7-2011

Real-world parameter tuning using factorial design with parameter decomposition

Aldy GUNAWAN
Singapore Management University, aldygunawan@smu.edu.sg

Hoong Chuin LAU
Singapore Management University, hclau@smu.edu.sg

Elaine WONG
EADS Singapore

Citation

GUNAWAN, Aldy; LAU, Hoong Chuin; and WONG, Elaine. Real-world parameter tuning using factorial design with parameter decomposition. (2011). *Advances in metaheuristics: Proceedings of the Ninth Meta-heuristics International Conference, Udine, Italy, 25-28 July 2011*. 53, 37-59.
Available at: https://ink.library.smu.edu.sg/sis_research/1612

Published in Advances in metaheuristics. Berlin: Springer. 2013 July, pp. 37-59.
http://doi.org/10.1007/978-1-4614-6322-1_3

Real-World Parameter Tuning Using Factorial Design with Parameter Decomposition

Aldy Gunawan, Hoong Chuin Lau, and Elaine Wong

Abstract In this paper, we explore the idea of improving the efficiency of factorial design for parameter tuning of metaheuristics. In a standard full factorial design, the number of runs increases exponentially as the number of parameters. To reduce the parameter search space, one option is to first partition parameters into disjoint categories. While this may be done manually based on user guidance, an automated approach proposed in this paper is to apply a fractional factorial design to partition parameters based on their main effects where each partition is then tuned independently. With a careful choice of fractional design, our approach yields a linear rather than exponential run time performance with respect to the number of parameters. We empirically evaluate our approach for tuning a simulated annealing algorithm that solves an industry spares inventory optimization problem. We show that our proposed methodology leads to improvements in terms of the quality of solutions when compared to a pure application of an automated parameter tuning configurator ParamILS.

1 Introduction

The performance of a metaheuristic algorithm largely depends on the expertise in tuning the algorithm's control parameters. For example, a simulated annealing algorithm yields good solutions only if several parameters such as initial temperature, cooling factor, number of iterations and so on are properly tuned. However, finding the best combination of parameter settings is a tedious and time-consuming task.

Aldy Gunawan · Hoong Chuin Lau
School of Information Systems, Singapore Management University, Singapore
e-mail: aldygunawan@smu.edu.sg, hclau@smu.edu.sg

Elaine Wong
Innovation Works, EADS Singapore, Singapore
e-mail: elaine.wong.kl@eads.net

In recent years, several automated approaches for finding good parameter settings have been proposed. These approaches can be classified into model-free algorithm configuration methods and model-based approaches. Model-free algorithms are simpler to implement than model-based approaches because the former can be applied out-of-the-box, while the latter requires iterations between fitting models and using them to make choices about which configurations to investigate [13]. Examples of model-free algorithm configuration methods are F-Race [4] and Iterated F-Race [5], ParamILS [10] and genetic algorithm GGA [2]; examples of model-based approaches are SPO+ [12] and SMAC [13]. SPO+ is an extension of the Sequential Parameter Optimization (SPO) framework, which focuses on tuning algorithms with continuous parameters for a single problem instance. SMAC, unlike SPO+, can be used to handle categorical parameters. Both, model-free and model based approaches have been used to optimize various algorithms for some classical combinatorial optimization problems, such as the propositional satisfiability problem (SAT), the travelling salesman problem (TSP) and MIP problems.

Many real-world problems make use of optimization algorithms that contain a set of parameters. For example, the CPLEX solver has 76 parameters that affect the search process [11]. In such problems, parameter search space reduction can be important in order for automated tuning to become computationally feasible. By decomposing the parameters into disjoint partitions and tuning them separately, the parameter search space will be significantly reduced. Lau and Xiao [14], for example, decompose the parameters set into disjoint graphs based on the correlation among the parameters, and the approach was used for tuning a genetic algorithm (GA) for the bandwidth minimization problem (BMP).

Another approach for decomposing parameters is found in Design of Experiments (DOE). DOE has been used to systematically find the best parameters values for a heuristic. Coy et al. [7] proposed a procedure, based on DOE and gradient descent, to find parameter settings for vehicle routing heuristics. The drawbacks of the approach are that: the linear approximation of the response surface and the average setting might not be appropriate if the class of problems is too broad. Adenso-Díaz and Laguna [1] developed CALIBRA, which employs a Taguchi fractional experimental design and a local search procedure to tune up to five parameters. CALIBRA only focuses on the linear assumption without examining interactions between parameters. Hutter et al. [13] proposed the use of response surface models to characterize the importance of parameters for future research direction. All the above-mentioned approaches show that factorial experimental design is particularly useful in the early stages of experimental work, when many parameters are likely to be investigated. Unfortunately, the caveat of a full factorial design is that the number of runs required increases exponentially with the number of parameters [8, 16].

In this paper, we propose a decomposition approach to reduce the parameter space of parameters. Consider an algorithm, called the target algorithm, that requires a number of parameters to be tuned. We divide the parameters into a number of disjoint categories. This is done either manually based on user guidance; or automatically using a fractional factorial design, which measures the main effect ranks among the parameters. Since our focus is to separate main effects and interactions, the so-called Resolution IV Design [15] is used.

We then apply the proposed tuning framework by Gunawan, Lau, and Lindawati [8] for each category of parameters. The proposed tuning framework is divided into three phases, namely screening, exploration and exploitation. Given a set of parameters to be tuned, the screening phase seeks to rank these parameters so as to determine unimportant parameters, whose values have insignificant impact on the solution quality. Values for unimportant parameters can be set to some constant numbers and thereby further reduce the resulting parameter space to be explored. In the exploration phase, a first-order polynomial model based on a response surface is then built to define the promising initial ranges for the important parameters. These promising initial ranges are then sent to an automated tuning configurator such as ParamILS [10] in the exploitation phase to find the desired parameter setting.

This work arises from a real optimization problem in the aerospace company EADS. We are concerned with tuning an existing algorithm that involves a set of parameters and intensive computations. The major contributions of this paper are summarized as follows:

- We propose the idea of parameter decomposition and describe two approaches.
- We apply our approach to tune a simulated annealing algorithm that is used to solve a computationally-intensive spares inventory optimization problem.

The remainder of this paper is organized as follows. The automated tuning framework of [8] is outlined in Sect. 2. Section 3 presents the spares inventory optimization problem and the target algorithm. Section 4 presents our decomposition approaches. Section 5 provides computational results of our proposed approach applied to the spares inventory optimization problem. Finally, we provide some concluding perspectives and future research directions in Sect. 6.

2 Automated Tuning Framework

The automated tuning problem is defined as follows: θ is the finite set of candidate parameter configurations; X is a set of parameters to be tuned (each parameter can be either discrete or continuous, over a numeric range); TA is a target algorithm; I_{tr} is a set of training instances; $H(x)$ is a function that measures algorithm $TA's$ performance under a fixed parameter setting x on a set of problem instances. The solution of the automated tuning problem is the configuration x^* such that:

$$x^* = argmin_x H(x) \tag{1}$$

In this paper, we define the performance metric as the average percentage deviation of the set of obtained solutions from the optimal (or best known) solutions. The goal of automated tuning is to find x that optimizes the performance metric w.r.t. an (unknown) instance distribution. While the true performance metric cannot be computed exactly, it can be estimated using the given set of training instances I_{tr}. Subsequently, to verify the quality of this parameter setting, we measure the performance against a set of test instances.

The framework proposed in [8] consists of three phases, namely screening, exploration and exploitation. In the screening phase, we determine which parameters exhibit significant main effects thereby reducing the number of parameters to consider. Parameters with statistically significant p-value (less than 10 %) are referred to important parameters, while parameters with p-value greater than 10 % are considered unimportant parameters. For this purpose, [8] proposes a 2^k factorial design which consists of k parameters, where each parameter p_i only has two levels (l_i and u_i). A complete design requires $(2 \times 2 \times \cdots \times 2) \times a = a \times 2^k$ observations where a represents the number of replicates for a particular parameter setting (see [15]).

