Analyzing Heuristic Performance with Response Surface Models: Prediction, Optimization and Robustness

Enda Ridge and Daniel Kudenko
Department of Computer Science, The University of York
Yo10 5DD England
ERidge@cs.york.ac.uk, Kudenko@cs.york.ac.uk

ABSTRACT

This research uses a Design of Experiments (DOE) approach to build a predictive model of the performance of a combinatorial optimization heuristic over a range of heuristic tuning parameter settings and problem instance characteristics. The heuristic is Ant Colony System (ACS) for the Travelling Salesperson Problem. 10 heurstic tuning parameters and 2 problem characteristics are considered. Response Surface Models (RSM) of the solution quality and solution time predicted ACS performance on both new instances from a publicly available problem generator and new real-world instances from the TSPLIB benchmark library. A numerical optimisation of the RSMs is used to find the tuning parameter settings that yield optimal performance in terms of solution quality and solution time. This paper is the first use of desirability functions, a well-established technique in DOE, to simultaneously optimise these conflicting goals. Finally, overlay plots are used to examine the robustness of the performance of the optimised heuristic across a range of problem instance characteristics. These plots give predictions on the range of problem instances for which a given solution quality can be expected within a given solution time.

Categories and Subject Descriptors

C.4 [**PERFORMANCE OF SYSTEMS**]: Design studies

General Terms

Performance

Keywords

Design of Experiments, Minimum Run Resolution V Design, Response Surface Model, Overlay Plots, Ant Colony Optimization

1. INTRODUCTION AND MOTIVATION

Heuristics are a class of approximate algorithms that are often applied to problems of search and optimization. Many heuristics are stochastic in nature. This means that repeated runs of a heuristic produce different behaviors. Their approximate nature means that while they should produce good solutions relatively quickly, they will often continue to produce improved solutions over extended run times.

These characteristics of randomness and approximation pose a significant challenge to the engineer and researcher. He/she must have some prediction of heuristic performance for the heuristic to be of any practical use. The difficulty of this challenge is exacerbated by the conflicting nature of the two most important heuristic performance measures, (1) solution quality and (2) the time to produce a solution of a given quality (solution time). Longer solution times yield better solution quality but at some point there is a trade off between the decreasing likelihood of further solution improvements and the increasing opportunity cost of expended computational resources . Furthermore, many heuristics have large numbers of tuning parameters. It is an open issue how these parameters affect performance and how parameter settings and performance relate to classes of problem instances.

Swarm intelligence heuristics are particularly difficult to model analytically because of their randomness and their large number of interacting components. An alternative to an analytical approach is to use an empirical analysis. Design of Experiments (DOE) [12] is a well-established field that involves experiment designs and analysis tools for the empirical modeling of processes. DOE empowers the engineer with the ability to make the best use of experimental resources, guidelines on good experimental procedure and a mathematical precision to the conclusions that can be drawn from collected data.

The heuristics engineer faces a situation where limited computational resources, sometimes of a varying specification, must be used to make performance predictions of a stochastic heuristic with many tuning parameters and to place boundaries on those performance predictions. There is a growing awareness of the need for methodical empirical approaches to address these challenges [9]. This awareness is not yet prevalent within Ant Colony Optimization.

This paper introduces DOE methodology and design analysis techniques to the Ant Colony Optimization (ACO) [8] community, illustrating their application with the Ant Colony System (ACS) heuristic. We build a predictive model of ACS performance where performance is measured in terms of two conflicting responses, solution quality and time to solution. Numerical optimization techniques are applied to this model to determine the ACS parameters that achieve optimal performance. For these optimized parameter settings, robustness of performance is analyzed across a range of problem instances. All reported research uses a publicly available algorithm implementation[1] and problem generator[2].

The paper begins with a background on the problem domain, the heuristic and experiment design. Sections 3 and 4 describe the experimental methodology and model fitting procedure, followed in Section 5 by a confirmation of the model's accuracy. Section 6 discusses the model's predictions and performs a numerical optimization of the model to determine optimum heuristic parameter settings. Section 7 performs a robustness analysis of these optimum settings. The paper concludes with a discussion of related work, the paper's conclusions and directions for future work.

2. BACKGROUND AND TERMINOLOGY

We give an overview of the Traveling Salesperson Problem (TSP) problem domain and the Ant Colony System (ACS) algorithm to better illustrate the DOE decisions discussed later. The reader is directed to other works for a comprehensive discussion of ant systems [8] and of the TSP problem [11].

2.1 The TSP Problem Domain

The TSP involves finding the shortest route between a set of cities such that no city is visited more than once. The TSP is often represented as a graph structure in which the nodes represent the cities to be visited and the lengths of links represent the costs associated with traveling from one node to another. The TSP is a very popular abstraction for many types of *discrete combinatorial optimization* problems.

