An Analysis of the Taguchi Method for Tuning a Memetic Algorithm with Reduced Computational Time Budget

Düriye Betül Gümüş(✉), Ender Özcan, and Jason Atkin

ASAP Research Group, School of Computer Science, University of Nottingham, Wollaton Road, Nottingham NG8 1BB, UK
{betul.gumus,ender.ozcan,jason.atkin}@nottingham.ac.uk

Abstract. Determining the best initial parameter values for an algorithm, called parameter tuning, is crucial to obtaining better algorithm performance; however, it is often a time-consuming task and needs to be performed under a restricted computational budget. In this study, the results from our previous work on using the Taguchi method to tune the parameters of a memetic algorithm for cross-domain search are further analysed and extended. Although the Taguchi method reduces the time spent finding a good parameter value combination by running a smaller size of experiments on the training instances from different domains as opposed to evaluating all combinations, the time budget is still larger than desired. This work investigates the degree to which it is possible to predict the same good parameter setting faster by using a reduced time budget. The results in this paper show that it was possible to predict good combinations of parameter settings with a much reduced time budget. The good final parameter values are predicted for three of the parameters, while for the fourth parameter there is no clear best value, so one of three similarly performing values is identified at each time instant.

Keywords: Evolutionary algorithm · Parameter tuning · Design of experiments · Hyper-heuristic · Optimisation

1 Introduction

Many real-world optimisation problems are too large for their search spaces to be exhaustively explored. In this research we consider cross-domain search where the problem structure will not necessarily be known in advance, thus cannot be leveraged to produce fast exact solution methods. Heuristic approaches provide potential solutions for such complex problems, intending to find near optimal solutions in a significantly reduced amount of time. Metaheuristics are problem-independent methodologies that provide a set of guidelines for heuristic optimization algorithms [18]. Among these, memetic algorithms are highly effective population-based metaheuristics which have been successfully applied to

T. Czachórski et al. (Eds.): ISCIS 2016, CCIS 659, pp. 12–20, 2016.
DOI: 10.1007/978-3-319-47217-1_2

a range of combinatorial optimisation problems [2,8,10,11,14]. Memetic algorithms, introduced by Moscato [12], hybridise genetic algorithms with local search. Recent developments in memetic computing, which broadens the concept of memes, can be found in [13]. Both the algorithm components and parameter values need to be specified in advance [17], however determining the appropriate components and initial parameter settings (i.e., parameter tuning) to obtain high quality solutions can take a large computational time.

Hyper-heuristics are high-level methodologies which operate on the search space of low-level heuristics rather than directly upon solutions [4], allowing a degree of domain independence where needed. This study uses the Hyper-heuristics Flexible Framework (HyFlex) [15] which provides a means to implement general purpose search methods, including meta/hyper-heuristics.

In our previous work [7], the parameters of a memetic algorithm were tuned via the Taguchi method, under a restricted computational budget, using a limited number of instances from several problem domains. The best parameter setting obtained through the tuning process was observed to generalise well to unseen instances. A drawback of the previous study was that even testing only the 25 parameter combinations indicated by the L_{25} Taguchi orthogonal array, still takes a long time. In this study, we further analyse and extend our previous work with an aim to assess whether we can generalise the best setting sooner with a reduced computational time budget. In Sect. 2, the HyFlex framework is described. Our methodology is discussed in Sect. 3. The experimental results and analysis are presented in Sect. 4. Finally, some concluding remarks and our potential future work are given in Sect. 5.

2 Hyper-Heuristics Flexible Framework (HyFlex)

Hyper-heuristics Flexible Framework (HyFlex) is an interface proposed for the rapid development, testing and comparison of meta/hyper-heuristics across different combinatorial optimisation problems [15]. There is a logical barrier in HyFlex between the high-level method and the problem domain layers, which prevents hyper-heuristics from accessing problem specific information [5]. Only problem independent information, such as the objective function value of a solution, can pass to the high-level method [3].