Let m be the total number of important parameters ($m \leq k$) determined in the screening phase. The exploration phase runs the target algorithm with respect to $2^m + 1$ possible parameter settings with an additional parameter setting defined by the centre point value of each parameter. A first-order (planar) model is then built to represent the relationship among parameters and the objective function value.

In order to test the significance of the planar model, interaction and curvature tests have to be conducted. The interaction test is used to test the significance of any interaction between parameters by looking at the significance of the estimated coefficient between two parameters. The curvature test tests whether the planar model is adequate to represent the local response function. The surface of parameters can still be approximated by a planar model as long as the existence of either interaction or curvature is not significant.

If the planar model assumption still holds, the process is then continued by applying steepest descent that moves rapidly to the vicinity of the optimum. From a statistical point of view, the region close to the optimum has been reached if the planar model assumption does not hold anymore. The final range for each important parameter is used as the input in the exploitation phase in which a configurator is used to find the optimal point in the region. In this study (as in [8]), ParamILS is applied to tune the target algorithm.

3 Case Description

In this section, we provide a description of the target algorithm used in this paper. Our problem is an aircraft spares inventory optimization problem arising from maintenance, repair and overhaul (MRO) operations faced by the aircraft total service support provider. The service provider is required to provide necessary spares to meet target service levels of customers of performance-based contracts operating out of a network of airports. The goal is to determine the optimal inventory allocation strategy that can fulfill target services levels. Optimality is defined in terms of total life cycle costs for spares comprising inventory holding cost, part purchasing and repair cost, logistics delivery cost, while service levels in terms of spares fill-rates.

The inclusion of logistic policies and strategies into the inventory optimization model allow better representation of real world operations. This problem incorporates (a) the use of lateral transshipment between warehouses, (b) service-time commitments for spares delivery, (c) inventory rationing for selected customers, and (d) option to scrap non-serviceable parts. Lateral transshipments are used when a serviceable spare part is not available at a local warehouse where the demand is triggered. In such cases, the serviceable spare part is shipped from an alternative warehouse to fulfill the urgent demand. As a result, the one (serviceable)-for-one (unserviceable) exchange would incur logistics transport costs. The optimized inventory allocation is required to make strategic trade-offs between dispatching from a central warehouse (lower warehousing costs) versus distributing inventory across the entire network (lower logistics transport costs) (For further reference, refer to [3]).

Service-time commitments are applicable in cases where spares demand is met from an alternative warehouse via lateral transshipments. The logistics transport has to deliver the serviceable part in the requested time in order to fulfill the demand. Failing to do so results in a penalty imposed on the service provider. Higher penalty rates encourage tighter conformance to the target service level, and possibly to the extent of even surpassing the targets required by the customers. Inventory rationing is a strategy adopted to prioritize different classes of customers. Unlike a simple pooling strategy, this strategy involves reserving inventory for customers requiring higher target service levels, as part of a higher priced contract and higher penalty rates [6]. Scrapping of parts is a resupply strategy to handle unserviceable parts. When deciding on scrap rates, the service provider takes into account repair shop turn-around times and costs, as well as a new part unit price and purchase lead time. The above problem is solved by a Simulated Annealing algorithm (see Fig. 1).

The parameters that need to be tuned are summarized in Table 1. The SA solution-acceptance criteria, which we term as oracle function, accepts a solution (x_k) that is worse than the incumbent solution (x) with the probability $Pr(x, x_k, T_k) = e^{(F(x)-F(x_k))/T_k}$, where T_k is the temperature at iteration k. In the oracle function, the exponential function is close to one if the new objective value $F(x_k)$ is close to $F(x)$, and approaches zero as the difference increases. An additional parameter oracle strictness (*Oracle*) is introduced into the SA algorithm. Oracle strictness adjusts the probability of acceptance as follows:

$$\hat{Pr}(x, x_k, T_k) = \left\{ \frac{Pr(x, x_k, T_k) - Oracle/100}{1 - Oracle/100} \right\}^{+} \tag{2}$$

The probability of accepting solution x_k is zero if $Pr(x, x_k, T_k)$ is less than *Oracle*, hence it is possible to tune the algorithm to a simple local search algorithm by setting *Oracle* to 100. Unlike the cooling function, which dynamically adjusts T_k and hence the likelihood of acceptance according to the iteration count, the oracle strictness is fixed throughout the algorithm.

