A TWO PHASE OPTIMISATION STRATEGY FOR DEDS: APPLICATION TO A MANUFACTURING SYSTEM

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

D. Rodríguez
Univ. of Zaragoza, c/Maria de Luna 3,
Zaragoza 50015 Spain
dierodri@posta.unizar.es

M. Silva
Univ. of Zaragoza, c/Maria de Luna 3,
Zaragoza 50015 Spain
silva@posta.unizar.es

Keywords: Optimisation, DES, Simulated Annealing

Abstract Model-based optimisation can make the design of complex DEDS more efficient. The process of optimising a manufacturing system is considered in this paper as an application, where the main problem lies in the computational effort required for a series of long simulation runs. We propose a two-phase optimisation method starting with a fast preoptimisation. This first step is done by computing rough approximations based on interpolation of upper and lower bounds of the performance indexes, instead of long run simulations (or analytical computations). Petri nets are used for the modelling of these systems, enabling the application of linear programming techniques for the bounding analysis in polynomial time on the size of the model. The second phase is a post-optimisation, in which every evaluation is conducted by means of simulations. A comparison of the achieved results and computation times with the ones obtained by standard techniques shows the usefulness of the proposed approach.

1. INTRODUCTION

The optimisation of Discrete Event Dynamic Systems (DEDS) is in most of the cases *NP-hard*. In order to alleviate this computational problem some authors [1, 8] have introduced *meta heuristics* to try to solve it in a reasonable time and with reasonable accuracy.

Petri Nets (PN) are used as modelling paradigm for the dynamical system. The tool used for modelling, analysis and simulation of our systems is TimeNET [6]. In a previous paper [9], the optimisation of Petri Net Performance Models is done by combining TimeNET and a meta heuristic optimiser, ASA [7], using a simulated annealing technique.

The aim of this paper is to reduce the computational cost of the optimisation. The proposed method is divided in two phases. The first one, **Fast Preoptimisation**, tries to get a "reasonable" initial solution, at low computational cost. This phase is inspired by *Ordinal Optimisation* ideas [2]. The main idea beyond this approach is to compute an "initially good enough" solution. The first phase takes advantage of the results by computing bounds that can be obtained in polynomial time on the size of the Petri Net model. Some "roughly approximated" values of performance measures, like throughput or mean number of tokens in a place, can be obtained in a very efficient way.

The second phase, **Fine Grain Optimisation**, is started once the fast preoptimisation process has reached certain conditions. The aim of this phase is to improve the best solution found in the first one using a more accurate performance measure computation. Simulation will be the chosen technique in most of the cases because, in complex systems, analytical results are either impossible or the computational cost is excessive.

The Optimisation technique used in both phases is called Adaptive Simulated Annealing (ASA) [7]. During the preoptimisation phase the computation of bounds from a linear description of the net requires the solution of several linear programming problems (LPP). For this task an interface to the program `lp-solve` has been implemented. Throughout the final optimisation phase, TimeNET [6] has been employed for the simulations.

2. GETTING ROUGH APPROXIMATIONS

Stochastic PN are used as modelling formalism for the dynamic of the system to be optimised. For the notation of these performance models the reader is referred to the literature (e.g. [3, 4]). The computation of approximated values can be done through several techniques (e.g.

Response Time Approximation, RTA [5]), where "accurate" results have been reported at the expense of relatively high computational cost.

Opposed to this, a rough computation of performance indexes is enough for the approach presented here. A weighted sum of upper and lower bounds for those measures is taken as an approximation. For certain classes of Petri nets, efficient algorithms based on linear programming problems (LPP) exist for the bounds computation. In the following, we will denote with $\chi_+[t_i]$ ($\chi_-[t_i]$) the upper (lower) bound of the throughput of transition t_i, and with $M_+[p_i]$ ($M_+[p_i]$) the upper (lower) bound of the mean number of tokens in place p_i in steady state.

