Sequential experiment designs for screening and tuning parameters of stochastic heuristics

Enda Ridge and Daniel Kudenko

Dept. of Computer Science, University of York, York YO10 5DD, UK

Abstract. This paper describes a sequential experimentation approach for efficiently screening and tuning the parameters of a stochastic heuristic. Stochastic heuristics such as ant colony algorithms often use a large number of tuning parameters. Testing all combinations of these factors is prohibitive and inefficient. The sequential procedure recommended by this paper uses resolution IV fractional factorial designs with fold-over and centre points as an efficient way to screen the most important tuning parameters. The effects of the most important parameters are then modelled using a central composite design and optimised with standard numerical methods. All designs, their analyses and interpretation are illustrated using the Ant Colony System algorithm.
The use of standard designs and methods has the benefit that the presented procedure can easily be followed with commercial software rather that relying on custom methodologies and tools that have only been developed in an academic context. Such a procedure has not been applied to ant colony algorithms before.

1 Introduction and Motivation

A need for greater scientific rigour in the Operations Research (OR) and heuristics community was called for in the 1970s [1], and again in the 1990s [2], [3]. Ant colony optimisation (ACO) algorithms [4] are a relatively new class of stochastic heuristic for typical OR problems of combinatorial optimisation. Reported research has generally been innovative and has yielded important insights into ACO and its relation to other heuristics. However, few if any papers have described a disciplined use of either established Design of Experiment (DOE) techniques or statistical tools to explore data and test hypotheses. These are prerequisites for scientific work in other fields such as psychology and biology. Encouragingly, heuristics research rarely needs to ask more than three types of question:

- Is there a difference in performance between some group of algorithms and if so, how large is the difference between each of these algorithms?
- How does a modification to an algorithm (such as using a local search method) affect its performance?
- What are the most important parameters for tuning an algorithm and what are their values for optimal algorithm performance?

These questions can be answered with mathematical precision using statistical tools. However, few researchers have the time to become experts in the associated technical details before they can even begin to conduct their experiments. Given the small number of common research questions, it should be possible to create standard experimental designs as well as recommended statistical tests and rules of thumb for interpreting them. Unfortunately, the large number of tuning parameters typical of many heuristics, of which ant colony algorithms are a prime example, makes the usual full factorial design prohibitively expensive. A heuristic with 10 parameters (not uncommon) tested at the minimum of 2 levels would require $2^{10} = 1024$ treatment conditions in a full factorial design. This would be further increased by the number of replicates required for each condition. Furthermore, we do not have a sufficient body of knowledge to use our judgement to screen out the less important parameters in all situations. We may find, having conducted a full and expensive testing programme, that many of our questions did not have interesting answers.

This paper describes an established *sequential experimental procedure* for algorithm *parameter screening* and *tuning* (the third research question above). The appropriate experiment designs at each stage of the sequential procedure are described. The diagnostic tools and rules-of-thumb for interpreting them are given. Appropriate statistical tests and their interpretation are explained. The methods are illustrated using an Ant Colony algorithm.

The use of standard designs and methods has the benefit that the procedure can easily be followed with available commercial software rather than the custom academic tools advocated by some authors. Such a procedure with commercial tools has not been applied to ant colony algorithms before.

The next Section gives a brief overview of the Ant Colony algorithm used for illustration. Section 3 covers some preliminaries. Section 4 is the recommended sequential experimentation procedure. The paper concludes with some related work[1].

2 Ant Colony algorithms

Ant Colony algorithms are optimisation algorithms inspired by the activities of natural ants. This paper is illustrated with reference to Ant Colony System for the Travelling Salesperson Problem (TSP) [5]. Space restrictions prevent a detailed description of ACS. The interested reader can consult recent review texts [6].

Broadly, the algorithm works by placing a set of artificial ants on the nodes of a TSP graph. The ants build TSP solutions by moving along the graph edges. These movements are probabilistic and are influenced both by a heuristic function and the levels of a real-valued marker called a pheromone. Their movement decisions also favour nodes that are part of a candidate list. The probability that ants will explore new solutions is influenced by a threshold. Ants remove the pheromone marker from edges they traverse with a *local pheromone decay*. When all ants have produced a solution, the best solution is used to strengthen the amount of the pheromone marker on the best solution's edges with a *global pheromone update*. The parameters that control this algorithm are summarized in Table 1. The last four columns are not relevant at this stage but are referenced in Section 4.

