IEMS Vol. 4, No. 2, pp. 184-191, December 2005.

Parameters Influencing the Performance of Ant Algorithms Applied to Optimisation of Buffer Size in Manufacturing

Matthias Becker†
Department of Computer Science, FG Simulation and Modeling
University Hannover, 30167 Hannover, Germany
Tel: +49 (0)511-762-3295, E-mail: xmb@sim.uni-hannover.de

Helena Szczerbicka
Department of Computer Science, FG Simulation and Modeling
University Hannover, 30167 Hannover, Germany
E-mail: hsz@sim.uni-hannover.de

Abstract. In this article we study the feasibility of the Ant Colony Optimisation (ACO) algorithm for finding optimal Kanban allocations in Kanban systems represented by Stochastic Petri Net (SPN) models. Like other optimisation algorithms inspired by nature, such as Simulated Annealing/Genetic Algorithms, the ACO algorithm contains a large number of adjustable parameters. Thus we study the influence of the parameters on performance of ACO on the Kanban allocation problem, and identify the most important parameters.

Keywords: Optimisation, Kanban, Petri Net, Ant Colony Optimisation

1. INTRODUCTION

Many companies implement the Just-In-Time (JIT) philosophy in order to survive amongst the competitors. The basic idea of the JIT philosophy is to produce only what is needed, that means minimising the stock and inventory, and letting the production be triggered by the current demands.

One way to control the material flow in a JIT manufacturing environment consisting of several manufacturing cells is the use of Kanbans (Sugimori *et al*. 1977), first used in Japan, Kanban meaning 'card'. These cards serve as production orders and also determine the buffer size in each manufacturing cell. The information flow in such a manufacturing line consisting of individual manufacturing cells is directed upstream. That means, a demand at the last production cell triggers a production order only in the cell before. If there is the desired part on stock, then the order can be fulfilled, otherwise the part is manufactured. This philosophy is called 'pull-production' (parts are pulled from the cell before only when there is a demand) as opposed to 'push-production' where raw parts are pushed into the first manufacturing cell independently of the demands (production to stock). A crucial point in the planning and design of a manufacturing line is to determine the number of Kanbans per cell. The higher the number of Kanbans, the higher the achievable throughput usually will be, but also the inventory/work in process (WIP) and the corresponding costs increase. Thus an optimal Kanban allocation to cells has to be found, that maximises a goal function that takes the costs (production, inventory, transport costs and losses due to demands that could not be satisfied) and the gain (caused by sold products) of an individual Kanban allocation into account. The so-called *Kanban allocation problem* has been approached by different heuristic optimisation algorithms in the past, amongst them Genetic Algorithms and Hill-climbing (Szczerbicka *et al*., 1998). Since there is no universal best optimisation algorithm (Wolpert and McReady, 1997), it is important to assess the quality of optimisation algorithms for specific problems. Ant Colony Optimisation (ACO) (Dorigo and Gambardella, 1997) is a new heuristic optimisation algorithm, that has not been applied to the Kanban allocation problem yet. Regarding application in manufacturing environments, up to now, ACO has mainly been used for scheduling optimisation (Price *et al*. and Kumar *et al*., 2003). Since the ACO is

† : Corresponding Author

particularly applicative for optimisation problems with an integer search space it is interesting to study the application of ACO to the Kanban allocation problem. This has been done in Becker *et al*. (2004). The following text is an extension of the afore mentioned previous work. First, the ACO algorithm is explained in the next section, then we introduce the manufacturing system and the optimisation problem (namely the Kanban allocation problem). In section 4 it is shown how the ACO algorithm can be applied to find solutions of the Kanban allocation problem. Finally, results about the performance of the ACO algorithm, the quality of found solutions and the dependency of both on the parameters of the ACO algorithm are presented and a conclusion is drawn.

2. ANT COLONY OPTIMISATION

In Dorigo and Gambardella the Ant Colony Optimisation algorithm is introduced (developed from the Ant System (AS) by Dorigo, 1992). The ACO is inspired by the mechanism with which natural ants find shortest paths between a food source and their nest.