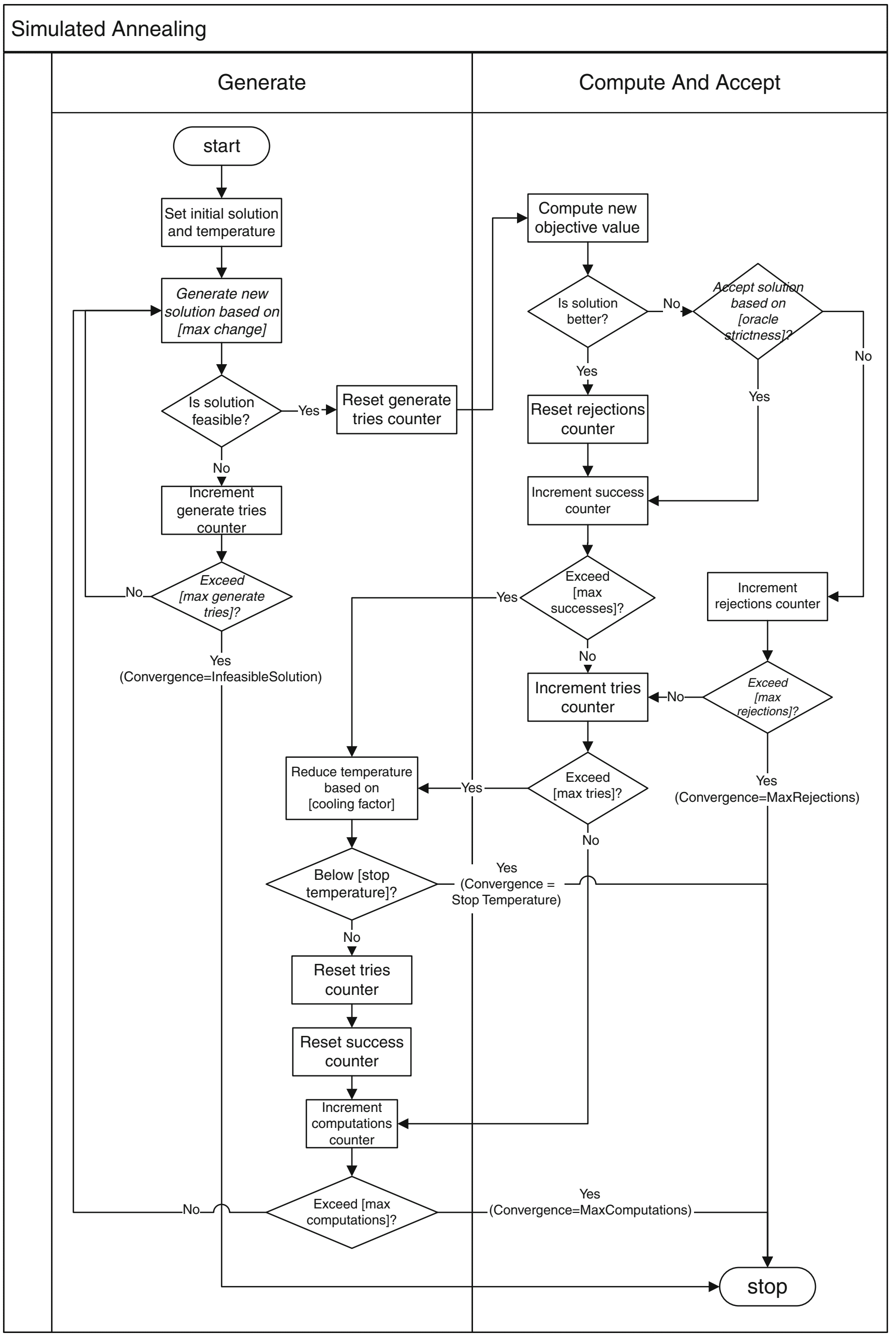

Simulated Annealing
Generate
Compute And Accept
start
Set initial solution and temperature
Generate new solution based on [max change]
Is solution feasible?
Yes
Reset generate tries counter
No
Increment generate tries counter
Exceed [max generate tries]?
No
Yes (Convergence=InfeasibleSolution)
Compute new objective value
Is solution better?
No
Accept solution based on [oracle strictness]?
No
Yes
Yes
Reset rejections counter
Increment success counter
Exceed [max successes]?
Yes
No
Increment rejections counter
Exceed [max rejections]?
No
Yes (Convergence=MaxRejections)
Increment tries counter
Exceed [max tries]?
Yes
No
Reduce temperature based on [cooling factor]
Below [stop temperature]?
Yes (Convergence = Stop Temperature)
No
Reset tries counter
Reset success counter
Increment computations counter
Exceed [max computations]?
No
Yes (Convergence=MaxComputations)
stop

Fig. 1: Flow chart of simulated annealing algorithm

Table 1: Parameter definition

Parameters (p_i)	Abbreviation	Definition
Max successes	Success	Maximum number of successes within one temperature
Max tries	Tries	Maximum number of tries within one temperature
Max computations	Comp	Maximum number of solutions generated
Max rejections	Reject	Maximum number of consecutive rejections
Max change	Change	Maximum change in a variable value when generating a new solution
Max generate tries	Generate	Number of tries to generate a feasible solution
Cooling factor	Cooling	Factor to reduce the temperature by during each Temperature change
Oracle strictness	Oracle	A value to depict the strictness of the oracle function in accepting a new solution that has an objective value worse than the current one. A higher value would result in a higher rejection rate (e.g. a value of 100 would accept only better solutions)

4 *s*-step Decomposition Approach

The main idea of the s-step decomposition approach is to decompose n parameters into s disjoint categories, namely CAT_1, CAT_2, ... CAT_s. Each category consists of $n_1, n_2, \ldots n_s$ parameters respectively where

$$n = n_1 + n_2 + \ldots + n_s \tag{3}$$

We assume that each category comprises at most 5 parameters [1]. Figure 2 gives a pictorial view of our decomposition approach, which is based on a s-step decomposition of factorial design. As described in Sect. 1, the n parameters to be tuned are initially divided into s different categories. By performing tuning separately on each category, we effectively reduce the parameter space from $2^{n_1+n_2+\ldots+n_s}$ to $2^{n_1} + 2^{n_2} + \ldots + 2^{n_s}$ possible parameter value settings.

Our approach proceeds as follows. In the First Step, we focus on tuning the n_1 parameters, while setting the other $n - n_1$ parameters to their default values. To do that, we apply the three phases described in Sect. 2 on the training instances. At the end of this step, we obtain the best parameter setting for these n_1 parameters. Then, for the subsequent step i (from 2 to s), we tune n_i parameters by setting the best values for the parameters obtained from steps 1 to $i - 1$. Once we obtain the best parameter configuration for all n parameters, we run the target algorithm with that particular configuration on the test instances.

To evaluate the performance of our proposed decomposition approach, we compare the results obtained by our approach against those obtained using the respective default parameter settings. We will show that our approach can lead to improvements in terms of the average objective values of the solutions. In the following sub-sections, we present two different decomposition approaches - one based on user guidance, and the other based on automated decomposition.

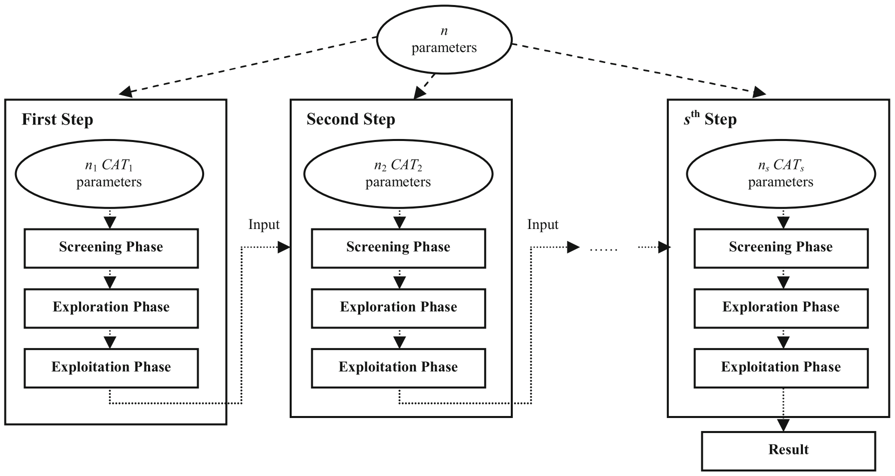


Fig. 2: s-step decomposition of factorial design

4.1 User-Guided Decomposition

In user-guided decomposition, we rely on information provided by the user for decomposing the parameters. This assumes that the user has conducted preliminary experiments a-priori and obtained observations and insights about the parameter importance levels and sensitivity to solution qualities. We then proceed to apply the s-step decomposition approach described above for each parameter category.

4.2 Automated Decomposition

A 2^k full factorial design would require a large number of runs when the number of parameters increases. An algorithm with 8 parameters would require 256 parameter settings where only 8 degrees of freedom correspond to main effects of parameters, while the rest correspond to interactions. At this point, since we have little interest in the interaction effects, we focus on decomposing the parameters based on their main effects.

For this purpose, we use a fractional factorial design to derive the parameter importance. There are several kinds of fractional factorial design classes based on existing alias relationships in the design, also known as the resolution. The concept of design resolution is to catalog fractional factorial designs according to the alias patterns they produce [15]. Resolution describes the degree to which estimated main effects are aliased (or confounded) with estimated 2-level interactions, 3-level interactions, etc. Here, we consider a Resolution IV ($2^{k-p}{}_{IV}$ design) where no main effect is aliased with any other main effect or two-parameter interactions. Note that the value of p depends on the highest possible design resolution (see Table 2) [15].

Table 2: Resolution IV design

Number of parameters (k)	p	Minimum number of runs
4	1	$2^{4-1} = 8$
5	1	$2^{5-1} = 16$
6	2	$2^{6-2} = 16$
7	3	$2^{7-3} = 16$
8	4	$2^{8-4} = 16$
9–16	Varying with k	32
17–32	Varying with k	64
33–64	Varying with k	128
...	...	...

A Resolution IV design can be used first since we assume that the parameters have monotonic effects on the response variable [15]. We take $k = 6$ parameters as an example. In a ${2^{6-2}}_{IV}$ design, only 16 parameter settings are required. To generate this design, a 2^4 design with the parameters A, B, C, D are set as the basic design, and only two additional columns are required for parameters E and F. To get the new columns, two design generators $E = A \times B \times C$ and $F = B \times C \times D$ are selected. Details of this process can be found in [15].

Note that the advantage of the above scheme is that even as the number of parameters k grows large, the minimum number of runs grows only linearly with k (rather than exponentially with the size of the original parameter space $\Theta(2^k)$). This demonstrates the scalability of our approach.

5 Experimental Results

In this section, we present the experimental results of tuning the simulated annealing algorithm (SA) applied to the spares inventory optimization problem described above. All the experiments were conducted on a PC running Windows XP with 2.33 GHz CPU and 1.96GB RAM.