It is common practice to hybridise ACS with local search procedures. This study focuses on ACS as a constructive heuristic and so omits any such procedure.

3 Preliminaries

Space restrictions prevent a comprehensive coverage of the general issues relevant to these experiments. Others have covered these in detail [7] for general heuristics. A familiarity with common DOE terminology is assumed.

Stopping Criterion. The choice of stopping criterion for an experiment run is difficult when algorithms can continue to run and improve indefinitely. CPU time is certainly not a reproducible metric and some independent metric such as a combinatorial count of an algorithm operation is required. The problem then becomes how many of these to use and how to scale them with problem size/difficulty so that comparisons are fair. We believe that a measure of algorithm stagnation is appropriate since this reflects the practical need of algorithms that continue to improve within reasonable time. In ACS, appropriate stagnation measures would be a fixed number of iterations or ant cycles without any improvement in solution quality.

Factor Coding. It is often convenient to code the experiment factors such that their values represent the high, medium, low, etc. levels of the factors. Analysis with coded variables can give

[1] The paper assumes some background knowledge of Design of Experiments due to space restrictions and the EMAA workshop's target audience. A significantly expanded version of this paper will appear as a University of York technical report, available from `http://www.cs.york.ac.uk/ftpdir/reports/`.

Param.	Meaning	Measurement Level	Factor	Low	High
m	Number of ants as fraction of problem size	Scale	A	10	100
q_0	Exploration/exploitation threshold	Scale	B	0.1	0.9
α	Influence of pheromone trails	Scale	J	0.5	8
β	Influence of heuristic	Scale	F	0.5	8
Cl	Ant candidate list length as fraction of problem size	Scale	K	10	75
ρ_{local}	Local pheromone decay	Scale	D	0.1	0.9
ρ_{global}	Global pheromone deposition	Scale	E	0.1	0.9
Q	Global pheromone update term	Scale	G	1	100
	Ant initial location method	Categorical	H	All at single random city	Randomly scattered

Table 1. Nine parameters for tuning ACS. Note that we have expressed the parameters m and cl as fractions of the problem size.

different numerical results to analysis with engineering units. Coded variables are mainly useful for determining relative size of factor effects.

Blocking on Seed. The algorithms in question achieve their stochastic behaviour using a pseudo-random number generator. For ACS, this impacts the initial location of ants, the exploration/exploitation probabilities and the movement decision. All variability between runs of these algorithms is directly attributable to the seed used in the generator and so seed should be treated as a known and controllable nuisance factor. A *nuisance factor* is a design factor that probably has an effect on the response but we are not interested in that effect [8]. The standard approach to dealing with such nuisance factors is *blocking*. This means that the same seed is used for all treatment conditions. Any replication must appear in a separate block defined by a new seed. This reduces the influence of variability due to seed.

Blocking on instance. There seems to be a general consensus that we should block on problem instances with authors who employ DOE also employing blocking on instances [9], [10], [7].

In the ACS scenario, this means that when we choose say a high level of the problem size factor, we use the same instance with this size in every treatment for which this size appears. This eliminates the variability between different instances with the same size. However, it weakens the scope of the conclusions we can draw from our data. Strictly speaking, our conclusions only apply to the instances on which we conducted the experiments. This can be mitigated by cross-validating our models and conclusions on new instances within the same levels of problem size as used in the experiments.

Power. Power is the probability of rejecting the null hypothesis when it is in fact false. Power should be at least 0.80 by convention. Power of a test depends on the expected effect size of interest and the significance level of the study. Power has generally been ignored in reported work. In brief, at the beginning of a design, the experimenter should fix the expected effect size and significance level and then increase the sample size with replicates until a minimum expected power of 80% is achieved. The observed power *after* analysis should be checked to ensure the predicted 80% has in fact been reached. Failing this, the design should be augmented with further replicates. Table

2 illustrates the increase in power to detect three effect sizes that is achieved by adding replicates to a 2^{5-2}_{III} design.

