A Two Phase Optimisation Method for Petri Net Models of Manufacturing Systems

A. Zimmermann (azi@cs.tu-berlin.de) *
Techn. Univ. Berlin, Franklinstr. 28/29, Sekr. FR 2-2, 10587 Berlin Germany

D. Rodriguez (dierodri@posta.unizar.es) † and M. Silva (silva@posta.unizar.es)
Univ. of Zaragoza, c/Maria de Luna 3, Zaragoza 50015 Spain

Abstract.
Optimisation is a key issue in the design of large manufacturing systems. An adequate modelling formalism to express the intricate interleaving of competition and cooperation relationships is needed first. Moreover, robust and efficient optimisation techniques are necessary. This paper presents an integrated tool for the automated optimisation of DEDS, with application to manufacturing systems. After a very quick overview of optimisation problems in Manufacturing Systems, it presents the integration of two existing tools for the modelling and evaluation with Petri nets and a general-purpose optimisation package based on Simulated Annealing. The consideration of a cache and a two phase technique for optimisation allows to speed-up the optimisation by a factor of about 35. During the first preoptimisation phase, a rough approximation of the optimal parameter set is computed based on performance bounds. Two application examples show the benefits of the proposed technique.

Keywords: Manufacturing systems, Modelling, Optimisation, Petri nets, Simulated Annealing

1. Introduction

The design of modern manufacturing systems is a complex task. High investments necessitate to make sure that the planned system will fulfil the requirements. Methods and computer tools for the modelling, performance evaluation and optimisation of manufacturing systems are therefore important.

Discrete event dynamic systems (DEDS) correspond to a view of systems where the state space is discrete (i.e. states are countable) and state changes are driven by (external or internal) events. The DEDS view of systems currently gains importance due to the development of computer based technologies. Manufacturing systems design and

* This work was performed while A. Zimmermann was visiting the University of Zaragoza

† This work has been partially developed within the project TAP. 98-0679 of the Spanish CICYT

operation is one of the technological fields where the DEDS view is widely used (Silva and Teruel, 1997).

The complexity of the behaviour of DEDS requires formal means for their modelling. In this paper we consider Petri nets (PN) for this task. As it is required for manufacturing systems, this allows to model systems with intricate interleaving of cooperation and competition, thanks to the ability of nets to model conflicts and synchronisations. Combined with appropriate interpreted extensions, PNs lead to different formalisms useful in the different phases of the life-cycle of the system under design or operation. Petri net performance models (PNPM) are obtained through interpreted extension of autonomous models in which a time duration is associated with transitions, and routing policies are defined to solve conflicts. The complex interleaving of choices and synchronisations in manufacturing systems may lead to systems where paradoxical behaviours are exhibited. For example, increasing the number of resources (i.e. tokens in the net model) can lead to a dead-locked system, and replacing a machine for a faster one can decrease the global productivity. It is thus clear that formal techniques and computer tools are required for their design and optimisation. The main contribution of this paper is methodological: a two phase optimisation strategy leading to reasonable improvement of efficiency.

After some generalities on optimisation in manufacturing systems (Section 2), two case studies (an assembly line and a flexible manufacturing cell) are introduced (Section 3). Single phase optimisation, improved by the adition of a cache (Section 4) and two phase optimisation (Section 5), introduce a technichal and a methodological contribution. Section 6 present some concluding remarks.

2. On Design Optimisation in Manufacturing

Design problems for manufacturing systems usually involve the selection of one out of several options (e.g. a machine or material handling system selection), or a dimension. Numerical values can be discrete (e.g. the size of a buffer) or continuous (e.g. a production mix). In practice, mixtures of these problems have to be considered in an optimisation. Typical optimisations deal with complex non-linear evaluative models in high-dimensional search spaces.

The optimisation problems arising for manufacturing system design can be solved using *evaluative* techniques, leading to some iterative computations, or *generative* techniques where a solution is obtained directly given a set of criteria and dynamic constraints. Among the latter are problems that allow to fit well-known mathematical programming templates like linear- (integer, mixed, ...) programming problems. Unfortunately those efficient algorithms are not applicable for more complex models. Evaluative techniques only require an algorithm that computes the value of an optimisation function from a parameterised model. The optimisation algorithm iteratively generates new parameter sets and controls the search for the optimum.

Discrete optimisation problems in Manufacturing Systems are usually NP-hard. For real-life problems it is not possible to analyse the full parameter space. Therefore search techniques have been developed that in general do not guarantee to find the global optimum, but often lead to a "good" (or just "acceptable") solution. Modern optimisation techniques approach the problem through some *meta-heuristics*, e.g. *tabu search* (Reeves, 1993), *genetic algorithms* (Reeves, 1993; Gen and Cheng, 1997), and *simulated annealing* (Aarts and Korst, 1989; Ingber, 1996).

In this paper we adopt Simulated Annealing as an evaluative metaheuristic, but Tabu Search could be used instead. First, in a one phase optimisation the simulated annealing toolkit ASA (Ingber, 1989; Ingber, 1996) is used. In order to avoid costly recalculations of profit functions for similar parameter sets, a kind of *cache* is implemented in the interface between ASA and the modelling and evaluation algorithm. In several examples this allows to improve the computational time by almost one order of magnitude. In order to reduce the computational cost further without losing too much on the result quality, a second optimisation approach is designed in two phases. During the first preoptimisation phase the profit function is computed by a very fast approximation technique based on upper and lower performance bounds. The result of the preoptimisation is used as the starting point of the second fine-grain optimisation phase.

The necessary performance measures can be obtained by *analytical* techniques based on either exact or approximate computations (e.g. product form solutions, if possible), *numerical* techniques (Markov-chain, brute force or net-driven generated) or *simulation* techniques. In most practical cases, expensive model evaluations are required. A broad perspective of PNPM performance evaluation techniques is contained in (Balbo and Silva (Eds.), 1998). In this paper an approximate evaluation technique based on performance bounds is used (Section 5as well as the simulation module of the software package TimeNET (Zimmermann et al., 2000).

