# Designing and comparing multiple portfolios of parameter configurations for online algorithm selection

Aldy Gunawan[1], Hoong Chuin Lau[1], Mustafa Mısır[2,3]

[1]School of Information Systems, Singapore Management University, Singapore
[2]Department of Computer Science, University of Freiburg, Germany
[3]College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China
aldygunawan@smu.edu.sg, hclau@smu.edu.sg, misir@informatik.uni-freiburg.de

**Abstract.** Algorithm portfolios seek to determine an effective set of algorithms that can be used within an algorithm selection framework to solve problems. A limited number of these portfolio studies focus on generating different versions of a target algorithm using different parameter configurations. In this paper, we employ a Design of Experiments (DOE) approach to determine a promising range of values for each parameter of an algorithm. These ranges are further processed to determine a portfolio of parameter configurations, which would be used within two online Algorithm Selection approaches for solving different instances of a given combinatorial optimization problem effectively. We apply our approach on a Simulated Annealing-Tabu Search (SA-TS) hybrid algorithm for solving the Quadratic Assignment Problem (QAP) as well as an Iterated Local Search (ILS) on the Travelling Salesman Problem (TSP). We also generate a portfolio of parameter configurations using best-of-breed parameter tuning approaches directly for the comparison purpose. Experimental results show that our approach lead to improvements over best-of-breed parameter tuning approaches.

## 1 Introduction

Algorithm Selection [1] concentrates on choosing the best algorithm(s) from a set of algorithms for a given problem instance. The key idea is to build a model that provides a mapping between instance features and performance of a group of algorithms on a set of instances. The resulting model is used to make performance predictions for new problem instances. In relation to algorithm selection, i.e. Algorithm Portfolios [2] primarily focus on determining a set of algorithms for an algorithm selection process. The goal is to choose these algorithms in a way that their strengths complement each other or provide algorithmic diversity that hedge against heterogeneity in problem instances in pretty much the same spirit as investment portfolios to reduce risks in economics and finance [3]. Meta-learning [4] has also been proposed as a unified framework for considering the algorithm selection problem as a machine learning problem.

The idea of algorithm selection has also been investigated in the context of parameter tuning or configuration [5]. The goal is to determine a configuration for a target algorithm that will (hopefully) work well for given instances. Hyper-parameter tuning [6] has the same tuning objective but only for machine learning algorithms. The evolutionary algorithm and meta-heuristic community categorises such methods as parameter tuning and parameter control [7]. Parameter tuning occurs offline, while parameter control is concerned with the strategies for adapting parameters in an online manner via some rules or learning algorithms.

SATZilla [8] is a successful example of applying algorithm portfolios to solve the SAT problem, which has consistently ranked top in the various SAT competitions. Its success lies in its ability to derive an accurate runtime prediction model makes effective use of the problem-specific features of SAT. Hydra [9] is another portfolio-based algorithm selection method that combines with automatic configuration to solve combinatorial problems such as SAT effectively.

Several configurators have been proposed for optimisation algorithms. CALIBRA [10] combines Taguchi fractional experimental design and local search. ParamILS [5], as explained above, applies iterated local search to find a single parameter configuration. Racing algorithms like F-Race [11] look for effective parameter values by performing a race between different configurations. Instance-Specific Algorithm Configuration (ISAC) [12] incorporates a G-means clustering algorithm for clustering instances with respect to the features with an existing parameter tuning method, i.e. GGA. GGA is used to configure an algorithm for each instance cluster and works like other case-based reasoning related algorithm selection approaches. [13] proposed Randomized Convex Search (RCS) with the underlying assumption that the parameter configurations (points) lie inside the convex hull of a certain number of the best points. FocusedILS, derived from ParamILS, has been used to provide a number of different parameter configurations for a given single algorithm. It has also been applied for designing multiple parameter configurations for a planner called Fast Downward, in [14]. The results are used for seven portfolio generation methods to build sequential portfolios. [15] proposed a model-based approach, namely SMAC, that can be used to handle categorical parameters. AutoFolio [16] was developed for automated configuration at a higher level by applying SMAC to algorithm selectors. ADVISER[17] was introduced as a web-based platform for algorithm portfolio generation.

This paper seeks to extend the literature on automatic algorithm configuration. The experiments were conducted for combinatorial optimization problems. Rather than providing a single parameter configuration that works well in general or a pre-set schedule of algorithms, we work in the space of online algorithm selection and feeds this process with a portfolio of parameter configurations derived from Design of Experiments (DOE). Our aim is to develop a generic approach for designing algorithm portfolio of parameter configurations for use within an online algorithm selection process to solve combinatorial optimization problems. In particular, we consider generating different parameter configurations for a given target algorithm as the algorithm portfolio.

Our contributions are listed as follows:

· We apply DOE to build algorithm portfolios of parameter configurations for a given target algorithm. Unlike configurators like ParamILS, F-Race or CALIBRA that provide single value for each parameter, DOE provides a subregion of values for each parameter (that are statistically important compared to other regions).
· We propose a random sampling approach to determine a portfolio of parameter configurations from the subregions. Even though methods like ISAC [12] and Hydra [9] already deliver portfolios of configurations, these techniques run a tuner for

multiple times, resulting in huge computational overheads. In our case, DOE and sampling is done once, which reduces computational overheads tremendously.

