Proceedings of the 2009 Winter Simulation Conference
M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls, eds.

BUFFER CAPACITY ALLOCATION USING ANT COLONY OPTIMISATION ALGORITHM

Ivan V. Vitanov
Advanced Technology Centre
BAE Systems
Chelmsford, UK

Valentin I. Vitanov
School of Engineering and Computing Sciences
Durham University
Durham, DH1 3LE, UK

David K. Harrison
School of Engineering and Computing
Glasgow Caledonian University
Glasgow, G4 0BA, UK

ABSTRACT

This paper presents an algorithm for the near-optimal allocation of buffer space to an assembly line by means of the ant colony optimisation (ACO) paradigm. Uniquely, the algorithm has been designed to work in conjunction with a simulation model and is adapted to have both combinatorial and stochastic problem-solving capability. The simulation model was developed using the WITNESS simulation package and serves the purpose of an objective function evaluator, encapsulating the dynamics of the line and enabling the production rate for a particular buffer configuration to be determined. Finally, the WITNESS Optimiser module was used as a benchmark in validating the ACO algorithm's performance. In the simulation experiments conducted, ACO attained slightly better results overall.

1 INTRODUCTION

Often research into production lines focused on using variety of complex analytical and stochastic simulation models for predicting (evaluating) the performance characteristics of a given system configuration. Over the last decades a number of publications on the optimisation of such systems have appeared. Extensive literature reviews can be found in Gershwin (1994), Papadopoulos and Heavey (1996) and Altiok (1996). These sources provide clear indication that many of the practical simulation-optimisation problems are *NP*-hard and only limited number of "what-if" scenarios can be examined in a simulation experiment. Therefore to obtain practical solutions for large instances modellers often use approximate methods returning near-optimal solutions in relatively short time. In addition to computational complexity simulation activity usually require significant upfront investment amounting to hundred thousand pounds for the acquisition of one of the software packages and training and maintaining appropriate staff. This severely limits the number of companies that can afford simulation as decision making tool in their everyday planning activity. It appears that for well defined practical problems like the allocation of buffers in production lines it might be beneficial to focus attention on developing appropriate simulation heuristics and methaheuristics that will increase the ability of finding high-quality solutions to practically relevant combinatorial optimisation problems in real-time and at low cost.

This paper study the applicability to buffer capacity allocation of a particularly successful methaheuristic named Ant Colony Optimisation Dorigo and Stutzle (2004) inspired by the foraging behaviour of real ants. The research challenge was related to the selection and realisation of an appropriate ACO algorithm and evaluation procedure (objective function) guiding the optimisation process. In practical situations often it is only possible to determine estimates of the expected objective function by means of sampling or simulation which is known as simulation optimization.

In the context of simulation optimisation, various methods have been examined in the literature. These include standard non-linear programming techniques, response surface methodology, stochastic approximation techniques and random search. Moreover, commercial simulation optimisation software has been dominated by metaheuristic approaches, which though lacking in theoretical convergence guarantees have nevertheless proven robust in overcoming local optima.
Comprehensive reviews of the literature on simulation optimisation can be found in Swisher et al (2000) and Fu (2002). A systematic survey article on metaheuristics is provided by Blum and Roli (2003). Results on the application of genetic algorithms and simulated annealing to the buffer allocation problem are presented in Spinellis and Papadopoulos (1999 and 2000) and Bulgak et al (1995). A thorough overview of the ant colony metaheuristic can be found in Dorigo and Di Caro (1999).

The production line, considered in this paper is one of the most commonly encountered types of manufacturing systems. It comprises a serial arrangement of machines or workstations which are clusters of one or more machines served by single

operator. Theoretically the operator of machine begins work on a job as soon as one becomes available. On Completion of the service, the job immediately moves to the next station as long as there is room for it. Thus an operator or machine can become starved (no job available) or blocked (no space to put a completed job). As a preventative measure against loss of productive capacity due to starvation and blocking, production lines typically contain buffer spaces. These are storage locations between successive machines and usually take the form of conveyors of different length, floor space to stack parts or special purpose storage units. The buffer capacity allocation in manufacturing systems design is a combinatorial optimisation problem of allocating limited resource to different alternatives. It is *NP* hard optimisation problem and the main goal is to find quickly acceptable quality solution. Using Ant Colony Optimisation heuristic. The difficulty is that very often the objective function cannot be represented as an explicit mathematical expression. Practical situations of determining the size and the position of the buffers usually face a complex system governed by random disturbances or events, and discrete-event simulation is used to fulfil this role. The simulation model is viewed as a stochastic objective function and is integrated with ant colony optimisation procedure in a 'closed-loop' arrangement, using feedback from the simulator to estimate the effect of parameter changes in a decision making procedure. In the present study, an ant colony algorithm has been developed to determine and optimise the buffer sizes in an asynchronous assembly line of closed-loop type comprising ten machines and as many buffers. The line itself was modelled using the WITNESS simulation software and used as an objective function in the optimisation process. The ACO algorithm was eventually tested against WITNESS's own thermostatistical simulated annealing optimisation routine.