50 instances were generated to represent different spares support scenarios. All instances have identical problem size, comprising three part numbers, two warehouses and three customers. A small problem size was intentionally adopted so as to allow us to have a more rigorous study of a computationally intensive spares optimization problem. Each iteration in SA involves simulating transshipment policies and inventory rationing strategies. However, the instances differ in terms of part specifications (such as unit price, failure rate, and repair turn-around-time), logistics (namely warehousing cost, transport time and cost), and contract terms (target fill rate and penalty). To decide on default parameter values, a basic inventory optimization problem, without sophisticated logistics policies or inventory rationing strategies, was used. The parameters for SA were manually tuned such that the algorithm

Table 3: Input scenarios on spares optimization problem

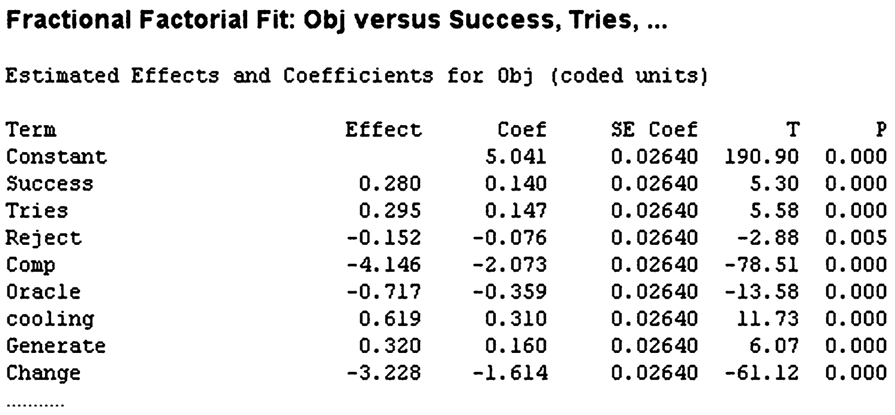
Specifications used to distinguish input scenarios	Range
Part specifications	
Unit price ($)	[1,000, 10,000]
Average repair cost (% of unit price)	[10, 20]
Expected repair shop turnaround time (years)	[0.01, 0.5]
Average number of spares demand at airport per year	[1. 50]
Logistics	
Average holding cost (% of unit price)	[10, 20]
Expected replenishment lead time at warehouse (years)	[0.01, 0.5]
Expected interval for transport between warehouses (h)	[1, 3]
Expected transport time between warehouses (h)	[1, 3]
Expected transport cost (% of unit price)	[10, 20]
Contract terms	
Target fill rate for operator (% of total annual demand)	[85, 100]
Delivery deadline at airport (h)	[1, 10]
Penalty rate ($)	[500, 5,000]

resulted in the best final objective value for the basic problem. The respective ranges of values are listed in Table 3.

Half of the instances were used as training instances and the rest as test instances. To account for the stochasticity involved in the SA algorithm, each instance was run 5 times and the performance of each instance was reported in terms of the average best objective value obtained and best objective value of 5 runs.

For the automated decomposition, a resolution IV fractional factorial design ($2^{8-3}{}_{IV}$ design) was adopted. The basic parameters are *Success*, *Tries*, *Reject*, *Comp* and *Oracle*. By applying principles described in [15], we define the complete alias relationships for the other three parameters: *Cooling*, *Generate* and *Change* where *Cooling* = *Success*×*Tries*×*Reject*, *Generate* = *Success*×*Tries*×*Comp* and *Change* = *Tries*×*Reject*×*Comp*×*Oracle*.

The result of the $2^{8-3}{}_{IV}$ design is summarized in Fig. 3. We observe that all parameters have p-value less than 10 %. We then proceed to classify the parameters according to the absolute main effect values. A parameter whose absolute effect value is among the top 4 would be classified as important, and less-important otherwise. Table 4 compares categories obtained by the automated-decomposition versus the user-guided one. Note that both approaches have categorized *Comp* and *Change* as CAT_1. Differences are that the user-guided decomposition categorized *Success* and *Tries* as CAT_1; and only the automated decomposition categorized *Cooling* and *Oracle* as CAT_1.



Fractional Factorial Fit: Obj versus Success, Tries, ...

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

Term                   Effect      Coef     SE Coef        T       P
Constant                          5.041     0.02640   190.90   0.000
Success                 0.280     0.140     0.02640     5.30   0.000
Tries                   0.295     0.147     0.02640     5.58   0.000
Reject                 -0.152    -0.076     0.02640    -2.88   0.005
Comp                   -4.146    -2.073     0.02640   -78.51   0.000
Oracle                 -0.717    -0.359     0.02640   -13.58   0.000
cooling                 0.619     0.310     0.02640    11.73   0.000
Generate                0.320     0.160     0.02640     6.07   0.000
Change                 -3.228    -1.614     0.02640   -61.12   0.000
...........
```

Fig. 3: Statistical results of a $2^{8-3}{}_{IV}$ design

Table 4: Parameter space for SA on spares optimization problem

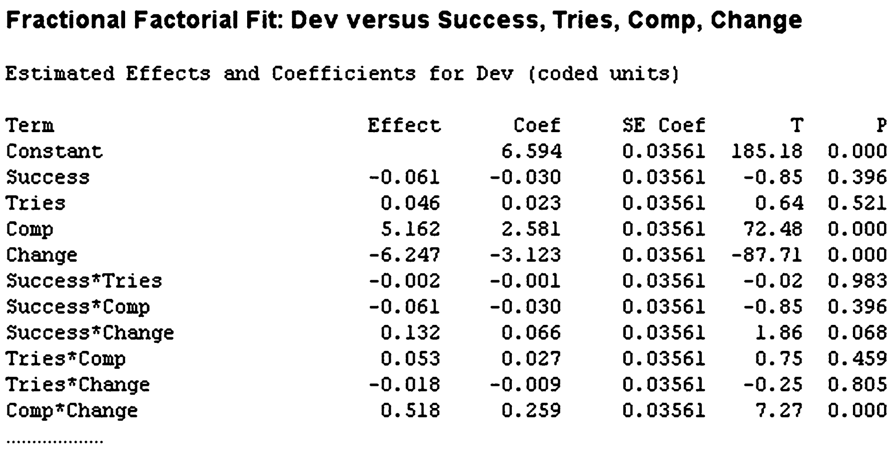
Parameters (p_i)	Default value	Range	User-guided	Automated	ParamILS step-size
Success	100	[100 , 1,000]	CAT_1	CAT_2	100
Tries	100	[100 , 1,000]	CAT_1	CAT_2	100
Comp	5,000	[1,000 , 5,000]	CAT_1	CAT_1	100
Reject	100	[100 , 1,000]	CAT_2	CAT_2	100
Change	2	[1 , 5]	CAT_1	CAT_1	1
Generate	300	[100 , 1,000]	CAT_2	CAT_2	100
Cooling	0.95	[0.5 , 1]	CAT_2	CAT_1	0.05
Oracle	30	[0 , 99]	CAT_2	CAT_1	10

5.1 *First Step*

The best values for CAT_1 parameters will be derived in this step, while setting CAT_2 parameters to their default values. Intermediate and overall results across all phases for the user-guided and automated decomposition are described below.

5.1.1 Screening Phase

Figures 4 and 5 provide results for the user-guided decomposition. Note that the dotted line in Fig. 5 represents the cut-off limit associated with the 10 % significance level. The bars at the left side of the dotted line represent insignificant parameters as well as insignificant interactions. We observe that the effect of *Comp* and *Change* are statistically significant, while both *Success* and *Tries* are insignificant. For further analysis, values for insignificant parameters can be set to a constant value. Since the spares optimization problem is a minimization problem, we set parameters with



```
Fractional Factorial Fit: Dev versus Success, Tries, Comp, Change

Estimated Effects and Coefficients for Dev (coded units)