Manufacturing systems are set up in order to make profit from producing and selling parts. A *profit function* has to be specified and later maximised by the optimisation. Typical profit functions consider the money earned from selling finished parts minus the costs arising in the production process. The price of raw parts, the money spent for work-in-process, machine and transport systems amortisation, and utilisation dependent costs are examples. All of them can be determined by a performance analysis of a model. The complexity of the profit function depends on the needs of the modeller and has to include every significant influence. More complex functions could e.g. capture human factors and costs as well.

3. Case modelling and their rationale

In this section we will consider two examples of Manufacturing Systems that involve three different types of design problems. These concern the determination of the Production Policy, the Buffer Dimensioning and the Machine Selection.

3.1. Example 1 – Assembly Line

An assembly line with five machines is considered in this example. Three different parts A, B and C are assembled for one final product. Customer demands and waiting times are also considered, and one of the optimisation goals is to find the *best production policy* out of three classical manufacturing control strategies ("push", "on demand" and "kanban"). Under the "push" strategy, customer waiting time is decreased by producing parts until the buffers are full, accepting a high work in process. Following the "on demand" (or "pull") strategy, production starts contrary to the first approach only after a customer demand. The "kanban" strategy allows to control work in process for production stages with the number of available kanban cards.

The second important parameter of the planned system is the number of parts (or pallets) in the system. This is controlled by *dimensioning the buffers* or by selecting the number of kanban cards. Finally, a *machine selection* problem is considered. We assume that there are three different options for machine 1. Each of them has a different processing speed, and a faster machine is more expensive.

Figure 1 shows the Petri net model of the system with *push strategy.* Machines are modeled by a resource place with the name of the machine, e.g. `M1`, and a sequence of an immediate transition, an operation place (like `m1A`), and a transition (like `M1A`). There are nine intermediate buffers (named `B1`...`B9`), which all have a capacity of B parts. The D customers are modelled with the upper right part. Transition `ok` fires when an order arrives and a complete part is available in this buffer.

Figure 2 shows the model of the system with "on demand" strategy. Every order starts the production of a new part by adding tokens to places `inA`, `inB`, and `inC`.

The model of the assembly system with kanban strategy is shown in figure 3. There are now two stages of assembly operations, which are connected by a central immediate transition. The operation is controlled by kanban cards. For the first stage their number is given by $K1$ and for the second by $K2$, both being parameters of the optimisation.

The profit function to be maximised by the optimisation computes the profit per day. It considers profit for selling parts, which decreases if the time from order until delivery is higher. The costs include work in process, costs for machine 1 depending on the machine selection, the buffer sizes, and an estimation of constant costs. A more detailed definition is omitted here. The formula of this profit function can be expressed like this:

$$\text{Profit} = \frac{108000\, T_{ok}}{1 + \frac{M_{wait}}{200\, T_{Dem}}} - 30(K1 + K2) - \frac{500}{\text{M1Delay}} - 1000$$

where T_t denotes the throughput of transition t, and M_p the mean number of tokens in place p. K1, K2 and M1Delay are the changing parameters of the model.

3.2. Example 2 – Flexible Manufacturing Cell

A flexible manufacturing cell (see figure 4) is used as a second example. The robot takes raw parts from the input buffer and places them on pallets at the loading station. The transport system includes conveyors and automated guided vehicles.