For the details of this computation the reader is referred to the literature [4]. The computational effort is linear in the size of the net structure, and efficient LPP solvers are freely available.

In order to compute an approximation of the throughput and mean marking, the following weighted sums of upper and lower bounds are computed.

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

Numerical experiences show that usually the throughput upper bound is much better (nearer to the actual value). This is however not surprising from the "trivial" formula of the throughput lower bounds. The value of α has therefore been set here to 0.9 for the examples presented later on and often results in a reasonable approximation of the throughput.

An approximated value for the mean number of tokens in the places can now be computed analogously.

$$M_{\simeq} = \beta \cdot M_+ + (1 - \beta) \cdot M_- \tag{2}$$

For the examples considered so far, the marking bounds where in general not very close to the actual values. No observation could be made whether the upper or lower bound is systematically better. Here β has been set to 0.5.

The lower bounds for the transition throughputs are usually not very accurate. The marking lower bounds are computed using them and the marking upper bounds depend on the marking lower bounds. Here we are interested in an approximation and not a true bound. Estimations of the marking bounds can thus be computed by assuming that $\chi_{\simeq}$ (as computed in equation 1) equals the correct throughput values. The bounds on the throughput that are being used in the original formulas [4] can then be substituted by $\chi_{\simeq}$, leading to some "approximated" values. This leads to a better marking approximation in all cases considered so far.

3. APPLICATION EXAMPLE

In this section a manufacturing system model with its design problems is presented. Figure 1 shows a sketch of the layout of the system. The robot takes raw parts from the input buffer and places them on pallets at the loading station. Each part circulates through the cell mounted on a pallet. From the loading station parts are taken to one of the two machines by a transport system with automated guided vehicles. After being processed in one of the machines, the work pieces are transported to the manual workplace and later to the assembly station and the unloading station by one of the subsequent conveyor belts.

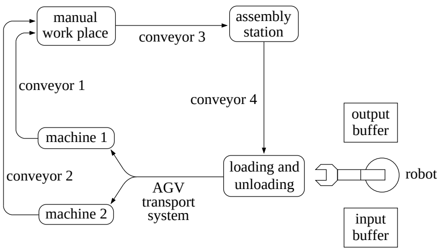

Figure 1 Flexible manufacturing cell

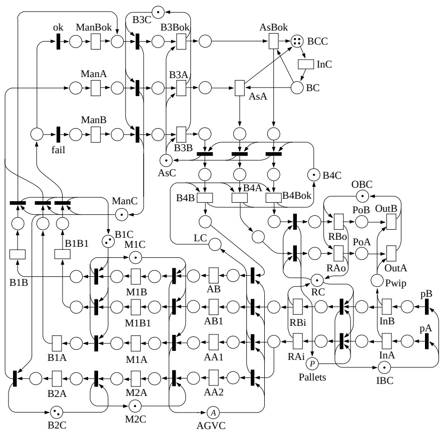

Figure 2 GSPN model of the cell

We assume that the structure of the manufacturing cell is fixed. Two products (A and B) have to be produced using it, following different production sequences.

A Petri net model for the example is shown in Figure 2. The different processing tasks (modelled by transitions) and parts in buffers (modelled by tokens and places) had to be unfolded due to the use of uncoloured nets. Transitions `InA`, `InB`, `InC`, `OutA`, and `OutB` model the cells input and output of parts. The production mix (percentage of parts of both types) is specified by the firing probabilities associated with the immediate transitions `pA` and `pB`. 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. The four conveyors act as intermediate buffers between their connected stations. Their names therefore begin with a `B` followed by the number. The number of AGV vehicles is set by the model parameter A, and P defines the number of pallets. In general, model elements with trailing `A` (`B`) refer to parts of type A (B).

The design decisions are the following: the number of pallets P available in the system, the number of vehicles of the AGV transport system A (in the range from one to four), the probability for a part of type A to be transported to machine 1 by the AGV (as opposed to transporting it to the alternative machine 2), and the production mix.