2.2 ACS Algorithm

The Ant Colony System (ACS) algorithm has 3 Stages. **Stage 1** involves initializing pheromone on all edges in the problem description. Pheromone is an abstraction of the chemical markers used by real ants. **Stage 2** involves all ants building their tours using a decision rule. The decision rule is influenced by the distance to the next city being considered and the pheromone level on the edge connecting to this city. Nearer cities with a high pheromone level are more likely to be chosen by an ant. The type of decision at any given move is governed by an exploration/exploitation threshold q_0. Consider an ant at a city i choosing the next city j, with a set of N_l cities that remain to be visited. A uniform-random exploration probability q is generated and the next city j is chosen as follows.

$$j = \begin{cases} \max_{l \in N_l} \left\{ [\tau_{il}]^{\alpha} [\eta_{il}]^{\beta} \right\}, \; if\, q \leq q_0 \\ J \end{cases}$$

where J is a next city j chosen with a probability p_{ij} according to a random proportional rule:

$$p_{ij} = \frac{[\tau_{ij}]^{\alpha} [\eta_{ij}]^{\beta}}{\sum\limits_{l \in N_l} [\tau_{ij}]^{\alpha} [\eta_{ij}]^{\beta}} \qquad j \in N_l$$

[1]http://iridia.ulb.ac.be/~mdorigo/ACO/aco-code/public-software.html
[2]www.research.att.com/~dsj/chtsp/download.html

The case of $q \leq q_0$ therefore represents *exploitation* while that of $q > q_0$ represents *exploration*. ACS is often augmented with *candidate lists* to improve performance. A candidate list for a given node is a sorted list of the nearest neighbors to that node. ACS first restricts its decisions to elements of its candidate list. If all elements of the candidate list have been visited, the decision process is applied to the full set of unvisited cities N_l as usual.

The *local pheromone update* involves decaying the pheromone level on an edge traversed by an ant by a small amount. Specifically, for an edge ij with pheromone τ_{ij}, the locally decayed pheromone value is

$$\tau_{ij} = (1 - \rho_{local})\, \tau_{ij} + \rho_{local} \tau_0$$

where ρ_{local} is a local pheromone decay parameter and τ_0 is the pheromone level on the edge at the start of the algorithm. This discourages subsequent ants from exploiting the exact same route and so promotes the exploration of a diverse range of solutions. In **Stage 3**, pheromone is deposited along the tour of one ant only in a *global pheromone update*. Specifically, for an edge ij with pheromone τ_{ij}, the globally updated pheromone value is:

$$\tau_{ij} = (1 - \rho_{global})\, \tau_{ij} + \frac{\rho_{global}}{\text{Tour cost}}$$

Stages 2 to 3 then repeat with all ants building new tours while performing local pheromone decays.

Note that ACO algorithms are often augmented with a *local search procedure* such as 3-opt local search. We omit such augmentations here. Performance increases due to the local search would confound the performance effects of variations in the ACO parameters. Furthermore, modeling all possible alternative local search modules would be prohibitive. There are several open questions regarding this common ACS implementation.

1. **Ant Placement.** Should ants be randomly scattered across the graph or should they be all placed at a single randomly chosen city? Presumably the former would be more computationally expensive. Is this expense commensurate with any gains in performance?

2. **Solution Construction.** We identify two possibilities,

   (a) Sequential. An ant completes its whole tour and associated local pheromone decays before processing moves to the next ant.

   (b) Parallel. An ant completes a single step of its tour and associated local pheromone decay before processing moves to the next ant.

   The intuitive difference is that in parallel tour building, the ant bases its decision on more up-to-date pheromone levels from the local pheromone decays of the other ants. Whether this difference has a positive, negative or negligible effect on performance is an open question [8, p. 78]

3. **Global Pheromone Deposit Ant.** Two possibilities are mentioned in the literature [8, p. 77].

   (a) Best Ant So Far. This approach uses the tour from the best ant so far during the entire algorithm.

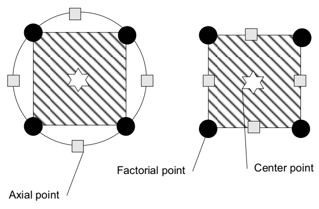

Figure 1: Central Composite Designs for two factors. On the left is a Circumscribed Central Composite (CCC) and on the right a Face-Centred Composite (FCC).

(b) Best Ant This Iteration. This approach uses the tour from the best ant of the current iteration.