2 FORMULATION OF THE SIMULATION SCENARIO

The system considered in this study is an assembly line with closed-loop material handling known as asynchronous assembly lines (AALs) which are common for the automotive industry. The original line adopted for the purposes of the study is (loosely) based on a real-axle assembly line used in BMW manufacturing plant. Closed-loop AALs are perhaps rarer than ordinary serial lines and so have not received as much attention in the literature on buffer optimisation. Nevertheless, their effective operation is vital to many sectors of industry, and they arguably possess a number of unique characteristics which necessitate independent study. In the AAL considered here, a set of assembly stations are arranged in tandem in the order of assembly operations performed. The stations are connected by conveyors.

Assemblies are transported on work carriers or pallets with fixtures and/or jigs to hold component parts. In closed-loop asynchronous assembly systems, a fixed number of pallets or carriers are circulated ad infinitum by the material handling system. As soon as a job is finished at the last machine, M_m, a new job (raw part) is immediately placed on the pallet, after which it returns to machine M_1. The number of circulating pallets is p. Clearly, the number of pallets affects the production rate of the line. By reason of their asynchronicity, such lines allow for the independent movement of pallets and are therefore unreliable due to random fluctuations in processing times (as well as machine failures).

The buffer allocation problem consists in finding an optimal configuration of buffer sizes so as to maximise a certain performance measure such as the steady-state production rate of the line.

In the assembly line shown in figure 2 below, a set of assembly stations are arranged in series in the order of assembly operations performed. The stations are connected by conveyors.

As can be seen from the figure, the modelled line is a single-loop system having ten machines and ten intermediate buffers, labelled M_1, M_2 ,…, M_{10} and b_1, b_2 ,…, b_{10} respectively. There is also an unload station before the first machine. A fixed number of pallets (this number can be varied in the simulation model) circulate perpetually in the line. Assemblies are carried around by the pallets, following a routing that takes them from the first through to the last station in sequence, with the finished assembly being removed at the unload station, leaving an empty pallet to return to the first machine where the assembly process starts over. Clearly, the number of pallets affects the production rate of the line.

The pallet dimensions and the span of the conveyor between any two adjacent stations determine the number of pallets that can be accommodated between these two adjacent stations. The maximum quantities for these work-in-process (WIP) inventories between each pair of stations represent the buffer sizes (capacities).

The task of buffer allocation in conveyor-driven systems such as the one investigated is complicated by the need to account for random fluctuations due to transport delays, in addition to fluctuations in production and failure rates. Transportation delays are proportional to buffer size, since a longer conveyor revolves over a greater distance and the speed of revolution is constant across all conveyors. The danger of having buffer sizes that are too large is that transport delays will be excessive and more in-process inventories must be input into the system to mitigate the large buffer sizes. When the buffer sizes are too small, then small processing delays will cause the buffer to

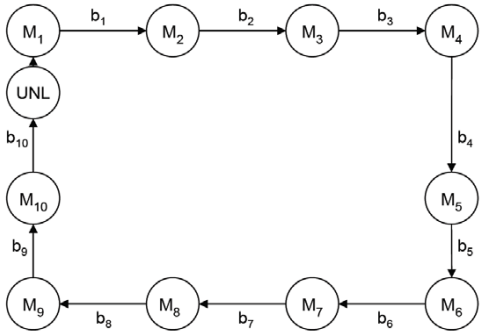

Figure 1:Schematic of the investigated assembly line

fill up. With a fixed number of pallets in the system, there is always an optimal buffer configuration able to limit blocking and starvation effects considerably so as to generate a maximum possible steady-state production rate.

The simulation model is characterised by the following parameters:

Processing or cycle times This is the time required for a station to perform an assembly task. Processing times are taken to be normally distributed random variables sampled according to a truncated normal distribution, with a variance of 10 seconds for all machines and ± 20 seconds upper and lower limit. Two separate line configurations (balanced and unbalanced) were tested, distinguished by different mean processing times as laid out in the table below. The values for the unbalanced line were randomly generated within realistic bounds.

Table 1 Machine processing times and breakdown intervals

	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}
Unbalanced (sec.)	110	110	120	90	100	90	80	80	120	90
Balanced (sec.)	100	100	100	100	100	100	100	100	100	100
Number of operations between failures	427	58	357	221	425	276	369	243	187	135

Jamming failure rate Stations are subject to jamming due to various reasons such as system breakdowns, defective parts being assembled or a robotic assembly station accidentally dropping the part it is holding. Jam occurrences are random events. They are modelled as an average breakdown interval for each machine, calculated in terms of parts produced. The negative exponential distribution was used to describe the jam rate. The individual jam times were again randomly generated within realistic limits.

• *Mean time to repair* This is the average time it takes to clear up a jam at a workstation expressed in seconds. The jam clear times are random variables modelled on the Erlang distribution. They were similarly generated randomly between realistic limits.