· We employ two online algorithm selection methods, namely Simple Random and Learning Automata. Again, the aforementioned portfolio-based methods that make use of parameter configurations are usually performed offline without any solution sharing, while our approach combines the strengths of multiple configurations by selecting them online and operating them on the same solution, which is different from standard algorithm configuration scenarios. Although dynamic portfolio methods [18] perform online selection, they also ignore solution sharing. The empirical results on two NP-hard problems, namely the Quadratic Assignment Problem (QAP) and Travelling Salesman Problem (TSP), show the advantage of using multiple configurations and solution sharing in algorithm selection.

## 2 Algorithm Configuration Problem

The algorithm configuration problem (ACP) [19] is about configuring a given target algorithm $TA$ to perform well on a set of problem instances. The goal is to configure $k$ parameters to set, $P = \{pr_1, \ldots, pr_k\}$, where each parameter has a range of values to be set, $pr_i \in D_i$. The configuration space involves $C = D_1 \times \ldots \times D_k$ many possible configurations. The objective is to come up with a configuration from such a, usually, large set to provide the best performance on an instance set, $I$. Thus, the ACP can be considered an optimisation problem where a solution is a configuration $c_i$ of the algorithm $TA$ on $I$. One issue with this idea is on solution evaluation. For assessing the quality of a $c_i$, $TA$ with $c_i$ should run on $I$. Although the required computational time for this task varies w.r.t. $TA$, $I$ and the problem domain of $I$, it is computationally expensive in general. Heuristic-based search and optimisation techniques such as GGA [20] and ParamILS [5] have been employed in order to overcome this issue.

Such tuning methods are eligible to deliver an effective configuration for a given algorithm. The idea of algorithm portfolios [2] have been used to take advantage of such techniques for building strong algorithm sets including algorithms with different configurations. Existing tuning based portfolio approaches like ISAC [12] and Hydra [9] were designed to address the offline algorithm selection problem. They pursue to the goal of specifying the best single algorithm configuration for solving a particular problem instance. These systems require a set of features representing instances to select algorithms after delivering a set of configurations derived from a computationally expensive training phase. For instance, Hydra mentions that it took 70 CPU days to construct a portfolio of configurations. A similar tool used for a SAT solver, i.e. SATenstein [21], spent 240 CPU days.

Unlike these cases, the aim of this study is to build a portfolio of configurations that can be used in an online setting. The online nature of our approach can allow changing configurations while a selected configuration is fixed for the offline ones. Our system performs like a parameter tuning tool where any domain specific features are not needed. Besides that, the tuning process is faster since the tuning operation is performed once while the tuners used in the aforementioned portfolio approaches run for multiple times. Although our approach is not directly comparable with these offline portfolio methods due to its distinct design, a state-of-the art parameter tuning approach, i.e. ParamILS which is also used in Hydra, is experimented for comparison.

## 3 Design of Experiments (DOE)

DOE is a well-studied statistical technique used in scientific/engineering decision-making to select and determine the key parameters of a particular process [22]. Some typical applications of DOE include 1) evaluation and comparison of basic design configurations, 2) evaluation of different materials, and 3) selection of design parameters.

Let us consider, in order to solve a particular problem, an algorithm (called the target algorithm) that requires a set of parameters to be set prior to the execution of the algorithm, a DOE-based framework was proposed in [23] to find ranges of parameters values which serve as input to existing configurators such as ParamILS [5]. The main goal is to find a parameter setting that performs best over a set of training instances and subsequently verifies the quality of this setting on a set of testing instances. In this paper, we utilize the first two phases of the framework, namely screening and exploitation phases, to provide promising sub-regions for the parameter configurations. These phases are briefly explained below.

### 3.1 Screening Phase

Suppose we have $k$ parameters of a target algorithm to be tuned, where each parameter $p_i$ (discrete or continuous) lies within a numeric interval. In the screening phase, a complete $2^k$ factorial design is applied to identify $m$ parameters ($m \leq k$) which have significant effects to the performance of the target algorithm (the "important" parameters). This requires $n \times 2^k$ observations where $n$ represents the number of replicates. Experiments are replicated to help identify the sources of variability and to better estimate the true effects of treatments.

In a $2^k$ factorial design, we examine the magnitude and direction of the effects to determine which parameters are likely to be important. The importance of a particular parameter $p_i$ can be defined by conducting the test of significance on the main effect of the parameter with a significance level, e.g. $\alpha = 10\%$. Furthermore, the ranking of the critical parameters is determined by the absolute values of the main effects of those parameters. The direction of the parameter effects are determined by the sign of the values of the main effects. For instance, if the objective function of the target algorithm is a minimizing function, the value of a particular parameter should be set to a low value if its coefficient of the main effect is positive. The output of this phase consists of a reduced range for each important parameter and all unimportant parameters will be set to a constant value.

### 3.2 Exploration Phase

In the exploration phase, we treat the $m$ important parameters determined from the screening phase, with the aim to find a promising range for them. We start exploring a larger space where the linear relationship is held and apply the standard approach for linear model checking and diagnosis [22].

The target algorithm is run with respect to the parameter configuration space $\theta$ which contains ($2^m$+1) possible parameter configurations with an additional setting defined by the centre points of the $m$ important parameters. By adding centre points, protection against curvature is provided.