Term                    Effect     Coef    SE Coef       T       P
Constant                          6.594    0.03561  185.18   0.000
Success                 -0.061   -0.030    0.03561   -0.85   0.396
Tries                    0.046    0.023    0.03561    0.64   0.521
Comp                     5.162    2.581    0.03561   72.48   0.000
Change                  -6.247   -3.123    0.03561  -87.71   0.000
Success*Tries           -0.002   -0.001    0.03561   -0.02   0.983
Success*Comp            -0.061   -0.030    0.03561   -0.85   0.396
Success*Change           0.132    0.066    0.03561    1.86   0.068
Tries*Comp               0.053    0.027    0.03561    0.75   0.459
Tries*Change            -0.018   -0.009    0.03561   -0.25   0.805
Comp*Change              0.518    0.259    0.03561    7.27   0.000
...................
```

Fig. 4: Statistical results (First Step)—user-guided decomposition

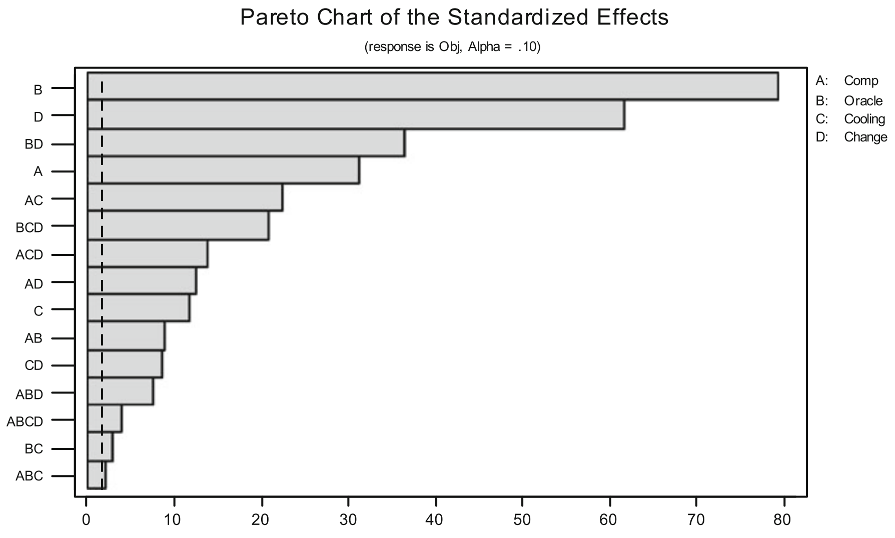
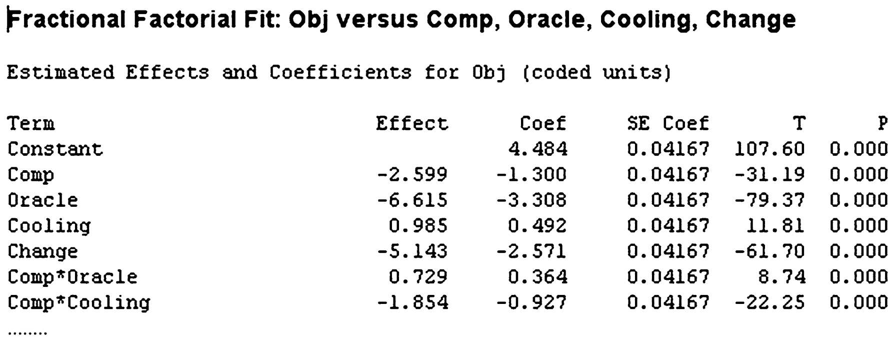


Fig. 5: Screening phase (First Step)—user-guided decomposition

negative effects to the upper-bound of the range, and those with positive effects to the lower-bound. In this case, *Success* is set to $1,000$, while *Tries* is set to 100.

Figures 6 and 7 provide results for the automated decomposition. We observe that all p-values are approximately zero, implying that all important parameters (*Comp*, *Change*, *Cooling* and *Oracle*) are statistically significant. Therefore, it is not possible to define constant values for the parameters; instead, we reduce the parameter range based on the main effect value. For parameters with negative effects, the new range will span the upper half of the original range (e.g. *Comp* is adjusted from



```
Fractional Factorial Fit: Obj versus Comp, Oracle, Cooling, Change

Estimated Effects and Coefficients for Obj (coded units)

Term                        Effect     Coef    SE Coef       T      P
Constant                              4.484    0.04167  107.60  0.000
Comp                        -2.599   -1.300    0.04167  -31.19  0.000
Oracle                      -6.615   -3.308    0.04167  -79.37  0.000
Cooling                      0.985    0.492    0.04167   11.81  0.000
Change                      -5.143   -2.571    0.04167  -61.70  0.000
Comp*Oracle                  0.729    0.364    0.04167    8.74  0.000
Comp*Cooling                -1.854   -0.927    0.04167  -22.25  0.000
........
```

Fig. 6: Statistical results (First Step)—automated decomposition

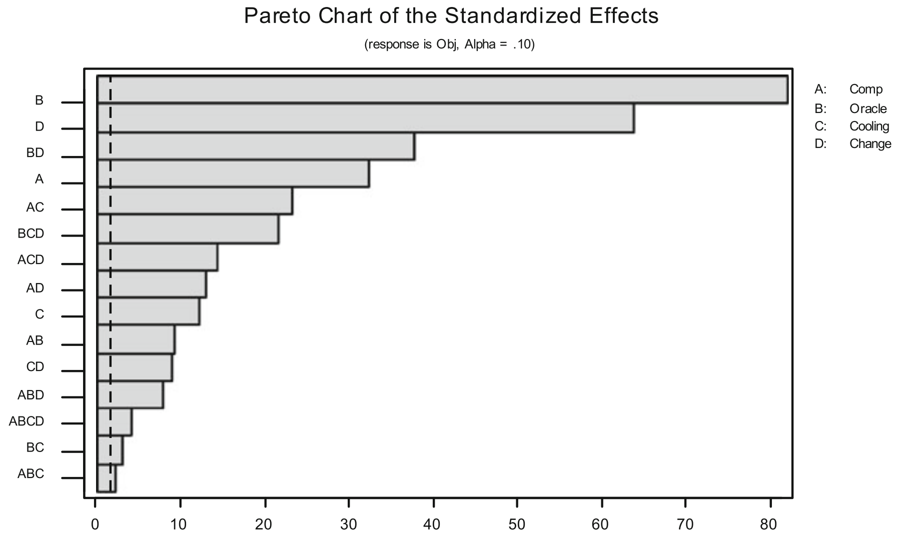


Fig. 7: Screening phase (First Step)—automated decomposition

[1,000, 5,000] to [2,500, 5,000]), and for parameters with positive effects, the new range will be limited to the lower half of the original range.

5.1.2 Exploration Phase

In this phase, we adopt the factorial experiment design to build the first-order Response Surface Model: $Y = \beta_0 + \beta_1 x_1 + \cdots + \beta_m x_m + \varepsilon$. Recall that from the screening phase, the user-decomposition approach has been simplified to consist of two parameters (*Comp* and *Change*), and four parameters (*Comp*, *Change*, *Cooling*, *Oracle*) with reduced ranges for parameter values for the automated decomposition.

Table 5: Final parameter space for SA algorithm from exploration phase (First Step)

	Final parameter space	
Parameters (p_i)	User-guided	Automated
Success	1,000	100
Tries	100	100
Comp	[1,000 , 3,000]	5,000
Reject	100	100
Change	[3 , 5]	[3 , 5]
Generate	300	300
Cooling	0.95	[0.5 , 0.75]
Oracle	30	[50 , 99]

To test the significance of the first-order model, two statistical tests, interaction and curvature tests, as described in Sect. 2, were run. It is assumed that the error component has a normal distribution with mean zero and unknown variance σ^2.

Figure 8 presents a sample of the statistical results from the statistical testing of the automated decomposition approach. At the end of the exploration phase, parameter *Comp* is no longer significant since p-value is greater than 10 %; therefore, we set its value to a high constant value. From the Analysis Variance output, each row represents the significance of each group of terms (main effects, 2-parameter interactions and so on). We observe that there are strong main effects of parameters. The row entitled "main effects" refers to the main effects of parameters, which are mainly affected by three significant parameters: *Oracle*, *Cooling* and *Change*.

The row labeled "2-Way Interactions" refers to the overall 2-way interaction between two different parameters. There may be some interaction between two parameters, such as *Oracle* and *Cooling*, *Oracle* and *Change*. There is no significant interaction between three and four parameters as well. We also conclude that the curvature test is significant. There is an evidence of curvature in the response over the region of exploration. It means that the region of the optimum has been found and then proceed to the exploitation phase to locate the best parameter setting.

In order to test the normality assumption, we generate the normal probability plot of the errors/residuals (Fig. 9). Since the errors lie approximately along a straight line, we do not suspect any problem with normality assumption. For the rest of the experiments, we conduct the same approach to test the normality of the errors.

Table 5 summarizes the final parameter space of parameters along the path of the steepest descent for user-guided decomposition and automated decomposition, respectively. This range is used as input for the exploitation phase.

5.1.3 Exploitation Phase

In this phase we apply ParamILS to tune the SA parameters based on the parameter space derived in the exploration phase. Table 6 summarizes the best parameter values obtained as a result of combining ParamILS and DOE. Since ParamILS requires

