Improving Ant Colony Optimization Performance through Prediction of Best Termination Condition

M. Veluscek[1], T. Kalganova[1], P. Broomhead[1]

[1] *Electronic and Computer Engineering, School of Engineering and Design*
Brunel University
Kingston Lane, Uxbridge, UB8 3PH, United Kingdom

***Abstract*— The Ant Colony System (ACS) is a well-known bio-inspired optimization algorithm which has been successfully applied to several NP-hard optimization problems, including transportation network optimization. This paper introduces a method to improve the computational time required by the algorithm in finding high quality solutions. The purpose of the method is to predict the best termination iteration for an unseen instance by analyzing the performance of the optimization process on solved instances. A fitness landscape analysis is used to understand the behavior of the optimizer on all given instances. A comprehensive set of features is presented to characterize instances of the transportation network optimization problem. This set of features is associated to the results of the fitness landscape analysis through a machine learning-based approach, so that the behavior of the optimization algorithm may be predicted before the optimization start and the termination iteration may be set accordingly. The proposed system has been tested on a real-world transportation network optimization problem and two randomly generated problems. The proposed method has drastically reduced the computational times required by the ACS in finding high quality solutions.**

Keywords*— *Transportation Network Optimization, Ant Colony Optimization, Termination Condition Adaptation, Hardness Prediction, Instance Difficulty;

I. INTRODUCTION

In the past decade or so, meta-heuristics approaches have been successfully applied to many NP-hard optimization problems. These approaches are popular due to their adaptability and application potential across differing problem domains. They are known for the reduced effort involved in their application and for their ability to find high-quality solutions to the most complex of combinatorial optimization problems. Equally, they are also known for their high computational complexity. In this work, we address the task of improving the runtime complexity of the Ant Colony System (ACS) when applied to a real-world transportation network optimization problem. The improvement is achieved by adopting a machine learning approach. Given an unseen problem instance, the best termination point for the optimization process is predicted by analysing its behaviour on previously solved instances which are the most similar to the current one.

This paper is structured as follow: in section II we explain the motivations and analyse related work. In section III we present the fitness landscape analysis used to gather information on the optimizer behaviour. In section IV we define the features used to characterize different problem instances. Two differing class definitions are presented in the section. Such definitions are based on knowledge from the fitness landscape analysis and allow the setting of a termination criterion for a given problem instance. In section V we present the results of the optimization experiments carried out using the proposed method. In section VI we draw the conclusions and discuss future research directions.

II. MOTIVATIONS AND RELATED WORK

According to Arisha et al. [1] the most frequently adopted approaches used when solving supply chain optimization problems are *gradient-based methods*, *metamodel-based methods*, *statistical methods* and *random search / metaheuristics*. Arisha et al. [1] also discusses the limitations of traditional techniques such as linear programming, integer programming and mixed-integer programming when handling the inherent interdependencies found in the current generation of supply chain networks. A review conducted by Ogunbanwo et al. [2] identified a trend towards the use of meta-heuristic approaches as the solution basis for solving transportation networks problems. The most common approaches include (Multi Objective) Genetic Algorithm, Ant Colony Optimization (ACO), and Swarm Particle Optimization. The ACS is a variation of the ACO and is defined in Dorigo et al. [3]. A successful application to the problem of transportation network optimization may be found in Musa et al. [4].

In the context of Ant Colony System and Meta-heuristics approaches, much work has been done in reducing the runtime requirements of the methods. Typically, the most common approaches either employ methods to reduce the search steps and arrive more quickly at higher quality solutions or they exploit parallelization / hardware acceleration techniques. Tseng et al. [5] presented a novel method to generally speed up the Ant Colony Optimization (ACO) for the Travelling Salesman Problem (TSP), by reducing redundant steps in its search. Pedemonte et al. [6] present a survey of recent advances in the parallel implementation of Ant Colony Optimization. In this work, we approached the problem of runtime reduction by focusing on the optimal setting of termination criteria to minimize the runtime required for a given instance.

According to Dorigo et al. [7], for all meta-heuristics, there is no general termination criterion. In practice, a number of rules of thumb have been used: the maximum CPU time elapsed, the maximum number of solutions generated, the percentage deviation from an optimum lower/upper bound, and the maximum number of iterations without improvement in solution quality are examples of such rules [7]. Lv et al. [8] analysed recent reviews of Ant Colony Optimization applications with a view to answer the questions "how to evaluate improvement?" and "what are the termination conditions?". However, their survey did not provide concrete answers, they found that all termination criteria are described with vague phrases, such as "no improvement is possible", or "termination conditions are met" [8]. More generally, Lv et al. [8] considered some of the earlier fundamental work on meta-heuristics without finding a consensus about termination criteria. More recently, Zhang et al. [9] analysed the approximate termination condition for the ACO applied to TSP. They found that many of the termination condition are only used in experimentation and are often too difficult or uneconomic for deployment in solving practical problem [9].

The approach taken in this work learns from the behaviour of the optimization process on previous problem instances in setting the termination criteria. It was observed that in many instances the optimization algorithm finds the best solution early in its search and then stalls, continuing the search for many more iterations without finding a better solution. This phenomenon is referred to as the *stalling effect*, Stomeo et al. [10] state: "The problem of stalling effect in fitness functions is related to the non-improvement of the fitness values during the evolutionary process". Figure 1 shows an example of the stalling effect. In this work we search for a relationship between the problem characteristics and the performance of the optimization process, with the intent of predicting how the solver will perform on a given instance and set the termination criteria to minimize the solver search time.