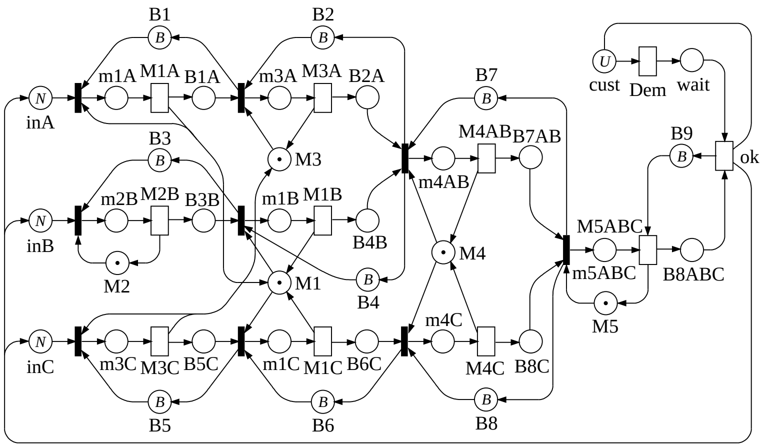

Figure 1. Assembly line with five machines: push strategy

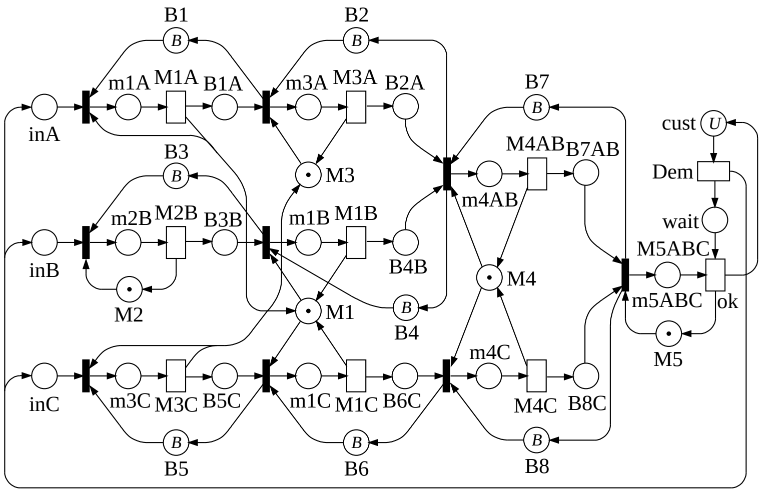

Figure 2. Assembly line with "on demand" strategy

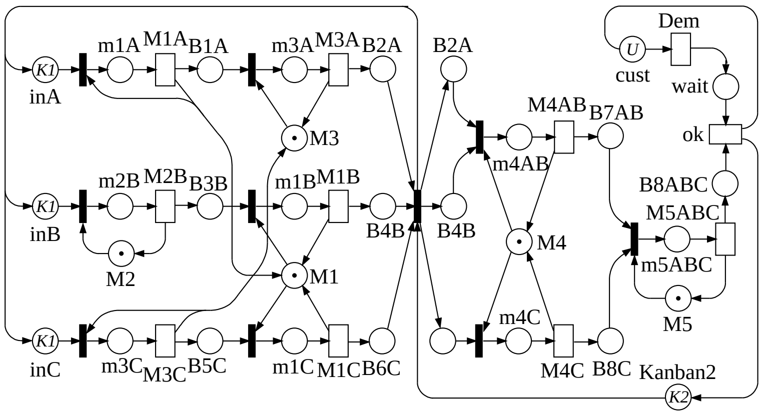

Figure 3. Assembly line with kanban strategy

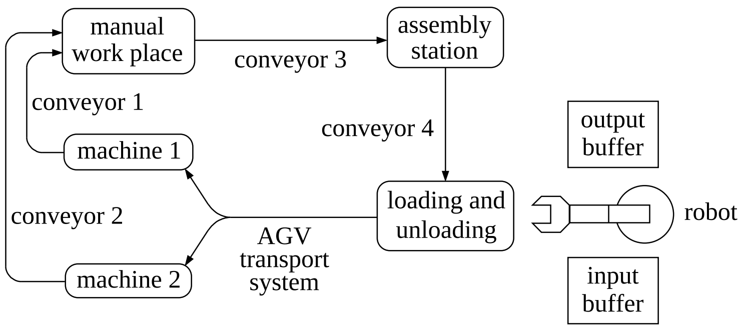

Figure 4. Flexible manufacturing cell

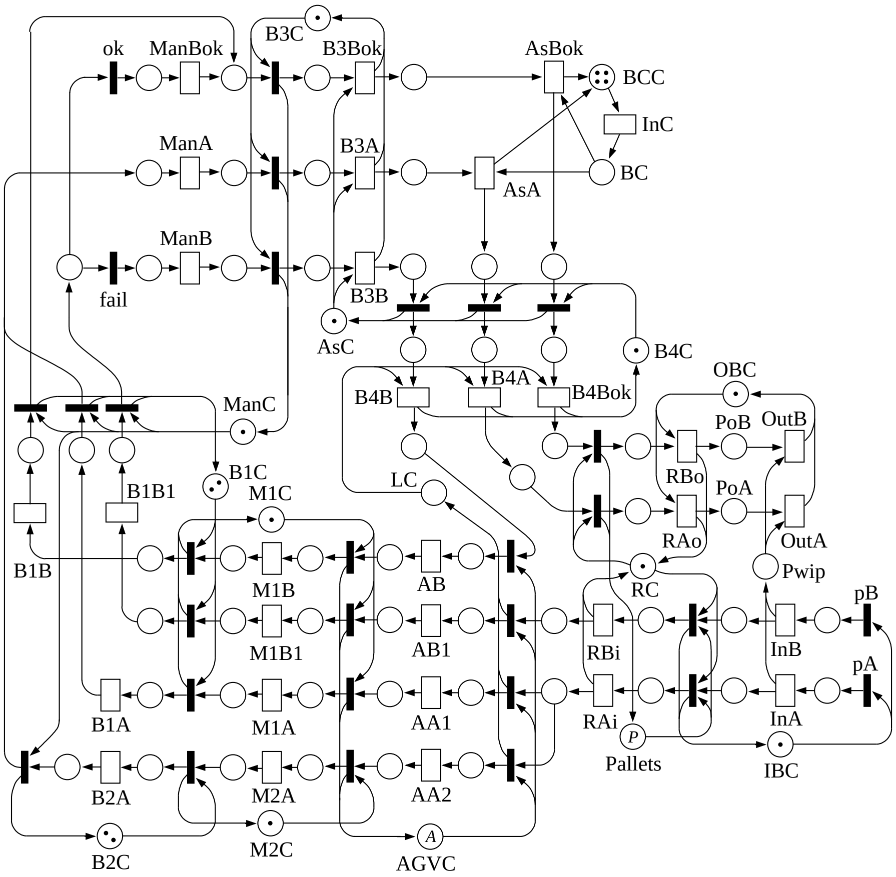

Figure 5. Petri net model of flexible manufacturing cell

Two types of products A and B have to be produced. Parts of type A can first be processed by one of the two machines. A manual operation and an assembly of an additional part have to follow, before the product is finished. B-type parts are first processed by machine 1. Afterwards they are tested at the manual work place. Parts that have been correctly processed are transported to the assembly station. After an assembly operation the product is finished. However, 5% of the parts have to be reworked at machine 1, which is detected at the manual work place.

A Petri net model for the example is shown in figure 5. The robot at the loading and unloading station is described with the transitions `RAi`, `RBi`, `RAo`, and `RBo` as well as the surrounding places and immediate transitions. Places whose names end with `C` ensure that the capacity of buffers and machines is not exceeded. `M1` and `M2` model the two machines, `Man` the manual work place and `As` the assembly station. Whether a part of type B has been correctly processed is decided by the firing of the immediate transitions `ok` and `fail`. The four conveyors act as intermediate buffers, their names begin with a `B`. The number of AGV vehicles is set by the model parameter A, and P defines the number of pallets.