The form of the relationship between the objective function and the parameters is initially unknown. Thus, the first step is to assume a first-order (planar) model of the response surface. The planar model is given by the following approximating function:

$$Y = X\beta + \varepsilon \tag{1}$$

where:

$Y$ is the vector of $(n \times 2^k)$ responses/objective function values
$X$ is the $((n \times 2^k) \times m)$ matrix
$\beta$ is a vector of size $(m \times 1)$
$\varepsilon$ is the $((n \times 2^k) \times 1)$ error vector

Model adequacy checking involves two statistical tests, namely the interaction and curvature tests, are required. The planar model can still be applied as long as either one of them is not statistically significant. Otherwise the region of planar local optimality has been reached and the promising region has been found. We then continue the process by applying the steepest descent, in order to bring the parameter to the vicinity of the optimum values. Once the region of the optimum has been found (e.g. one of two statistical tests is statistically significant), the planar model is invalid and we can assume that we are in the promising range for each important parameter. The details of this framework can be referred to [23].

## 4 Solution Approach

Algorithm portfolios are often used with offline algorithm selection strategies. This work was inspired by hyper-heuristic and operator selection studies which encourage to use online algorithm selection [24]. Our approach is basically in two parts - portfolio generation and online selection. The portfolio generation part involves finding varying instantiations, as configurations, of the same algorithm with diverse problem solving capabilities. A resulting portfolio involving configurations of a particular algorithm is then used by online algorithm selection.

### 4.1 Portfolio Design

Recall that DOE provides a promising range for each parameter $p_i$ based on steepest descent of a linear response surface. More precisely, we have the promising interval [$l_i$, $u_i$] for each parameter $p_i$. Given these intervals, we propose three different methods to generate a portfolio of parameter configurations. The first is to simply use a constant step size, namely, $[l_i, l_i + \delta, l_i + 2\delta, \ldots, u_i]$ where $\delta$ is a constant step size.

**Algorithm 1:** Online Portfolio-Based Algorithm Selection

```
  TA: target algorithm with $k$ parameters, $\theta$: configuration space defined by the initial
  ranges of each parameter $p_i$ ($\forall i = 1,2,\ldots,k$), $I$: instances, $z$: portfolio size
  Portfolio Design
1 Run DOE to obtain a promising range $[l_i, u_i]$ for each parameter $p_i$
2 Generate a portfolio of $z$ parameter configurations from the promising ranges
  Online Algorithm Selection
3 Apply a selection method (Simple Random (SR) or Learning Automata (LA)
```

The second method is to perform intensification on the promising configuration space via random sampling. First, we generate $n$ random samples of parameter configurations from the promising space. A contour plot is then generated where all sampled points (parameter configurations) having the same response are connected to produce the contour lines of the surface. This contour plot provides an approximate fitness landscape from which we can sample $z$ points randomly, with a probability that decreases from one contour line to the next. More precisely, if the contour plot is divided into $y$ contour lines, then we draw $z_i$ samples from the region bounded by contour lines $i$ and $i+1$, where $z = z_1 + z_2 + \ldots + z_y$ and $z_i > z_{i+1}$ for all $1 \leq i \leq y-1$.

As a more informed strategy compared to using the contour plots, a clustering approach is utilized as the final approach. The $k$-means clustering is employed while $k$, i.e. the number of clusters, is determined by the Silhouette score [25]. Inspired from OSCAR[26], the idea is to cluster configurations w.r.t. their performance, i.e. solution quality, on the training instances. The performance measure used in our study is the percentage of deviation of the objective function value obtained by a particular parameter configuration from the best known solution. Normalized performance values are used as features characterizing configurations, similar to landmarking [27]. Finally, the configuration with the highest average rank from each cluster is then included in the portfolio.

## 4.2 Online Algorithm Selection

Even though the No Free Lunch theorem [28] states that there is no one algorithm performs well on all possible problem instances that are closed under permutation, it is usually the case that the search spaces of target problem instances do not have the property of 'closure under permutation'. Using multiple mutation operators in an evolutionary algorithm setting is theoretically shown to be effective by [29]. The advantage of using more than one algorithm in a hyper-heuristic environment is theoretically explained in [30]. As a consequence, both experimental and theoretical studies suggest that online algorithm selection is useful for better performance.

In this section, we propose two approaches to perform online algorithm selection. First, Simple Random (SR) [31] randomly chooses a parameter configuration at each iteration. Although this is very naive approach, it can effectively manage a small-sized algorithm set. Second, Learning Automata (LA), a.k.a. stateless reinforcement learning, has been used to perform heuristic selection in [32] due to its nice convergence property to a Nash equilibrium. Formally, a learning automaton is described by a quadruple $\{\tilde{A}, \beta, p, U\}$. $\tilde{A} = \{\tilde{a}_1, \ldots, \tilde{a}_n\}$ is the set of actions available. $p$ maintains the probabilities of choosing each of these actions. $\beta(t)$ is a random variable between 0 and 1 for the environmental response. $U$ is a learning scheme used to update $p$ [33].