In the course of the paper, we will show how this method answers the concerns raised by Lv et al. [8] and Zhang et al. [9]. We will provide a definite procedure to evaluate improvement and set proper termination conditions. Using well-known machine learning algorithms for prediction and existing and open source libraries for the implementation, the difficulty of adoption of the proposed system is kept low, making it economic for deployment in real-world application. Complexity wise, both the learning and prediction steps do not significantly affect the performances: the learning step, which is required to be performed only once, is expected to be fast due to the small number of features involved, and including the prediction step into the optimization process will significantly reduce the time requirements as the termination criteria are dynamically set to the optimum of each instance.

Understanding the relationship that exists between the problem instances and the optimization algorithm has led to improvements in the optimization process. Smith-miles et al. [11] used a knowledge discovery approach to seek insight into the relationship between the Scheduling Problem structure and the effectiveness of heuristics. Rules from a decision tree were used to select the best heuristic from a portfolio. Similar work has been undertaken in Smith-miles et al. [12] for the Travel Salesman Problem. We present a similar approach where instead of using the acquired knowledge to select the most promising algorithm from a portfolio, we use it to improve the performance of the current one.

III. Fitness Landscape Analysis

Fitness landscape analysis [13] provides a vivid metaphor of the search space as perceived by an optimization process [14]. Metaphors of a landscape are commonly used to aid the understanding of heuristic search methods when solving combinatorial optimization problems. Furthermore, the concept has been shown to be useful for understanding the behaviour of combinatorial optimization algorithms, and can help in predicting their performance [15].

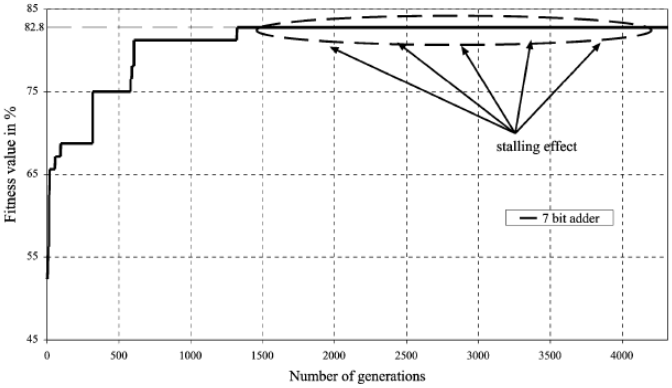

Figure 1 – The stalling effect in fitness function analysis refers to the phenomenon where the fitness values do not improve during most of the optimization process. Source of the figure is Stomeo et al. [10].

Given a vector of variables $x \in R^n$ and a vector of cost coefficients $c \in R^n$, a combinatorial optimization problem may be defined as:

$$v^{¿} = min\{c^T x \,|\, Ax = b \wedge x \geq 0\}, \qquad (0)$$

where $A \in R^{m \times n}$ is a matrix of coefficients, $b \in R^m$ is a vector of coefficients and $v^{¿} \in R^n$ is a vector of assignments for the variables x such that the value of the objective function $c^T x$ is minimum. The matrix A and the vector b define the constraints over the decision variables x and define the problem search space. Therefore, an optimization problem is defined by the tuple $lp := (c, A, b)$.

The fitness landscape of an optimization problem lp is the tuple $fl := (S, f, d)$, where $S(A,b) = \{v \in R^n | Av = b\}$ is the set of all possible solutions, $f : R^n \longrightarrow R$ is the fitness function defined as $f : v \mapsto c^T v$ and $d : R^n \times R^n \longrightarrow R$ is the distance between two feasible solutions. In Evolutionary Computation, for binary coded problems, the distance measure is usually the Hamming distance between bit strings [15]. For problems where the solution is a vector of real number, the Euclidean distance may be applied.

Usually, the fitness landscape is interpreted as a graph $G = \{V, E\}$ with vertex set $V = S$ and edge set $E = \{(v, v') \in S \times S \vee d(s, s') = d_{min}\}$ with d_{min} denoting the minimum distance between two points in the search space [15]. Such interpretation allows for effective analysis and visualization of the search space. However, for the purpose of this work, we are interested in analysing how the search for the optimal solution evolves over time and in predicting the best termination point based on instance features. Let us define the search process or *walk* on a landscape [16] as the t -tuple $\Gamma = (v_0, v_1, \ldots, v_{t-1})$ being the sequence of visited solutions during the search/optimization process. The fitness landscape analysis adopted in this work is, therefore, the sequence of fitness function evaluations at each iteration:

$$\Phi_\alpha = (f(v_0), f(v_1), \ldots, f(v_{t-1})) \quad (0)$$

.

The performance of the search process may be measured as the number of iterations required to find the optimal solution.