```
Fractional Factorial Fit: Obj versus Comp, Oracle, Cooling, ......

Estimated Effects and Coefficients for Obj (coded units)

Term                          Effect      Coef     SE Coef        T       P
Constant                                0.3314     0.01803    18.39   0.000
Comp                         -0.0391   -0.0196     0.01803    -1.08   0.286
Oracle                       -0.6303   -0.3151     0.01803   -17.48   0.000
Cooling                      -0.4848   -0.2424     0.01803   -13.45   0.000
Change                       -0.3617   -0.1808     0.01803   -10.03   0.000
.................
Oracle*Cooling                0.4617    0.2309     0.01803    12.81   0.000
Oracle*Change                 0.3563    0.1782     0.01803     9.88   0.000
Cooling*Change                0.2615    0.1307     0.01803     7.25   0.000
........

Analysis of Variance for Obj (coded units)

Source                 DF       Seq SS      Adj SS       Adj MS      F       P
Main Effects            4       9.1755     9.17553      2.29388 147.09   0.000
2-Way Interactions      6       4.9248     4.92482      0.82080  52.63   0.000
3-Way Interactions      4       0.7313     0.73132      0.18283  11.72   0.000
4-Way Interactions      1       0.0005     0.00052      0.00052   0.03   0.856
Curvature               1       0.0842     0.08422      0.08422   5.40   0.026
Residual Error         34       0.5302     0.53024      0.01560
  Pure Error           34       0.5302     0.53024      0.01560
Total                  50      15.4467
```

Fig. 8: Statistical results (First Step) for exploration phase—automated decomposition

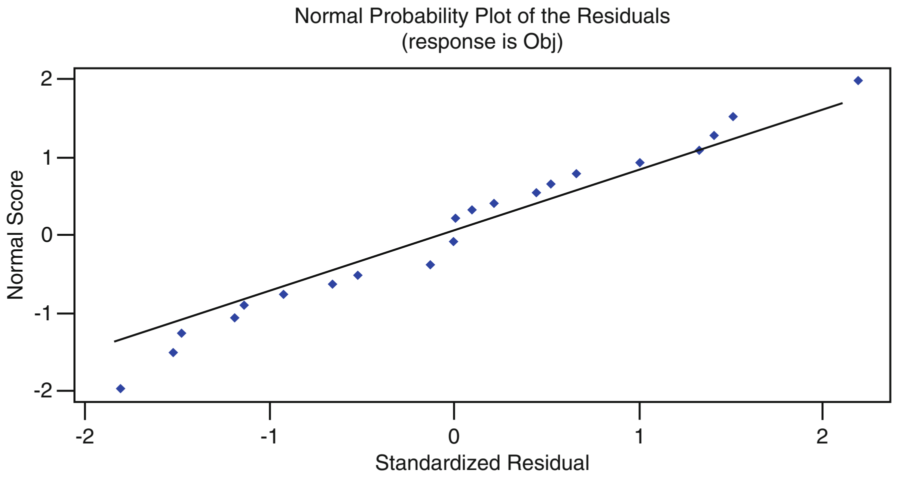


Fig. 9: Normal probability plot of errors—user-guided decomposition

all parameter domains to have discrete values, the parameters have to be discretized with a fixed step size (see last column of Table 4). Using the best parameter values, the average objective values from solving the spares optimization problem for the

Table 6: Parameter setting for SA on spares optimization problem (First Step)

Parameters (p_i)	Default value	ParamILS + DOE (user-guided decomposition)	ParamILS + DOE (automated decomposition)
Success	100	1,000	100*
Tries	100	100	100*
Comp	5,000	1,000	5,000
Reject	100	100*	100*
Change	2	3	3
Generate	300	300*	300*
Cooling	0.95	0.95*	0.6
Oracle	30	30*	90

*represents default values

25 training instances are provided in Table 7. The number of iterations of ParamILS is set to a constant number (e.g. 100). In this case, the number of iterations refers to how many times the target algorithm is being called.

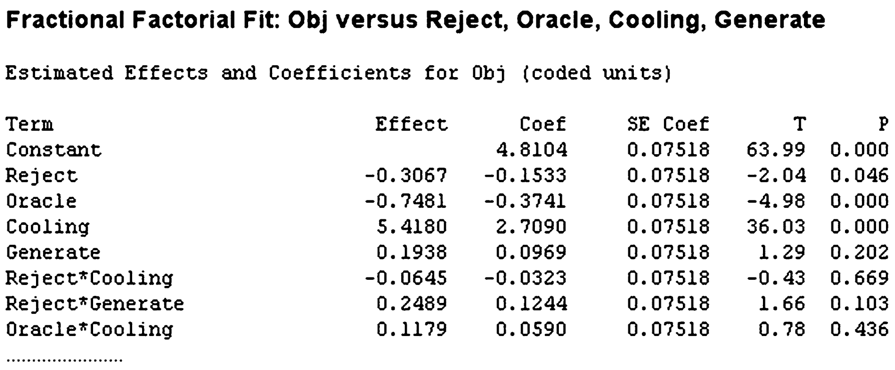
We have also tried to run ParamILS on a smaller number of parameters. More precisely, we selected four parameters in CAT_1 of the user guided decomposition (Table 3), and apply ParamILS directly without exploration phase. The best total life cycle cost obtained is \$1,492,117, which is larger compared with ParamILS+DOE under the user guided decomposition (\$1,262,087—as shown in Table 6).

We observe that user-guided decomposition results in higher values for parameters *Success* and *Cooling*, while the automated decomposition results in higher values for parameters *Comp* and *Oracle*. Note that the effect of *Comp* varies in different phases of the SA algorithm. In the early phase of the search, the focus is on diversification (random walk accepting worse moves), while in the later phase, the emphasis tends towards intensification. Hence, if the parameters are tuned such that cooling is slow (i.e. high *Cooling* and high *Success*) or oracle strictness low (i.e. worse solution is more likely accepted), the range of values for *Comp* may cause the search to already terminate in the diversification phase, where there is no guarantee that a larger *Comp* value produces better solutions. On the contrary, if the SA were tuned differently such that the range of values for *Comp* extends into the intensification phase, then larger *Comp* values may probably lead to better solutions.

In our context, we observe the former case for the user-guided decomposition results from the First Step (Table 6). The solution obtained by the user-guided decomposition is 50 % more than the automated decomposition, revealing that the combination of slow cooling (0.95), lengthy exploration at each temperature (1,000 success criteria), and low oracle strictness (30), prematurely terminates the search (with maximum *Comp* value of 3,000) in the diversification phase. Coupled with the relatively large neighbourhood (change set to 3 units) for each solution, the search oscillates unpredictably and as such a *Comp* value of 3,000 for user-guided decomposition did not generate better solutions than setting *Comp* to 1,000.

Table 7: Best total life cycle cost obtained from running SA with tuned parameters (First Step) on training instances

Algorithms	Average total life-cycle cost ($)
Default value	$995,185
ParamILS + DOE - user guided decomposition	$1,262,087
ParamILS + DOE - automated decomposition	$840,921
Screening + ParamILS - user guided decomposition	$1,492,117



Fractional Factorial Fit: Obj versus Reject, Oracle, Cooling, Generate

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

Term                            Effect       Coef     SE Coef        T       P
Constant                                   4.8104     0.07518    63.99   0.000
Reject                         -0.3067    -0.1533     0.07518    -2.04   0.046
Oracle                         -0.7481    -0.3741     0.07518    -4.98   0.000
Cooling                         5.4180     2.7090     0.07518    36.03   0.000
Generate                        0.1938     0.0969     0.07518     1.29   0.202
Reject*Cooling                 -0.0645    -0.0323     0.07518    -0.43   0.669
Reject*Generate                 0.2489     0.1244     0.07518     1.66   0.103
Oracle*Cooling                  0.1179     0.0590     0.07518     0.78   0.436
.......................
```