2.3 Designs for Response Surface Models

The overall goal of this research is to investigate how changes in algorithm tuning parameters and problem characteristics affect algorithm performance. The tuning parameters and characteristics we vary are called *factors* and the performance measures we gather are called *responses*. When investigating the effect of factors on some response(s), the general approach is to vary those factors over some number of *levels*. The experimental region covered by our chosen ranges of factors is called the *design space*. We measure the responses and interpolate these measurements into a *response surface* over the design space. This is the *Response Surface Methodology* (RSM) [13]. When individual factors have an effect on the responses, we say there is a *main effect*. We may also encounter *higher-order interactions* between factors. That is, the effect of factor A can only be understood when the level of an interacting factor B is known and fixed. There is a large body of well established work on RSM. This overview is restricted to those aspects of most interest to the heuristic researcher and is necessarily brief.

One of the most popular families of designs for building response surfaces are known as the *Central Composite Designs* (CCD). A CCD contains an embedded factorial design[3] augmented with design points at the centre of the design space (*centre points*) and so-called *axial points* located at some distance α from the design centre. The two most popular CCD designs are illustrated in Figure 1.

The *Circumscribed Central Composite* (CCC) sets α at such a value that the axial points create a circle that circumscribes the square defined by the embedded factorial. The *Face-Centered Composite* (FCC) sets α such that the axial points lie on the faces of the square defined by the embedded factorial. For statistical reasons, the rotatability of the CCC design makes it preferable for analysis. However, when parameter ranges have a practical limit, the FCC design may be the only alternative.

The use of a full factorial in the embedded part of a CCD is expensive since a crossing of say 12 factors, each at only 2 levels, would require $2^{12} = 4096$ design points for the embedded factorial alone. The advantage of a full factorial is that it permits estimating all interactions between factors. *Fractional factorial designs* provide a manageable alternative to full factorials. Under the assumption that higher order interactions are insignificant and can be safely ignored, it is possible to judiciously choose a subset of the full factorial's design points such that lower order interactions can still be estimated. The price we pay is that higher order interactions become *aliased* with one another. When two terms A and B are aliased, it is impossible to tell whether a significant effect is due to A or to B.

Recently, a state-of-the-art design called a *Minimum Run Resolution V* design has been introduced [14]. This provides a further saving in experiment runs while still allowing all main and second-order interactions to be estimated.

[3] A full factorial design crosses all levels of all factors with one another.

2.4 Prediction Intervals

It is the unfortunate reality that the vast majority of meta-heuristics are stochastic in nature. The implication for the experimenter is that two repeated runs of a meta-heuristic will produce different behaviour and consequently different results. Furthermore, responses such as CPU times are susceptible to noise and therefore will vary from run to run. How can we evaluate a response surface built from the averages of algorithm runs given that any individual run will likely vary somewhat from the response surface model's predictions? *Prediction Intervals* define with mathematical precision, the range around the response surface inside which a percentage of runs will fall.

3. METHODOLOGY

Some difficult design issues arise that are particular to experiments with heuristics.

3.1 Stopping criterion

It is common in the ACO field, and in heuristics in general, to halt experiments after some combinatorial count. This count is typically some algorithm operation or an iteration. With regard to reproducibility, this is certainly preferable to the use of time as a stopping criterion. However, a *hard* iteration stopping criterion risks biasing results. A fixed number of iterations will not solve a more difficult problem as well as an easier problem. This is of particular importance as we vary problem characteristics that are hypothesized to affect performance. This research takes a practical view that once an algorithm is no longer producing improved results regularly, it is better to halt its execution and better employ the computational resources. This leads to using *stagnation* as a stopping criterion where stagnation is a number of iterations in which no solution improvement is observed. This offers the reproducibility of a combinatorial count while the *softer* stagnation mitigates the likelihood of bias. This research uses a stagnation stopping criterion of 250 iterations without improvement.

3.2 Responses

Performance measures must reflect the conflicting heuristic goals of high solution quality and low time to solution. The response that reflects time-to-solution is the CPU time from the beginning of an algorithm run to the time that the algorithm has stagnated and halted. We were careful not to include time to write output and the time to calculate output data that is not essential to the algorithm's functioning.

There are several choices of response to reflect solution quality and it is a matter of debate within the community as to which are the more appropriate. This research uses *relative error*. Relative error of a solution is the difference between the algorithm solution and the optimal solution expressed as a percentage of the optimal solution. The optimal solution value was calculated using the Concorde solver [2].

3.3 Problem Instances

This research uses problem size as a factor. It also uses problem edge cost standard deviation as this has been shown to affect problem difficulty for ACO algorithms [20]. We need a method to produce instances with controllable levels of these characteristics. Unfortunately, publicly available benchmark libraries [19] do not have the breadth to provide such instances. For reproducibility, we used our own Java implementation of the 8th DIMACS Implementation Challenge [10] problem generator. This was informally verified to produce the same instances as the original C code. Our generator was then modified to draw its edge costs from a log-normal distribution where the standard deviation of the resulting edge costs could be controlled while their mean was fixed at 100. The use of a log-normal distribution with these properties was inspired by previous investigations on problem difficulty for an exact TSP algorithm [5]. Histograms of the normalized cost matrix of real instances in the TSPLIB [19] do indeed demonstrate a log-normal shape.