Let us define the speed of the search process and its acceleration, respectively as the improvement of the best known solution over the first one and the rate of change in the speed. Given the iteration $i \in (0, t)$, the speed of the optimization process for the tuple Φ is:

$$+¿ \to R \quad (0$$
$$s : Z^{¿},$$
$$i \mapsto (\Phi(i) - \Phi(0))/i,$$

and the acceleration is:

$$+¿ \to R \quad (0)$$
$$a : Z^{¿},$$
$$i \mapsto (s_\Phi(i) - s_\Phi(0))/i.$$

Such definitions of speed and acceleration describe the rate of improvement of the best known solution at any given iteration. Figure 2 shows an example of the result of this analysis.

As a variation, the fitness landscape analysis can be modified to include the topology of the search space by considering the mean pair-wise distance of visited solutions at any given iteration. The updated definition of fitness landscape analysis would be as follow:

$$\Phi_\beta = (p(0), p(1), \ldots, p(t-1)) \quad (0)$$

.

where let p be the function that measure the mean pair-wise distance of the visited solutions:

$$+¿ \to R \quad (0)$$
$$p : Z^{¿},$$
$$i \mapsto \frac{\sum_{k \in [0,i]} \sum_{j \in ¿} d(v_k, v_j)}{i * (i-1)/2},$$

and consequently the standard deviation on the pair-wise distance is:

$$+¿ \to R \quad (0)$$
$$p_{sd} : Z^{¿},$$
$$i \mapsto \sqrt{\frac{\sum_{k \in [0,i]} \sum_{j \in ¿} (d(v_k, v_j) - p(v_k))^2}{i * (i-1)/2}},$$

Figure 3 shows an example of the modified fitness landscape analysis.

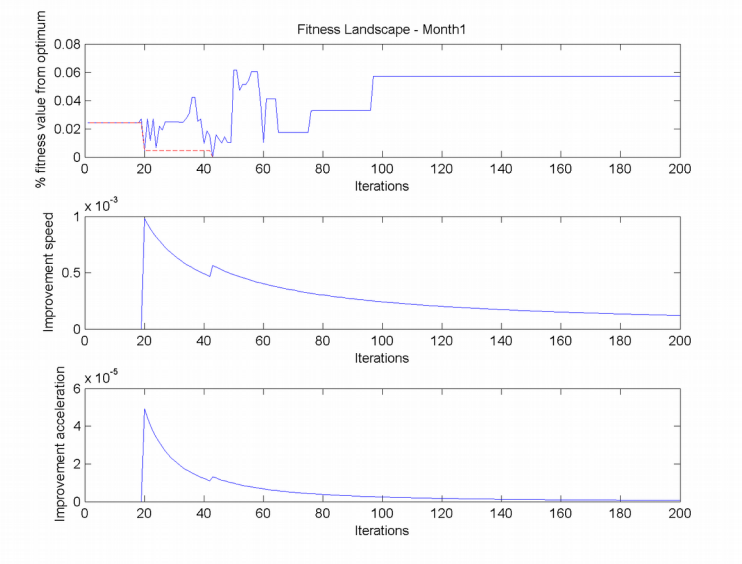

Figure 2 - Example of fitness landscape analysis as defined in Eq. (0) with speed and acceleration improvement. The definition of speed and acceleration is respectively in Eq. (0) and (0).

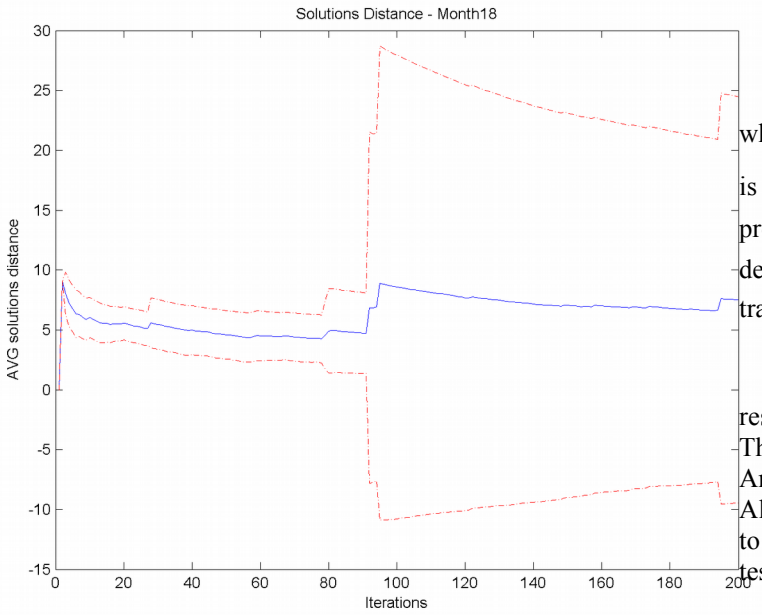

Figure 3 – Example of fitness landscape analysis as defined in Eq. (0). The solid line is the mean pair-wise distance of the visited solutions as defined in Eq. (0) and the dash-dot lines are the standard deviation on such mean as in Eq. (0).