A learning automaton operates iteratively by evaluating the feedback provided as the result of a selected action. The feedback from the environment is stated as the environmental response ($\beta(t)$) referring whether a selected action is favorable ($\beta(t) = 1$) or unfavorable ($\beta(t) = 0$). This feedback is then used to update the corresponding action probabilities. The Boolean feedback is only used in this online algorithm selection. The update operation will depend on the choice of the update scheme ($U$), namely *linear reward-penalty*, *linear reward-inaction* and *linear reward-$\varepsilon$-penalty* are the common update schemes which vary in the degree of rewarding or penalising a selected action with respect to the environmental response. All these update schemes use the following equations:

$$\begin{aligned} p_i(t+1) = p_i(t) &+ \lambda_1 \, \beta(t)(1 - p_i(t)) \\ &- \lambda_2 (1 - \beta(t)) p_i(t) \\ &\text{if } \tilde{a}_i \text{ is the action taken at time step } t \end{aligned} \tag{2}$$

$$\begin{aligned} p_j(t+1) = p_j(t) &- \lambda_1 \, \beta(t) p_j(t) \\ &+ \lambda_2 (1 - \beta(t))[(r-1)^{-1} - p_j(t)] \\ &\text{if } \tilde{a}_j \neq \tilde{a}_i \end{aligned} \tag{3}$$

where $r$ is the number of actions in $\tilde{A}$.

The $\lambda_1$ and $\lambda_2$ values are the learning rates used to update the selection probabilities. The first one is used to reward an action while the latter parameter is to penalise an unfavorable action. The aforementioned three update schemes are determined based on how these two learning rates are set. If they are equal, the update scheme is described as linear reward-penalty $(L_{R-P})$. When the second rate is set to zero, the system is defined as linear reward-inaction $(L_{R-I})$. In the case of $\lambda_2 < \lambda_1$, it is defined as linear reward-$\varepsilon$-penalty $(L_{R-\varepsilon P})$.

Applying the above to online algorithm selection, the actions are associated with the choice of algorithms during run time. In this paper, we have chosen the $(L_{R-I})$ update scheme, which means that $\lambda_2$ is set to 0. Moreover, the probability update process is performed based on two feedbacks, which are finding a new best solution and delivering an improved solution respectively. This means that two values are chosen for $\lambda_1$ depending on which of two feedbacks are received.

## 5 Experimental Results

In order to empirically show the performance of our proposed algorithm selection methods, we perform experiments on two classical combinatorial optimization problems - the Quadratic Assignment Problem (QAP) and Traveling Salesman Problem (TSP).

For each experiment, we perform six different selection methods: 1) SR with constant step size (SR-Constant), 2) SR with random sampling from the contour plot (SR-Contour), 3) LA with constant step-size (LA-Constant), 4) LA with random sampling from the contour plot (LA-Contour), 5) SR with clustering (SR-Cluster), and 6) LA with clustering (LA-Cluster). The learning rate ($\lambda_1$) for LA is set to 0.1 and 0.01 respectively for the feedbacks of finding a new best solution or improving the current solution.

## 5.1 Quadratic Assignment Problem

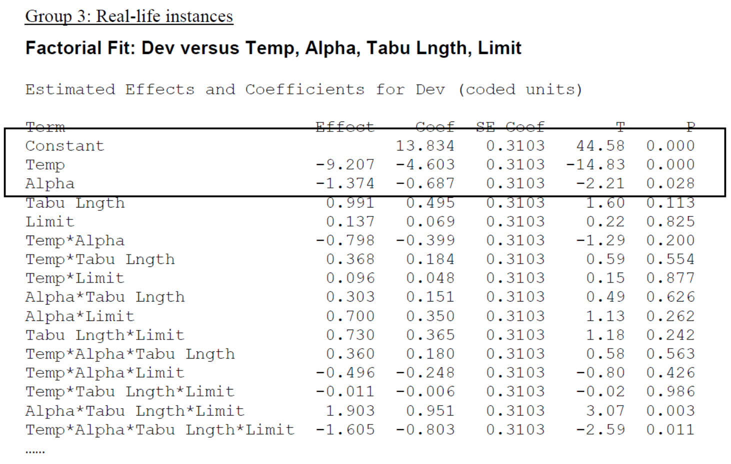
The QAP is interested in the minimum cost allocation of facilities to locations, taking the costs as the sum of all distance-flow products. A Simulated Annealing - Tabu Search (SA-TS) hybrid meta-heuristic [34] is used as the target algorithm with four parameters. Table 1 gives the details about the parameter configurations from [23].

Table 1: The parameter space of the QAP

| **Parameters** | **Initial range** | **DOE Range** | | **Step size** | **Contour Plot** | **Clustering** |
|---|---|---|---|---|---|---|
| Init. temperature (*Temp*) | [100, 7000] | Group I | [4378, 6378] | 250 | 5 values | 2 values |
| | | Group II | [4238, 6238] | 250 | 5 values | 2 values |
| | | Group III | [4000, 6000] | 250 | 5 values | 6 values |
| Cooling factor (*Alpha*) | [0.5, 0.95] | Group I | [0.935, 0.945] | 0.005 | 5 values | 2 values |
| | | Group II | [0.935, 0.945] | 0.005 | 5 values | 2 values |
| | | Group III | [0.85, 0.95] | 0.05 | 5 values | 6 values |
| Tabu list length (*TabuLngth*) | [5, 10] | Group I | 5 | - | - | - |
| | | Group II | 6 | - | - | - |
| | | Group III | 6 | - | - | - |
| Diversification factor (*Limit*) | [0.01, 0.1] | Group I | 0.01 | - | - | - |
| | | Group II | 0.1 | - | - | - |
| | | Group III | 0.1 | - | - | - |

The QAP benchmark instances are from QAPLIB [35]. The instances are grouped into four classes: unstructured instances (Group I), grid-based distance matrix (Group II), real-life instances (Group III) and real-life-like instances (Group IV). Due to the limitation of the target algorithm that can only handle symmetrical distance matrix, we only focus on instances from the first three classes. By referring to [23] for classifying instances into training and testing instances, we conduct the experiments for two different set of instances: 1) testing instances and 2) all instances (training + testing instances). Instance classes consist of 11, 24, 14 training and 5, 11, 7 testing instances, respectively. Those instances are selected randomly.

