TIMED PETRI NET SIMULATION AND RELATED SCHEDULING METHODS: A BRIEF COMPARISON

Gašper Mušič[(a)]

[(a)]University of Ljubljana
Faculty of Electrical Engineering
Tržaška 25, Ljubljana, Slovenia

[(a)]gasper.music@fe.uni-lj.si

ABSTRACT

The paper deals with Petri net based modelling and optimization of scheduling problems. Timed Petri net models are derived and used in conjunction with various optimization strategies. Standard Petri net scheduling approaches are applied, which include heuristic dispatching rules as well as heuristic based search through the reachability tree. As an alternative a simulation-based optimization is implemented to optimize the input sequences. Various conflict resolution strategies are used in order to compare and evaluate possible operation schedules in the modelled system. The strategies are based on predefined ordering rules, i.e. sequences of transition firings, which are changed during optimization procedure. The optimization problem is then solved by heuristic algorithms, including genetic algorithms, simulated annealing and threshold accepting. All these methods are implemented in the so called MATLAB PetriSimM toolbox which offers, among others, an implementation of an automated model building for the scheduling purposes. A number of benchmark tests is performed in order to compare various optimization strategies and to illustrate the suitability of the related approaches for solving practical problems.

Keywords: Petri nets, simulation, optimization, scheduling

1. INTRODUCTION

Recently, many research results related to planning and scheduling systems are reported in the engineering literature. This is due to increased demands for maximizing capacity utilization, minimizing throughput times, minimizing delays and work in progress, as well as necessity for meeting agreed delivery dates as close as possible.

Many reported results are based on formal, theoretically oriented approaches. The problems are often idealized in order to suit the constraints imposed by the chosen optimization technique and they have to ignore many practical constraints in order to solve scheduling problems efficiently. Classical mathematical programming approaches are computationally demanding and often cannot achieve feasible solutions to practical problems (Jain and Meeran 1999). This limits the applicability of classical scheduling methods in real cases found in the industrial environment.

On the other hand, simulation based scheduling is not restricted to idealized simple models. Underlying discrete-event model allows the inclusion of many process specific details and constraints. Heuristic optimization algorithms in connection with simulation systems are a suitable alternative to usual analytical methods in practical cases with high complexity (Weigert, Horn and Werner 2006). Therefore, simulation-based scheduling has potential abilities to deal with actual large-scale and complex problems (Arakawa, Fuyuki and Inoue 2002).

Advantages of discrete-event simulation include, among others, the ability to represent a system's uncertainty and dynamicity, and more generally to produce realistic (valid) representations of the real system (Semini and Fauske 2006). Nevertheless, derivation of an adequate process model may be a complex and cumbersome task. A suitable modelling framework should be chosen, allowing for a systematic model development and possible software support in model building.

Petri nets (PN) form a modelling framework that can be used through several phases of the manufacturing system life cycle (Silva and Teruel 1997). They represent a powerful graphical and mathematical modelling tool. The different abstraction levels of Petri-net models and their different interpretations make them suitable to model many aspects of manufacturing systems, including scheduling problems (Lee and DiCesare 1994, Xiong and Zhou 1998, Yu, Reyes, Cang and Lloyd 2003b). Attempts to automate the model building with Petri nets for scheduling purposes have been reported recently (Gradišar and Mušič 2007).

In the paper a Petri net modelling approach is used that is tailored to model typical scheduling problems. Various optimization strategies are then implemented using derived models. A recently proposed way of controlling the Petri net model during optimization by imposing a set of transition sequences and priorities is compared to other Petri net based scheduling approaches.