During the optimisation, the number of pallets and of AGVs can be changed. More of them may lead to higher throughput, but increases work in process and costs. Different production possibilities for parts of type A are considered by adjusting the probablity whith which they are processed by machine 1 or 2. Finally, the production mix of the two products can changed.

The *profit function* for this example considers the type-dependent amount of money earned by selling one of the parts. Work in process as well as costs associated with the number of pallets and AGVs are included as well.

The complete profit function including the conversion from seconds to one day is defined as follows:

$$\text{Profit} = 172800 T_{OutA} + 345600 T_{OutB} - 10 M_{Pwip} - 2000 - 250A - 20P$$

where T_t denotes the throughput of transition t, and M_p the mean number of tokens in place p.

4. First Schema of the Optimisation Approach

The first approach is based on a *single phase*. It uses an implementation of the simulated annealing algorithm called ASA (Ingber, 1989; Ingber, 1996). This technique is known to be useful for applications where local optima lead to problems with simpler algorithms. The ASA algorithm generates new parameter sets in a region around the last accepted one. The cost (or profit) function is computed for it and checked if it is better than the last accepted one. With a certain probability, it is possible to accept a worse solution to avoid being trapped in a local optimum. The cost "temperature" T^{cost} influences the probability with which worse solutions are accepted. The parameter "temperature" T^{par} controls how far away from the last accepted parameter the new parameter value is selected. The speed of "cooling down" the temperatures (and thus the convergence and run length of the algorithm) is controlled by the constant TAnnealScale = 100 of the ASA algorithm. In the second approach, TAnnealScale is changed to speed up the convergence.

Simulation is the method used for the computation of the performance measures, because in most considered examples, analytical techniques are unable to obtain a result for the measures. The Petri net modelling and performance evaluation package TimeNET (Zimmermann et al., 2000) is used. It allows firing times of the transitions to be zero, deterministic, exponentially, or more generally distributed. Simulation runs can be executed in parallel on a cluster of workstations. Statistical techniques guarantee a reliable variance estimation and derivation of valid confidence intervals.

The computational effort for computing the value of the cost function from a parameter set is high because every cost function is computed by simulation. For the examples considered in Section 3 this can take some minutes to complete, depending on the model complexity or the confidence interval. During the optimisation, the same parameter sets (or only slightly differing ones) are often generated. It is therefore very important for efficiency to avoid re-computations. Every result is thus stored in a cache-like table, together with its corresponding parameter set by the interface procedure.

Before an optimisation can be started, the *original model* and a *configuration file* have to be specified. The configuration file contains the objective function and the parameters to consider in the optimisation. The system model is constructed using the TimeNET graphical user interface. It contains the definition of a cost or profit function as a performance measure. ASA calls its user-defined cost function, which is now the interface procedure to TimeNET, with a parameter set.

In the case that the parameter set has not been evaluated, the interface procedure prepares a parameterised model from the original Petri net model by substituting the actual parameter values in the model description. The TimeNET simulation component is called afterwards. The resulting file with the computed value of the profit or cost function is read by the interface procedure after the TimeNET process has finished. The new value is stored in the cache together with the parameter set and afterwards returned to the ASA optimiser. In the next step ASA tests whether convergence is reached and if so, exits with the final optimisation result. A new parameter set is generated otherwise and a new iteration begins.

4.1. Results for Example 1

The profit function has been evaluated for the range of possible parameter sets in order to test the optimisation algorithm. Figure 6 shows the profit function for the kanban strategy versus the numbers of kanban cards in the two assembly stages. There is one mesh of plot data for each of the three possible processing delays of machine 1. The best profit results of almost 9000 are achieved using machine 1 with delay 1, and kanban card numbers of 2 and higher. The surface plots for the other two strategies are omitted here, because the optimal profit is achieved with kanban strategy.

During the optimisation, one special type of parameter selects between the three available models. The automatic optimisation finds a profit result of 8965 by selecting the kanban strategy, $K1 = 2, K2 = 8$ and $M1Delay = 1$.

4.2. Results for Example 2

Figure 7 shows a plot of the profit function versus the number of AGV vehicles and pallets. Different meshes are drawn for three selected production mix values. The mesh forms are very similar, while producing more parts B gives the best results. The optimal profit of 6397 is gained for two AGV cars, 8 pallets, 80% parts B, and a probability of sending parts A to machine 1 of 10%.

An automatic optimisation computes a profit function of 6338, which is an error of less than 1%, for 2 AGVs, 9 pallets, 79% parts B, and probability 22% of parts A to machine 1.

Table I compares the computational cost for an optimisation using exhaustive search versus the presented ASA/TimeNET tool combination with and without cache. The efficiency of the cache of already computed results is demonstrated with the results in the last column. For each example the percentage of results which could be answered