The application of DOE screening phase yields the following result for Group III (Figure 1). It reveals that two parameters (*Temp* and $Alpha$) are statistically significant (with p-value $\leq 5\%$), while the effect of other parameters: *TabuLngth* and *Limit* are insignificant. Based on the coefficient value obtained, we determine the constant value



Group 3: Real-life instances

**Factorial Fit: Dev versus Temp, Alpha, Tabu Lngth, Limit**

```
Estimated Effects and Coefficients for Dev (coded units)

Term                          Effect     Coef  SE Coef       T      P
Constant                               13.834   0.3103   44.58  0.000
Temp                          -9.207   -4.603   0.3103  -14.83  0.000
Alpha                         -1.374   -0.687   0.3103   -2.21  0.028
Tabu Lngth                     0.991    0.495   0.3103    1.60  0.113
Limit                          0.137    0.069   0.3103    0.22  0.825
Temp*Alpha                    -0.798   -0.399   0.3103   -1.29  0.200
Temp*Tabu Lngth                0.368    0.184   0.3103    0.59  0.554
Temp*Limit                     0.096    0.048   0.3103    0.15  0.877
Alpha*Tabu Lngth               0.303    0.151   0.3103    0.49  0.626
Alpha*Limit                    0.700    0.350   0.3103    1.13  0.262
Tabu Lngth*Limit               0.730    0.365   0.3103    1.18  0.242
Temp*Alpha*Tabu Lngth          0.360    0.180   0.3103    0.58  0.563
Temp*Alpha*Limit              -0.496   -0.248   0.3103   -0.80  0.426
Temp*Tabu Lngth*Limit         -0.011   -0.006   0.3103   -0.02  0.986
Alpha*Tabu Lngth*Limit         1.903    0.951   0.3103    3.07  0.003
Temp*Alpha*Tabu Lngth*Limit   -1.605   -0.803   0.3103   -2.59  0.011
......
```

Fig. 1: Screening Phase (Group III)

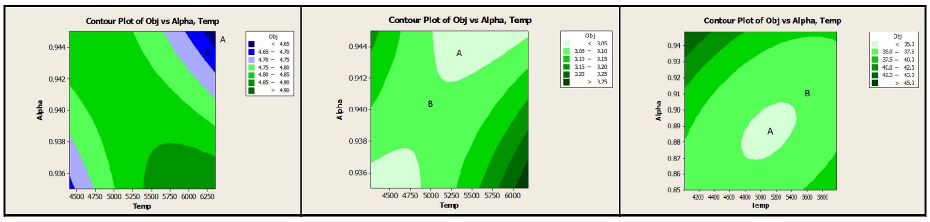


Fig. 2: Contour Plot (Groups I, II and III, respectively)

for each insignificant parameter, e.g. the effect of parameter *Limit* is 0.137, so the value of this parameter is set to its lower bound value, which is 0.01 (Table 1).

Using this information in the DOE exploration phase, we find the promising planar region for both parameters (*Temp* and *Alpha*). The final ranges for both parameters are summarized in Table 1. The contour plots generated from random sampling for instances are as shown in Figure 2. From the plots, we pick three and two different parameter configurations randomly from two promising regions, *A* and *B*, respectively.

Unlike the contour plots case, in the clustering based portfolio generation approach, the number of configurations used in the resulting portfolio is automatically determined. The last three columns show details on step sizes, number of random samples used to generate the portfolio and number of parameter settings generated by the clustering method.

In order to compare the performance of our proposed approach, we also run the target algorithm with constant parameter values generated by RCS and ParamILS. Both configurators also use the same inputs from DOE range (Table 1). The parameter values are obtained from [23].

For each instance within a particular group, we perform 10 runs and compare the percentage deviations of the average objective function value of the solutions obtained

Table 2: The performance of the tested approaches on the QAP instances with respect to the best known solutions (P: ParamILS, Cs: Constant, Ct: Contour, Cl: Cluster)