• *Delayed buffers* The buffers are intended to imitate the actions of conveyors and have a minimum delay time which is proportional to the buffer space: each additional unit of buffer space adds 7 seconds to the delay time. The buffer sizes between pairs of stations are the decision variables and can vary between 1 and 10 (with a step size of 1) for each buffer. For a line with 10 buffers this means there are 1010 possible buffer configurations.

The expected value of the production rate is represented by the number of parts produced in 100 000 simulation seconds (the duration of a simulation run), after a warm-up period of 10 000 seconds to remove the initial transient. At the end of each run the model is rewound to time zero in preparation for the next run. Independence of runs is maintained through incrementing the random number stream and substream for each random variable in the model.

3 STRUCTURE OF THE SOFTWARE APPLICATION

The software implementation of the optimisation algorithm and its synchronisation with the simulation model involved the use of a number of software packages besides WITNESS; namely, Matlab, Visual Basic and Microsoft Excel. The software application consists of the assembly line model developed in the WITNESS simulation environment, the optimisation program - written in Matlab code and executed from a Visual Basic module, and an Excel front end to display the results of the optimisation and simulation routines. The data exchange interfaces between the separate components of the application are displayed in figure 2. The optimisation algorithm had to be developed independently of the WITNESS environment because the capabilities of WITNESS do not extend to general-purpose numerical programming. Instead, Matlab, a high-performance language for technical computing, was chosen as a suitable implementation platform.

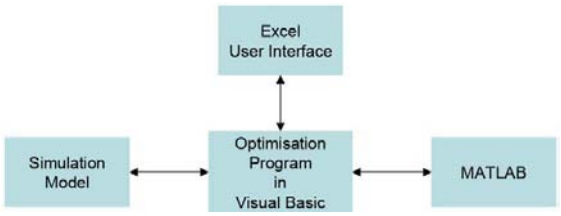

Figure 2: Schematic of the optimisation methodology

The OLE automation protocol was employed in order to connect the simulation model (WITNESS) to the optimisation procedure (Matlab). OLE automation is a way of letting programs control each other. WITNESS and Matlab are both 'OLE Automation Servers' and both can be controlled by other programs. Visual Basic is an example of an 'OLE Automation Controller', meaning that it can control other programs. WITNESS and Matlab are not 'OLE Automation Controllers', and so cannot control other programs. Because Visual Basic is an 'OLE Automation Controller' and WITNESS and Matlab are 'OLE Automation Servers', it is possible to control WITNESS and Matlab from Visual Basic. This, however, is but one likely scenario, as there are other languages and programs besides Visual Basic that are 'OLE Automation Controllers'.

The way the integration of WITNESS and Matlab is achieved within the software application is by executing the main program from Visual Basic, which runs the simulation model and optimisation procedure in a synchronised fashion, exchanging instructions and data between the two as required. The main program consists mostly of Matlab commands framed in Visual Basic code which are sent to Matlab for execution. The program code enables ant solution construction to take place within Matlab and for the solutions (buffer allocation vectors) to be fed into the WITNESS model for evaluation. The simulation is then run with each solution configuration and the corresponding value for the production rate returned to Matlab via the main program. The simulation results are subsequently used to update the algorithm's memory and so guide future solution construction. The cycle begins afresh in the next iteration until a given number of iterations are completed or a stopping condition is fulfilled. The intermediate results of the optimisation are displayed in an Excel spreadsheet concurrently with the running of the application. The final results are displayed in Excel at the close of program execution. figure 3, shows the program screens at run-time. In the diagram, the 'simulation' frame encloses the WITNESS GUI with the assembly line model in run mode. The 'optimisation' frame contains the following screenshots from top to bottom: the Excel front end, the Visual Basic module with the main program, and the Matlab runtime command window.

4 ANT COLONY OPTIMISATION PROCEDURE

The program code and scheme for the algorithm are given in Vitanov (2006). The algorithm draws on the work of Gutjahr and the S-ACO algorithm developed by him (Gutjahr, 2004) which represents the first general-purpose ant algorithm that tackles stochastic combinatorial problems, i.e. combinatorial optimisation under uncertainty. The buffer algorithm performs a search in the feasible solution space looking for an optimal solution. In this context, a solution can be any configuration of buffer sizes (b_1, b_2 ,…, b_{10}), as long as the values they take do not exceed the limits set on individual buffer capacity, which, as previously noted, could range from 1 to 10 pallet spaces.

The stepwise construction of a solution is represented by a random walk in the construction graph C, beginning in the start node. Following the definition of the construction graph encoding given in figure 4, the walk must satisfy the condition that each node is visited at most once; already visited nodes are infeasible. When there is no feasible unvisited successor node available, the walk stops and is decoded as a complete solution for the problem. The conceptual unit performing the walk is termed an artificial ant.