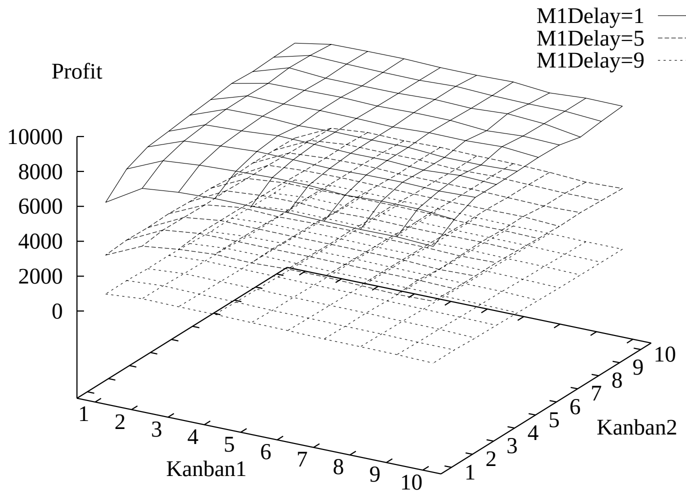

Figure 6. Profit function for assembly line with kanban strategy

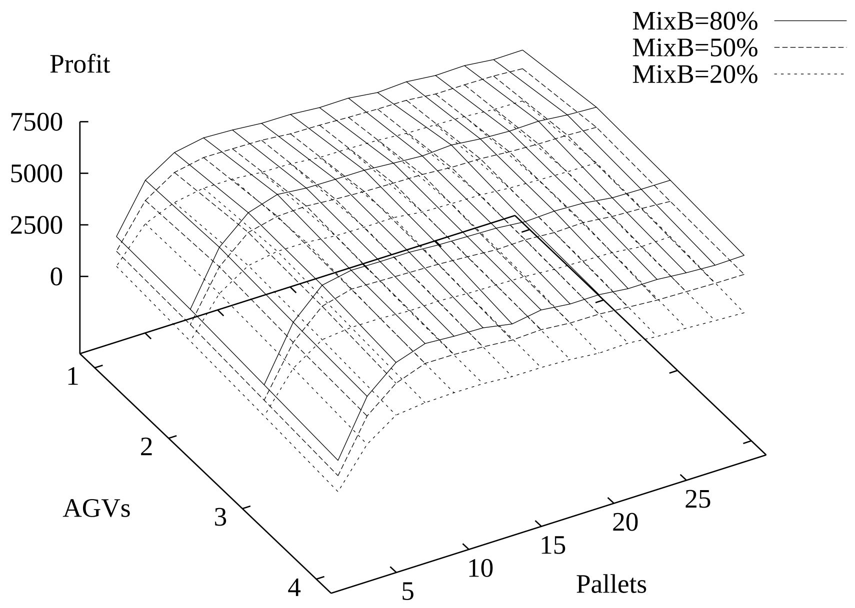

Figure 7. Profit function for flexible manufacturing cell

Table I. Reduction of computational complexity by using the cache

	Number of TimeNET calls			Cache
	Exhaustive	standard ASA	ASA with cache	hit rate
Example 1	900	387	74	81%
Example 2	144768	2008	274	86%

from the cache is given. The cache usage speeds up the simulated annealing algorithm by almost one order of magnitude in addition to the speedup due to the use of simulated annealing.

5. A two phase Optimisation Strategy

The main idea of the two-phase optimisation method is to compute a first guess of a "promising area" of the parameter search space very fast; then use this as a starting point for a more thorough optimum search (during the first preoptimisation phase). Speed is more important than accuracy during the first phase preoptimisation phase. Thus an approximation method for performance measures is adopted here that makes use of results on bounds for Petri net models.

5.1. Previous Results on Bounds

The method used to obtain the different performance measures involved in the objective function is based on the results obtained by Campos et al. (Balbo and Silva (Eds.), 1998; Campos, 1990; Campos and Silva, 1992; Campos el al., 1991), who found upper and lower bounds for the throughput of the transitions as well as the mean number of tokens in steady state. The formulas can be applied to any Petri net, but if they are restricted to the class of FRT-nets the results are more reliable than for other different classes of Petri nets. The main reason for this is that for models belonging to this class the relative routing rates of transitions at conflict can be computed directly from the net structure. Based on these results the visit ratios $\mathbf{v}[t_k]$ and average service demands $\overline{\mathbf{D}}^{(1)}[t_k]$ for all transitions t_k with respect to t_1 can be computed. For the exact definition of FRT nets as well as proofs for the validity of the bounds the reader is referred to the above mentioned references.

¿From the average service demands it is possible to compute upper bounds for the throughput of one transition t_i in the model by solving the LPP problem (Campos and Silva, 1992):

$$\begin{array}{ll} \chi_+[t_i] = \dfrac{1}{h_i} & \\ \text{with } h_i = & \min\{y \cdot \mathbf{Pre} \cdot \overline{\mathbf{D}} / y \cdot \mathbf{C} = 0,\ y \cdot M_0 = 1, y \geq 0\} \end{array} \tag{1}$$

where y is a P-semiflow and $\mathbf{C} = \mathbf{Pre} - \mathbf{Post}$ denotes the incidence matrix. The bound for all other transitions can be directly calculated from the result and the relative corresponding visit ratios.

After that the throughput lower bounds are computed:

$$\begin{aligned} \chi_-[t_1] &= \frac{1}{\sum_{t \in T} \overline{\mathbf{D}}^1[t]} \\ \chi_-[t_i] &= \chi_-[t_1] \cdot \mathbf{v}^{(1)}[t_i] \end{aligned} \tag{2}$$

A lower bound for the mean place marking (mean number of tokens in the places in steady state) can be computed using the results of the throughput lower bounds (Campos and Silva, 1992).

$$M_- = \mathbf{Pre} \cdot \overline{\mathbf{D}}^{(1)} \cdot \chi_-^{(1)}[t_1] \tag{3}$$

For the computation of the upper bound for the mean number of tokens a LPP problem has to be solved for each place in the system:

$$\begin{array}{l} M_{+}[p_i] = \min\{M / Y^T \cdot C = 0, Y^T \cdot \mathbf{e}_i = 1, Y \geq 0\} \\ M \;\; = \;\; M_{-}[p_i] + Y^T \cdot (M_0 - M_{-}) \end{array} \tag{4}$$

with

$$if \;\; i = k \text{ then } 1 \text{ else } 0 \tag{5}$$

where M_0 is the initial marking vector. With this approach only two LPPs must be solved to obtain the different bounds for the throughput and the mean marking of places. For the examples this is efficiently done using the software tool `lp-solve`.