| **Instances** | **Metric** | **Methods** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **RCS** | **P** | **SR-P** | **LA-P** | **SR-Cs** | **LA-Cs** | **SR-Ct** | **LA-Ct** | **SR-Cl** | **LA-Cl** |
| Group I | % Dev Avg (Test) | 0.606 | 0.692 | 0.779 | 0.509 | 0.535 | 0.492 | 0.473 | **0.471** | 0.512 | 0.534 |
| | % Dev Best (Test) | 0.314 | 0.345 | 0.416 | 0.350 | 0.378 | 0.340 | **0.301** | 0.325 | 0.331 | 0.359 |
| | % Dev Avg (All) | 0.880 | 0.973 | 1.011 | 0.756 | 0.737 | 0.734 | 0.716 | **0.700** | 0.709 | 0.748 |
| | % Dev Best (All) | 0.505 | 0.581 | 0.618 | 0.470 | 0.449 | **0.444** | **0.444** | 0.476 | 0.556 | 0.465 |
| Group II | % Dev Avg (Test) | 0.214 | 0.210 | 0.394 | 0.139 | 0.168 | **0.134** | 0.149 | 0.151 | 0.157 | 0.136 |
| | % Dev Best (Test) | 0.030 | 0.024 | 0.061 | 0.025 | 0.022 | 0.018 | **0.012** | 0.032 | 0.028 | 0.029 |
| | % Dev Avg (All) | 0.262 | 0.247 | 0.417 | **0.183** | 0.192 | 0.189 | 0.189 | **0.183** | 0.188 | 0.195 |
| | % Dev Best (All) | 0.068 | 0.060 | 0.103 | 0.038 | 0.045 | 0.031 | **0.030** | 0.031 | 0.033 | 0.043 |
| Group III | % Dev Avg (Test) | 1.231 | 1.196 | 1.990 | 0.667 | 0.744 | 0.767 | **0.636** | 0.935 | 0.704 | 0.866 |
| | % Dev Best (Test) | **0.000** | 0.191 | **0.000** | **0.000** | **0.000** | **0.000** | **0.000** | **0.000** | **0.000** | **0.000** |
| | % Dev Avg (All) | 2.921 | 2.848 | 3.414 | 1.620 | 2.563 | 1.824 | 2.516 | 1.700 | 1.737 | **1.590** |
| | % Dev Best (All) | 1.114 | 0.873 | 0.278 | 0.252 | 1.170 | 0.241 | 0.266 | **0.231** | 0.239 | 0.251 |

and the best objective function value obtained against the best known/optimal solutions. In order to ensure the fairness among approaches, we use the same computational budget for each one. For example, ParamILS uses $z$ time units so others also use $z$ time units.

The results are summarized in Table 2. In general, we see that we can obtain better results by generating a portfolio of algorithms with different parameter configurations, either by applying Simple Random (SR) or Learning Automata (LA), compared against constant parameter values (RCS or ParamILS). The best performers are SR-Contour and LA-Constant. SR and LA with clustering (SR-Cluster and LA-Cluster) are also comparable with others. Those constant parameter values (RCS and ParamILS) do not perform well.

Table 3: Wilcoxon Signed Rank Test of Table 2 results

| **Methods** | **Group I** | **Group II** | **Group III** |
|---|---|---|---|
| **RCS** | 5 | 5 | 5 |
| **ParamILS** | 6 | 5 | 5 |
| **SR-ParamILS** | 7 | 6 | 6 |
| **LA-ParamILS** | 3 | 1 | 1 |
| **SR-Constant** | 4 | 4 | 2 |
| **LA-Constant** | 2 | 1 | 2 |
| **SR-Contour** | 1 | 2 | 1 |
| **LA-Contour** | 1 | 2 | 4 |
| **SR-Cluster** | 3 | 3 | 1 |
| **LA-Cluster** | 4 | 1 | 3 |

We also run ParamILS five times in order to generate five parameter configurations. Both selections methods, simple random and learning automata (SR-ParamILS and LA-ParamILS), are used to compare with others. The purpose of this comparison is to show how generating a portfolio of algorithms with different parameter values generated from the contour plot and the clustering method outperforms a portfolio of algorithms using constant step-size and best-of-breed parameter tuning approaches (e.g. ParamILS). The results are summarized in SR-ParamILS and LA-ParamILS columns of Table 2.

For further analysis, Wilcoxon Signed Rank Test is used to test all pairwise differences between each algorithm selection approach in terms of %Dev Avg (Test) values. The ranks are summarized in Table 3. Some methods have the same rank values, meaning that those methods are statistically indifferent. We observe that algorithm selection with LA methods outperforms other methods in all groups of testing instances. SR also performs well in Group III instances in terms of the percentage of average deviations for testing instances. In general, using the contour plot to generate a portfolio of promising parameter values outperforms other approaches.

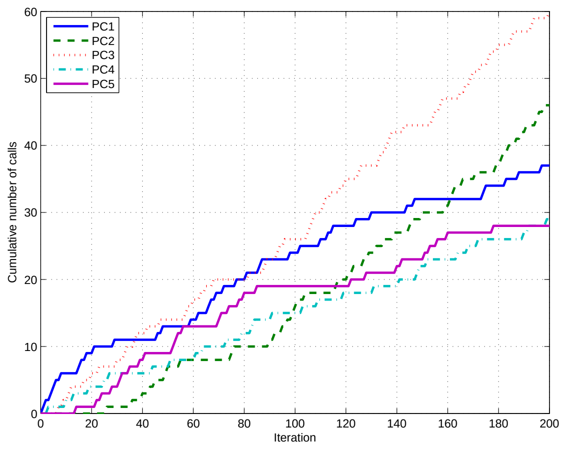


Fig. 3: The effect of learning automata on parameter configuration selection while solving a QAP instance, tho150

Lastly, we provide a glimpse of the effectiveness of the generated portfolio by examining the frequency distribution of selection, as shown in Figure 3. As shown in the figure, the cumulative frequency of choosing each parameter configuration vary over iterations, suggesting that different configurations are effectively used throughout the online selection process. And considering that LA outperformed both ParamILS and SR, we can conclude that the LA's learning process pays off.

## 5.2 Travelling Salesman Problem

The Travelling Salesman problem (TSP) requires finding a tour that visits all cities exactly once that minimises the total distance travelled. In our experiment, Iterated Local Search (ILS) with a 4-Opt perturbation [36] is used as the target algorithm.