Roughly, this works as follows: Initially, ants randomly find different paths of different lengths. An individual ant leaves a trace of pheromone where it is walking. A pheromone is a kind of scent which is left for the orientation of other ants. Ants prefer paths with a higher pheromone concentration with a higher probability. Pheromone is also subject to evaporation. Thus after completing a tour to a food source and back, the pheromone of the longer paths has had more time to evaporate and thus has a weaker concentration compared to shorter paths. When ants are at a decision point at which different paths start, they choose with higher probability a path with a higher concentration of pheromone. By using this path the pheromone concentration is being increased again. This mechanism reinforces the pheromone concentration on short paths between two decision points. A complete short path can be constructed by adding the short paths between all decision points. In the end a shortest path has been established from the starting point to the end and is used by the majority of ants.

The ant algorithm can be illustrated easily be a typical application, the Travelling Salesman Problem (TSP). There a salesman has to visit a number of cities in an order that minimises the overall travel distance. The mathematical representation is a graph with the N cities as nodes, and weighted edges between all cities. The weight of an edge between cities i and i represents the distance $d(i,j)$ between these cities. The ant approach to the TSP works as follows:

- Initial step: As first step, let a number of ants find random tours. Each ant leaves an amount of pheromone on the edges belonging to the tour (local pheromone update rule). This amount is proportional to the length of the tour. Additionally the best tour found up to now (or alternatively during one iteration) is marked with some amount of extra pheromone (global pheromone update).
- Iteration step: A fraction of the pheromone is evaporated. The evaporation speed is an adjustable parameter. Let again find a number of tours by a swarm of ants. Now, each ant decides at each node which node to choose next. Ants prefer shorter edges and edges with a higher pheromone concentration in a probabilistic manner. Adjustable parameters are the weights with which heuristic information (here: the length of path) and the amount of pheromone influence the probabilistic decision.

Several common termination conditions can be used, such as a pre-determined number of generations, or if there is no improvement of the solution over a number of iterations. In the next section the manufacturing system to be optimised is introduced.

3. THE KANBAN MANUFACTURING SYSTEM

As mentioned before the Kanban mechanism is used in a JIT environment where major principles are production on demand and keeping inventory low. The Kanban mechanism is a way to implement a pull-production (production is triggered by demand) and controlling the inventory. Kanban systems are organised in several cells. The Kanbans control the inventory in each cell. Each part that enters a cell i needs to find a free Kanban at the bulletin board Bi that is then attached to the part (cf. Figure 1).

Then the part is stored in the input buffer IBi and waits for being processed at machine Mi. Then the part waits in the output buffer OBi until the next cell issues a demand (in form of a free Kanban in that cell). When a part leaves a cell, the Kanban is detached from the part and placed in the bulletin board, and serves as production order to the previous cell. By this mechanism the number of Kanbans in a cell determines the maximum inventory of this cell. The Kanban allocation problem deals with assigning an optimal number of Kanbans to each cell. The more Kanbans circulate in each cell, the more throughput is achievable. The negative effect of many Kanbans is an increase in maximum as well as mean inventory that leads to larger storage costs and more inflexibility. Finding the optimal number of Kanbans for each cell with respect to a cost function that calculates the balance of gain by throughput of produced parts on the one hand and costs of production on the other hand is called the *Kanban allocation problem*.