2. PETRI NETS

In the paper, Petri nets (Murata 1989, Cassandras and Lafortune 1999) are represented as Place/Transition (P/T)

nets in a form of a five-tuple (P, T, I, O, M), where

- $P = \{p_1, p_2, \ldots, p_m\}, m > 0$ is a finite set of places.
- $T = \{t_1, t_2, \ldots, t_n\}, n > 0$ is a finite set of transitions (with $P \cup T \neq \emptyset$ and $P \cap T = \emptyset$).
- $I : (P \times T) \rightarrow \mathbb{N}$ is the input arc function. If there exists an arc with weight k connecting p to t, then $I(p,t) = k$, otherwise $I(p,t) = 0$.
- $O : (P \times T) \rightarrow \mathbb{N}$ is the output arc function. If there exists an arc with weight k connecting t to p, then $O(p,t) = k$, otherwise $O(p,t) = 0$.
- $M : P \rightarrow \mathbb{N}$ is the marking, M_0 is the initial marking.

Let ${}^\bullet t \subseteq P$ denote the set of places which are inputs to transition $t \in T$, i.e., there exists an arc from every $p \in {}^\bullet t$ to t. Transition t is enabled by a given marking if, and only if, $M(p) \geq I(p,t), \forall p \in {}^\bullet t$. An enabled transition can fire, and as a result removes tokens from input places and creates tokens in output places. If transition t fires, then the new marking is given by $M'(p) = M(p) + O(p,t) - I(p,t), \forall p \in P$.

In order to enable a timed analysis of the modelled system behaviour, a P/T Petri net has to be extended with time information. The concept of time is not explicitly given in the original definition of Petri nets. As described in Bowden (2000), there are three basic ways of representing time in Petri nets: firing durations, holding durations and enabling durations. The firing-duration principle says that when a transition becomes enabled it removes the tokens from input places immediately but does not create output tokens until the firing duration has elapsed. In Zuberek (1991) a well-defined description of this principle is given. When using holding-duration principle, a created token is considered unavailable for the time assigned to transition that created the token. The unavailable token can not enable a transition and therefore causes a delay in the subsequent transition firings. This principle is graphically represented in Figure 1, where the available tokens are schematized with the corresponding number of undistinguishable (black) tokens and the unavailable tokens are indicated by empty circles. The time duration of each transition is given beside the transition, e.g., $f(t_1) = t_d$. When the time duration is 0 this denotation is omitted. In Figure 1, t denotes a model time represented by a global clock and t_f denotes the firing time of a transition.

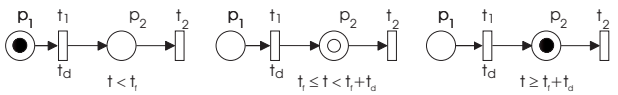

Figure 1: Timed Petri net with holding durations

With enabling durations the firing of the transitions happens immediately and the time delays are represented by forcing transitions that are enabled to stay so for a specified period of time before they can fire. When enabling duration policy is used, the firing of one transition can interrupt the enabling of other transitions, as the marking, which has enabled previous situation, has changed (Bowden 2000). It is natural to use holding durations when modelling most scheduling processes as transitions represent starting of operations, and generally once an operation starts it does not stop to allow another operation to start in between.

By using holding durations the formal representation of the timed Petri net is extended with the information of time, represented by the multiple:

$TPN = (P, T, I, O, s_0, f)$, where:

- P, T, I, O are the same as above,
- s_0 is the initial state of a timed Petri net.
- $f : T \rightarrow \mathbb{R}_0^+$ is the function that assigns a non-negative deterministic time-delay to every $t_j \in T$.

The state of a timed Petri net is a combination of three functions $s = (m, n, r)$, where,

- $m : P \rightarrow \mathbb{N}$ is a marking function of available tokens.
- $n : P \rightarrow \mathbb{N}$ is a marking function of unavailable tokens.
- r is a remaining-holding-time function that assigns values to a number of local clocks that measure the remaining time for each unavailable token (if any) in a place.

A transition t_j is enabled by a given marking if, and only if, $m(p_i) \geqslant I(p_i, t_j), \forall p_i \in {}^\bullet t_j$. The firing of transitions is considered to be instantaneous. A new local clock is created for every newly created token and the initial value of the clock is determined by the delay of the transition that created the token. When no transition is enabled, the time of the global clock is incremented by the value of the smallest local clock. An unavailable token in a place where a local clock becomes available and the clock is destroyed. The enabling condition is then checked again.