	% power for 25 replicates			% power for 95 replicates		
Effect size (Std. Dev.)	0.15	0.25	0.5	0.15	0.25	0.5
A	6.4	20.5	82.2	30.4	80.6	99.9
B	6.4	20.5	82.2	30.4	80.6	99.9
C	6.4	20.5	82.2	30.4	80.6	99.9
D	6.4	20.5	82.2	30.4	80.6	99.9
E	6.4	20.5	82.2	30.4	80.6	99.9
BC	6.4	20.5	82.2	30.4	80.6	99.9
BE	6.4	20.5	82.2	30.4	80.6	99.9

Table 2. Power of a 2^{5-2}_{III} design to detect 3 effect sizes at the 1% significance level.

Problem instances. Experiments in Evolutionary Computation (EC) have generally performed separate analyses for individual problems. This may be because the continuous functions EC optimises can differ greatly. ACS for the TSP (and related problems) always operates on fully connected graphs. These can only differ in their size and edge lengths and so we consider problem size as a 10^{th} factor (see Table 1) in the experiments to follow. The same instance is used for each level of problem size to remove this nuisance factor.

4 Recommended sequential procedure

The sequential experiment designs and statistical tests presented here are well established and in regular use in other fields. Their availability in commercially available software is a strong advantage. Their application to heuristics, and in particular Ant Colony algorithms, is new. The following was developed by analogy with designs and methods from other fields [8].

4.1 Screening

The first challenge is to reduce the number of tuning parameters under consideration. In this case, a naïve full factorial design for the 10 factors will require a prohibitive $2^{10} = 1024$ treatments multiplied by the necessary replicates for power. We are primarily interested in main effects (factor A, factor B...) and low-order interaction effects (AB, AD). Assuming higher order interactions are negligible, we can run a fraction of the 1024 treatments to yield insights into only the lower order effects. This is a **fractional factorial (FF)**. The most appropriate of these designs for screening is a so-called **Resolution IV** FF. The price we pay is that some second-order effects are indistinguishable from one another. They are **aliased.** These aliased effects can be disentangled by judicious choice of additional treatments to run.

Experiment Design

1. **Factors.** Choose the factors that are to be studied, identify their measurement and decide on appropriate high and low levels for each (see Table 1). Resource restrictions limit our choice of levels for problem size to 1000 and 2000.

2. **Significance level.** Choose an appropriate significance (alpha) level for the study. A conventional level is 1% or 5%.
3. **Factor Coding.** Code the high factor levels as +1 and low levels as –1.
4. **Blocking of controllable nuisance factors.** Identify known and controllable nuisance factors. Reduce their impact with blocking. For stochastic heuristics, the most important of these is random seed.
5. **Measuring uncontrollable nuisance factors.** Identify known and uncontrollable (but measurable) nuisance factors and make sure to measure these during the analysis. For example, an experiment that aims to closely investigate run times should measure issues such as CPU load during an experiment run as the experimenter usually has no control of the operating systems processes.
6. **Responses.** Identify the response(s) to be measured. These should be measured in as raw a form as possible rather than summarising into statistics such as averages. If possible, responses should be conflicting, to test the system under different circumstances. For example, speed and accuracy are usually conflicting responses.
7. **Variability at centre points.** A small number of replicates should be run at the design's centre point to estimate the variability of the process. If the process is extremely variable then further analysis of this response will be difficult.
8. **Choose Design.** For the given number of factors and blocks, choose an appropriate resolution IV 2-level fractional factorial design. Where there are several available designs, check whether the design requiring a smaller number of treatments has a satisfactory aliasing structure. Note that any resolution IV design will have aliased two-factor interactions. However, knowledge of the system and its most likely interactions may make some of these aliases negligible. Furthermore, judicious assignment of factors will result in the most important factors having the least aliasing in the generated design.
   For example, in the case of 10 factors, two resolution IV designs are available: 2^{10-5}_{IV} with 32 treatments and $2^{10\text{ - }4}_{\mathrm{IV}}$ with 64 treatments. If we decide to use 2 replicates of each treatment requiring 2 blocks then we must perform 64 and 128 runs respectively. An examination of the aliasing structure for these two designs reveals that all 2-factor interactions in the $2^{10\text{ - }5}_{\mathrm{IV}}$ design are aliased with another 2-factor interaction. The $2^{10\text{ - }4}_{\mathrm{IV}}$ design has only six aliased 2-factor interactions. Since there have been no reported exploratory analyses of this algorithm, we opt to use the more expensive $2^{10\text{ - }4}_{\mathrm{IV}}$ design. In this design, AB, JK, AJ, BK, AK, BJ, DE, GH, DG, EH, DH and EG are all aliased with another 2-factor interaction. C and F are never involved in these aliases and so should be assigned to factors of most importance. In this example, we choose problem size and influence of heuristic for C and F respectively. Furthermore, we see that A, B, J and K form a chain and D, E, G and H form a chain. We should attempt to group factors accordingly. For this design, we will assign the ant-related parameters m, q_0, α and cl to factors A, B, J and K respectively and the algorithm parameters ρ_{local}, ρ_{global}, Q and location method to D, E, G and H respectively. A summary of the factors, their symbols and levels is given in Table 1.
9. **Check Power.** For the study's chosen significance level (5%), examine the design's power to detect a given effect size. If the power is not greater than 80%, introduce replicates into the design. Recall that each new replicate requires a new block. Repeat this step until sufficient power is expected.
10. **Run Order.** This is now the minimum design. Generate a random run order for the design. This counteracts unknown and uncontrollable nuisance factors.
11. **Gather Data.** Collect data according to the treatments and their run order.