HyFlex was used in the first Cross-domain Heuristic Search Challenge (CHeSC2011) for the implementation of the competing hyper-heuristics. Twenty selection hyper-heuristics competed at CHeSC2011. Details about the competition, the competing hyper-heuristics and the tools used can be found at the CHeSC website[1]. The performance comparison of some previously proposed selection hyper-heuristics including one of the best performing ones can be found in [9]. Six problem domains were implemented in the initial version of HyFlex: Maximum Satisfiability (MAX-SAT), One Dimensional Bin Packing (BP), Permutation Flow Shop (PFS), Personnel Scheduling (PS), Traveling Salesman (TSP) and Vehicle Routing (VRP). Three additional problem domains were

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

added by Adriaensen et al. [1] after the competition: 0-1 Knapsack (0-1 KP), Max-Cut, and Quadratic Assignment (QAP). Each domain contains a number of instances and problem specific components, including low level heuristics and an initialisation routine which can be used to produce an initial solution. In general, this routine creates a random solution.

The low-level heuristics (operators) in HyFlex are categorised as *mutation*, *ruin and re-create*, *crossover* and *local search* [15]. Mutation makes small random perturbations to the input solution. Ruin and re-create heuristics remove parts from a complete solution and then rebuild it, and are also considered as mutational operators in this study. A crossover operator is a binary operator accepting two solutions as input unlike the other low level heuristics. Although there are many crossover operators which create two new solutions (*offspring*) in the scientific literature, the Hyflex crossover operators always return a single solution (by picking the best solution in cases where the operator produces two offspring). Local search (hill climbing) heuristics iteratively perform a search within a certain neighbourhood attempting to find an improved solution. Both local search and mutational heuristics come with parameters. The *intensity of mutation* parameter determines the extent of changes that the mutation or ruin and re-create operators will make to the input solution. The *depth of search* parameter controls the number of steps that the local search heuristic will complete. Both parameter values vary in [0,1]. More details on the domain implementations, including low level heuristics and initialisation routines can be found on the competition website and in [1,15].

3 Methodology

Genetic algorithm are well-known metaheuristics which perform search using the ideas based on natural selection and survival of the fittest [6]. In this study, a steady state memetic algorithm (SSMA), hybridising genetic algorithms with local search is applied to a range of problems supported by HyFlex, utilising the provided mutation, crossover and local search operators for each domain.

SSMA evolves a *population* (set) of initially created and improved *individuals* (candidate solutions) by successively applying genetic operators to them at each evolutionary cycle. In SSMA, a fixed number of individuals, determined by the *population size* parameter, are generated by invoking the HyFlex initialisation routine of the relevant problem domain. All individuals in the population are evaluated using a *fitness function* measuring the quality of a given solution. Each individual is improved by employing a randomly selected local search operator. Then the evolutionary process starts. Firstly, two individuals are chosen one at a time for crossover from the current population. The generic *tournament selection* which chooses the fittest individual (with the best fitness value with respect to the fitness function) among a set of randomly selected individuals of *tournament size* (*tour size*) is used for this purpose. A randomly chosen crossover operator is then applied producing a single solution which is perturbed using a randomly selected mutation and then improved using a randomly selected local search.

Finally, the resultant solution gets evaluated and replaces the worst individual in the current population. This evolutionary process continues until the time limit is exceeded.

SSMA has parameters which require initial settings and influence its performance. Hence, the Taguchi orthogonal arrays method [16] is employed here to tune these parameter settings. Firstly, control parameters and their potential values (levels) are determined. Four algorithm parameters are tuned: population size (PopSize), tournament size (TourSize), intensity of mutation (IoM) and depth of search (DoS). The parameter levels of {0.2, 0.4, 0.6, 0.8, 1.0} are used for both IoM and DoS. PopSize takes a value in {5, 10, 20, 40, 80}. Finally, {2, 3, 4, 5} are used for TourSize. HyFlex ensures that these are problem independent parameters, i.e. common across all of the problem domains. Based on the number of parameters and levels, a suitable orthogonal array is selected to create a design table. Experiments are conducted based on the design table using a number of 'training' instances from selected domains and then the results are analysed to determine the optimum level for each individual control parameter. The combination of the best values of each parameter is predicted to be the best overall setting.