IV. Features of a Transportation Network Optimization Problem

The optimization of transportation networks commonly consists of finding the best route to send products from a set of suppliers to a set of customers/dealers. As a generic problem, transportation network optimization is defined by a set of suppliers, a set of dealers and a distribution network. Each supplier is associated with a production capacity and cost, each dealer has a product demand which may vary over time and the distribution network is defined in terms of transportation times and costs between network nodes. Solutions to such problem are usually sought by the application of mathematical programming and artificial intelligence techniques. A minimal model for the problem is as follow:

$$min \sum_{i=1}^{p} \sum_{j=1}^{q} w_{ij} y_{ij} \tag{0}$$

$$s.t.: \sum_{j=1}^{q} y_{ij} \leq S_i \qquad \forall i \in Z^{+¿} \wedge i \leq p \tag{0}$$

$$\sum_{i=1}^{p} y_{ij} = D_j \qquad \forall j \in Z^{+¿} \wedge j \leq q \tag{0}$$

$$y_{ij} \geq 0 \qquad \forall i \in Z^{+¿} \wedge \forall j \in Z^{+¿} \tag{0}$$

where $p \in Z^{+¿}$ is the number of manufacturers, $q \in Z^{+¿}$ is the number of dealers with demand, $S_i \in Z^{+¿}$ is the production capacity at manufacturer i, $D_j \in Z^{+¿}$ is the demand from dealer j, $y_{ij} \in Z^{+¿}$ is the number of units transported from manufacturer i to dealer j, $w_{ij} \in R$ is the cost of sending product from the source i to the dealer j. Equations (0) and (0) represent respectively the constraints about capacity and demand.

The optimization algorithm implemented in this work is the Ant Colony System [3], the Vogel's Approximation Method of Allocation as described by Samuel et al. in [17] has been used to establish the starting solution. The parameters used for the test cases are as reported in Table 1.

Parameter	Value
Number of Ants	20
Maximum N ° of Iterations	1,000
Pheromone Evaporation Rate (ρ)	0.1
Weight on Pheromone Information (α)	1
Weight on Heuristic Information (β)	20
Exploitation to Exploration Ratio ($Q0$)	0.9

Table 1 - Ant Colony System set of parameters for all tested problem instances. These parameters are from the original definition of the Ant Colony System in Dorigo M. et al. [18].

The following sections, IV.A and IV.B, characterize features of the problem instances and propose two class definitions related to the solver behaviour. The purpose of the class definitions is to provide an understanding of the complexity of a given instance by considering the behaviour of the optimization algorithm. These class definitions allow the termination condition to be set according to the difficulty level of the instance. These are mostly related to the maximum number of iterations or to the maximum number of visited solutions. As described above, this work focuses on termination condition since we are interested in addressing the *stalling effect* problem and improving the time complexity of the optimization process over a given set of instances. However, it is reasonable to assume the same principle may very well be adopted to set others parameters. Arguably, for example a more difficult instance might require a higher number of ants or a lower exploitation to exploration ratio.

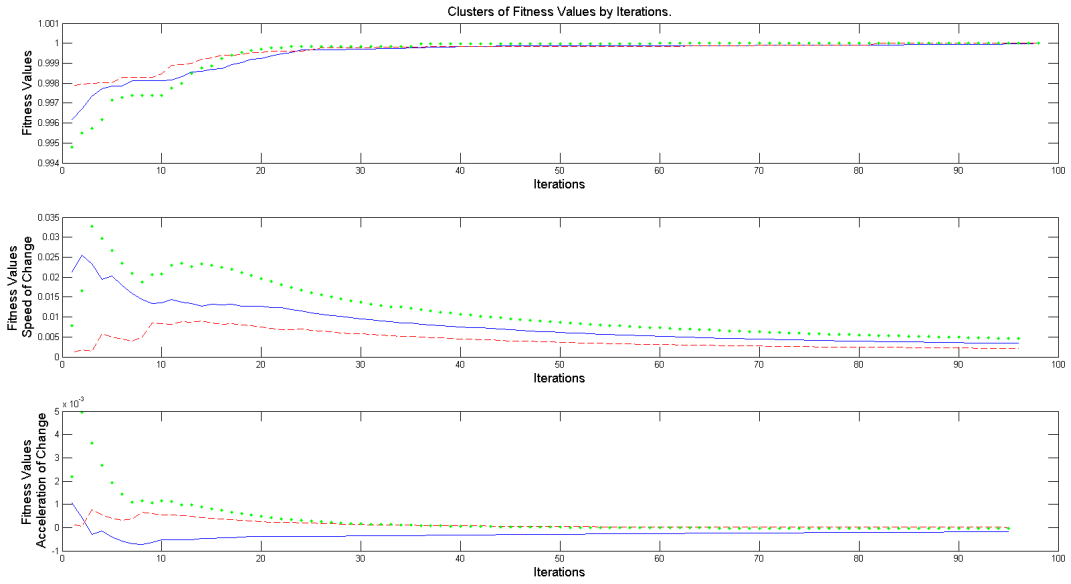

Figure 4 – Centroids result of the clustering of the fitness function analysis based on the definition in Eq. (0). The fitness function values are normalized for visualization purposes. The speed and acceleration of the resulting centroids is also measured according to Eq. (0) and (0).