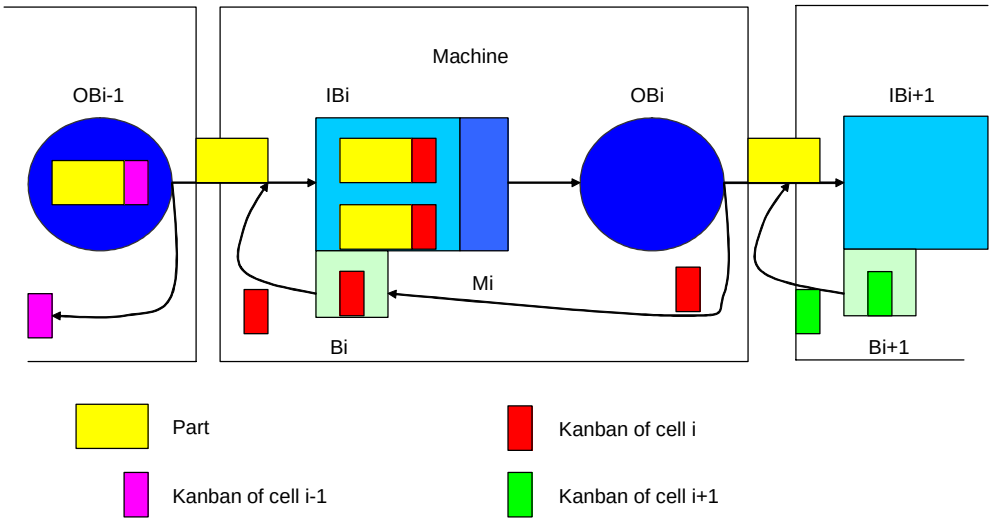

Figure 1. Flow of parts and Kanban in the system

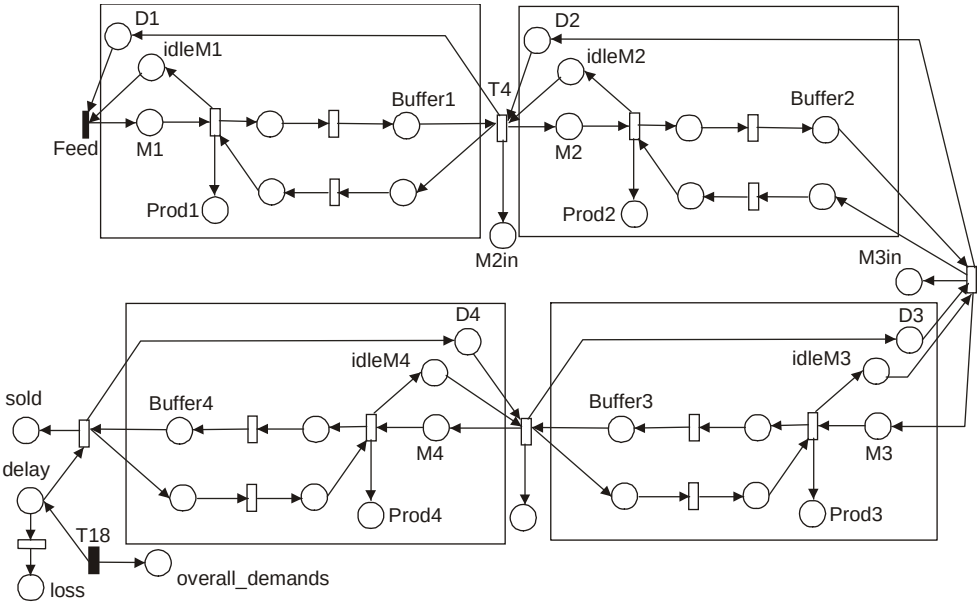

Figure 2. PN model of the four-stage Kanban system

Stochastic Petri nets (Marsan *et al.*, 1995) have proven to be an adequate mathematical modeling formalism for Kanban systems, since Petri nets are well suited to represent synchronisation and split operations (used for matching/detaching of parts and Kanbans). Petri nets can by analysed by using linear algebra techniques, solution of the underlying Markov chain as well as simulation.

3.1 Petri net model of the Kanban manufacturing system

In this section we present the Kanban system that we use for studying the applicability of the ACO algorithm on the Kanban allocation problem.

A detailed or formal introduction of Petri nets is out of scope here. Only some informal idea is given here, for more information see the relevant literature, e.g. in (Marsan *et al.*, 1995). A Petri net consists of passive elements (called places, drawn as circles) in which objects can be placed (called tokens, drawn as black dots inside places) and active elements (called transitions, drawn as rectangles) that can move tokens along arcs from one place to another. A transition can model a time span that is needed to move a token (unfilled rectangle) or a mere logical, timeless operation (filled rectangle).