3.4 Experiment Design

The reported research used a Minimum Run Resolution V Circumscribed Central Composite design with 3 replicates of the factorial and axial points and 3 centre points. This requires a total of 1560 runs. A 5% statistical significance level was used. At this level, a quadratic polynomial surface of both responses could detect effects with a size of 0.25 standard deviations with a power greater than 74%.

Table 1 lists the 12 experiment factors and their high and low factorial levels. Note that the design must be repeated for each level of the categoric factors. We emphasize that inclusion of a parameter as a factor does *not* imply the parameter has a significant effect on performance. This would require a screening analysis. However, once an RSM has been built, the sensitivity of the response to each factor can be determined from the RSM's equation.

For reasons of practicality, α was set to 1.5. The CCC is therefore not perfectly rotatable (see Section 2.3). An α of 1.5 results in the parameter Rho having high and low axial points of 0.05 and 0.95 for example.

- **Alpha and Beta.** These are the exponential terms in the ant's decision rule (Section 2.2) . Exponentiation with a non-integer exponent is an extremely expensive operation. This was confirmed with a profiling of the experimental code. We force Alpha and Beta to be integers, reasoning that any solution quality benefit offered by say, an alpha of 1.95, would be outweighed by the time saving of using alpha of 2.

- **Ants.** This is the number of ants expressed as a percentage of problem size. This is typically expressed as an absolute value in the literature. We use the percentage to improve the extent to which our results can be generalised.

- **NNAnts.** This is the length of the ant's candidate list, expressed as a percentage of problem size. Again, we avoid the more common absolute expression for the same reasons as above.

Table 1: Factors and factor levels at factorial points. N=numeric, C=categoric.

	Factor	**Type**	**Low (-)**	**High (+)**
A	Alpha	N	3.00	11.00
B	Beta	N	3.00	11.00
C	Ants	N	25.00	100.00
D	NNAnts	N	10.00	25.00
E	q0	N	0.20	0.80
F	Rho	N	0.20	0.80
G	RhoLocal	N	0.20	0.80
H	Problem Size	N	335	475
J	Problem StDev	N	20.00	60.00
K	Solution Construc.	C	parallel	sequential
L	Placement	C	random	same
M	Pheromone Update	C	Best So Far	Best Of Iteration

- q_0. This is the exploration/exploitation threshold. Its value must be between 0 and 1.

- **Rho and RhoLocal.** The pheromone related terms are also limited to be between 0 and 1. We make the distinction between the value used for local pheromone decay and global pheromone deposit as others have done [4]. It is an open question whether this distinction should be made, with the majority of research setting both values equal to one another.

- **Problem size and standard deviation.** These domain characteristics were discussed in Section 3.3.

- **Solution construction.** As observed in Section 2.2, ants can produce their solutions *sequentially* or in *parallel*.

- **Ant Placement.** The *random* setting scatters all ants across the graph. The *same* setting places them all at the same randomly chosen start city.

- **Pheromone Update.** As discussed in Section 2.2, there are at least two choices for the ant that performs the *global pheromone update* procedure (Figure Figure 1).

Factor levels were chosen so that they (1) covered the range of parameters with a limited range (e.g. rho) or (2) covered the ranges typically seen in the literature (e.g. alpha). The design has a sufficient resolution that all main effects and second order effects are not aliased. The crossings of factors H and J required 9 problem instances (including the centre points). The same instance was used for every occurrence of a given size and standard deviation combination. Since experiments were conducted on 5 similar machines, it was necessary to randomize the run order. This

Table 2: Fit summary for CPU Time.

Source	Sum of Squares	df	Mean Square	F Value
Linear	320.86	12	26.74	422.85
2FI vs Linear	13.62	66	0.21	3.63
Quadratic vs 2FI	**19.19**	**9**	**2.13**	**48.46**

Table 3: Fit summary for Relative Error.

Source	Sum of Squares	df	Mean Square	F Value
Linear	80.28	12	6.69	807.98
2FI vs Linear	7.05	66	0.11	27.77
Quadratic vs 2FI	2.13	9	0.24	98.11

is a standard DOE technique for dealing with unknown and uncontrollable nuisance factors [15].

4. MODEL FITTING

The data for Time and Relative Error were checked with the usual diagnostic tools[4]. A Box-Cox plot recommended a $\log_{10}$ transformation of both responses. 13 outliers (less than 1% of the data) were removed. The transformed data then passed all diagnostics satisfactorily.