4 Experimentation and Results

In [7], experiments were performed with a number of configurations for SSMA using 2 training instances from 4 HyFlex problem domains. An execution time of 415 seconds was used as a termination criterion for those experiments, equivalent to 10 nominal minutes on the CHeSC2011 computer, as determined by the evaluation program provided by the competition organisers. Each configuration was tested 31 times, the median values were compared and the top 8 algorithms were assigned scores using the (2003–2009) Formula 1 scoring system, awarding 10, 8, 6, 5, 4, 3, 2 and 1 point(s) for the best to the 8th best, respectively. The best configuration was predicted to be IoM = 0.2, DoS = 1.0, TourSize = 5 and PopSize = 5, and this was then applied to unseen instances from 9 domains and found to perform well for those as well. A similar process was then applied to predict a good parameter configuration across 5 instances from each of the 9 extended HyFlex problem domains, and the same parameter combination was found, indicating some degree of cross-domain value to the parameter setting. With 31 repetitions of 25 configurations, this was a time-consuming process.

The aim of this study is to investigate whether a less time consuming analysis could yield similar information. All 25 parameter settings indicated by the L_{25} Taguchi orthogonal array were executed with different time budgets, from 1 to 10 min of nominal time (matching the CHeSC2011 termination criterion), the Taguchi method was used to predict the best parameter configuration for each duration and the results were analysed. 2 arbitrarily chosen instances from each of the 6 original HyFlex problem domains were employed during the first parameter tuning experiments. Figure 1 shows the main effect values for each parameter level, defined as the mean total Formula 1 score across all of the

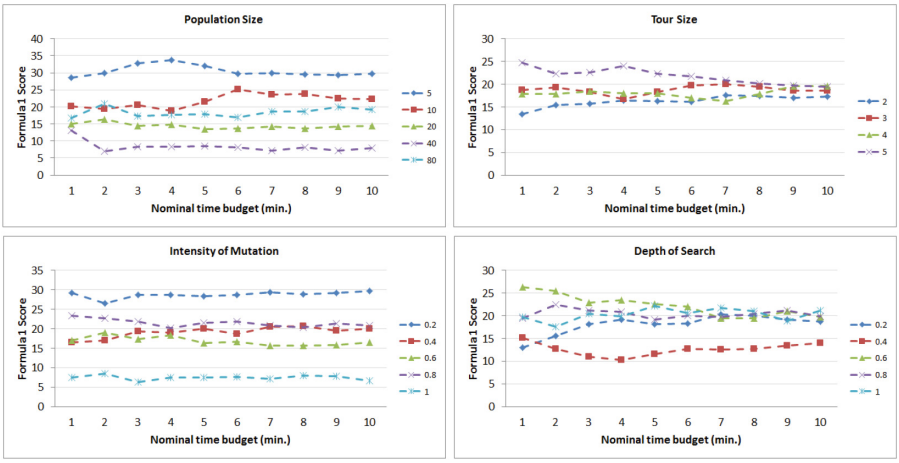

Fig. 1. Main effects of parameter values at different times using 2 training instances from 6 problem domains

settings where the parameter took that specific value. It can be seen that a population size of 5 has the highest effect in each case during the 10 nominal minutes run time. Similarly, the intensity of mutation parameter value of 0.2 performs well at each time. For the tour size parameter, 5 has the highest effect throughout the search except at one point: at 10 nominal minutes, the tour size of 4 had a score of 19.58 while tour size 5 had a score of 19.48, giving very similar results. The best value for the depth of search parameter changes during the execution; however, it is always one of the values 0.6, 0.8 or 1.0. 0.6 for depth of search is predicted to be the best parameter value for a shorter run time.