3. PETRI NETS AND SCHEDULING

In the following, a holding durations principle of time representation in the Petri net model is assumed.

3.1. Petri net modelling of scheduling problems

An important concept in PNs is that of conflict. Two events are in conflict if either one of them can occur, but not both of them. Conflict occurs between transitions that are enabled by the same marking, where the firing of one transition disables the other transition.

The conflicts and the related conflict resolution strategy play a central role when modelling scheduling problems. This may be illustrated by a simple example, shown in Figure 2. The example involves two machines $M = \{M_1, M_2\}$, which should process two jobs $J = \{J_1, J_2\}$, and where $J_1 = \{o_1(M_1) \prec o_2(M_2)\}$ and $J_2 = \{o_3(M_1)\}$. Job J_1 therefore consist of two operations, the first one using machine M_1 and the second one machine M_2, while job J_2 involves a single operation using machine M_1. Obviously, the two jobs compete for machine M_1. This is modelled as a conflict between transitions starting corresponding operations.

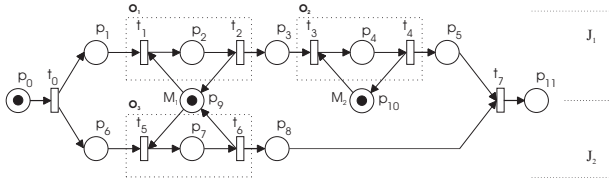

Figure 2: A PN model of a simple scheduling problem

Place p_9 is a resource place. It models the machine M_1 and is linked to t_1 and t_5, which start two distinct operations. Clearly, the conflict between t_1 and t_5 models a decision, whether machine M_1 should be allocated to job J_1 or J_2 first.

Similarly, other decisions are modelled as conflicts linked to resource places. The solution of the scheduling problem therefore maps to a conflict resolution in the given Petri net model.

3.2. Derivation of optimal or sub-optimal schedules

A derived Petri-net model can be simulated by an appropriate simulation algorithm. During the simulation, the occurring conflicts are resolved 'on the fly', e.g. by randomly choosing a transition in conflict that should fire. Instead, **heuristic dispatching rules**, such as Shortest Processing Time (SPT) or Longest Processing Time (LPT), can be introduced when solving the conflicting situations. In this way different evolutions of the Petri net can be simulated. When the marking of the places that represent resources is being considered, the schedule of process operations can be observed, i.e., when, and using which resource, a job has to be processed. Usually, different rules are needed to improve different predefined production objectives (makespan, throughput, production rates, and other temporal quantities).

To show the practical applicability of the Petri net models for the purposes of scheduling, an option for using various conflict resolution rules has been implemented within the PetriSimM tool for Matlab. With the simulation a marking trace of a timed Petri net can be achieved. The unavailable token marking trace of places that represent resources is then characterized as a schedule. In situations in which a conflict occurs, the simulator acts as a decision maker that solves the conflict. By introducing different heuristic dispatching rules (priority rules) decisions can be made easily. In this way, only one path from the reachability graph is calculated, which means that the algorithm does not require a lot of computational effort. Depending on the given scheduling problem a convenient rule should be chosen.

A more extensive exploration of the reachability tree is possible by **PN-based heuristic search method** proposed by Lee and DiCesare (1994). It is based on generating parts of the Petri net reachability tree, where the branches are weighted by the time of the corresponding operations. Sum of the weights on the path from the initial to a terminal node gives a required processing time by the chosen transition firing sequence. Such a sequence corresponds to a schedule, and by evaluating a number of sequences a (sub)optimal schedule can be determined. The method is further investigated in Yu et al. (2003), where a modified heuristic function is proposed and tested on a number of benchmark tests.

A complementary approach to the above mentioned reachability tree exploration methods is the **simulation-optimization approach**. Such a method is proposed in Löscher, Mušič and Breitenecker (2007), and is based on parametrized conflict resolution through sequences and priorities.