Transitions can also model conditions such as synchronisation, when two or more arcs lead from places (representing the conditions) to one transition (that is executed if all conditions are fulfilled). Two or more arcs leading away from one transition model a split operation. The system (see Figure 2) consists of four manufacturing cells. In each cell a number of containers with Kanbans attached circulates, this number is represented by the initial number of tokens in places *Buffer1, Buffer2, Buffer3, Buffer4*. The number of processed parts in each cell is counted in places *Prod1, ..., Prod4*. Stochastic demands are generated by transition *T18*. Fulfilled demands that contribute to the profit are counted in place *sold* while demands that could not be fulfilled and lead to a loss are counted in place *loss*.

3.2 Transient analysis

The Kanban system has been analysed by conducting a stochastic simulation with confidence level of 95 percent. The (partly stochastic) delays of single tasks in the system were in the order of minutes, the simulation covered 24 hours of operation. The simulation yields the performance parameters that are connected to WIP, fulfilled and lost demands, delay and transport costs (see Table 1) that will be needed in the cost function later.

Table 1. Performance measures calculated from the simulation model

Parameter	Explanation
Ai	produced amount in cell i
Bi	change of WIP in cell i
Ci	mean WIP in cell i
Di	number of transports between cell i and cell j
E	number of lost demands (because of reaching the time limit)
F	demands not yet met
G	mean delay of demands

4. APPLICATION OF ACO TO THE KANBAN ALLOCATION PROBLEM

In this section the Kanban allocation problem is reformulated, so that it is accessible to the ACO algorithm. The Kanban allocation problem needs to be represented by a graph with one start node where the ants start their tours. One valid Kanban allocation corresponds to a path through the graph ending at a final node. These considerations lead to the following problem representation in Figure 3 (Example for a three stage Kanban system with a maximum of three Kanbans at each stage).

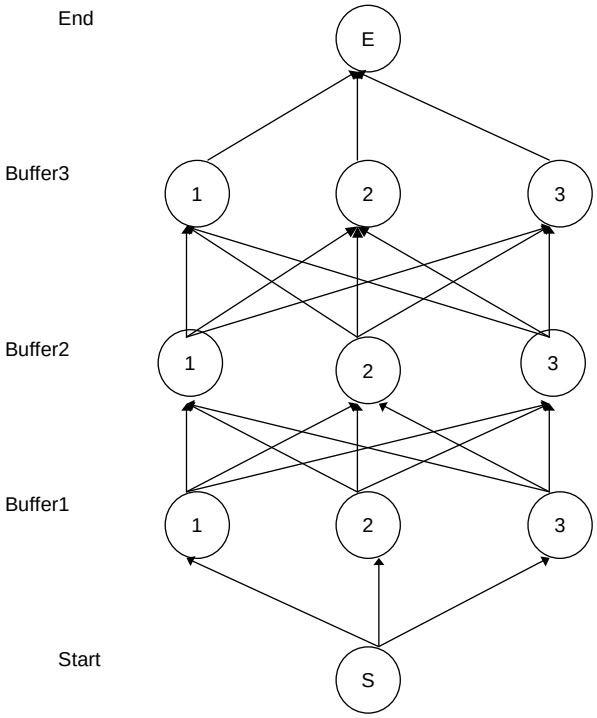

Figure 3. Problem representation for ACO

The ants start at node *S*. The choice of the next node determines the number of Kanbans in the first stage. From each of this nodes, one node of the next level can be chosen, again representing the number of Kanbans for this stage. This continues, until the ants reach the final node *E*. Each tour through this graph represents a valid Kanban allocation. Additionally a heuristic information (similar to the distance in the TSP problem) is needed by the ACO algorithm. We use the function

$$\eta(b_j) = \frac{1}{b_j \cdot l_j} \tag{1}$$

to describe the heuristic information, where *bj* are the number of Kanbans to be chosen and *lj* is the storage cost per Kanban (for the detailed use of the heuristic function η cf. one of the original papers e.g. Dorigo *et al*. 1996). With this function lower numbers of Kanbans are heuristically preferred (in the same manner in that ants heuristically tend to choose the shorter path at a decision point). Using solely the heuristic information (disregarding the pheromone information) leads to a greedy algorithm.