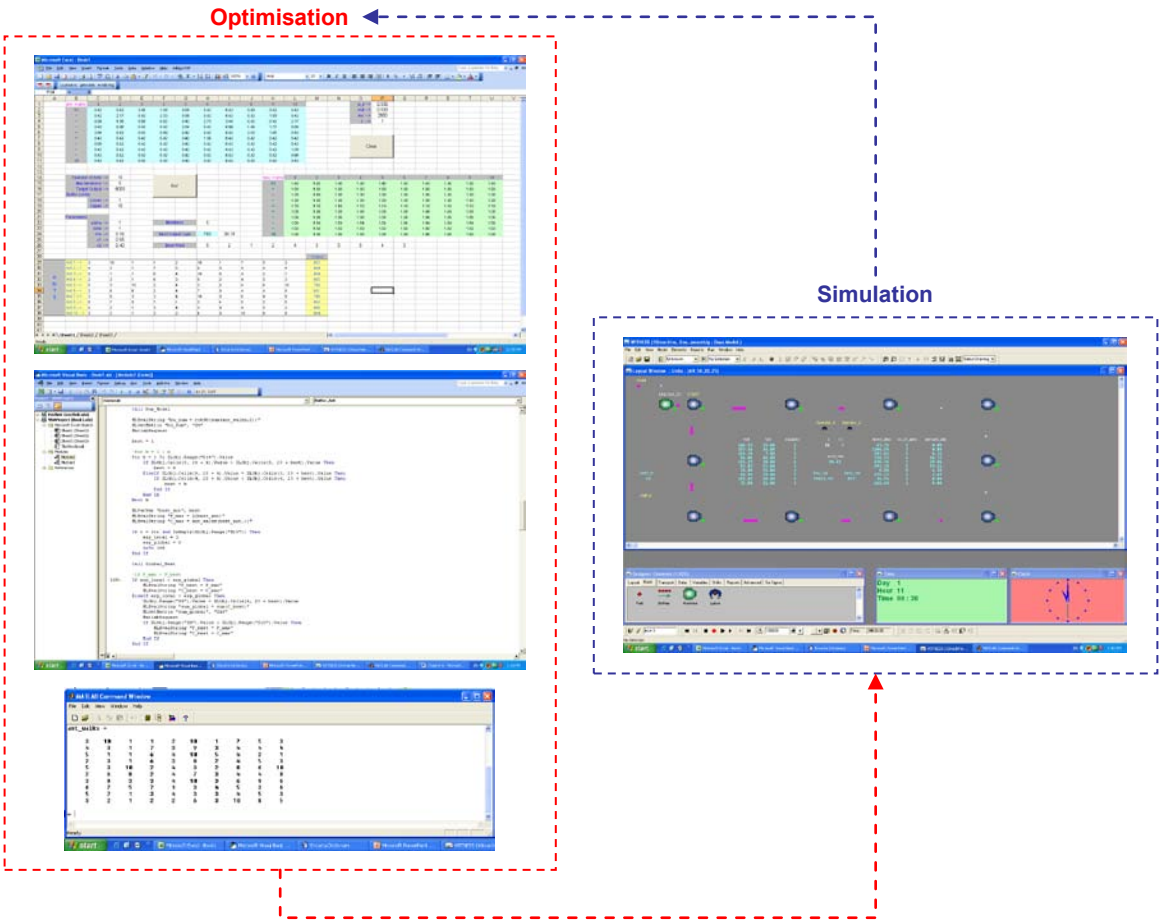

Figure 3 Example of the programme screens of the application at runtime

To each feasible walk in the sense above, there corresponds exactly one feasible solution. When constructing a walk in the construction graph, the probability $p_{i,j}$ to go from a node i to a feasible successor node j is chosen as proportional to $[\tau_{i,j}]^{\alpha}[\eta_{i,j}]\beta$ where $\tau_{i,j}$ is the pheromone value and $\eta_{i,j}$ is the heuristic value. The complete formula is:

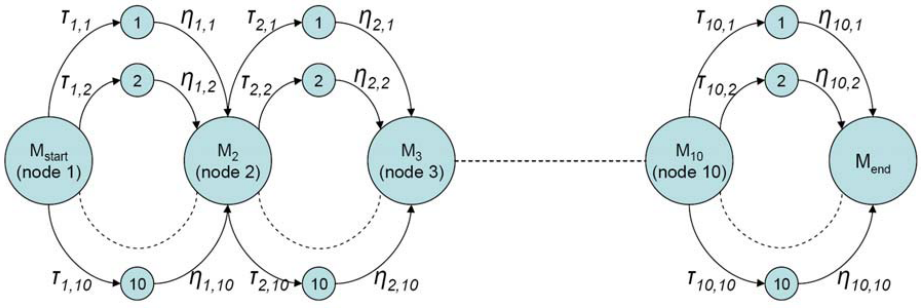

Figure 4: Construction graph for the buffer allocation problem

$$p_{i,j} = \begin{cases} 0, if\ (i,j) is\ not\ \ feasible, \\ \dfrac{\left[\tau_{i,j}\right]^{\alpha}\left[\eta_{i,j}\right]^{\beta}}{\sum_{(i,j)}\left[\tau_{i,k}\right]^{\alpha}\left[\eta_{i,j}\right]^{\beta}} \end{cases}$$