5.2. The Proposed Approach

In order to compute an approximation of the throughput, a weighted sum of upper and lower bounds is computed.

$$\chi_{\simeq}[t_i] = \alpha \cdot \chi_{+}[t_i] + (1 - \alpha) \cdot \chi_{-}[t_i] \tag{6}$$

Experiences show that usually the throughput upper bound is much better (closer to the actual value) and more sensitive to parameter changes. This is however not surprising because of the "trivial" formula of the throughput lower bounds. The value of α has therefore been set to 0.9 for the examples presented later on and often results in a "reasonable" good approximation of the throughput.

A similar improvement in the context of a more general linear description of a Petri net has been presented in (Balbo and Silva (Eds.), 1998) for the marking upper bounds. For each transition t_r with only one input place p_i the following inequality holds:

$$M_{+}[p_i] \leq K \cdot \chi_{+}[t_r] \cdot S[t_r] + (K - 1) \tag{7}$$

where $K = \mathbf{Pre}(p_i, t_r)$ and $S[t_r]$ is the service time of transition t_i.

An approximated value for the mean number of tokens in the places can now be computed in the same manner as for the throughput:

$$M_{\simeq} = \beta \cdot M_{+} + (1 - \beta) \cdot M_{-} \tag{8}$$

where β is a weighting factor similar to α for the throughput approximation. For the examples considered so far, the marking bounds were in general not very close to the actual values. No observation have been made whether the upper or lower bound is systematically better. Therefore β has been set to 0.5 for the example computations, resulting in an equal importance of marking lower and upper bound.

It has already been stated that the lower bounds for the transition throughputs are usually not very exact. The marking lower bounds are computed using them and the marking upper bounds depend on the marking lower bounds. Therefore the marking bounds are also not very tight in most cases. However, we are mainly interested in a good approximation of the throughput and mean marking. An approximation of the marking bounds can then be computed by assuming that $\chi_{\simeq}$ (as computed in equation (6)) equals the correct throughput values. The bounds on the throughput that are being used in formulas (7) and (3) can then be substituted by $\chi_{\simeq}$.

This technique can be applied to the two application examples because they belong to the special class of Petri nets called FRT-nets. The results for the examples show that the improved approximation based on bounds can actually be closer to the real value; however, one should keep in mind that the upper and lower values are no bounds anymore.

5.3. Results for Example 1

The profit function for the range of possible parameter sets has been evaluated in order to test the optimisation algorithm. Because of the dimension of the parameter space, only selected parts can be shown in the plots. Figure 8 shows the profit function for the *push strategy* versus machine selection and buffer size. The shape of the simulated curve can also be found in the approximated ones. Improving the bounds by approximation makes the difference to the simulated results even smaller.

Table II. Optimal results for different evaluation techniques

Parameter	Simulation	Approximation	Improved Appr.
Buffer size	2.0	1.0	1.0
Kanban cards	8.0	1.0	1.0
Machine 1 speed	1.0	1.0	1.0
Control strategy	kanban	kanban	kanban
Profit	8965	6256	7761

The optimal profit (among the parameter sets analysed here) is achieved for the kanban strategy and machine 1 selected with delay 1. Keeping them fixed, Figure 9 shows the influence of the remaining two parameters of the kanban model, namely the numbers of kanban cards in the two production stages. This plot is shown here to visualise the limits of approximating the performance measures using bounds. While again the improved approximation is much closer to the simulated

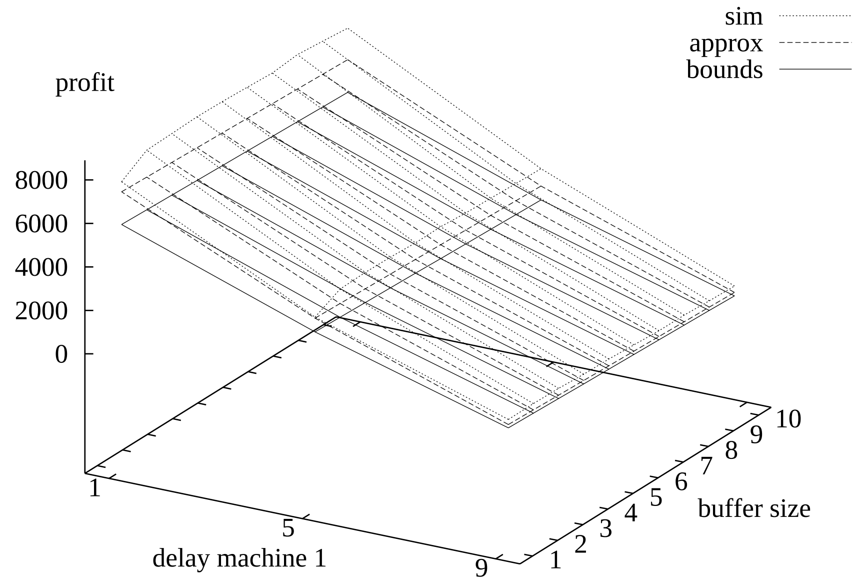

Figure 8. Profit function for assembly line with push strategy

results, none of both bounds based functions capture the influence of kanban cards on the system performance. The reason for this is that as long as there are enough kanban cards available to reach the throughput upper bound, the bounds calculation is not able to realize the buffering function of additional cards. More cards therefore lead only to a higher associated cost, decreasing the overall profit. However, the best approximated solution (both number of kanban cards 1, machine 1 speed 1, kanban strategy) is not too far away from the best simulated one (2 resp. 8 kanban cards, machine 1 speed 1, kanban strategy). The experiences from the models analysed so far lead to the conclusion that in this example, the approximation works well for structural changes, while this might not be the case for "fine tuning" parameters like buffer sizes.

As an exercise Table II shows the optimal results of a "full" exploration in the parameter space for simulation, bounds approximation, and improved approximation. As long as no other local optimum results in profit function values of similar quality like in this case, the "optimal" approximated parameter values should be a very good starting point for the later fine-grain optimisation.