The analysis of variance (ANOVA) is commonly applied to the results in the Taguchi method to determine the percentage contribution of each factor [16]. This analysis helps the decision makers to identify which of the factors need more control. Table 1 shows the percentage contribution of each factor. It can be seen that intensity of mutation and population size parameters have

Table 1. The percentage contribution of each parameter obtained from the Anova test for 6 problem domains

par. \n.t.b. (min.)	1	2	3	4	5	6	7	8	9	10
IoM	37.6 %	22.6 %	28.8 %	24.6 %	28.2 %	29.9 %	32.4 %	32.6 %	34.1 %	36.3 %
DoS	14.8 %	13.2 %	9.3 %	11.0 %	9.5 %	6.6 %	6.3 %	6.4 %	5.4 %	4.0 %
PopSize	20.5 %	34.0 %	35.6 %	38.2 %	38.5 %	38.3 %	37.7 %	39.4 %	39.4 %	35.1 %
TourSize	10.7 %	3.7 %	3.2 %	5.0 %	2.8 %	3.0 %	2.0 %	0.8 %	0.8 %	0.5 %
Residual	16.3 %	26.5 %	23.0 %	21.1 %	21.0 %	22.2 %	21.5 %	20.8 %	20.2 %	24.1 %

Table 2. The p-values of each parameter obtained from the Anova test for 6 domains. The parameters which contribute significantly are marked in bold.

par. \n.t.b. (min.)	1	2	3	4	5	6	7	8	9	10
IoM	**0.019**	0.191	0.090	0.105	0.078	0.077	0.060	0.054	**0.045**	0.060
DoS	0.171	0.406	0.497	0.384	0.450	0.633	0.635	0.614	0.669	0.825
PopSize	0.090	0.086	0.056	**0.037**	**0.036**	**0.042**	**0.041**	**0.033**	**0.031**	0.065
TourSize	0.188	0.746	0.741	0.568	0.757	0.749	0.836	0.945	0.947	0.977

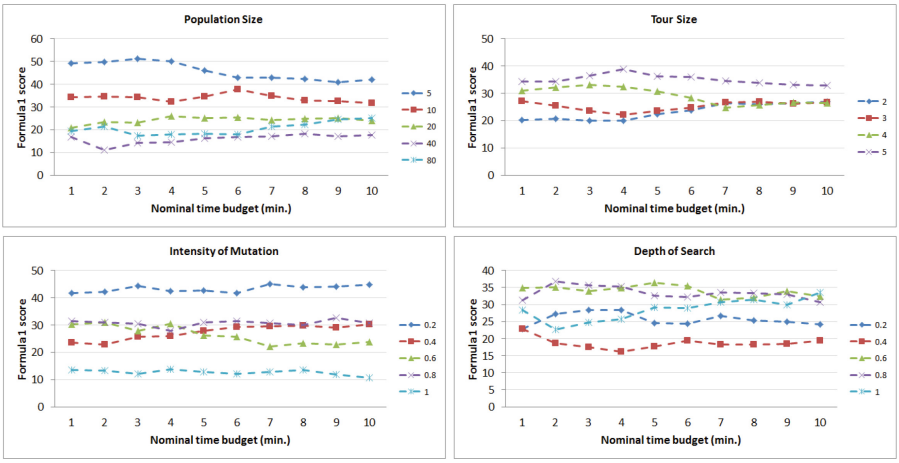

Fig. 2. Main effects of parameter values at different time using 2 training instances from 9 problem domains

highest percentage contribution to the scores. P-values lower than 0.05 means that the parameter is found to contribute significantly to the performance with a confidence level of 95 %. Table 2 shows the p-values of the parameters at each time. The contribution of the PopSize parameter is found to be significant in 6 out of 10 time periods, whereas the intensity of mutation parameter contributes significantly in only 2 out of 10 time periods and the contribution of the other parameters was not found to be significant.