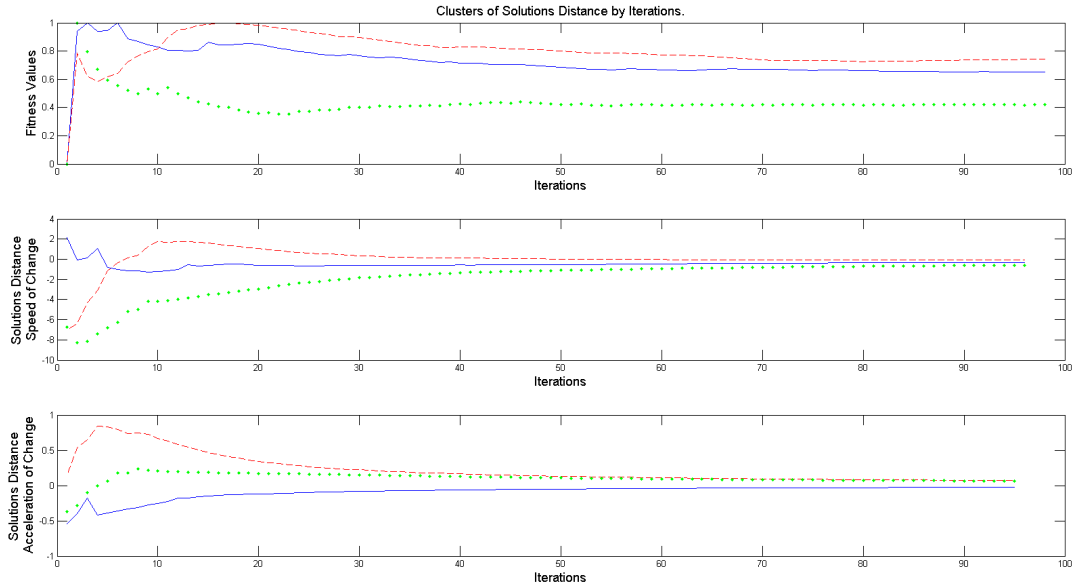

Figure 5 – Centroids result of the clustering of the fitness landscape analysis based on the definition in Eq. (0), (0), and (0). The speed and acceleration of the resulting centroids is also measured.

A. Problem Features

Supply chain optimization problems usually differ in the demand, the production capacity and some details of the distribution network.

The features we have adopted to summarize variations in demand and production capacities are:

- *Percentage of active dealers*. The total number of dealers is known from the definition of the full distribution network. Instances with more active dealers typically will be more difficult to solve and probably require more iterations.
- *Mean and standard deviation of the demand*. Such statistics briefly summarize the distribution of the demand through the network.
- *Mean and standard deviation of the capacity*. As with the demand, this feature describes the distribution of the capacity throughout the network.
- *Mean and standard deviation of the capacity per demand*. The purpose here is to measure how much

capacity is available on average to satisfy the demand of a given dealer.

- *Ratio of total demand to total production capacity*. This feature is a generalization of the previous one.

The features to describe the distribution network are:

- *The ratio of production sources to dealers*. This highlights how many production sources are available to satisfy a given dealer's demand.
- *The total number of connections between production sources and dealers*. This describes the level of connectivity in the underlying network.
- *Mean and standard deviation of the values in the heuristic information matrix*. In this context, the heuristic information refers to the information held on the routes in the network which guide the solver in building the distribution plan. For instance, if the goal is to maximizing the profit of a distribution plan, then the heuristic information is likely to be the transportation cost on the routes. Such a feature should distinguish between instances with different variations in transportation costs. Instances with uniformly distributed costs are likely to be easier to solve as small variations in the distribution plan will not fundamentally affect the overall profit.

B. Class Definition

As the intent of this work is to reduce the task of finding the best termination condition to a classification problem, the following class definitions need to be discrete and preferably of nominal type. A discretization of the class features has been achieved by applying a simple clustering on their values using instances of the training set. Three obvious class values $\{easy, medium, hard\}$ may be produced from the application of K-means [19] with 3 clusters.

1) Fitness Function Values Through Iterations

The first class definition is based on the fitness landscape analysis as define in Eq. (0). A smaller sample containing a sequence of fitness values is considered for each problem instance. The number of samples is 10% of the total number of iterations and the sampling rate is quadratic so that more iterations at the beginning of the search process are considered and hence more details are collected prior to the best solution being found. These sequences are the input to the clustering step and the output centroid are themselves sequences of fitness values. The speed and acceleration of the centroid sequences are measured as in Eq. (0) and (0). Figure 4 shows an example depicting the result of clustering the sequences of fitness function values. The termination condition is the average value of the following criteria:

- The first iteration when the best solution is found, $\left(argma\,x_{i\in[0,t)}\Phi_{\alpha}(i)\right)_0$.
- The highest iteration when the speed of change falls below the average speed. Let I be the set of iteration indexes where the speed is closer to the average value $I=argma\,x_{i\in[0,t)}\left(s_{\Phi_\alpha}(i)\cong\left(\sum_{j\in[0.t)}s_{\Phi_\alpha}(j)\right)/t\right)$. The iteration of termination is $I_{|I|}$.
- The highest iteration when the acceleration falls below to the average acceleration. Similarly to the step above the set I is defined as ¿ and the iteration of termination $I_{|I|}$.

$$a_{\Phi_\alpha}(j)$$
$$\sum_{j\in ¿}$$
$$(¿/t¿)$$
$$a_{\Phi_\alpha}(i)\cong ¿$$
$$I=argma\,x_{i\in[0,t)}¿$$

