithm can generate a dynamic chromosome size as the results of using a modified hyper-heuristic algorithm. The fitness and accuracy of the GCUK have been improved and the hyper-heuristic algorithm has been modified to suit the automatic clustering problem. For future work, machine learning algorithms can be used to enhance the hyper-heuristics selection operators aiming to improve the solution fitness, accuracy and robustness and reduce the execution time by learning from historical data.

Acknowledgments
The authors would like to thank the Ministry of Higher Education (MOHE), Malaysia (ERGS Grant No: 0153AB-I27, 203/PKOMP/6730002 and MyBrain15), Universiti Pendidikan Sultan Idris and Universiti Teknologi PETRONAS for supporting this study.

References
[1] Park J, Mei Y, Nguyen S, Chen G and Zhang M J 2018 An investigation of ensemble combination schemes for genetic programming based hyper-heuristic approaches to dynamic job shop scheduling Applied Soft Computing 63 72-86
[2] Zamli K Z, Din F, Kendall G and Ahmed B S 2017 An experimental study of hyper-heuristic selection and acceptance mechanism for combinatorial t-way test suite generation Information Sciences 399 121-53
[3] H D J, Matthew H, Khaled I and Ender O 2014 A genetic programming hyper-heuristic for the multidimensional knapsack problem Kybernetes 43 1500-11
[4] Yu L, Maoguo G, Licheng J and Yi Z 2015 A memetic algorithm based on hyper-heuristics for examination timetabling problems International Journal of Intelligent Computing and Cybernetics 8 139-51
[5] Ferreira A S, Goncalves R A, Pozo A and Ieee 2017 A Multi-Armed Bandit Selection Strategy for Hyper-heuristics (New York: Ieee)
[6] Muhamad Hariz Adnan, Wahidah Husain, Nur’Aini Abdul Rashid and Hassan M F 2017 Preliminary Analysis to Investigate Accuracy of Data Mining for Childhood Obesity and Overweight Predictions
[7] Adnan M H, Husain W and Rashid N A A 2012 Hybrid approaches using decision tree, naive Bayes, means and euclidean distances for childhood obesity prediction International Journal of Software Engineering and Its Applications 6 99-106

[8] Adnan M H, Husain W and Rashid N A A 2012 A hybrid approach using Naïve Bayes and Genetic Algorithm for childhood obesity prediction. In: Computer & Information Science (ICCIS), 2012 International Conference on: IEEE (pp 281-5)

[9] Tsai C W, Chang W L, Hu K C and Chiang M C 2017 Quality, Reliability, Security and Robustness in Heterogeneous Networks, ed J H Lee and S Pack (New York: Springer) pp 97-109

[10] Gonçalves R, Almeida C, Venske S, Delgado M, Pozo A and Ieee 2017 A New Hyper-Heuristic based on a Restless Multi-Armed Bandit for Multi-Objective Optimization (Los Alamitos: Ieee Computer Soc)

[11] Prestes L, Delgado M R, Gonçalves R, Almeida C P, Pozo A and Ieee 2017 A Hyper-Heuristic in MOEA/D-DRA Using the Upper Confidence Bound Technique (Los Alamitos: Ieee Computer Soc)

[12] Rahbari D, Kabirzadeh S, Nickray M and Ieee 2017 A Security Aware Scheduling in Fog Computing by Hyper Heuristic Algorithm (New York: Ieee)

[13] Nguyen S and Ieee 2017 Ieee 2017 Iecon 2017 - 43rd Annual Conference of the Ieee Industrial Electronics Society, (New York: Ieee) pp 4796-801

[14] Nguyen S, Zhang M J and Ieee 2017 A PSO-based Hyper-heuristic for Evolving Dispatching Rules in Job Shop Scheduling (New York: Ieee)

[15] Tyasnurita R, Ozcan E, John R and Ieee 2017 Learning Heuristic Selection using a Time Delay Neural Network for Open Vehicle Routing (New York: Ieee)

[16] Fontoura V D, Pozo A T R, Santana R and Ieee 2017 Automated Design of Hyper-Heuristics Components to Solve the PSP Problem with HP Model (New York: Ieee)

[17] Rosales-Perez A, Gutierrez-Rodriguez A E, Ortiz-Bayliss J C, Terashima-Marín H, Coello C A C and Ieee 2017 Evolutionary Multilabel Hyper-Heuristic Design (New York: Ieee)

[18] Sabar N R, Turky A, Song A, Sattar A and Ieee 2017 Optimising Deep Belief Networks by Hyper-heuristic Approach (New York: Ieee)

[19] Din F, Alsewari A A, Zamli K Z and Ieee 2017 A Parameter Free Choice Function based Hyper-Heuristic Strategy for Pairwise Test Generation (New York: Ieee)