As in some other frequently used ACO variants for deterministic problems, in each round a round-winner is determined. In the stochastic context, this is done by comparing all walks that have been performed in this round on a single random scenario, drawn specifically for this round. In an ACO implementation for a deterministic problem, it is always reasonable to store the best solution seen so far in a special variable. A crucial difference to the deterministic case is that in a stochastic context it is not possible anymore to decide with certainty whether a current solution is better than the solution presently considered as the best found. Only a 'good guess' can be made by sampling. After a current round-winner has been determined, it is compared with the solution considered currently as the overall best solution. This is done based on a sample of $\omega = 5$ randomly drawn scenarios used by both solutions. Also, these scenarios are round-specific, i.e. in the next round new scenarios will be drawn. The winner of the comparison is stored as the new 'global best' solution. The round winner and global best are the only solutions whose constituent arcs are updated with pheromone. Description of the S-ACO algorithm in pseudocode is exhibited in Figure 5.

The optimisation algorithm itself has certain parameters which need to be adjusted in order to get the best performance out of it. These are: alpha and beta, the pheromone and heuristic weights respectively; rho, the pheromone evaporation rate; and c_1 and c_2, the global-best and local-best reinforcement. These parameters were tuned using designed experiments and the determination of a response surface metamodel at each stage the results were compared against the WITNESS simulation model.

5 EXPERIMENTAL PROCEDURE

5.1 Summary of ACO parameters

The way an ant colony optimisation algorithm exploits the ants collective search experience is by updating the pheromone trail as a function of the solution quality achieved by each of the participating ants [Dorigo, and Stuetzle, 2004]. In the BAP algorithm presented in this thesis, this is controlled by five parameters: Alpha(α), Beta(β), Rho(ρ), C1(c1) and C2(c2). These determine the relative importance of heuristic and pheromone values, as well as the deposition and evaporation rate of the pheromone trails of the algorithm.

5.1.1 Decision policy control parameters

Alpha(α) and Beta(β) are parameters of the algorithm that determine the relative importance of pheromone intensity and the predefined desirability of solutions in the ants' decision making process. The pheromone intensity represents recently acquired knowledge, in other words the learned attractiveness of an option, and the desirability signifies the inherent attractiveness of an option. By changing the ratio between Alpha(α) and Beta(β) one is in a position to make the ACO optimisation process more exploratory or more conservative when exploring new domains of the case study response surface in search for an optimum. When Alpa>>Beta, the algorithm will make decisions based mainly on the learned information as represented by the pheromones. In cases when Beta>>Alpha, the algorithm will act as a classic stochastic greedy algorithm, selecting mainly nominally optimal solutions and largely ignoring the outcome on final solution quality.

5.1.2 Pheromone evaporation parameter

The pheromone persistence factor Rho(ρ) controls the probability for the selection of decision paths that are not regularly selected by ants in the decision making process. This parameter is important because it makes the algorithm adaptive to recently acquired knowledge, allowing the latest information to guide the optimisation process. The suggested values from the literature are for $0.5 < \text{Rho} < 0.95$. For $\text{Rho}(\rho) \rightarrow 1$, only small amounts of pheromone will evaporate between iterations of the algorithm, which slows down convergence. For $\text{Rho}(\rho) \rightarrow 0$, the p heromone evaporates faster resulting in faster convergence.

set $\tau_{k,l} = 1$ for all (k,l);
for round $m = 1, 2, ..., \{$
  for ant $\sigma = 1, ..., s \{$
    set k, the current position of the ant, equal to the start node of C;
    set u, the current walk of the ant, equal to the empty list;
    while (a feasible continuation (k,l) of the walk u of the ant exists)
      select successr node l with probability $p_{k,l}$ where

$$p_{kl} = \begin{cases} 0, if\ (k,l)\ is\ \text{infeasible}, \\ \tau_{k,l}\,\eta_{k,l}(u) / (\sum_{(k,r)} \tau_{k,r}\,\eta_{k,r}(u)), else, \end{cases}$$

      the sum being over all feasible (k,r);
      set $k = l$, and apend l to u;
    }
    set $x_\sigma = u$;
}
based on one random scenario w, select the best walk x out of the
  walks $x_1, ..., x_s$;
if $(m = 1)$ set $\hat{x} = x$; / / $\hat{x}$ is the candidate for the best solution
else {
  based on N_m random scenarios w_v, compute a sample estimate

$$\varepsilon(Fx) - F(\hat{x})) = \frac{1}{N_m} \sum_{v=1}^{Nm} (f(x, w_v) - f(\hat{x}, w_v))$$

  for the difference between the costs of x and $\hat{x}$,
  if $((\varepsilon(Fx) - F(\hat{x})) < 0)$ set $x = \hat{x}$;
}
evaporation: set $\tau_{k,l} = (1-\rho)\tau_{k,l}$ for all (k,l);
global-best reinforcement: *set* $\tau_{k,l} := \tau_{k,l} + c_1$ for all $(k,l) \in \hat{x}$;
local-best reinforcement: set $\tau_{kl} := \tau_{kl} + c_2$ for all $(k,l) \in x$;
}

Figure 5: Pseudo code description of the adopted S-ACO algorithm following (Gutiahr , 2004)

5.1.3 Pheromone increment parameters

The pheromone increment parameters C1 and C2 determine the amount of pheromone increase on global-best and round-best walks respectively. Both parameters are in the range C1 > 0 and C2 > 0. Literature sources suggest that C2 should be smaller than C1, but the cases when C2 has a very small positive value produce better results in comparison to when it is set equal to 0 [Gutjahr, 2004].