In order to investigate the effect of Depth of Search (DoS) further, we increased the number of domains considered to 9 (and thus used 18 training instances). The main effects of the parameter values are shown in Fig. 2 and Tables 3 and 4 show the percentage contributions and p-values for each parameter. It can be observed from Fig. 2 that the best parameter value does not change over time for the PopSize, TourSize and IoM parameters. The best parameter setting could be predicted for these three parameters after only 1 nominal minute of run time. However, for the depth of search parameter, the best setting indicated in [7] is found only when the entire run time has been used. The best

Table 3. The percentage contribution of each parameter obtained from the Anova test for 9 domains

par. \n.t.b. (min.)	1	2	3	4	5	6	7	8	9	10
IoM	27.7 %	23.6 %	24.0 %	20.3 %	26.3 %	30.0 %	39.1 %	37.3 %	43.4 %	46.0 %
DoS	7.1 %	12.3 %	9.6 %	11.7 %	12.4 %	10.4 %	10.1 %	12.3 %	12.5 %	10.8 %
PopSize	47.3 %	44.5 %	40.8 %	38.2 %	35.3 %	35.6 %	30.9 %	28.3 %	25.0 %	25.5 %
TourSize	8.5 %	7.3 %	9.9 %	14.0 %	8.9 %	7.2 %	4.8 %	4.1 %	3.2 %	2.6 %
Residual	9.4 %	12.3 %	15.7 %	15.8 %	17.1 %	16.7 %	15.1 %	18.1 %	15.9 %	15 %

Table 4. The p-values of each parameter obtained from the Anova test for 9 problem domains. The parameters which contribute significantly are marked in bold.

par. \n.t.b. (min.)	1	2	3	4	5	6	7	8	9	10
IoM	**0.009**	**0.032**	0.057	0.086	0.056	**0.038**	**0.013**	**0.026**	**0.011**	**0.008**
DoS	0.232	0.144	0.317	0.241	0.248	0.310	0.278	0.274	0.217	0.251
PopSize	**0.002**	**0.005**	**0.013**	**0.017**	**0.026**	**0.024**	**0.027**	0.054	0.053	**0.044**
TourSize	0.109	0.219	0.201	0.112	0.263	0.336	0.453	0.587	0.628	0.677

setting for DoS at different times still changes between 0.6, 0.8 and 1.0. When all 9 domains are used, the number of times that the parameters settings contribute significantly is increased. Again it seems that the best setting for DoS depends upon the runtime, but the effect of the parameter is much greater at the longer execution times with the addition of the new domains.

These three values combining with the best values of other parameters were then tested separately on all 45 instances from 9 domains, with the aim of finding the best DoS value on all instances. According to the result of experiments, each of these three configurations found the best values for 18 instances (including ties), considering their median performances over 31 runs. This indicates that these three configurations actually perform similarly even though there are small differences overall. Hence, using only one nominal minute and 2 instances from 6 domains was sufficient to obtain the desired information about the best configuration, reducing the time needed for parameter tuning significantly.

5 Conclusion

This study extended and analysed the previous study in [7], applying the Taguchi experimental design method to obtain the best parameter settings with different run-time budgets. We trained the system using 2 instances from 6 and 9 domains separately and tracked the effects of each parameter level over time. The experimental results show that good values for three of the parameters are relatively easy to predict, but the performance is less sensitive to the value of the fourth (DoS), with different values doing well for different instances and very similar,

"good", overall performances for three settings, making it hard to identify a single "good" value. In summary, these results show that it was possible to predict a good parameter combination by using a much reduced time budget for cross domain search.