A *fit analysis* was conducted for each response to determine the highest order statistically significant unaliased model that could be fit to the responses. A fit analysis begins by fitting the lowest order model, a linear model, to the response. The next higher order model is then fit to the same response. If no additional terms from the higher order model are statistically significant, it is not necessary to use the higher order model. This procedure continues until the highest required order is found. The Minimum Run Resolution V design is aliased for cubic models (see Section 2.3). This leaves a linear model, a 2 factor interaction model and a quadratic model to be considered. Summaries of the fit analyses for both responses are found in Table 2 and Table 3.

Based on these results, two **quadratic models** were thus built for each response and for each combination of the categoric factors (Table 1). Insignificant terms were removed from the models. Note, this is distinct from screening where insignificant *factors* and all their associated terms are removed from the original experiment design. Unfortunately, space requirements prevent reproducing these models here. Before drawing conclusions from the resulting response surface models for each response, we must confirm the accuracy of the models.

[4]Normal plot of Internally Studentized Residuals; Residuals versus Predicted; Residuals versus Run; Predicted Versus actual; Externally Studentized Residuals versus Actual; and Leverage, DFFITS, DFBETAs and Cooks Distance versus Run. See the literature [13] for details.

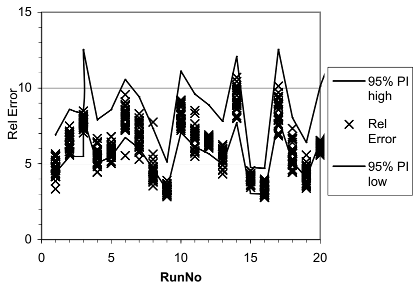

Figure 2: Prediction intervals and actual values of relative error from 25 replicates of 20 randomly generated runs.

5. CONFIRMATION

How do we confirm the accuracy of a response surface model without reproducing the entire response surface model again? A common approach in DOE [12] is to randomly sample points within the design space, run the actual process (in this case, the algorithm) at those randomly generated points, and compare the model's predictions to the measurements from the randomly generated algorithm runs. Since response surface models are interpolative, we should only expect them to perform well within the design space on which they were built. For example, the models presented here use problem sizes between 300 and 500 and so should not be expected to perform well on instances of size 1000. However, it is always worthwhile to assess performance outside the design space and we do so here. Recall that the CCC's design space lies circumscribed within its axial points. We should expect predictive performance to be better within the factorial ranges of Table 1 than between the factorial and axial ranges. Failing a completely accurate prediction of performance, we identify two criteria upon which our satisfaction with the model (and thus confidence in its predictions) can be judged.

- **Conservative:** we should prefer models that provide consistently higher predictions of relative error and higher solution time than those actually observed.
- **Matching Trend:** we should prefer models that match the trends in algorithm performance. The prediction of the parameter combinations that give the best and worst performance should match the actual parameter combinations that give the observed best and worst performance.

We randomly generate new experiment runs on combinations of factors within the design space. For the randomly chosen combinations of problem characteristics, completely new instances were generated. Each experiment run was replicated 20 times. The measured responses were then compared to the models 95% prediction interval (Section 2.4). Figure 2 and Figure 3 illustrate the results for the Relative Error and Time responses on new problem instances.

We see that both RSM models follow the trends of the actual responses. The RSM of relative error is conservative. The RSM of solution time made poor predictions of a small number of runs as evidenced by the outliers that fell above the upper limit of the 95% prediction interval. These times

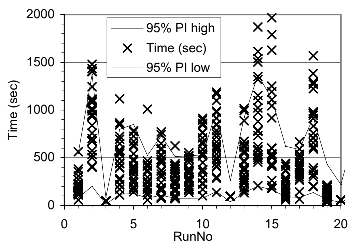

Figure 3: Prediction intervals and actual values of solution time from 25 replicates of 20 randomly generated runs.

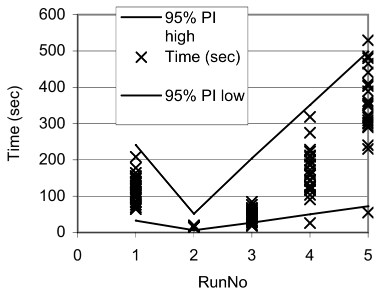

Figure 4: Prediction intervals and actual values of solution time from 25 replicates of 5 TSPLIB instances.

may be due to variation in the hardware of the 5 experimental machines used and the fact that these machines were not in a strictly controlled computational environment.

We applied this procedure to several instances from TSPLIB [19] that were close to the model design space in terms of problem characteristics. Figure 4 shows the prediction intervals and actual data on solution time. The model is an excellent predictor of solution times of TSPLIB instances, despite those instances being outside the design space. However, the predictions for relative error were not good, although they were consistently conservative. This may be because the instances are outside the model design space or because these instances have some characteristic that is not present in the instances produced by the DIMACS [10] problem generator.