2) Pair-wise Distance Between Visited Solutions

The definition of the second class is based on the fitness landscape analysis as defined in Eq. (0). The purpose of this definition is to avoid visiting solutions that are the same or very close to each other. In almost all practical applications, the optimization process is stop after a finite number of search operations, regardless of whether the optimal solution has been found or not. An approximation to the optimal solution is generally acceptable provided the quality is reasonably high. Arguably, if one of the main concerns is reducing the computational time, then one may be willing to accept lower quality solutions. This definition attempts to terminate the optimization process as soon as the difference between visited solutions does not significantly improve the quality of the found solution; that is the tested solutions are not very different from each other and those perturbations do not lead to an improvement in the solution. As for the previous class definition, for each instance, the sequence of fitness function values is sampled according to a quadratic rate. These sequences are the input to the clustering step and an example of centroids is shown in Figure 5. Again, the termination condition is the average value of the following criteria:

- The first iteration when the pair-wise distance between the visited solutions is the highest, $\left(argma\,x_{i\in[0,t)}\Phi_{\beta}(i)\right)_0$.
- The highest iteration when the speed of change falls below the average speed. The iteration of termination is $I_{|I|}$ where the

set I is defined as

$$s_{\Phi_\beta}(j)$$
$$\sum_{j \in ¿}$$
$$¿$$
$$(¿/t¿)$$
$$s_{\Phi_\beta}(i) \cong ¿$$
$$I = argma\, x_{i \in [0,t)} ¿$$

.

- The highest iteration when the acceleration falls below to the average acceleration. Similarly to the step above, the termination iteration is $I_{|I|}$ where the set I is defined as

$$I = argma\, x_{i \in [0,t)} \left(a_{\Phi_\beta}(i) \cong \left(\sum_{j \in [0.t)} a_{\Phi_\beta}(j) \right) / t \right)$$

.

V. Numerical Experiments

The two class definitions described in IV.B have been tested on a real-world transportation network optimization problem. The details of the problem can be found in Veluscek et al. [20]. The profit maximization problem has been extended to consider inventory policy and stochastic variability in transportation costs (see [21] for examples of models that consider inventory policy and stochastic variability). As in [20], the data sets were provided by a real-world manufacturing company with a worldwide dealership network and an interest in logistic optimization. The company provided the transportation network map, demand data for 432 dealers in the period from January 2010 to December 2011, and data relating to the manufacturing costs, production capacities and regional sale prices. The problem complexity is quite significant due to the fact that the underlying transportation network is made up of 8 production facilities, 432 dealer locations and 48 shipping ports. The network representation is a four layer graph where:

1. The production facilities are connected both to the outbound shipping ports and the dealer locations.
2. At the outbound shipping ports it is possible to send product to the set of inbound shipping ports.
3. And the inbound shipping ports are connected to the dealer locations.

This network design resulted in almost 8 million potential routes between production facilities and dealer locations. In the 24 months (from January 2010 to December 2011) the dealer demands have been split into two independent problem instances and used as training and test sets. We adopted the data mining framework Weka [22] to implement and test several classification systems. The implemented classification systems are *0R*, *1R*, *Naïve Bayes*, *Bayes Network*, *J48*, *Random Forest* and *SMO with a polynomial kernel of degree 3*, all of which have been used to build a classification committee. Witten et al [23] in their book describe the theory and implementation details of all these algorithms. The algorithm *0R* is often used to set a baseline for classification accuracy, as its predicted class is simply the most frequent one. As the amount of data available for training and testing is limited, the performance of each single model has been assessed through a 10-fold cross validation scheme. In Table 2, the classification accuracy of the single models is shown.

The method presented in this paper also has been tested on two randomly generated problems. In both instances, the number of dealers, production facilities and shipping ports is the same as in the original problem; it is only the demand figures, the production capacities, the transportation times and costs and the sale prices that have been randomly generated. In the first problem, the figures have been generated according to a normal distribution with the same mean and standard deviation as in the original data set (e.g. the demand figures have the same mean and standard deviation as those found in the original problem). The figures for the second problem are randomly generated in an interval between 0 and an upper limit which is a random increase over the maximum value in the original data, according to a negative exponential distribution. Table 3 shows the accuracy measures for the first generated problem, whereas Table 4 shows them for the second.

Class definition	0R (baseline)	1R	Naïve Bayes	Bayes Net	J48	Random Forest	SMO Poly3	Average (STD)
IV.B.1	45.8	83.3	79.2	83.3	91.7	75.0	83.3	82.6 (5.06)
IV.B.2	41.7	83.3	75.0	95.8	87.5	87.5	79.2	84.7 (6.66)

Table 2 – Classification accuracy for 1000 iterations and K-means with 3 clusters to define the class. The 0R system provides a baseline for the classification problem. The average and standard deviation do not include 0R. 0R stands for ZeroR, the baseline classifier, 1R [24] for OneR and SMO [25] for sequential minimum optimization, the algorithm for training support vector machine.