[20] Kabirzadeh S, Rahbari D, Nickray M and Ieee 2017 Proceedings of the 2017 21st Conference of Open Innovations Association, (New York: Ieee) pp 148-55

[21] Damasevicius R and Wozniak M 2017 Artificial Intelligence and Soft Computing, Icaisic 2017, Pr I, ed L Rutkowski, et al. (Cham: Springer International Publishing Ag) pp 337-46

[22] Muthuraman S, Venkatesan V P and Ieee 2017 A Comprehensive Study on Hybrid Meta-Heuristic Approaches Used for Solving Combinatorial Optimization Problems (New York: Ieee)

[23] Ong J H, Teo J and Ieee 2017 Empirical Analysis of Hyper-heuristic Search Algorithms in Expensive Numerical Optimization (New York: Ieee)

[24] van Lon R R S, Branke J and Holvoet T 2018 Optimizing agents with genetic programming: an evaluation of hyper-heuristics in dynamic real-time logistics Genet. Program. Evol. Mach. 19 93-120

[25] El Yafrani M, Martins M, Wagner M, Ahiod B, Delgado M and Luders R 2018 A hyperheuristic approach based on low-level heuristics for the travelling thief problem Genet. Program. Evol. Mach. 19 121-50

[26] Sabar N R, Yi X and Song A 2018 A Bi-objective Hyper-Heuristic Support Vector Machines for Big Data Cyber-Security Ieee Access 6 11

[27] Rojas-Dominguez A, Padierna L C, Valadez J M C, Puga-Soberanes H J and Fraire H J 2018 Optimal Hyper-Parameter Tuning of SVM Classifiers With Application to Medical Diagnosis Ieee Access 6 7164-76
[28] Obin J H, Alfred R and Abdalla M H 2017 A PSO Inspired Asynchronous Cooperative Distributed Hyper-Heuristic for Course Timetabling Problems *Advanced Science Letters* 23 11016-22
[29] Lin J, Wang Z J and Li X D 2017 A backtracking search hyper-heuristic for the distributed assembly flow-shop scheduling problem *Swarm and Evolutionary Computation* 36 124-35
[30] Kheiri A and Keedwell E 2017 A Hidden Markov Model Approach to the Problem of Heuristic Selection in Hyper-Heuristics with a Case Study in High School Timetabbling Problems *Evol. Comput.* 25 473-501
[31] Burke E K, Gendreau M, Hyde M, Kendall G, Ochoa G, Özcan E and Qu R 2013 Hyper-heuristics: a survey of the state of the art *Journal of the Operational Research Society* 64 1695-724
[32] Hong L B, Drake J H, Woodward J R and Ozcan E 2018 A hyper-heuristic approach to automated generation of mutation operators for evolutionary programming *Applied Soft Computing* 62 162-75
[33] Gudino-Penaloza F, Gonzalez-Mendoza M, Mora-Vargas J and Hernandez-Gress N 2013 Fuzzy Hyperheuristic Framework for GA Parameters Tuning. In: 2013 12th Mexican International Conference on Artificial Intelligence, pp 53-8
[34] Kumari A C, Srinivas K and Gupta M P 2013 *Proceedings of the 2013 3rd Ieee International Advance Computing Conference*, ed B M Kalra, et al. (New York: Ieee) pp 813-8
[35] Eiben A E, Michalewicz Z, Schoenauer M and Smith J E 2007 *Parameter Setting in Evolutionary Algorithms*, ed F G Lobo, et al. (Berlin, Heidelberg: Springer Berlin Heidelberg) pp 19-46
[36] Chaabouni S, Jammoussi S, Benayed Y and Ieee 2013 *2013 5th International Conference on Modeling, Simulation and Applied Optimization*, (New York: Ieee)
[37] Cobos C, Mendoza M, Leon E and Ieee 2011 *A Hyper-Heuristic Approach To Design And Tuning Heuristic Methods For Web Document Clustering* (New York: Ieee)
[38] José-Garcia A and Gómez-Flores W 2016 Automatic clustering using nature-inspired metaheuristics: A survey *Applied Soft Computing* 41 192-213
[39] Wang H, Kang Z and Wang L 2016 Performance-Aware Cloud Resource Allocation via Fitness-Enabled Auction *IEEE Transactions on Parallel and Distributed Systems* 27 1160-73
[40] Zamli K Z, Alkazemi B Y and Kendall G 2016 A Tabu Search hyper-heuristic strategy for t-way test suite generation *Applied Soft Computing* 44 57-74
[41] Yue S, Wang J-S, Wu T and Wang H 2010 A new separation measure for improving the effectiveness of validity indices *Information Sciences* 180 748-64
[42] Adnan M H, Hassan M F and Aziz I A 2017 Limitations and Future Directions for Cloud Service Negotiations using Double Auctions. In: *ICIMU 2017*,