Fig. 10: Statistical results (Second Step)—user-guided decomposition

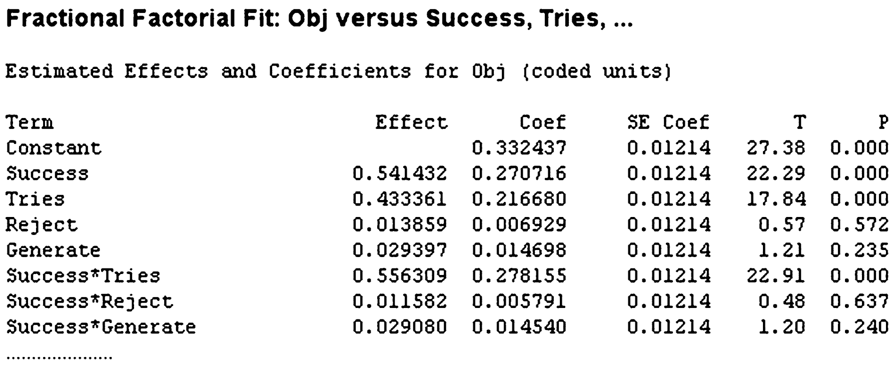
5.2 Second Step

In this step, values for CAT_2 parameters as listed in Table 4 will be determined. The values of CAT_1 parameters are set to the best values obtained by ParamILS + DOE in Step 1 (Table 6).

5.2.1 Screening Phase

Figure 10 presents the results of CAT_2 parameters for the user-guided decomposition approach. All parameters except parameter *Generate* are found to be significant. Since *Generate* has a positive main effect, it will be set to the lower bound of the original range. The intervals for the statistically significant parameters are reduced such that parameters with negative effects are assigned a new range covering the upper half of the original range (e.g. *Oracle* is adjusted to [50, 99]), and parameters with positive effects will be limited to the lower half of the original range.

Similar observations can be obtained from the results of the automated decomposition (Fig. 11). Only parameters *Success* and *Tries* are statistically significant. The other two parameters, *Reject* and *Generate*, are set to their lower bound. In



Fractional Factorial Fit: Obj versus Success, Tries, ...