4.1 Basic parameters

Since we want to study the effect of a single pa-

rameter on the quality of the found solution and on the performance of ACO, we need a basic setting of parameters. Table 2 shows all relevant parameters, an explanation and the value in the basic setting. When studying the effect of one parameter, the others are kept constant.

Table 2. Basic parameters of ACO

Name	meaning	init.value
N	number of ants	5
g	number of generations	700
τ_0	initialisation of pheromone	0.5
α	weight of pheromone on decision	0.5
β	weight of heuristic data on decision	0.5
q_0	degree of random choice at decision point	0
Q	amount of pheromone to be deposited along a tour	2
ρ	percentage of pheromone evaporation during one step	0.95

4.2 Cost function

The cost or goal function is used in an optimisation algorithm to rate the quality of a found solution. The optimisation algorithm tries to find a solution that maximises or minimises the value of the goal function. The goal function that is to be optimised by the ACO algorithm calculates the costs that a given Kanban allocation generates during the production period. These costs comprise costs for raw material, transport costs, storage cost, production cost, and penalty costs for delay of demands and demands that could not be satisfied. The production rate is triggered by external demands arriving in a stochastic manner according to an exponential distribution. The goal function used here is equation 2:

$$Z = A_1 \cdot k_0 + \sum_{i=1}^{W} (A_i(f_i + t_i) + B_i(k_i + t_{pi}) + C_i \cdot l_i + D_i \cdot t_{mi}) + (E+F)k_{op} + G \cdot k_s \quad (2)$$

The meaning of the variables of the different costs are found in Table 3, while the performance measures needed from the simulation model have already been mentioned in Table 1.

Table 3. Parameters of the cost function

cost	Explanation
k_0	cost for raw material
f_i	production cost in cell I
k_i	sum of cost of product up to cell i
t_{pi}	transport cost in cell I
l_i	storage cost in cell I
t_{mi}	transport cost between cell i and j
k_{op}	cost for demands not met
k_s	cost for delay of demands

5. RESULTS

In order to assess the effect of the parameters of ACO on the performance of the algorithm, a relatively small Kanban system has been chosen. Thus it is possible to do an exhaustive search and determine the global optimum, so that the results of ACO can be rated and compared to the global optimum. Figure 4 shows the number of solutions (y-axis) within a certain cost interval (x-axis). Overall 10000 configurations have been evaluated (four cells with a maximum of ten Kanbans allowed.).

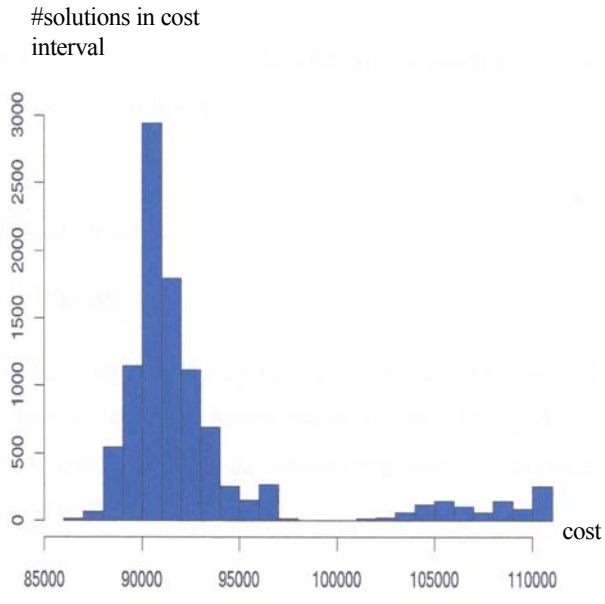

Figure 4. Costs of all possible solutions

The best configuration is (1,3,3,4) (in accordance

to the theoretical results found in Tayur 1992). In the following subsections we discuss the influence of several adjustable parameters of the ACO algorithm on the quality of the solution and on the speed of convergence. The parameters and their meaning have already be shown in table 2. We will not present all used formulae where these parameters are needed. We refer the interested reader to the original literature e.g. in Dorigo and Gambardella 1997, since we stick to the denotation of the parameters there. Curves display moving averages of mean costs from 20 runs over 700 generations.