Analysis

1. **Check blocking.** Examine a scatter plot of response values against blocks. These should exhibit a random scatter to demonstrate that blocks do not interact with the response.

2. **Correlation.** Using a scatter plot of each response against another response, check that the responses are not highly correlated with one another. If they are highly correlated then one of the responses need not be analysed. For example, in ACS we might expect the number of ant tours performed and the accuracy of the final solution to be highly correlated.
3. **Note Best Responses.** Examine a scatter plot of the response variable against each of the factors. Note any overall difference in response between the two levels of a factor. Equally, note if there seems to be little difference between factor levels. This suggests the relevant factors and their better levels.
4. **Find important effects.** Various plots can be used to identify the most important effects that should be included in a model of the data. A particularly useful one is the Half-Normal Percentage probability plot[2]. If none are important, then the experiment does not need to continue.
5. **Rank Important Effects.** A Pareto chart of t-value of effect against each possible effect permits ranking the important effects.
6. **ANOVA test.** Perform an ANOVA on the model containing these most important effects and any effects needed to maintain model hierarchy[3].
7. **Diagnosis.** The usual diagnosis tools are used to verify that the model is correct and that its assumptions have not been violated.
   - **Normality.** A Normal Plot of Studentised Residuals should be approximately a straight line. Deviations from this may indicate that a transformation of the response is appropriate.
   - **Constant Variance.** A plot of Studentised Residuals against predicted response values should be a random scatter. Patterns such as a 'megaphone' may indicate the need for a transformation of the response.
   - **Time-dependent effects.** A plot of Studentised Residuals against run order should be a random scatter. Any trend indicates the influence of some time-dependent nuisance factor that was not countered with randomisation.
   - **Model Fit.** A plot of predicted values against actual response values will identify particular treatment combinations that are not well predicted by the model. Points should align along the 45^0 axis.
   - **Leverage and Influence.** Leverage measures the influence of an individual design point on the overall model. A plot of leverage for each treatment indicates any problem data points.
     A plot of Cook's distance against treatment measures how much the regression changes if a given case is removed from the model.

   If the model passes the diagnoses then it is correct and we can proceed to draw conclusions from it.
8. **Model power.** Check the observed power of the ANOVA performed on the actual data. If the power is less than 0.80, add replicates to the design, gather more data and begin the analysis stage again.
9. **Model significance.** If the ANOVA has sufficient power, check that the overall model is significant. If not, add more terms to the model and return to the ANOVA step.
10. **Model Reduction.** Check the significance of each of the terms in the ANOVA. Those terms with a p-value less than the study's threshold alpha level are deemed significant. Those terms with a p-value greater than this threshold are insignificant and can be removed from the model. If insignificant terms are detected, remove them from the model and perform the ANOVA again on the reduced model.
11. **Model Fit.** Check that the predicted R-Squared value is in reasonable agreement with the Adjusted R-Squared value and that both of these are close to 1. Check that the model has a signal to noise ratio greater than about 4.