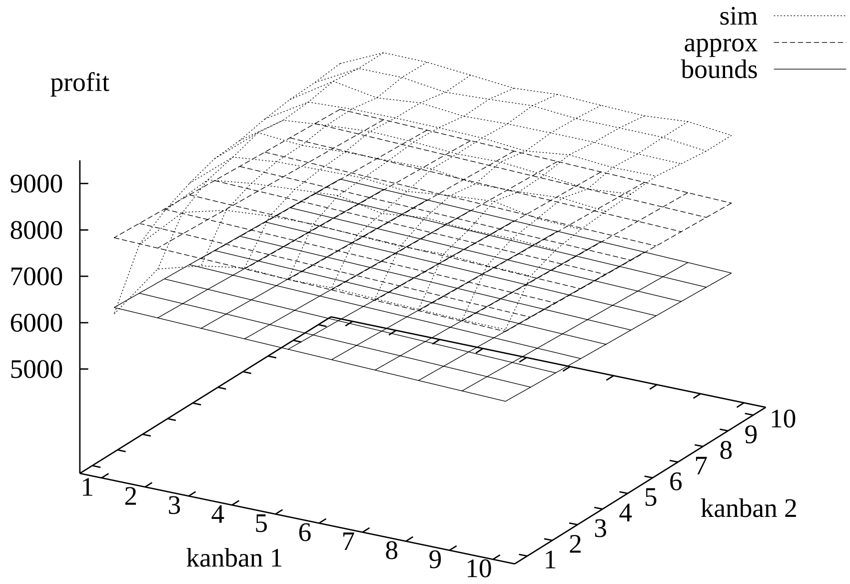

Figure 9. Profit function approximation for assembly line with kanban strategy

For a comparison of the results and run times first the application example was optimised using the one-phase ASA algorithm with cache (see column "ASA (cache)" in Table III).

Afterwards the preoptimisation was started. It took two minutes to complete, while for 388 overall cost function calls 59 different ones resulted in an approximation. One approximation typically took between one and two seconds to complete. The "optimal" parameter set is Kanban1 = 1, Kanban2 = 1, machine 1 delay = 1, and kanban strategy, giving a profit value of 7761 (column "Phase I " in Table III).

The second optimisation step is started with the final results of the preoptimisation phase as starting values. Table III shows results (best found cost and corresponding parameter set, result value from simulation) and computation times for these different selections of TAnnealScale between 100 (the default only for Phase I) and 10, 5 and 1 (for Phase II), influencing the cooling factor speed. The result in brackets is computed by approximation. It seems to be possible to make the cooling process faster without loosing much of the result quality. However, there is of course a limit on how fast the optimisation can be made without a significant loss on the result quality. For

Table III. Tradeoff between speedup and result quality

	ASA (cache)	Phase I	Phase II		
TAnnealScale	100	100	10	5	1
Kanban1	4	1	4	4	1
Kanban2	6	1	1	7	1
Machine 1 speed	1	1	1	1	1
Control Strategy	kanban	kanban	kanban	kanban	kanban
Profit	8911	6115 (7761)	8702	8649	6115
Time (minutes)	64	2	14	7	2
Speedup (Ph. I+II)			4.0	7.1	16.0

TAnnealScale = 1 the simulated annealing process does not move away from the initial solution, resulting in a clearly non-optimal parameter set.

5.4. Results for Example 2

In the case of this example the influence of the work in process on the profit function is not very important, and therefore the approximate bounds computation does not change the results significantly. In the following thus only the "correct" bounds computation is used for the approximate calculation of the performance measures.

To check the quality of the automatic optimisation algorithm, the profit function has been computed for a selected number of parameter sets that systematically cover the whole range of possible solutions. Figure 10 shows a plot of the profit function versus the number of AGV vehicles and the production mix in terms of parts B. Two different meshes are drawn for results obtained by simulation and approximation based on bounds. For the picture, the number of pallets is set to 12 and the probability of parts A to be sent to machine 1 is set to 30%. The actual values are in the range for which the optimal values are achieved with simulation. The shape of the functions is quite similar – the main difference is that the approximated profit is higher for one AGV, while simulation achieves the optimal result with two AGV.

Table IV shows a comparison of some results for simulation and approximation using bounds for this example. The bounds calculation results of the profit function are quite different from the simulated ones. More important, however, is the very good approximation of where the global maximum can be found. The best simulated profit of 6397 for a systematic "full" search is achieved for 2 AGVs, 8 pallets, 80% parts

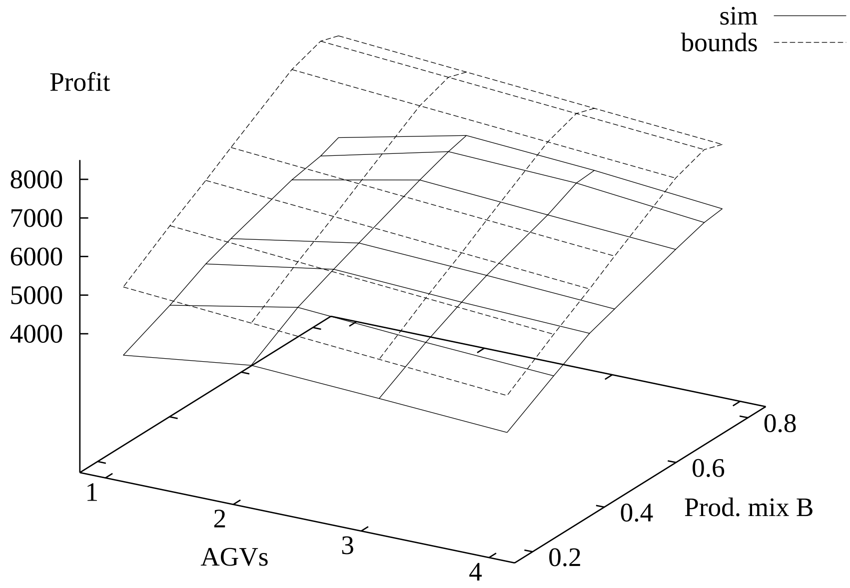

Figure 10. Profit function for the flexible manufacturing cell

Table IV. Results comparison for simulation and approximation

Parameter	Simulation	Approximation
AGV vehicles	2	1
Pallets	8	6
Part B prod. mix	80%	80%
Part A to machine 1	10%	10%
Profit	6397	8717

of type B, and a probability of 10% for parts of type A to be sent to machine 1.