5.2 Sensitivity analysis procedure

While tuning ACO parameters when the analytical value of the objective function is unknown the optimisation runs are performed for ranges of the algorithm parameters determined on the basis of initial experiments. Experimental settings of the parameters are determined through Response Surface Methodology (RSM) using Box-Behnken and D-optimal plans.
The proposed procedure consists of the following stages:

1. Using design of experiments software "Design Expert" Response Surface Methodology option to determine an initial experimental plan for the specified parameter ranges.
2. Performing 50 iterations with the optimisation algorithm for each instance of the parameter settings.

3. Performing a series of (5 to 10) replicated simulation runs to estimate the average value of the best production rate achieved with a given set of parameter values.
4. Using average values obtained as above to fit an appropriate polynomial model using the least squares method (with Design Expert software).
5. Performing numerical and graphical analysis of the solutions obtained using analysis of variances (ANOVA) and 3D graphical representation of the response surface polynomial approximation.
6. Tuning the ACO parameters to achieve maximum performance of the response variable (production rate) using one of the Design Expert optimisation algorithms.
7. Setting the ACO control parameters to the 3 best sets of optimal values calculated above, and performing a further 50 iterations for each and analysing the findings.

Two sets of experiments were studied in accordance with the methodology described above:

A)α=1, and varying the parameter ranges for Beta(β), Rho(ρ), C_1 and C_2.
B)α=1, Beta(β)=0 and varying the parameter ranges for Rho(ρ), C_1 and C_2. (i.e. the heuristic matrix is neutralised)

5.3 Optimisation experiments

The line model considered in this text has a solution space comprising 1010 solutions. Because of its magnitude, it was not possible to know in advance what an optimal solution might look like. To validate the ACO algorithm's performance therefore and to be sure of finding a close-to-optimal design for the system under study, it was decided to compare the algorithm against an established simulation-based metaheuristic technique with known performance guarantees. This served the double purpose of determining the fitness of the ACO algorithm in solving the buffer allocation problem and gauging its suitability for simulation optimisation. As already stated, these issues were addressed by running tests with both the WITNESS Optimiser module and the ACO algorithm on the assembly line model developed in WITNESS. To this end, the WITNESS Optimiser and ACO algorithms were compared over a set of 10 simulation experiments each, for both the balanced and unbalanced line configurations. In each case, five different pallet inputs were considered. To ensure a fair comparison, both algorithms were allowed to complete exactly 1000 solution evaluations in each experimental instance.
In each experimental instance, after the algorithms had completed the set number of evaluations and had yielded their respective best solutions, i.e. best buffer configurations found, they were then compared on the basis of 10 replications executed with the simulation model, from which an average response value was calculated for each solution. The replications were performed using the same set of random number scenarios for both solutions being compared.
For the simulation model used, thirteen was found to be a critical number of pallets in relation to system performance. Including fewer pallets meant getting a trivial result, with the smallest buffer configuration, i.e. all buffer sizes equal to one, being found optimal. This occurred, since queuing effects were rendered almost non-existent because of too few pallets, so that transportation times, which are contingent on buffer size, had an overwhelming impact on production rate. With pallets > 13, the randomness and general noise in the system seemed to increase cumulatively with each additional pallet, so that thirteen could be seen almost as a tipping point in terms of stability.

With respect to actual running time, the WITNESS Optimiser is at a distinct advantage because it is integrated into the WITNESS environment itself while the ACO algorithm is spread across four different applications, so running times are not comparable. That being said, overall running times were not that different between the two algorithms: WITNESS Optimiser took approximately 1hour 30minutes to complete 1000 evaluations, whereas ACO took around 1hour 45 minutes. Although both algorithms were run for the same number of evaluations, that should not be taken to mean that they each performed the same number of simulation runs as part of the optimisation process. For the 1000 evaluations completed in each experiment, ACO performed 2000 simulation runs of the model, while WITNESS Optimiser performed 5000. The number of simulation runs performed depends partially on the sample size used in calculating average values. In the experiments described, a sample size equal to 5 replications was used by both algorithms. In addition, the ant algorithm used 10 ants throughout, which is the equivalent of 10 solution evaluations per iteration. In each experimental instance, after the algorithms had completed the set number of evaluations and had yielded their respective best solutions, i.e. best buffer configurations found, they were compared on the basis of 10 replications executed with the simulation model, from which an average response, i.e. production rate, value was calculated for each solution. The replications were performed using the same set of random number scenarios for both solutions being compared.