References

1. Adriaensen, S., Ochoa, G., Nowé, A.: A benchmark set extension and comparative study for the hyflex framework. In: IEEE Congress on Evolutionary Computation, CEC 2015, 25–28 May 2015, Sendai, Japan, pp. 784–791 (2015)
2. Alkan, A., Özcan, E.: Memetic algorithms for timetabling. In: The 2003 Congress on Evolutionary Computation, CEC 2003, vol. 3, pp. 1796–1802. IEEE (2003)
3. Burke, E.K., Gendreau, M., Hyde, M., Kendall, G., McCollum, B., Ochoa, G., Parkes, A.J., Petrovic, S.: The cross-domain heuristic search challenge – an international research competition. In: Coello, C.A.C. (ed.) LION 2011. LNCS, vol. 6683, pp. 631–634. Springer, Heidelberg (2011)
4. Burke, E.K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Qu, R.: Hyper-heuristics: a survey of the state of the art. J. Oper. Res. Soc. **64**(12), 1695–1724 (2013)
5. Cowling, P.I., Kendall, G., Soubeiga, E.: A hyperheuristic approach to scheduling a sales summit. In: Burke, E., Erben, W. (eds.) PATAT 2000. LNCS, vol. 2079, p. 176. Springer, Heidelberg (2001)
6. Gendreau, M., Potvin, J.Y.: Metaheuristics in combinatorial optimization. Ann. Oper. Res. **140**(1), 189–213 (2005)
7. Gümüş, D.B., Özcan, E., Atkin, J.: An investigation of tuning a memetic algorithm for cross-domain search. In: 2016 IEEE Congress on Evolutionary Computation (CEC). IEEE (2016)
8. Ishibuchi, H., Kaige, S.: Implementation of simple multiobjective memetic algorithms and its applications to knapsack problems. Int. J. Hybrid Intell. Syst. **1**(1), 22–35 (2004)
9. Kheiri, A., Özcan, E.: An iterated multi-stage selection hyper-heuristic. Eur. J. Oper. Res. **250**(1), 77–90 (2015)
10. Krasnogor, N., Smith, J., et al.: A memetic algorithm with self-adaptive local search: TsP as a case study. In: GECCO, pp. 987–994 (2000)
11. Merz, P., Freisleben, B.: A comparison of memetic algorithms, tabu search, and ant colonies for the quadratic assignment problem. In: Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999, vol. 3, pp. 2063–2070 (1999)

12. Moscato, P., et al.: On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms. Caltech Concur. Comput. Prog. C3P Rep. **826**, 1989 (1989)
13. Neri, F., Cotta, C.: Memetic algorithms and memetic computing optimization: a literature review. Swarm Evol. Comput. **2**, 1–14 (2012)
14. Ngueveu, S.U., Prins, C., Calvo, R.W.: An effective memetic algorithm for the cumulative capacitated vehicle routing problem. Comput. Oper. Res. **37**(11), 1877–1885 (2010)
15. Ochoa, G., et al.: HyFlex: a benchmark framework for cross-domain heuristic search. In: Hao, J.-K., Middendorf, M. (eds.) EvoCOP 2012. LNCS, vol. 7245, pp. 136–147. Springer, Heidelberg (2012)
16. Roy, R.: A Primer on the Taguchi Method. Competitive Manufacturing Series. Van Nostrand Reinhold, New York (1990)
17. Segura, C., Segredo, E., León, C.: Analysing the robustness of multiobjectivisation approaches applied to large scale optimisation problems. In: Tantar, E., Tantar, E.-E., Bouvry, P., Del Moral, P., Legrand, P., Coello-Coello, C.A., Schütze, O. (eds.) EVOLVE-A Bridge between Probability, Set Oriented Numerics and Evolutionary Computation. SCI, vol. 447, pp. 365–391. Springer, Heidelberg (2013)
18. Sörensen, K., Glover, F.W.: Metaheuristics. In: Gass, S.I., Fu, M.C. (eds.) Encyclopedia of Operations Research and Management Science, pp. 960–970. Springer, New York (2013)