Although the approximately found parameter set is a neighbouring solution to the one found by simulation, the simulated cost function value of 5464 for the best approximated parameter set reaches only 85% of the optimum of the simulated parameter sets. This underlines the importance of the second fine-grain optimisation step, although the

Table V. Tradeoff between speedup and result quality

	ASA (cache)	Phase I	Phase II		
TAnnealScale	100	100	10	5	1
AGV vehicles	2	1	2	1	1
Pallets	9	5	9	10	6
Part B prod. mix	79%	78%	78%	74%	78%
Part A to machine 1	22%	10%	16%	10%	10%
Profit	6338	5069 (8841)	6326	5575	5388
Time (minutes)	802	2	89	38	14
Speedup (Ph. I+II)			8.8	20.0	50.0

parameter sets are very close. For a comparison of the results and run times the same set of experiments were carried out for this example as for the first one.

Table V shows results and computation times for the standard setup (ASA with cache), and the two phases with different selections of TAnnealScale. Profit results are computed by simulation from the parameter sets, while the result in brackets is the bounds based approximation result.

The second optimisation step is started with the final results of the preoptimisation phase as starting values. For TAnnealScale values of 5 and 1, the fine-grain optimisation was not able to find the better solution with AGV vehicles equal to two. Again, this shows that a significant speedup can be achieved, but the heuristic choice of the faster temperature scheme is important. For both models a speedup factor of at least four could be achieved while a near-optimal solution is still found. There is no guarantee for a better solution if the computational cost is higher; only the probability increases. In addition to that, the reason for the difference of less than 1% could also be the simulation that computes the results.

6. Conclusion

The optimisation of complex systems is computationally expensive, even when iterative meta heuristics - like simulated annealing - are applied. This is due to the cost of the ddynamic model evaluation e.g. by simulation. In this paper two techniques are presented. The first one allows a speedup of 7 while obtaining an “optimal” solution. The two phase method adds another speedup of 5, by first obtainig a “near optimal” solution using a fast and approximated evaluation of the model of the plant. Both approaches together allow a speedup factor of more than 35 for the examples considered so far. The paper demonstrates the benefits of the techniques with two manufacturing system examples, that are modelled with stochastic Petri nets.

References

Aarts, E. and J. Korst: 1989, *Simulated Annealing and Bolzmann Machines*. Wiley.

Ajmone Marsan, M., G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis: 1995, *Modelling with Generalized Stochastic Petri Nets*, Series in parallel computing. John Wiley and Sons.

Balbo, G. and M. Silva. (Eds.): 1998, *Performance Models for Discrete Event Systems with Synchronisations: Formalisms and Analysis Techniques*. Universidad de Zaragoza, Spain. MATCH Advanced School.

Campos, J.: 1990, 'Performance Bounds for Synchronized Queueing Networks'. Ph. D. Dissertation, Universidad de Zaragoza.

Campos, J., G. Chiola, and M. Silva: 1991, 'Properties and Performance Bounds for closed Free Choice synchronized Monoclass Networks '. *IEEE Transactions on Automatic Control (special issue on Multidimensional Queueing Networks)* **36(12)**, 1368–1382.

Campos, J. and M. Silva: 1992, 'Structural Techniques and Performance Bounds of Stochastic Petri Net models'. In: G. Rozenberg (ed.): *Advances in Petri Nets 1992*, Vol. 609 of *Lecture Notes in Computer Science*. Springer Verlag, pp. 352–391.

Ciardo, G., R. German, and C. Lindemann: 1994, 'A Characterization of the Stochastic Process Underlying a Stochastic Petri Net'. *IEEE Transactions on Software Engineering* **20**, 506–515.

Dallery, Y., Z. Liu, and D. Towsley: 1994, 'Equivalence, Reversibility, Symmetry and Concavity Properties in Fork-Join Queueing Networks with Blocking'. *Journal of the ACM* **41**, 903–942.

Gen, M. and R. Cheng: 1997, *Genetic Algorithms and Engineering Design*. Wiley.

Gordon, W. J. and G. F. Newell: 1967, 'Closed Queueing Systems with Exponential Servers'. *Operations Research* **15**, 254–265.

Ingber, L.: 1989, 'Very fast simulated re-annealing'. *Journal of Mathematical Computer Modelling* **12**(8), 967–973.

Ingber, L.: 1996, 'Adaptive simulated annealing (ASA): Lessons learned'. *Journal of Control and Cybernetics* **25**(1), 33–54.

Molloy, M. K.: 1982, 'Performance Analysis Using Stochastic Petri Nets'. *IEEE Transactions on Computers* **31**(9), 913–917.

Reeves, C. L.: 1993, *Modern Heuristic techniques for Combinatorial Problems*. Wiley.

Silva, M. and E. Teruel: 1997, 'Petri Nets for the Design and Operation of Manufacturing Systems'. *European Journal of Control* **3**(3), 182–199.

Zimmermann, A., J. Freiheit, R. German, and G. Hommel: 2000, 'Petri Net Modelling and Performability Evaluation with TimeNET 3.0'. In: *11th Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation*. Schaumburg, Illinois, USA, pp. 188–202. LNCS 1786.