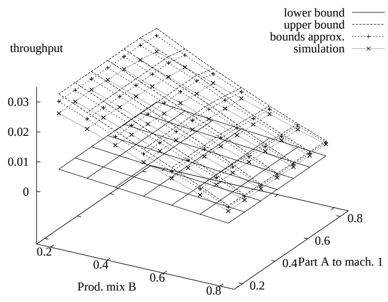

Figure 3 Throughput computed with the different methods

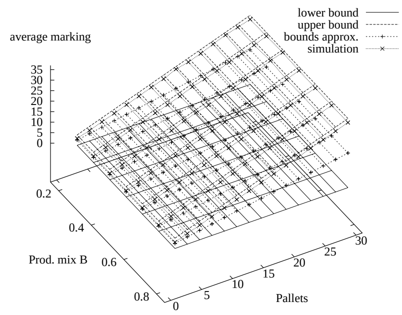

Figure 4 Quality of marking approximation

Figure 3 shows a plot of the throughput comparing the different techniques considered in this paper. The approximation used in this approach is near from the throughput real results. For an evaluation of the approximated values for average markings, Figure 4 contains plots of the values computed by simulation and approximation. The approximated values are not very close to the simulated ones. However, the similarity of the functions shows that the approximated values depend in the same way on the changing parameters as the simulated ones.

4. TWO-PHASE OPTIMISATION METHOD

The main idea is to compute an initial parameter set for the problem during a fast preoptimisation phase. Later, it is used as the starting point of the second (fine-grain) optimisation phase. The implementation uses the ASA (Adaptive Simulated Annealing) tool [7], a freely available simulated annealing program.

An important question is when to change from the preoptimisation phase to the second one. As each calculation in the preoptimisation phase only takes some seconds, the number of annealing iterations during this phase does not play such a big role. Therefore the original ASA parameters have been used. When the annealing algorithm has reached its end, the second step is started with some adjusted parameters and temperatures. It should be noted that for the annealing algorithm the duration can be more or less arbitrarily changed by choosing different temperature schemes. However, the question is then how good the found solution will be.

The second phase is a normal ASA optimisation based on the simulation optimisation, as it has been employed previously [9]. The speed of the ASA algorithm (leading to the number of parameter sets for which the model has to be analysed) depends essentially on the temperature scheme. For an acceleration of the second phase the following (heuristic) changes have been made:

- The initial cost temperature T_0^{cost} is decreased from 1 to 0.1, thus reducing the acceptance probability of worse solutions.
- The cooling speed control constant c is made smaller, reducing the value of TAnnealScale from 100 to other values (20, 10, 5, 1), resulting in a faster temperature reduction process.

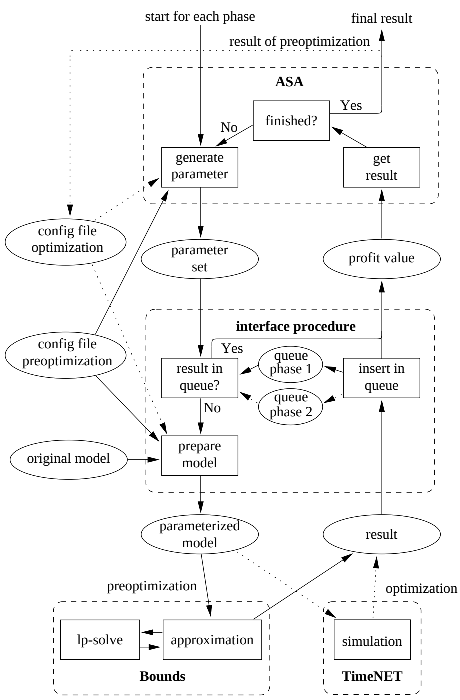

Figure 5 Interaction of ASA, bounds computation and TimeNET

Figure 5 sketches the interaction of the different program parts. To avoid re-computations, every result is stored in a cache-like table (called queue) together with its corresponding parameter set

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. ASA calls its user-defined cost function, which is now the interface procedure to the bounds approximation or TimeNET, with a parameter set. The parameters given to the interface procedure are first checked whether they have already been computed.