6. PREDICTIONS AND OPTIMISATION

Satisfied with the model, we can begin to explore its relationships and move to the region of the design space that optimizes the responses. We express the multiple responses in terms of a single desirability function[5] [7]. For each response $Y_i(x)$, its desirability function $d_i(Y_i)$ maps values between 0 to 1 to the possible values of $Y_i(x)$. $d_i(Y_i) = 0$ is completely undesirable and $d_i(Y_i) = 1$ is an ideal response. The *overall desirability* for all k responses is the geometric mean of the individual desirabilities:

$$D = \left(\prod_{1}^{k} d_i \right)^{\frac{1}{k}}$$

[5]www.itl.nist.gov/div898/handbook/pri/section5/pri5322.htm

If the goal is to minimize a responses (as in this research), the desirability functions take the following form where L_i, U_i and T_i are the lower bound, upper bound and target values of the desired response.

$$d_i\left(\hat{Y}_i\right) = \begin{cases} 1.0 & \text{if } \hat{Y}_i(x) < T \\ \left(\frac{\hat{Y}_i(x) - U_i}{T_i - U_i}\right) & \text{if } \mathrm{T}_i \le \hat{Y}_i(x) \le U_i \\ 0 \quad \text{if } \hat{Y}_i(x) > U_i \end{cases}$$ Note that the fitted response value $\hat{Y}_i$ is used in place of Y_i.

A well-established multi-objective numerical optimization,, the Nelder-Mead downhill simplex [18, p. 326], is then performed on the response surface model's equations such that the desirability of all responses is maximized. We specify the optimization here with the dual goals of minimizing both quality and solution time, while allowing all algorithm-related factors to vary within their design ranges. The two problem characteristics are also factors in the model since we want to establish the relationship between these problem characteristics, the algorithm parameters and the response. It does not make sense to include these factors in the optimization. We therefore perform the numerical optimizations with the two problem characteristics fixed at three-level factorial combinations. The results of these 9 optimizations are presented in Table 4.

Recall that we are forcing Alpha and Beta to only take on integer values because of the expensive cost of non-integer exponentiation. Recall that two problem characteristics were included as factors in our model. In general, the user of the algorithm is presented with a problem of given characteristics and wishes to determine the algorithm parameter settings that will optimize the desirability of all responses. The above numerical optimization above could of course be rerun with new constraints on problem size and problem cost matrix standard deviation that match the characteristics of the problem to hand. However, such optimization tools may not be available to the user. We should therefore like to determine how robust the recommended parameters settings are to variations in the problem size and problem cost matrix standard deviation.

7. ROBUSTNESS

Overlay plots effectively lay the contour plots of each response on top of one another so that a common region of operability can be identified. Operability in this case is defined by some boundary constraints on problem characteristics that can be solved within given ranges of the quality and time responses. Figure 5 illustrates an overlay plot of the responses that are within 1% and 3% relative error and under 35 seconds solution time. The white area represents combinations of problem size and problem standard deviation that can be solved subject to these constraints using the optimal parameter settings from the desirability optimization. Expectedly, the region is concentrated around smaller problem sizes and smaller problem standard deviations. Clearly, more relaxed constraints will allow a wider range of problems to be solved.

The significance of the DOE techniques presented is as follows. We have been able to efficiently model the performance of a stochastic heuristic in terms of two performance measures with a Central Composite Design. Prediction Intervals have allowed us to judge the accuracy of the model. Desirability functions have allowed us to simultaneously optimize performance in terms of both responses.

Table 4: Results of optimisations on 9 combinations of problem characteristics. Each result is the best of 25 numerical optimisations. Last row gives the overall parameter setting recommendations. -, 0 and + are low, medium and high levels respectively.

Size	StDev	alpha	beta	ants	NNAnts	Q0	rho	rhoLocal	Solution Construction	Ant Placement	Pheromone Update
-	-	3.00	10.19	25.01	10.00	0.80	0.20	0.54	parallel	random	bestOfIteration
-	0	4.82	11.00	25.01	10.00	0.80	0.60	0.58	sequential	random	bestOfIteration
-	+	3.16	10.34	25.00	10.00	0.80	0.79	0.59	sequential	random	bestOfIteration
0	-	6.45	10.92	25.00	10.00	0.80	0.20	0.53	parallel	random	bestSoFar
0	0	3.58	11.00	25.00	10.08	0.79	0.20	0.57	sequential	random	bestOfIteration
0	+	3.05	11.00	25.01	10.11	0.74	0.20	0.56	sequential	random	bestOfIteration
+	-	3.70	11.00	25.00	10.03	0.80	0.20	0.64	parallel	random	bestOfIteration
+	0	5.91	11.00	25.00	10.01	0.80	0.21	0.60	parallel	random	bestOfIteration
+	+	8.48	11.00	25.00	10.00	0.21	0.20	0.52	parallel	Same	bestOfIteration
		3	**11**	**25**	**10**	**.8**	**.2**	**.55**	**Parallel**	**Random**	**bestOfIteration**