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

Term                        Effect      Coef    SE Coef       T      P
Constant                             0.332437   0.01214   27.38  0.000
Success                   0.541432   0.270716   0.01214   22.29  0.000
Tries                     0.433361   0.216680   0.01214   17.84  0.000
Reject                    0.013859   0.006929   0.01214    0.57  0.572
Generate                  0.029397   0.014698   0.01214    1.21  0.235
Success*Tries             0.556309   0.278155   0.01214   22.91  0.000
Success*Reject            0.011582   0.005791   0.01214    0.48  0.637
Success*Generate          0.029080   0.014540   0.01214    1.20  0.240
......................
```

Fig. 11: Statistical results (Second Step)—automated decomposition

Table 8: Final parameter space for SA algorithm from exploration phase (Second Step)

	Final parameter space	
Parameters (p_i)	User-guided	Automated
Success	1,000	[100 , 500]
Tries	100	[100 , 500]
Comp	1,000	5,000
Reject	800	100
Change	3	3
Generate	100	100
Cooling	[0.5 , 0.6]	0.6
Oracle	[80 , 99]	90

the next phase, we only focus on parameters *Success* and *Tries* for the automatic decomposition.

5.2.2 Exploration Phase

In this phase, we apply the same approach as we have used in the First Step until we reach the region of the optimum. The final parameter space of parameters for both decomposition approaches are summarized in Table 8. For the user-guided decomposition, parameter *Reject* was found to be insignificant at the end of the exploration phase; hence it is set to a constant value equal to the lower bound from the screening phase.

Table 9: Parameter setting for SA on spares optimization problem (Second Step)

Parameters (p_i)	Default value	ParamILS + DOE (user-guided decomposition)	ParamILS + DOE (automated decomposition)
Success	100	1,000	500
Tries	100	100	300
Comp	5,000	1,000	5,000
Reject	100	800	100
Change	2	3	3
Generate	300	100	100
Cooling	0.95	0.55	0.6
Oracle	30	80	90

5.2.3 Exploitation Phase

Table 9 summarizes the best parameter values obtained as a result of combining ParamILS and DOE for the Second Step. As in the First Step, the three sets of parameter values were used for solving the training instances with SA. We conducted 5 runs on each instance and the average as well as the best found objective value for each instance is determined (Table 10). For each instance, the percentage improvement of each proposed decomposition approach from the default case (denoted as % improvement over average and best results, respectively) are calculated. We observe that the best objective values obtained in the Second Step outperform those of the First Step. In addition, by applying ParamILS+DOE, further improvements can be obtained for both training and test instances. The grand mean of average total life cycle costs of our proposed approaches are statistically different from those of default value setting (with the 10 % significance level).

In general, it is found that automated decomposition consistently yields better solutions than user-guided decomposition in terms of the grand mean of the average and best total life cycle cost. All training and test instances solved by the automated decomposition outperform the default case, while the user-guided decomposition provided better solutions for 18 training and 20 test instances only. Furthermore, the percentage improvement by using the automated decomposition is higher, achieving improvements of the best values of 16.28 % and 14.30 % for training and test instances, respectively, while the user-guided cases only achieved 1.33 % and 3.27 % respectively. Similar results are obtained for improvements in average objective values—the automated decomposition achieved 12.88 % and 12.01 % improvements in training and test instances, respectively, compared with the user-guided decomposition achieving 1.81 % and 4.53 % respectively. Operationally, improvements achieved by the automated-decomposition amounts to an average total life cycle cost savings of more than 10 %, which in our test cases is worth \$100,000 as compared to the default case.

The standard deviation (stdev) and average of coefficient of variance (CV) reaffirms the superior performance of the automated decomposition. As seen from Ta-

Table 10: Parameter tuning and results for SA on spares optimization problem (Second Step)

Results	Default value	ParamILS + DOE (user-guided decomp.)	ParamILS + DOE (automated decomp.)
Training instances			
Grand mean of average total life cycle cost ($)	995,608.80	969,816.90	837,820.40
p-value	–	0.079	0
N instances with better results	–	18	25
N instances with worse results	–	7	0
% improvement over average results	–	1.33 %	16.28 %
Average of best total life cycle cost ($)	956,164.70	929,390.30	837,109.00
p-value	–	0.025	0
N instances with better results	–	17	25
N instances with worse results	–	8	0
% improvement over best results	–	1.81 %	12.88 %
Average of std dev total life cycle cost	38,869.3	33,089.2	570.6
Average of CV total life cycle cost	3.86 %	3.63 %	0.00 %
Test instances			
Grand mean of average total life cycle cost ($)	1,169,127.00	1,125,680.00	1,004,810.00
p-value	–	0	0.001
N instances with better results	–	20	25
N instances with worse results	–	5	0
% improvement over average results	–	3.27 %	14.30 %
Average of best total life cycle cost ($)	1,137,053.00	1,082,371.00	1,002,520.00
p-value	–	0	0
N instances with better results	–	20	25
N instances with worse results	–	5	0
% improvement over best results	–	4.53 %	12.01 %
Average of std dev total life cycle cost	33,072.7	39,142	2,369.3
Average of CV total life cycle cost	2.90 %	3.62 %	0.19 %
Execution time	14 h	10 h	15 h

ble 10, stdev for the automated decomposition are relatively low for both training and test instances (at 570.6 and 2,369.3, respectively). A low standard deviation indicates that the results of several runs tend to be consistent and close to the average total life cycle cost. We also calculated the average of CV in order to measure the dispersion of results against the average total life cycle cost. All three approaches achieved low values of CV (less than 5 %). The automated decomposition is the lowest (best) compared with the default case and the user-guided decomposition approaches. In terms of average execution time for solving an instance, the automated decomposition requires 1 h and 5 h more than the default case and user-guided decomposition, respectively.

Table 11: ParamILS versus ParamILS + DOE (automated decomposition)

Results ($)	ParamILS	ParamILS+DOE (automated)	% Improvement
Training instances			
Grand mean of average total life cycle cost($)	1,007,763	837,820	16.86
Average of best total life cycle cost($)	961,296	837,109	12.92
Test instances			
Grand mean of average total life cycle cost($)	1,236,578	1,004,810	18.74
Average of best total life cycle cost($)	1,198,951	1,002,520	16.38

Finally, we compare the results obtained by the automated decomposition with the pure application of ParamILS (i.e. without the decomposition of parameters). In this case, the computational budget allocated (i.e. the number of iterations) is fixed. For instance, suppose the numbers of iterations of ParamILS and DOE are x and y respectively, the number of iterations of the first scenario is x+y, while the number of iterations of the second scenario is set to z, with $z = x$+y. Here, the number of iterations of ParamILS and DOE are 100 (as mentioned in Sect. 5.1.3) and $(5 \times 2^{n-3} + 5 \times 2^{n_1} + 5 \times 2^{n_2})$, respectively; while the number of iterations of the pure ParamILS is the total of both number of iterations.

Table 11 shows that for both training and test instances, the combination of ParamILS and DOE outperforms that of a direct application of ParamILS. We observe an improvement on the average of best total life cycle cost of 12.92 % and 16.38 % for training and test instances respectively.

6 Conclusion

In this paper, we explore the idea of improving the efficiency of factorial design for parameter tuning of metaheuristics. Our experimental results show that cost savings exceeding 10 % can be achieved. Due to the non-convexity of the search space, the solution converges to a local optimum when SA was manually configured. The tuned SA using our approach resulted in the significant annual cost savings because arbitrage opportunities, otherwise missed in the default SA settings, were identified.

For future extensions to this work, one can consider using other approaches such as Cluster Analysis to classify (decompose) parameters. Cluster analysis is a statistical approach used to divide the parameters into a set of objects based on their similarity (see [9]). The basic assumption in the first order (planar) model is that errors are assumed to be uncorrelated and distributed with mean 0 and constant (but unknown) variance. A normal plot of the errors should be approximately a straight line. Transformation of the response would be necessary if the normal plot is not a straight line [16]. We will also consider a more elaborate model such as the second order model in order to find the optimum region since interaction and curvature do exist [15].

In this paper, we make use of one automated parameter tuning configurator, namely ParamILS, and apply a 2-step (2 categories) decomposition approach. One can consider using other configurators, such as F-Race, for solving other real-world parameter tuning problems with larger number of parameters. Such a case study would also be interesting to explore a more general s-step decomposition.

Acknowledgements The authors wish to thank Lindawati and Nguyen Van Vinh for their valuable assistance in the technical implementation of the algorithm.

References

1. Adenso-Díaz, B., Laguna, M.: Fine-tuning of algorithms using fractional experimental design and local search. Oper. Res. **54**, 99–114 (2006)
2. Ansótegui, C., Sellmann, M., Tierney, K.: A gender-based genetic algorithm for the automatic configuration of solvers. In: Proceedings of the 15th International Conference on Principles and Practice of Constraint Programming, (CP 2009). Lecture Notes in Computer Science, vol. 5732, pp. 142–157. Springer, Heidelberg (2009)
3. Axsáter, S.: Modelling emergency lateral transshipments in inventory systems. Manag. Sci. **36**, 1329–1338 (1990)
4. Birattari, M., Stützle, T., Paquete, L., Varrentrapp, K.: A racing algorithm for configuring metaheuristics. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002), pp. 11–18. Morgan Kaufmann, San Francisco (2002)
5. Birattari, M., Yuan, Z., Balaprakash, P., Stützle, T.: F-race and iterated F-race: An overview. In: Bartz-Beislstein, T., Chiarandini, M., Paquete, L., Preuss, M. (eds.) Empirical Methods for the Analysis of Optimization Algorithms, pp. 311–336. Springer, Berlin (2010)
6. Cattani, K.D., Souza, G.C.: Inventory rationing and shipment flexibility alternatives for direct market firms. Prod. Oper. Manag. **11**, 441–457 (2002)
7. Coy, S.P., Golden, B.L., Runger, G.C., Wasil, E.A.: Using experimental design to find effective parameter settings for heuristics. J. Heuristics **7**, 77–97 (2000)
8. Gunawan, A., Lau, H.C., Lindawati: Fine-tuning algorithm parameters using the design of experiments approach. In: Coello, C.A.C. (ed.) Proceedings of Learning and Intelligent Optimization, 5th International Conference (LION 5). Lecture Notes in Computer Science, vol. 6683, pp. 278–292. Springer, Heidelberg (2011)
9. Han, J., Camber, M.: Data Mining: Concepts and Techniques, 2nd edn. Morgan Kaufmann, San Francisco (2006)
10. Hutter, F., Hoos, H.H., Leyton-Brown, K., Stützle, T.: ParamILS: an automatic algorithm configuration framework. J. Artif. Intell. Res. **36**, 267–306 (2009)
11. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Automated configuration of mixed integer programming solvers. In: Lodi, A., Milano, M., Toth, P. (eds.) Proceedings of International Conference on Integration of Artificial Intelligence and Operations Research techniques in Constraint Programming (CPAIOR). Lecture Notes in Computer Science, vol. 6140, pp. 186–202. Springer, Heidelberg (2010)
12. Hutter, F., Hoos, H.H., Leyton-Brown, K., Murphy, K.P.: Time-bounded sequential parameter optimization. In: Blum, C., Battiti, R. (eds.) Proceedings of Learning and Intelligent Optimization, 4th International Conference (LION 4). Lecture Notes in Computer Science, vol. 6073, pp. 281–298. Springer, Heidelberg (2010)

13. Hutter, F., Hoos, H.H., Leyton-Brown, K.: Sequential model-based optimization for general algorithm configuration. In: Proceedings of Learning and Intelligent Optimization, 5th International Conference (LION 5). Lecture Notes in Computer Science, vol. 6683, pp. 507–523. Springer, Heidelberg (2011)
14. Lau, H.C., Xiao, F.: Enhancing the speed and accuracy of automated parameter tuning in heuristic design. In: Proceedings of the 8th Metaheuristics International Conference (MIC 2009), Hamburg, Germany (2009)
15. Montgomery, D.C.: Design and Analysis of Experiments, 6th edn. Wiley, New York (2005)
16. Ridge, E., Kudenko, D.: Sequential experiment designs for screening and tuning parameters of stochastic heuristics. In: Proceedings of Workshop on Empirical Methods for the Analysis of Algorithms, pp. 27–34. Reykjavik, Iceland (2006)