If not, the interface procedure prepares a parameterised model from the original Petri net model. Depending on the optimisation phase, either the approximation component or the TimeNET simulator is called afterwards. The resulting value of the profit function is stored in the queue 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.

5. APPLICATION EXAMPLE

The application example is now optimised with the proposed two-phase algorithm, and the results are compared with those obtained in [9].

The profit function per day includes the profit from selling parts, work in process, and constant costs. The complete profit function is defined as Profit $= 172800 T_{OutA} + 345600 T_{OutB} - 10 M_{Pwip} - 2000 - 250A - 20P$.

Table 1 Tradeoff between speedup and result quality

	Standard	Phase I	Phase II			
TAnnealScale	100	100	20	10	5	1
AGV vehicles	2	1	2	2	1	1
Pallets	10	5	8	9	10	6
Part B prod. mix	78%	78%	78%	78%	74%	78%
Part A to M1	44%	10%	13%	16%	10%	10%
Profit	6635	5069	6267	6326	5575	5388
Time (minutes)	802	2	135	89	38	14
Speedup (Ph. I+II)			5.8	8.8	20.0	50.0

Table 1 shows results and computation times for the original algorithm, and the two phases with different selections of TAnnealScale. Without loosing too much accuracy, the two phase method finds very good results with a speedup of about nine. About the same speedups have been achieved for other examples. For a TAnnealScale values of 5 and 1, the fine-grain optimisation was not able to find the solution with AGV vehicles equal to two. This shows that a significant speedup can be achieved, but the heuristic choice of the faster temperature scheme is important.

6. CONCLUDING REMARKS

Iterative meta heuristics optimisation procedures (as Simulated Annealing) suffer when the cost evaluation of the model for a parameter set is expensive. In [9] we introduce a “queue” in order to avoid recomputation for a pattern of parameters, allowing frequently a computational speed-up around 6-7 (order of the hit rate in the queue 85%). Here, a two phase optimisation, where an interpolation of upper and lower bounds are taken as rough approximation in the first phase, allows to make a new speed-up around 9 for similar quality in the optimal solution in the considered manufacturing system.

References

[1] Aarst,E.; Korst, J. (1989). *Simulated Annealing and Bolzmann Machines.* Wiley.

[2] Y.C. Ho, R. Sreenivas, P. Valiki, “Ordinal Optimisation of DEDS”, *J. Discrete Event Dynamic Systems* **2** 1992, pp. 61–88.

[3] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, G. Franceschinis, *Modelling with Generalized Stochastic Petri Nets*, Wiley 1995.

[4] J. Campos, G. Chiola, M. Silva, “Properties and Performance Bounds for Closed Free-Choice Syncronized Monoclass Queueing Networks”, *IEEE Trans. Automatic Control*, **36 (12)**, 1991, pp. 1368–1382.

[5] H. Jungnitz, B. Sanchez, M. Silva, “Approximate Throughput computation of Stochastic Marked Graphs”, *J. Parallel and Distributed Computing* **2** 1992, pp. 282–295.

[6] R. German, C. Kelling, A. Zimmermann, G. Hommel, “TimeNET - A Toolkit for Evaluating Non-Markovian Stochastic Petri Nets”, *J. Performance Evaluation*, **24**, 1995, pp. 69–87.

[7] L. Ingber, “Very fast simulated re-annealing”, *J. Mathematical Computer Modelling*, **12 (8)** 1989, pp. 967–973.

[8] C.L. Reeves, *Modern Heuristic techniques for Combinatorial Problems*, Wiley 1993.

[9] A. Zimmermann, D. Rodriguez, and M. Silva, “Modelling and Optimisation of Manufacturing Systems: Petri Nets and Simulated Annealing”, *Proc. European Control Conference (ECC'99)*, Karlsruhe, 1999.