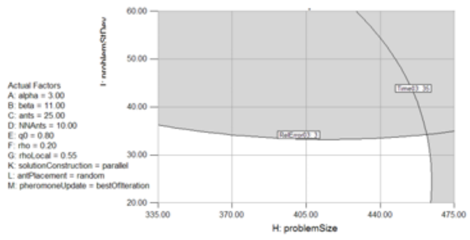

Figure 5: Overlay plot of responses within 1% and 3% relative error and under 35 seconds solution time for different combinations of problem size and problem standard deviation. All other parameters are set to the values recommend by the numerical optimisation of desirability.

Overlay plots have allowed us to generalize the robustness of the optimal parameter settings across variations in problem characteristics.

8. RELATED WORK

Factorial designs have been combined with a local search procedure to systematically find the best parameter values for a heuristic [1]. CALIBRA begins by finding a set of 'optimal' parameter values using the *Taguchi methodology* in a 2 level factorial design. This analysis is used only as a guideline to focus the search through the parameter space. An iterative local search then improves on the parameter values within a refined region of the design space. Unfortunately CALIBRA can only tune five algorithm parameters. The restrictive linear assumption precludes examining interactions between parameters. ACO algorithms require more than 5 parameters and our fitting analysis shows that interactions in higher order models are indeed important.

Coy *et al* [6] systematically find good settings for 6 tuning parameters on a set of Vehicle Routing Problems (VRP). Presented with a set of problems to solve, the procedure finds high quality parameter settings for a small number of problems in the problem set (the *analysis set*) and then combines these settings to achieve a good set of parameters for the complete problem set. A *fractional factorial design* is used to produce a response surface. The optimal parameter settings of the RSMs from the analysis set are averaged to obtain the final parameter settings for all problems in the set. The approach does not use higher-order models (such as quadratic) since different response surfaces are averaged over all analysis set instances. Coy *et al* acknowledge that their method will perform poorly if the representative test problems are not chosen correctly or if the problem class is so broad that it requires very different parameter settings. We see our work as a more general approach than Coy *et al* since problem characteristics are included in our RSM model.

Park and Kim [16] used a non-linear response surface method to find parameter settings for a simulated annealing algorithm. Parsons and Johnson [17] used a central composite design with embedded fractional factorial to build a response surface and improve the performance of a genetic algorithm on a test data set. Only two parameters were modeled.

Birattari [3] uses algorithms derived from a machine learning technique known as *racing* to incrementally tune the parameters of several metaheuristics including Max-Min Ant System for the TSP [22]. Tuning was achieved with a fixed time constraint where the goal is to find the best configuration of an algoirthm within this time. While the dual problem of finding a given threshold quality in as short a time as possible was acknowledged, the author does not pursue the idea of a bi-objective optimisation of both time and quality. We have shown here how a bi-objective optimisation can be achieved with a stagnation stopping criterion and the use of desirability functions.

9. CONTRIBUTIONS AND FUTURE WORK

The reported research makes several important contributions to the field of Ant Colony Optimization and the broader field of heuristics.

- **Efficient Experiment Designs.** To our knowledge, this is the first application of Minimum Run Resolution V designs to efficiently model a heuristic's performance. This design offers huge savings in experimental runs over full factorials and over fractional factorials. This saving is important for heuristics that typically have 8 or more tuning parameters.

- **Methodical Model Evaluation.** The paper illustrates the use of *prediction intervals* to confirm the predictive accuracy of a response surface model with a given confidence. Building response surface models and using DOE diagnostic tools involves some subjective judgements. This is unavoidable. Prediction intervals and new random experiment runs provide us with feedback on those subjective decisions.

- **Simultaneous Analysis of Conflicting Responses.** This is the first use of *desirability functions* to simultaneously analyse and optimise the conflicting responses of solution quality and solution time. These are critical to all heuristic performance analyses and should not be considered in isolation. While much good research has been done by running algorithms for a fixed time or up to a target quality, these choices necessarily bias one response in favour of the other.

- **Optimal Parameter Recommendations for ACS.** Established DOE numerical optimisation techniques allowed us to find parameter settings that optimise ACS performance for different combinations of problem characteristics. Results suggested answers to several open questions (Section 2.2).

  - It is worthwhile distinguishing between Rho and RhoLocal as different values were recommended.
  - Parallel solution construction seems preferable.
  - Random ant placement seems preferable.
  - Using the best ant from the current iteration is preferable.