Table 4: The parameter space of the TSP

| Parameters | Initial range | DOE Range | Step size | Contour Plot | Clustering |
|---|---|---|---|---|---|
| Maximum # iterations ($Iter_{max}$) | [100, 900] | [400, 600] | 50 | 5 values | 2 values |
| Perturbation strength ($P_s$) | [1, 10] | [1, 3] | 1 | 5 values | 2 values |
| Non-improving moves tolerance ($Tl_{nip}$) | [1, 10] | [4, 6] | 1 | 5 values | 2 values |
| Perturbation choice ($P_c$) | [3, 4] | 3 | - | - | - |

Table 4 summarizes the list of the parameters to be tuned, the initial and final ranges for each parameters after applying DOE. Similar to QAP, the last column provides how we generate the algorithm portfolio. We only compare with ParamILS since RCS does not perform well in solving the QAP (Section 5.1). 47 TSP instances out of 70 instances from TSPLIB are used as the training instances while the rest (23 instances) are treated as testing instances.

The experiment result is presented in Table 5. We observe that our approach works well compared to existing configurators. In particular, the selection method using LA-Contour outperforms others. The performance of algorithm selection methods are ranked based on Wilcoxon Signed Rank Test as follows: LA-Contour $\approx$ LA-Constant $\approx$ SR-Cluster $\succ$ LA-ParamILS $\approx$ SR-Constant $\succ$ LA-Cluster $\approx$ SR-ParamILS $\succ$ SR-Contour $\succ$ ParamILS.

Table 5: The performance of the tested approaches on the TSP instances with respect to the best known solutions (P: ParamILS, Cs: Constant, Ct: Contour, Cl: Cluster)

| Metric | Methods | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | P | SR-P | LA-P | SR-Cs | LA-Cs | SR-Ct | LA-Ct | SR-Cl | LA-Cl |
| % Dev Avg (Test) | 1.742 | 1.331 | 1.321 | 1.325 | 1.295 | 1.377 | **1.291** | 1.295 | 1.332 |
| % Dev Best (Test) | 0.852 | 0.752 | 0.787 | 0.792 | 0.704 | 0.768 | **0.664** | 0.749 | 0.880 |
| % Dev Avg (All) | 1.671 | 1.259 | 1.211 | 1.262 | 1.272 | 1.304 | **1.207** | 1.252 | 1.277 |
| % Dev Best (All) | 0.838 | 0.736 | 0.702 | 0.815 | 0.717 | 0.770 | **0.684** | 0.800 | 0.800 |

Similar to QAP, we also generate five parameter configurations using ParamILS and compare against five points generated from the contour plot, as shown in Table 5 (SR-ParamILS and LA-ParamILS columns). We again conclude that our proposed approach using the contour plot outperforms the portfolio with configurations generated by ParamILS.

## 6 Conclusion

This paper shows that Design of Experiments (DOE) coupled with random sampling can automatically generate good portfolios of parameter configurations that can be used by an online algorithm selection process. The computational results on two classical combinatorial optimisation problems showed the strength of our proposed method compared to state-of-the-art configurators such as ParamILS. We show that the proposed approach lead to improvements to two combinatorial optimization problems, QAP and TSP, compared against single configurations.

Many interesting problems arise from this research. For example, how to set the learning rates in the learning automaton? How to improve our proposed schemes at generating portfolios? Will our generic approach perform well in other problems? How to speed up the process through parallelization?

## ACKNOWLEDGEMENTS

This research is supported by the National Research Foundation, Prime Minister's Office, Singapore under its International Research Centres in Singapore Funding Initiative.

## References

1. Rice, J.: The algorithm selection problem. Advances in computers **15** (1976) 65–118
2. Gomes, C., Selman, B.: Algorithm portfolios. AI **126**(1) (2001) 43–62
3. Huberman, B., Lukose, R., Hogg, T.: An economics approach to hard computational problems. Science **275**(5296) (1997) 51
4. Smith-Miles, K.: Cross-disciplinary perspectives on meta-learning for algorithm selection. ACM Computing Surveys **41**(1) (2008) 1–25
5. Hutter, F., Hoos, H.H., Leyton-Brown, K., Stützle, T.: ParamILS: an automatic algorithm configuration framework. JAIR **36**(1) (2009) 267–306
6. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. JMLR **13** (2012) 281–305
7. Eiben, A., Michalewicz, Z., Schoenauer, M., Smith, J.: Parameter Control in Evolutionary Algorithms. Parameter Setting in Evolutionary Algorithms (2007) 19–46
8. Xu, L., Hutter, F., Hoos, H.H., Leyton-Brown, K.: SATzilla: portfolio-based algorithm selection for SAT. JAIR **32**(1) (2008) 565–606
9. Xu, L., Hoos, H.H., Leyton-Brown, K.: Hydra: Automatically configuring algorithms for portfolio-based selection. In: AAAI'10. (2010) 210–216
10. Adenso-Diaz, B., Laguna, M.: Fine-tuning of algorithms using fractional experimental designs and local search. OR **54**(1) (2006) 99–114
11. Birattari, M., Stützle, T., Paquete, L., Varrentrapp, K.: A racing algorithm for configuring metaheuristics. In: GECCO'02. (2002) 11–18
12. Kadioglu, S., Malitsky, Y., Sellmann, M., Tierney, K.: ISAC–instance-specific algorithm configuration. In: ECAI'10. (2010) 751–756
13. Lau, H.C., Xiao, F., Halim, S.: A framework for automated parameter tuning in heuristic design. In: MIC'09, Hamburg, Germany (June 13–16 2009)
14. Seipp, J., Braun, M., Garimort, J., Helmert, M.: Learning portfolios of automatically tuned planners. In: ICAPS'12, Atibaia/Sao Paulo, Brazil (2012)

15. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: LION'11. Volume 6683 of LNCS. Springer (2011) 507–523
16. Lindauer, M., Hoos, H.H., Hutter, F., Schaub, T.: AutoFolio: an automatically configured algorithm selector. Journal of Artificial Intelligence Research (2015) 745–778
17. Mısır, M., Handoko, S.D., Lau, H.C.: ADVISER: a web-based algorithm portfolio deviser. In Dhaenens, C., Jourdan, L., Marmion, M.E., eds.: Learning and Intelligent Optimization. Volume 8994 of LNCS. Springer (2015) 23–28
18. Kadioglu, S., Malitsky, Y., Sabharwal, A., Samulowitz, H., Sellmann, M.: Algorithm selection and scheduling. In: CP'11. Volume 6876 of LNSC., Springer (2011) 454–469
19. Hutter, F., Hoos, H.H., Stutzle, T.: Automatic algorithm configuration based on local search. In: AAAI'07. Volume 22. (2007) 1152
20. Ansótegui, C., Sellmann, M., Tierney, K.: A gender-based genetic algorithm for the automatic configuration of algorithms. CP'09 (2009) 142–157
21. KhudaBukhsh, A.R., Xu, L., Hoos, H.H., Leyton-Brown, K.: SATenstein: Automatically building local search sat solvers from components. In: IJCAI'09. (2009) 517–524
22. Montgomery, D.: Design and Analysis of Expeirments. John Wiley and Sons Inc, 6th Edition (2005)
23. Gunawan, A., Lau, H.C., Lindawati: Fine-tuning algorithm parameters using the design of experiments approach. In Coello Coello, C., ed.: LION'11. LNCS, Springer (2011) 278–292
24. Mısır, M., Verbeeck, K., De Causmaecker, P., Vanden Berghe, G.: An investigation on the generality level of selection hyper-heuristics under different empirical conditions. Applied Soft Computing **13**(7) (2013) 3335–3353
25. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of computational and applied mathematics **20** (1987) 53–65
26. Mısır, M., Handoko, S.D., Lau, H.C.: OSCAR: Online selection of algorithm portfolios with case study on memetic algorithms. In Dhaenens, C., Jourdan, L., Marmion, M.E., eds.: Learning and Intelligent Optimization. Volume 8994 of LNCS. Springer (2015) 59–73
27. Pfahringer, B., Bensusan, H., Giraud-Carrier, C.: Tell me who can learn you and i can tell you who you are: Landmarking various learning algorithms. In: Proceedings of the 17th international conference on machine learning. (2000) 743–750
28. Wolpert, D., Macready, W.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation **1** (1997) 67–82
29. He, J., He, F., Dong, H.: Pure strategy or mixed strategy? - an initial comparison of their asymptotic convergence rate and asymptotic hitting time. In Hao, J.K., Middendorf, M., eds.: (EvoCOP'12). Volume 7245 of LNCS. (2012) 218–229
30. Lehre, P.K., Özcan, E.: A runtime analysis of simple hyper-heuristics: To mix or not to mix operators. In: FOGA'13, Adelaide, Australia (2013)
31. Cowling, P., Kendall, G., Soubeiga, E.: A hyperheuristic approach to scheduling a sales summit. In: PATAT '00: Selected papers, Springer (2001) 176–190
32. Mısır, M., Wauters, T., Verbeeck, K., Vanden Berghe, G.: A new learning hyper-heuristic for the traveling tournament problem. In Caserta, M., Voss, S., eds.: Metaheuristics: Intelligent Problem Solving - MIC'09. Springer (2010)
33. Thathachar, M., Sastry, P.: Networks of Learning Automata: Techniques for Online Stochastic Optimization. Kluwer Academic Publishers (2004)
34. Ng, K.M., Gunawan, A., Poh, K.L.: A hybrid algorithm for the quadratic assignment problem. In: CSC'08, Nevada, USA (July 14–17 2008)
35. Burkard, R., Karisch, S., Rendl, F.: Qaplib–a quadratic assignment problem library. Journal of Global Optimization **10**(4) (1997) 391–403
36. Halim, S., Yap, R., Lau, H.: An integrated white+black box approach for designing and tuning stochastic local search. In: CP'07. Volume 4741 of LNCS. Springer (2007) 332–347

37. Lindauer, M., Hoos, H., Hutter, F.: From sequential algorithm selection to parallel portfolio selection. In Dhaenens, C., Jourdan, L., Marmion, M.E., eds.: Learning and Intelligent Optimization. Volume 8994 of LNCS. Springer (2015) 1–16
38. Hutter, F., Hoos, H., Leyton-Brown, K.: An efficient approach for assessing hyperparameter importance. In: Proceedings of International Conference on Machine Learning 2014 (ICML 2014). (June 2014) 754–762
39. López-Ibáñez, M., Dubois-Lacoste, J., Stützle, T., Birattari, M.: The I-Race package, iterated race for automatic algorithm configuration. Technical Report TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium (2011)