Class definition	0R (baseline)	1R	Naïve Bayes	Bayes Net	J48	Random Forest	SMO Poly3	Average (STD)
IV.B.1	37.5	100.0	95.8	95.8	87.5	100.0	87.5	94.4 (5.2)
IV.B.2	37.5	100.0	95.8	95.8	87.5	100.0	87.5	94.4 (5.2)

Table 3 - Classification accuracy for 1000 iterations and K-means with 3 clusters to define the class. The problem is randomly generated according to a normal distribution with mean and standard deviation as in the original data set. The average and standard deviation do not include 0R. 0R stands for ZeroR, the baseline classifier, 1R [24] for OneR and SMO [25] for sequential minimum optimization, the algorithm for training support vector machine.

Class definition	0R (baseline)	1R	Naïve Bayes	Bayes Net	J48	Random Forest	SMO Poly3	Average (STD)
IV.B.1	62.5	83.3	75.0	100.0	87.5	87.5	83.3	86.1 (7.4)
IV.B.2	70.8	87.5	87.5	95.8	92.0	92.0	92.0	91.0 (2.9)

Table 4 - Classification accuracy for 1000 iterations and K-means with 3 clusters to define the class. The problem is randomly generated where the figures are an interval between 0 and an upper limit which is a random increase over the maximum value in the original data, according to a negative exponential distribution. The average and standard deviation do not include 0R. 0R stands for ZeroR, the baseline classifier, 1R [24] for

OneR and SMO [25] for sequential minimum optimization, the algorithm for training support vector machine.

A. Performance Improvements

In this section, we compare the performance of the original optimization process with the one developed in this paper and deploy it to predict the best stopping iteration for each given instance. The metrics for the experiment are the profit and computation time required to find the distribution plan. Table 5 shows the performance improvements when the classification system is adopted to predict the best stopping iteration for each given instance. As before, the first random problem is generated according to a normal distribution with mean and standard deviation as in the original data set. The figures for the second random problem are generated in an interval between 0 and an upper limit which is a random increase over the maximum value in the original data, according to a negative exponential distribution. As shown in Table 5, the application of the class definition from IV.B.1 consistently reduced runtime of about a third, while the found solutions have no meaningful difference in profit. The application of the class from IV.B.2 decreased the runtime furthermore with, however, a more significant loss in profit.

	Profit	Δ profit	Time (s)	Δ time
Original problem				
Regular optimization	3,297,976,866	-	1231.13	-
Class IV.B.1	3,297,480,246	**0.015%**	366.69	70.215%
Class IV.B.2	3,292,985,613	0.151%	23.65	**98.079%**
Random problem 1				
Regular optimization	2,751,544,955	-	3959.05	-
Class IV.B.1	2,751,606,950	**0.002%**	1196.28	69.784%
Class IV.B.2	2,751,328,020	0.008%	839.25	**78.802%**
Random problem 2				
Regular optimization	173,440,895,200	-	7355.24	-
Class IV.B.1	173,418,451,300	**0.013%**	1703.66	76.837%
Class IV.B.2	173,355,955,800	0.049%	442.7	**93.981%**

Table 5 - Performance improvements when the classification system is adopted to predict the best stopping iteration for each given instance. The first random problem is generated according to a normal distribution with mean and standard deviation as in the original data set. The figures of the second random problem are generated in an interval between 0 and an upper limit which is a random increase over the maximum value in the original data, according to a negative exponential distribution.

VI. CONCLUSION

The aim of this work was to make improvements in the practical time complexity for the Ant Colony System when applied to a real-world transportation network optimization problem. The starting observation was that in many instances the optimization algorithm finds the best solution early in its search and then stalls, effectively searching over many more iterations without finding a better solution. We referred to this phenomenon as the *stalling effect*. We postulate that if the onset of the stalling effect could be predicted, then for that given instance the search can be terminated with the ensuing benefit that the overall optimization process might very well require less time to find a solution of equal or comparable quality.

The approach we presented learns from the behaviour of the optimization process itself on past instances in setting the termination criteria. A relationship between specific characteristics of the problem and the performance of the optimization process is sought. The relationship is used to predict how the solver will perform on a given instance and to set the termination criteria such that the time spent by the solver in its search is minimized.

A fitness landscape analysis has been performed to understand the behaviour of the optimization process. Two class definitions have been proposed to capture the behaviour of the process, classify the problem instances and predict the best termination iteration. Features not related to the optimization process have been used to characterize different problem instances. Several classical machine learning classification algorithms have been employed to learn the relationship between problem instances and termination classes.

The proposed algorithm has been tested on a real-world transportation network, plus two random generated problems. The runtime of the Ant Colony System in all the experiments has been significantly reduced while the overall difference in the quality of the solutions was deemed acceptable.

As future work, we will be investigating more features of the problem in order to gain a better understanding of the factors that mate the greatest contribution the performance of the optimization process. Moreover, we would like to improve either the class definitions or the classification system such that the found solutions for all intent and purposes are no difference to the original ones. Finally, as anticipated, we believe a variation of this approach could be employed to set other parameters to their appropriate 'best' value. Future work will investigate the effectiveness of the method in setting other parameters and determining the impact of their relationship with the problem features.

VII. ACKNOWLEDGEMENT

Authors would like to thank the supporter of this work: the Logistics Research & Innovation team of Caterpillar Inc.