[2] See `http://www.itl.nist.gov/div898/handbook/pri/section5/pri598.htm`.

[3] Model hierarchy refers to the principle that a model with a higher-order effect (e.g. AB) should also contain those lower order effects (e.g. A and B).

12. **Examine Aliasing.** The model will have revealed that some main and interaction effects are significant. Recall that the resolution IV design will have some effects aliased with one another. If a significant effect is aliased, we can first attempt to use engineering judgement to justify ignoring the alias. For example, an interaction of ant candidate list length with ant location method might safely be assumed negligible. Failing this, we must augment the design to de-alias the significant effects.

4.2 Augment the design to de-alias effects

A **foldover** procedure is a methodical and efficient way to introduce more treatments into a fractional design so that a particular effect can be de-aliased. The foldover procedure produces double the number of new treatments for which data must be gathered (once for each replicate).

1. A foldover is performed on selected model terms. The fewer terms selected, the better the de-aliasing that will be created by the new treatments. The augmented design should foldover on those most significant model terms that we wish to de-alias
2. One of each replicate from the new treatments is assigned to one of each of the existing blocks.

The screening analysis procedure can now repeat. Further augmentation may be required.

4.3 Check for curvature

At this stage, the relative importance of each term in the model has been assessed. A linear relationship between the factors and the response has been assumed. Adding centre points to a design allows us to determine whether the response surface is not planar but actually contains some type of curvature. The average response value from the actual centre points is compared to the estimated value of the centre point that comes from averaging all the factorial points. If there is curvature of the response surface in the region of the design, the actual centre point value will be either higher or lower than predicted by the factorial design points. Generally, five centre points should be sufficient to assess the existence of curvature. If the analysis with centre points reveals the possibility of curvature in the actual data then further experiments are required.

4.4 Response Surface Modelling

Response surface modelling is useful for finding the relationship between several factors and a response and then optimising this response.

Experiment Design

1. **Choose design space.** If a screening experiment has been performed to choose the relevant factors then there already exists a planar model of the response. The design space should be refined to cover the most interesting part of the design space from the screening experiment. The important factors from the screening experiment are accordingly given new levels.
2. **Define Central Composite Design (CCD).** A central composite design with face-centred axial points and 5 centre runs is an appropriate and efficient design for response surface modelling. If we have decided to use blocks then these centre points must be repeated for each block.
3. **Run Order.** Define a random run order for the treatments and replicates.

Analysis

1. **Blocks.** Examine scatter plots of each response against the blocks. Check that the responses are not highly correlated with the blocks.
2. **Examine and choose appropriate model.** The highest order model that can be generated from the CCD is quadratic. All lower order models (linear, 2-factor interaction and quadratic) are generated and then assessed on three aspects: their significance, their lack of fit and their R-squared values.
   Begin with the linear model. If the model is not significant, it is removed from consideration. Next, tests of lack of fit are performed on each model. Any model that yields a significant result should be rejected from consideration. Finally, the Adjusted R-squared and predicted R-squared are examined. These should be within 0.2 of one another and as close to 1 as possible.
3. **Calculate model coefficients.** A linear regression is performed on the chosen model to estimate its coefficients. The usual tests (see above) of the linear regression model are performed. If the model passes these tests then its proposed coefficients can be accepted. These coefficients can be expressed in coded terms or the original natural units.
4. **Validation.** The ability of the model to predict new responses within the design space can be assessed by choosing new combinations of factors that were not treatments in the original design. The difference between the model's predicted values and the actual values from the validation are compared. The model is interpolative and it is not advisable to use it to extrapolate outside the design space.
5. **Response surface.** The regression equation can now be used to produce a response surface in terms of any two factors while the other factors are held constant.
6. **Perturbation plot.** A perturbation plot shows how the response perturbs as a chosen factor is varied, while all other factors are held at a constant value. It is usually convenient to set this constant value at the centre of the design space. Factors that result in smaller perturbations can be held constant in a contour or surface plot while other more interesting factors are actually plotted.

If other responses were measured and were not found to be highly correlated, these responses should now be analysed in the same way.