5.1 Weight of pheromone information α

Figure 5 shows the mean quality of each generation for different values of α (in the following referred to as 'generation mean'), while Figure 6 shows the mean quality of the best solution found up to a certain generation (in the following referred to as 'global best'). Low values of α (α = 0; 0.1) let the pheromone information nearly be ignored so that mainly the heuristic information and a certain amount of randomness is used. With this settings the mean quality of the found solutions does not improve. However, both very good and very bad solutions are found, resulting only in a poor or medium 'generation mean' but in a very good 'global best' solution. Medium values of α (α = 0.5; 0.9; 1.0) lead to rapid improvement of the mean generation quality. The higher α the earlier the improvement stops at a suboptimal solution, reaching only a poor 'global best'. The highest value α = 5 shows the fastest convergence towards a good generation mean. However the algorithms gets stuck very fast and finds only a poor 'global best' solution. A good compromise can be achieved by using values between 0.1 and 0.5. This configuration shows the fastest improvement of the 'global best' (a near optimal solution at approx. 250 generations), and with α = 0.1 the solution converges against the optimum.

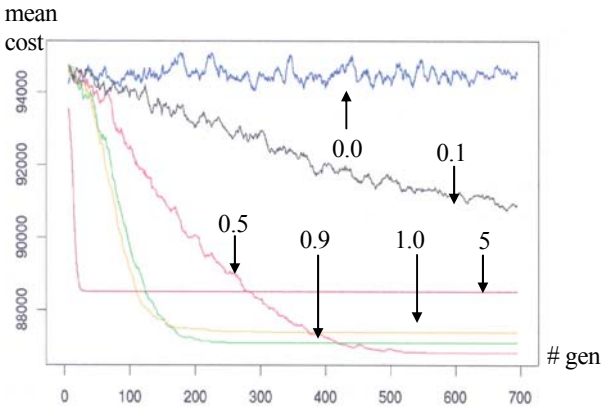

Figure 5. Mean costs of best solution in one generation dependent on α

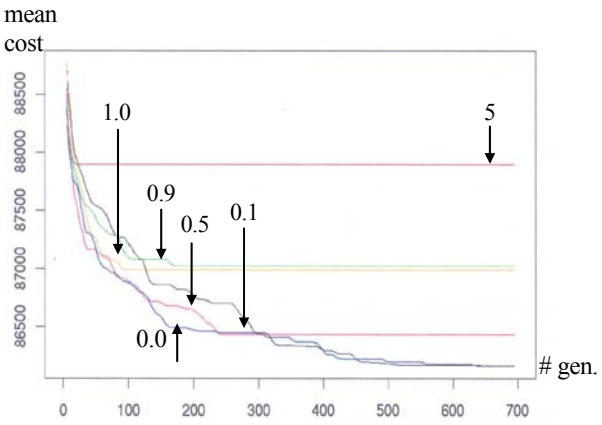

Figure 6. Mean costs of best solution of all generations dependent on α

5.2 Pheromone persistence ρ

ρ determines the percentage of pheromone that is left after the evaporation step. The higher the value of ρ the slower pheromone evaporates. Figure 7 shows that for all values of ρ a fast improvement of the best solution is achieved. But only for higher values of ρ the convergence last long enough to reach the optimum. This is a clear hint that the usage of the knowledge of predecessors contributes to the success of the optimisation.