5.4 The "unbalanced" line experiment

The results of the experiments for the unbalanced case are summarised in table 8.1, below. As can be seen from the table, tests were performed with the following pallet inputs: 13, 18, 23, 28, 33.
The coloured rows in the table present the results for the WITNESS Optimiser. The right-most column displays average production rate calculated over 10 replications, where production rate is taken to be number of parts produced per 100 000 seconds (the length of a simulation run).

Table 2 Test results for the unbalanced line configuration

ACO \ Witness Optimiser ==> Unbalanced Model											
Buffer 1	Buffer 2	Buffer 3	Buffer 4	Buffer 5	Buffer 6	Buffer 7	Buffer 8	Buffer 9	Buffer 10	Pallets	Averages
2	3	1	2	2	2	1	1	3	1	13	793.5
1	4	2	1	1	1	1	3	3	1	13	795.5
7	5	4	3	1	2	2	5	4	1	18	806.9
1	5	4	1	4	2	1	6	6	6	18	806.3
2	8	6	3	2	5	1	5	9	1	23	822.8
1	8	4	1	1	1	1	6	9	1	23	823.6
2	9	5	4	1	3	4	8	7	4	28	820.2
1	10	5	1	2	1	1	7	9	3	28	821.2
2	10	7	1	2	2	2	5	9	9	33	826.3
9	10	6	1	2	1	1	6	10	2	33	825.0

The results of the simulation experiment are shown in figure 6. For the unbalanced line, WITNESS Optimiser outperformed ACO in three of the five test instances, though its performance was bettered on the remaining two. Overall, the results were closely matched. The best system design of those tested would appear to be in the case of 33 pallets.

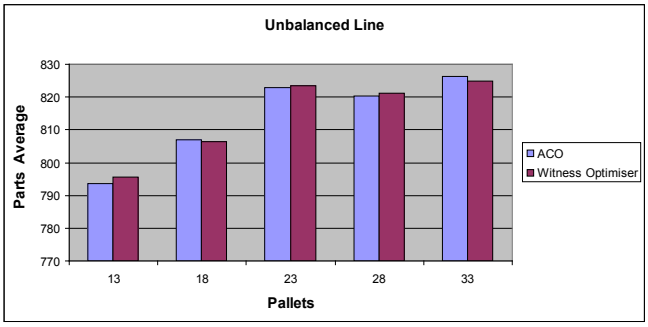

Figure 6:Best average production rate for unbalanced line

5.5 The "balanced" line experiment

For this scenario, the previous experimental settings were retained; only the line configuration was changed to 'balanced'. The results from the optimisation runs are displayed in table 3 and figure 7. The coloured rows indicate the WITNESS Optimiser results.
As revealed by the results, this batch of experiments is an improvement on the previous in relation to the performance of ACO, which came out on top in four of the five trials. This means that over the 10 test instances investigated, ACO leads WITNESS Optimiser 6:4. Of course, more tests would have to be conducted for the results to be conclusive, but even so, it could be justifiably said that the ACO algorithm has shown itself a capable optimisation heuristic.

A possible explanation as to the worse showing of ACO in the unbalanced-line case could reside with the fact that to select the round-best solution ACO relies on evaluations performed on a single random number scenario. Although this approach is economical, in cases where the model being optimised has low sensitivity, as in the unbalanced case, there are going to be different configurations that repeat the same response, so that multiple replications are needed to ascertain solution quality.

Table 3 The results for the balanced line configuration

ACO \ Witness Optimiser Balanced Model ==> Balanced Model											
Buffer 1	Buffer 2	Buffer 3	Buffer 4	Buffer 5	Buffer 6	Buffer 7	Buffer 8	Buffer 9	Buffer 10	Pallets	Averages
1	2	1	2	1	2	1	1	2	1	13	884.6
1	1	1	1	1	1	1	1	1	2	13	886.4
4	3	3	1	2	3	2	1	1	1	18	922.7
2	4	2	3	3	1	2	2	4	2	18	921.3
5	6	1	3	3	1	4	4	6	1	23	933.1
3	5	2	4	2	2	2	2	9	4	23	932.2
7	8	3	3	5	6	2	5	6	4	28	943.7
3	7	4	5	2	2	3	2	8	8	28	943.6
10	10	5	3	7	3	9	5	4	8	33	949.5
6	8	6	5	4	2	2	6	7	10	33	946.7

Although ACO applies replications when comparing round-best and global-best solutions, the initial filtering of evaluations to select a round-best solution foregoes multiple replications, from which average values can be derived. WITNESS Optimiser, on the other hand, can replicate each evaluation.

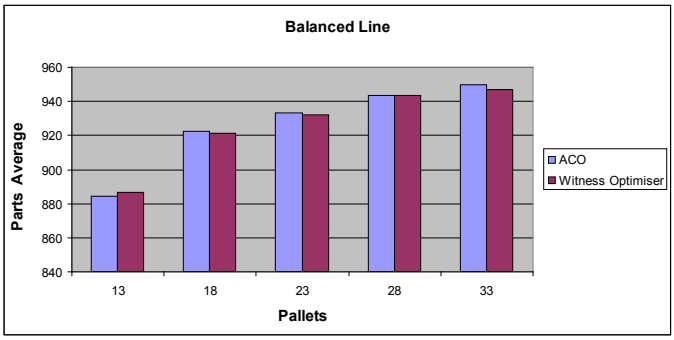

Figure 7:Best average production rate for balanced line

5 CONCLUSION

This paper has presented a new solution method for the buffer allocation problem in asynchronous assembly lines. The proposed solution incorporates both simulation and optimisation components: a specially developed stochastic ant colony algorithm is integrated with a simulation model of a closed-loop assembly line.