VIII. REFERENCES

[1] A. Arisha and W. Abo-Hamad, "Optimisation Methods in Supply Chain Applications: a Review," in *12th Annual Conference of the Irish Academy of Management, Galway Mayo Institute of Technology*, 2009.

[2] A. Ogunbanwo, A. Williamson, M. Veluscek, R. Izsak, T. Kalganova, and P. Broomhead, "Transportation Network Optimization," *Encyclopedia of Business Analytics and Optimization*, pp. 2570–2583, Feb. 2014.

[3] M. Dorigo and L. Gambardella, “Ant colony system: A cooperative learning approach to the traveling salesman problem,” *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.

[4] R. Musa, J.-P. Arnaout, and H. Jung, “Ant colony optimization algorithm to solve for the transportation problem of cross-docking network,” *Computers & Industrial Engineering*, vol. 59, no. 1, pp. 85–92, Aug. 2010.

[5] S.-P. Tseng, C.-W. Tsai, M.-C. Chiang, and C.-S. Yang, “A fast Ant Colony Optimization for traveling salesman problem,” *IEEE Congress on Evolutionary Computation*, pp. 1–6, Jul. 2010.

[6] M. Pedemonte, S. Nesmachnow, and H. Cancela, “A survey on parallel ant colony optimization,” *Applied Soft Computing*, vol. 11, no. 8, pp. 5181–5197, Dec. 2011.

[7] M. Dorigo and T. Stützle, *Ant Colony Optimization*. Cambridge, MA: MIT Press, 2004.

[8] Q. Lv and X. Xia, “Towards Termination Criteria of Ant Colony Optimization,” in *Third International Conference on Natural Computation (ICNC 2007) Vol V*, 2007, vol. 5, pp. 276–282.

[9] Z. Zhang, Z. Feng, and Z. Ren, “Approximate termination condition analysis for ant colony optimization algorithm,” in *2010 8th World Congress on Intelligent Control and Automation*, 2010, no. 60875043, pp. 3211–3215.

[10] E. Stomeo, T. Kalganova, and C. Lambert, “Generalized disjunction decomposition for evolvable hardware.,” *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, vol. 36, no. 5, pp. 1024–43, Oct. 2006.

[11] K. A. Smith-miles, R. J. W. James, J. W. Giffin, and Y. Tu, “Understanding the Relationship between Scheduling Problem Structure and Heuristic Performance using Knowledge Discovery,” in *Learning and Intelligent Optimization*, 2009.

[12] K. Smith-miles, J. Van Hemert, and X. Y. Lim, “Understanding TSP Difficulty by Learning from Evolved Instances,” in *Learning and Intelligent Optimization*, 2010, pp. 266–280.

[13] S. Wright, *The roles of mutation, inbreeding, crossbreeding, and selection in evolution*. 1932.

[14] E. Pitzer, M. Affenzeller, A. Beham, and S. Wagner, “Comprehensive and Automatic Fitness Landscape Analysis Using HeuristicLab,” in *Computer Aided Systems Theory – EUROCAST 2011*, vol. 6927, R. Moreno-Díaz, F. Pichler, and A. Quesada-Arencibia, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 424–431.

[15] B. Freisleben and P. Merz, “Fitness landscape analysis and memetic algorithms for the quadratic assignment problem,” *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 4, pp. 337–352, 2000.

[16] J. Humeau, A. Liefooghe, E.-G. Talbi, and S. Verel, “ParadisEO-MO: from fitness landscape analysis to efficient local search algorithms,” *Journal of Heuristics*, vol. 19, no. 6, pp. 881–915, Jun. 2013.

[17] A. E. Samuel and M. Venkatachalapathy, “Modified Vogel ’ s Approximation Method for Fuzzy Transportation Problems,” vol. 5, no. 28, pp. 1367–1372, 2011.

[18] M. Dorigo, V. Maniezzo, and a Colorni, “Ant system: optimization by a colony of cooperating agents.,” *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, vol. 26, no. 1, pp. 29–41, Jan. 1996.

[19] J. MacQueen, “Some methods for classification and analysis of multivariate observations,” in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, 1967, vol. 233, no. 233, pp. 281–297.

[20] M. Veluscek, A. Ogunbanwo, A. Williamson, T. Kalganova, P. Broomhead, and A. J. Grichnik, “Benchmarking of Meta-heuristic Algorithms for Real- World Transportation Network Optimization,” 2014.

[21] J. J. Bravo and C. J. Vidal, “Freight transportation function in supply chain optimization models: A critical review of recent trends,” *Expert Systems with Applications*, vol. 40, no. 17, pp. 6742–6757, Dec. 2013.

[22] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The WEKA Data Mining Software: An Update.” 2009.

[23] I. H. Witten, E. Frank, and M. A. Hall, *Data Mining Practical Machine Learning Tools and Techniques*, 3rd ed., vol. 40, no. 6. Elsevier Inc., 2001, p. 9823.

[24] R. C. Holte, “Very simple classification rules perform well on most commonly used datasets.,” *Machine Learning*, vol. 11, pp. 63–91, 1993.

[25] J. Platt, “Fast Training of Support Vector Machines using Sequential Minimal Optimization,” in *Advances in Kernel Methods - Support Vector Learning*, B. Schoelkopf, C. Burges, and A. Smola, Eds. MIT Press, 1998.