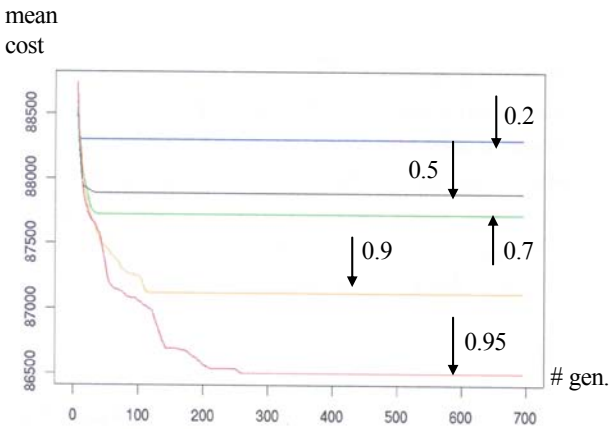

Figure 7. Mean costs of best solution in all generations dependent on ρ

5.3 Number of ants *N*

Not surprisingly the more ants are searching the better is the found solution, see figure 8. On the negative side more ants need more (time-consuming) model

evaluations.

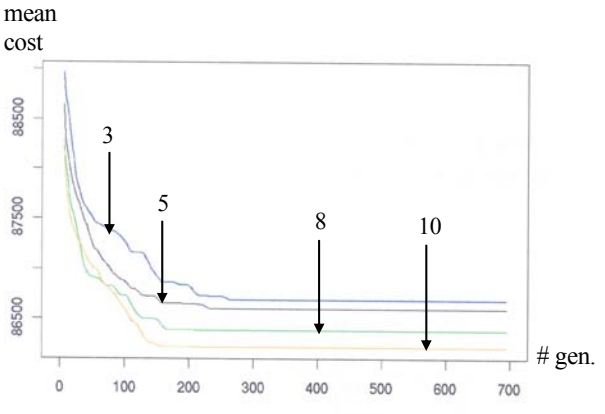

Figure 8. Mean costs of best solution in all generations dependent on number of ants

Note however that not only the quality of the best solution is better with more ants, but also the convergence towards this solution is faster. Thus it seems more appropriate to use more ants and lesser iterations instead of letting few ants search over a greater number of generations.

5.4 Weight of heuristic information β

Parameter β determines the influence of the heuristic information on the decisions of the ants. In our example the heuristic information mainly consists of the knowledge that more buffer space induces more costs. This means that with a higher value of β the ants tend to avoid Kanban allocations which use a relatively high amount of buffer space.

In Figure 9 it can be observed that the mean quality of the found Kanban allocation is better for low values of β.

This is true for the found Kanban allocation in early generations as well as in late generations. Therefore it seems reasonable to use low values for β, since the heuristic knowledge usually is only incomplete and probably neglects some more complex connections between other factors.

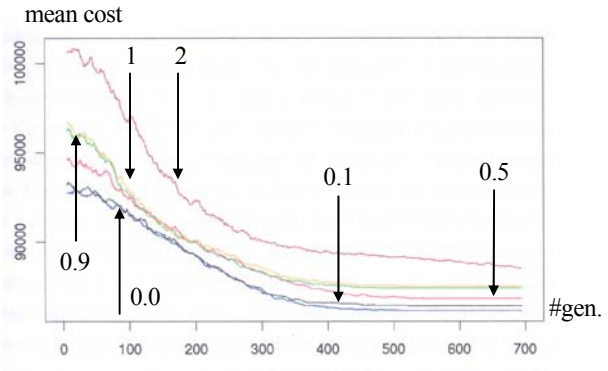

Figure 9. Mean costs of best solution in all generations dependent on β

5.5 Weight of heuristic information Q

Parameter Q determines the influence of the best ant in one generation. It determines how many additional pheromone the best ant adds to the found path.

Figure 10 shows how the quality of the best found solution evolves over the generations, depending on the value of Q. It can be seen that the value of Q should be well balanced to obtain optimal performance of the ant algorithm. Both high and low values result in approximately the same found quality of the solution, only a medium value (Q=5) lets the algorithm perform significantly better.