5 Related Work

Adenso-Díaz and Laguna [11] give a brief list of OR papers that have used statistical experiment design over the past 30 years. Some discussions are quite general and offer guidelines on the subject [12], [13], [14]. Experimental design techniques have been used to compare solution methods [15] and to find effective parameter values [16], [17]. Adenso-Díaz and Laguna [11] also report a custom methodology and tool, CALIBRA, for parameter tuning. However, this is limited to tuning a maximum of 5 parameters and cannot detect parameter interactions. Other related methodologies for evolutionary algorithms have also been proposed [18].

It seems there is a growing awareness of the need for experimental design in the broad field of evolutionary computation also, as reflected by some publications appearing in the field's main peer-reviewed fora [19], [20], [21].

Acknowledgements

The authors gratefully acknowledge the reviewers' helpful comments.

References

1. Lin, B.W., Rardin, R.L.: Controlled experimental design for statistical comparison of integer programming algorithms. Management Science **25**(12) (1979) 1258–1271

2. Hooker, J.N.: Testing heuristics: We have it all wrong. Journal of Heuristics **1** (1996) 33–42
3. McGeoch, C.C.: Toward an experimental method for algorithm simulation. INFORMS Journal on Computing **8**(1) (1996) 1–15
4. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence. Oxford University Press (1999)
5. Dorigo, M., Gambardella, L.M.: Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation **1**(1) (1997) 53–66
6. Dorigo, M., Stützle, T.: Ant Colony Optimization. The MIT Press (2004)
7. Johnson, D.S.: A theoretician's guide to the experimental analysis of algorithms. In Goldwasser, Johnson, McGeoch, eds.: Proceedings of the Fifth and Sixth DIMACS Implementation Challenges. American Mathematical Society (2002) 215–250
8. Montgomery, D.C.: Design and Analysis of Experiments. 6 edn. John Wiley and Sons Inc (2005)
9. Coffin, M., Saltzman, M.J.: Statistical analysis of computational tests of algorithms and heuristics. INFORMS Journal on Computing **12**(1) (2000) 24–44
10. Crowder, H.P., Dembo, R.S., Mulvey, J.M.: On reporting computational experiments with mathematical software. ACM Transactions on Mathematical Software **5**(2) (1979) 193–203
11. Adenso-Dıaz, B., Laguna, M.: Fine-tuning of algorithms using fractional experimental designs and local search. Operations Research **54**(1) (2006) 99–114
12. Barr, R.S., Golden, B.L., Kelly, J.P., Resende, M.G.C., Stewart, W.R.: Designing and reporting on computational experiments with heuristic methods. Journal of Heuristics **1** (1995) 9–32
13. Crowder, H.P., Dembo, R.S., Mulvey, J.M.: Reporting computational experiments in mathematical programming. Mathematical Programming **15** (1978) 316–329
14. Greenberg, H.: Computational testing: Why, how and how much? ORSA Journal on Computing **2**(1) (1990) 94–97
15. Amini, M.M., Racer, M.: A rigorous computational comparison of alternative solution methods for the generalized assignment problem. Management Science **40**(7) (1994) 868–890
16. Coy, S., Golden, B., Runger, G., Wasil, E.: Using experimental design to find effective parameter settings for heuristics. Journal of Heuristics **7**(1) (2001) 77–97
17. Xu, J., Chiu, S., Glover, F.: Fine-tuning a tabu search algorithm with statistical tests. International Transactions on Operations Research **5**(3) (1998) 233–244
18. Bartz-Beielstein, T.: Tuning evolutionary algorithms - overview and comprehensive introduction. Technical Report CI-148/03, Universität Dortmund (2003)
19. Hancock, E.R., Myers, R.: Empirical modelling of genetic algorithms. Evolutionary Computation **9**(4) (2001) 461–493
20. François, O., Lavergne, C.: Design of evolutionary algorithms-a statistical perspective. IEEE Transactions on Evolutionary Computation **5**(2) (2001) 129–148
21. Czarn, A., MacNish, C., Vijayan, K., Turlach, B., Gupta, R.: Statistical exploratory analysis of genetic algorithms. IEEE Transactions on Evolutionary Computation **8**(4) (2004) 405–421