  Note however that inclusion of a factor and its optimisation does not mean that the factor actually affects performance. Such a recommendation would require a so-called screening analysis [21].

- **Robustness of Parameter Recommendations.** This is the first use of the DOE technique of overlay plots to provide a visualisation of the robustness of recommended optimal parameter settings across a range of problem characteristics.

An immediate direction for our future work is the investigation of other designs to improve the accuracy and practicality of the response surface model. The Circumscribed Central Composite (CCC) is preferable from a statistical point of view but it is difficult to make its design space cover the whole range of parameter values (See Table 1). The more practical Face-Centered Composite should be investigated and compared to the CCC.

We have conducted methodical screening analyses of ACO parameters to decide on the minimal set of parameters to predict performance [21].

We hope to see the power and versatility of these well-established DOE methodologies and engineering tools becoming more widespread within the ACO community.

10. REFERENCES

[1] B. Adenso-Dıaz and M. Laguna. Fine-Tuning of Algorithms Using Fractional Experimental Designs and Local Search. *Operations Research*, 54(1):99–114, 2006.

[2] D. Applegate, R. Bixby, V. Chvatal, and W. Cook. Implementing the Dantzig-Fulkerson-Johnson algorithm for large traveling salesman problems. *Mathematical Programming Series B*, 97(1-2):91–153, 2003.

[3] M. Birattari. *The Problem of Tuning Metaheuristics.* Phd, Universit Libre de Bruxelles, 2006.

[4] H. M. Botee and E. Bonabeau. Evolving Ant Colony Optimization. *Advances in Complex Systems*, 1:149–159, 1998.

[5] P. Cheeseman, B. Kanefsky, and W. M. Taylor. Where the Really Hard Problems Are. In *Proceedings of the Twelfth International Conference on Artificial Intelligence*, volume 1, pages 331–337. Morgan Kaufmann Publishers, Inc., 1991.

[6] S. Coy, B. Golden, G. Runger, and E. Wasil. Using Experimental Design to Find Effective Parameter Settings for Heuristics. *Journal of Heuristics*, 7(1):77–97, 2001.

[7] G. Derringer and R. Suich. Simultaneous Optimization of Several Response Variables. *Journal of Quality Technology*, 12(4):214–219, 1980.

[8] M. Dorigo and T. Stützle. *Ant Colony Optimization.* The MIT Press, Massachusetts, USA, 2004.

[9] J. N. Hooker. Needed: An Empirical Science of Algorithms. *Operations Research*, 42(2):201–212, 1994.

[10] D. S. Johnson and L. A. McGeoch. Experimental analysis of heuristics for the STSP. In *The Traveling Salesman Problem and Its Variations*, pages 369–443. 2002.

[11] E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys, editors. *The Traveling Salesman Problem - A Guided Tour of Combinatorial Optimization.* John Wiley and Sons.

[12] D. C. Montgomery. *Design and Analysis of Experiments.* John Wiley and Sons Inc, 2005.

[13] R. H. Myers and D. C. Montgomery. *Response Surface Methodology. Process and Product Optimization Using Designed Experiments.* John Wiley and Sons Inc., 1995.

[14] G. Oehlert and P. Whitcomb. Small, Efficient, Equireplicated Resolution V Fractions of 2K designs and their Application to Central Composite Designs. In *Proceedings of 46th Fall Technical Conference.* American Statistical Association, Section on Physical and Engineering Sciences, 2002.

[15] B. Ostle. *Statistics in Research.* Iowa State University Press, 2nd edition, 1963.

[16] M.-W. Park and Y.-D. Kim. A systematic procedure for setting parameters in simulated annealing algorithms. *Computers and Operations Research*, 25(3):207–217, 1998.

[17] R. Parsons and M. Johnson. A Case Study in Experimental Design Applied to Genetic Algorithms with Applications to DNA Sequence Assembly. *American Journal of Mathematical and Management Sciences*, 17(3):369–396, 1997.

[18] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in Pascal: the art of scientific computing.* Cambridge University Press, 1989.

[19] G. Reinelt. TSPLIB - A traveling salesman problem library. *ORSA Journal of Computing*, 3:376–384, 1991.

[20] E. Ridge and D. Kudenko. An Analysis of Problem Difficulty for a Class of Optimisation Heuristics. In *Proceedings of the Seventh European Conference on Evolutionary Computation in Combinatorial Optimisation*, volume 4446 of *LNCS*, pages 198–209. Springer-Verlag, 2007.

[21] E. Ridge and D. Kudenko. Screening the Parameters Affecting Heuristic Performance. In *Proceedings of the Genetic and Evolutionary Computation Conference*, volume N/A, page N/A. ACM, 2007.

[22] T. Stützle and H. H. Hoos. Max-Min Ant System. *Future Generation Computer Systems*, 16(8):889–914, 2000.