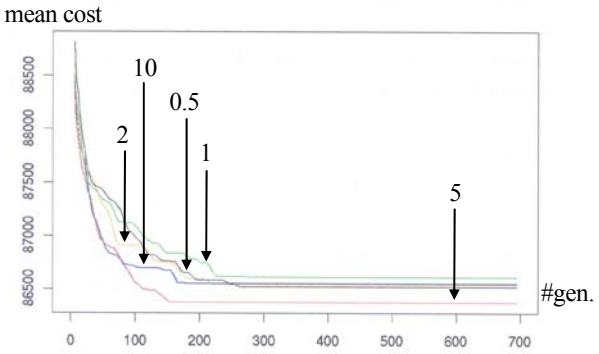

Figure 10. Mean costs of best solution in all generations dependent on Q

6. CONCLUSION

In this work we studied the applicability of the ACO algorithm and its parameters on the Kanban allocation problem. ACO shares a difficulty with most more sophisticated heuristic optimisation algorithms, that is to determine the right setting of the numerous parameters of the algorithm in order to work properly on the actual problem (Grefenstette, 1986). In this work we identified the impact of some parameters on the result. The good settings for ρ indicate that the pheromone should not evaporate too fast. The interpretation of this is, that the effect of learning from previous generations is important for a successful optimisation. However, also the heuristic information, that is the influence of factual knowledge, cannot be neglected but also should not be overestimated. The parameter studies of α and β showed that a good balance between heuristic and pheromone information produces best results. The influence of the best ant in one generation can improve the performance of the algorithm, however it is difficult to choose the right value for the parameter Q. Regarding the number of ants to be used it can be stated that a higher number of indi-

viduals achieve a higher quality of the solution in a corporate effort. Even if we consider that a higher number of ants result in a higher number of model evaluations (= simulation runs) per generation, we observe that the higher convergence of the solution against the optimum usually compensates for this. In future work, dynamic parameter adaptation could overcome the problem of having to find good parameter setting at the beginning.

REFERENCES

Becker, M. and Szczerbicka, H. (2004), Ant colony optimisation of buffer size in manufacturing, *Proceedings of the Fifth Asia Pacific Industrial Engineering and Management Systems Conference APIEMS 2004*, Surfers Paradise, Australia, 21.4.1-21.4.12

Dorigo, M. (1992), Optimization, learning and natural algorithms, Ph. D. thesis, Politecnico di Milano, Italy.

Dorigo, M. and Gambardella (1997), L., Ant colony system: A cooperative learning approach to the travelling salesman problem, *IEEE Transactions on Evolutionary Computation*, **1**(1).

Dorigo, M., Maniezzo, V., and Colorni (1996), A., The ant system: Optimization by a colony of cooperating agents, *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, **26**(1).

Grefenstette, J. (1986), Optimization of control parameters for genetic algorithms, *IEEE Trans. Syst. Man Cybern*, **16**(1), 122-128.

Kumar, R., Tiwari, M., and Shankar, R. (2003), Scheduling of flexible manufacturing systems: an ant colony optimisation approach, *Proceedings of the I MECH E Part B Journal of Engineering Manufactur*, **217**(10).

Marsan, M., Balbo, G., Conte, G., Donatelli, S., and Franceschinis, G. (1995), Modelling with Generalized Stochastic Petri Nets, Chichester England: John Wiley and Sons.

Price, W., Gravel, M., and Gagne, C. (2002), Scheduling casting operations using ant optimisation, In INFORMS Conference on OR/MS Practice, Analysing and Enhancing the Extended Enterprise, Montreal.

Sugimori, Y., Kusunoki, K., Cho, F., and Uchikawa, S. (1977), Toyota production system and kanban system materialization of just-in-time and respect-for-human system, *International Journal of Production Research*, **15**(6), 553-564.

Szczerbicka, H., Syrjakow, M., and Becker, M. (1998), Genetic algorithms, a tool for modelling, simulation and optimization of complex systems, *Cybernetics and Systems: An International Journal*, Special Issue: Intelligent modelling and simulation for complex systems **II** (7), 639-660.

Tayur, S.~R. (1992), Properties of serial kanban systems, *Queueing Systems*, **12**, 297-318.

Wolpert, D. and W.~Macready (1997), No free lunch theorems for optimisation, *IEEE Transactions on Evolutionary Computation*, **1**(1), 67-82.