The main contributions of the proposed algorithm can be summarised as follows.

- It produces optimal or near-optimal solutions in a reasonable amount of computer time for which upper bounds can be established. In the case of large lines, solution quality may have to be traded off against time.
- With minor adaptations, it can cope with different line topologies, lengths and limits on buffer capacities to be distributed.
- It takes into account the main characteristics of the buffer allocation problem, i.e. discreteness of the buffer sizes, presence of multiple local optima and lack of an explicit objective function to evaluate production rate with respect to the buffer sizes.
- The algorithm is capable of overcoming local optima to find a global optimal solution.
- It is a stochastic algorithm: the generation of solutions is randomised to an extent. It probabilistically maps the solution space of the problem.

Future research could incorporate further testing, including more experimental runs and a wider array of alternative algorithms. Further experimental work would better establish the validity of the algorithm's performance. Additionally, the convergence properties of the algorithm could be studied and compared against the experimental findings.

REFERENCES

Altiok, T. 1997. *Performance Analysis of Manufacturing Systems*. Springer-Verlag, New York.

Blum, C. and A. Roli. 2003. Metaheuristics in combinatorial optimization: overview and conceptual comparison. *ACM Computing Surveys* Vol. 35. No. 3. pp. 268-308.

Bulgak, A.A., P.D. Diwan, and B. Inozu. 1995. Buffer size optimization in asynchronous assembly systems using genetic algorithms. *Computers in Industrial Engineering* Vol. 28. No. 2. pp. 309-322.

Dorigo, M. and G. Di Caro. 1999. Ant algorithms for discrete optimization. *Artificial Life* 5: 137-172.

Dorigo,M., Stutzle, T.2004.*Ant Colony Optimization*. The MIT Press.

Fu, M.C. 2002. Optimization for Simulation: Theory vs. Practice. INFORMS *Journal on Computing* Vol. 14. No. 3. pp.192-215.

Gershwin, S.B. 1994. *Manufacturing Systems Engineering*. Prentice Hall, New Jersey.

Gutjahr, W. 2004. S-ACO: an ant-based approach to combinatorial optimization under uncertainty. Ants Workshop, pp. 238-249.

Papadopoulos, H.T. and Heavey C. 1996. Queuing theory in manufacturing systems analysis and design: A classification of models for production and transfer lines. *European Journal of Operational Research*. 92: 1-27.

Spinellis, T.P. and Papadopoulos, C.T. 1999. Production line buffer allocation: genetic algorithms versus simulated annealing. *Second International Aegean Conference on the Analysis and Modelling of Manufacturing Systems*, pp. 89-101.

Spinellis, T.P. and Papadopoulos, C.T. 2000. A simulated annealing approach for buffer allocation in reliable production lines. *Annals of Operations Research* 93. pp. 373 -384.

Swisher, J.R., P.D. Hyden, H.J. Sheldon and L.W. Schruben. A survey of simulation optimization techniques and procedures. In *Proceedings of the 2000 Winter Simulation Conference*, eds. J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, 119-128. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Vitanov, I.V. 2006. A simulation-based ant colony algorithm for assembly line buffer allocation. *MSc thesis*, Cranfield University.

AUTHORS BIOGRAPHIES

IVAN V. VITANOV is a Research Engineer in the BAES, Advanced Technology Centre. He received his engineering degree in Mechatronics from Kings Colledge London in 2003 and an MSc in Computational Methods from Cranfield University UK in 2006. Before joining BAE Systems he spent time working for BMW in Germany. His research interest include stochastic learning methods, optimisation, simulation and autonomous robotics. His email is <IvanVitanov@baesystems.com>.

VALENTIN I. VITANOV is a Professor of Design Manufacture and Management in The School of Engineering at the University of Durham, UK. He received his PhD in Industrial Automation and Robotics from the St. Petersburg Electrotechnical University (Russia). He is a Fellow of the Institute of Engineering and Technology, UK. His research interests include applied statistics and robust design, manufacturing systems simulation and optimisation, operational research and autonomous robotics. His email is <v.i.vitanov@durham.ac.uk>.

DAVID K HARRISON is a Professor at the School of Engineering and Computing at Glasgow Caledonian University. He received his PhD from the University of Manchester, UK. His research interest primarily concerned with improving overall manufacturing competitiveness in companies via Optimised Design, CADCAM and advances towards CIM. Most of his research has been in association with industrial partners. He is Fellow of the Institute of Engineering and Technology, Fellow of the Institute of Mechanical Engineers and President of the Institute of Engineers and Shipbuilders in Scotland. His email address is <D.Harrison@gcal.ac.uk>.