The main input parameters of a simulation run within the simulation-optimization approach are represented by the sequences of transition firings. A change of the input parameters leads to a different task ordering and to different schedule. Disjoint groups of transitions can be selected and to each group a firing rule can be assigned. The transitions are numbered within the group and a firing list is defined this way. All transitions of the group are deactivated except of the transition represented by the first number in the firing list. After the firing of the selected transition the next value of the list is taken.

If a priority is needed for sequence transitions a sequence priority can be defined. This sequence priority controls the behaviour of conflicts between transitions of different sequence groups. Priorities are necessary if conflicts between transitions should be solved in a special way.

When solving a scheduling problem, all possible permutations of transition sequences build the problem solution space. A fitness function is defined which assigns a value to each solution of the solution space and defines the quality of the selected solution. The evaluation of the fitness value is performed by Petri net simulation resulting in the overall cycle time of the system. The search through the solution space is driven by heuristics, like genetic algorithms, simulated annealing or threshold accepting (Vidal 1993, Dueck and Scheuer 1990, Goldberg 1989). All these methods are implemented in the so called MATLAB PetriSimM toolbox, which offers the capability of modelling, simulation, and optimisation of Timed, Coloured, and Stochastic Petri nets (Mušič, Löscher and Gradišar 2006).

4. EXAMPLES

A set of scheduling examples was chosen from the literature and tested in conjunction with the described Petri net based scheduling approaches. The examples are based on Lee and DiCesare (1994), Xiong and Zhou (1998), Yu, Reyes, Cang and Lloyd (2003b). Also two examples based on case studies were examined. These are a furniture fittings production example (Gradišar and Mušič 2007) and a production cell example (Löscher, Mušič and Breitenecker 2007).

The first example is based on semiconductor test facility studied in Xiong and Zhou (1998). The facility consists of three types of integrated circuit testers, two types of handlers and two types of hardware, which are combined into three workcenters that represent three machines in the scheduling context. There are four jobs to be scheduled, and job requirements and corresponding times are defined.

Based on the problem description, a Petri net model in Figure 3 is generated. The modelling approach follows the one suggested in Yu, Reyes, Cang and Lloyd (2003a).

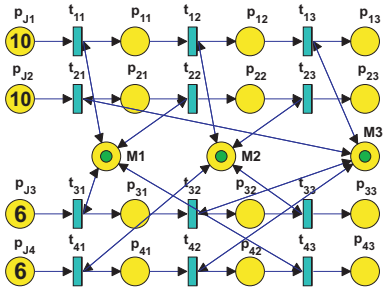

Figure 3: PN model of the first scheduling example

Places that explicitly model operation execution are omitted. Operation execution can be observed by looking for unavailable tokens in the places that follow operation triggering transitions. Time delays adjoined with these transitions model operation duration. Unavailable tokens also appear in resource places, modelling the allocation of resources.

The model has been used in conjunction with the three previously described scheduling strategies, all attempting to minimize makespan for the given lot sizes, indicated by the initial marking in the model. With the dispatching rules strategy, a so called SPT rule has been used, meaning that among any set of transitions in conflict, the one with the shortest time assignment is chosen. In case of several transitions with the same shortest time, one is randomly selected. This explains the interval 134-138 in the results presented in Table 1. Because of the occasional random selection the results of several subsequent optimization runs differ. The table also shows results of PN-heuristic search and simulations based optimization run employing simulated annealing (SA), which both result in makespan of 134. This matches the result reported in Xiong and Zhou (1998) and indicates that the given example is too simple to show any significant difference among strategies.

Next, examples from Lee and DiCesare (1994) are adopted, that have also been studied in Yu, Reyes, Cang and Lloyd (2003b). The second example (example 3 in Lee and DiCesare (1994)) deals with five jobs, each consisting of four operations in a sequence, and sharing three machines, i.e, resources (Figure 4). Lot size of 10 is assigned to each job. The main difference comparing to previous example is the flexibility of operation execution. Namely, the same operation can be executed on different machines, which also results in different execution times. This significantly increases the complexity of the optimization problem.

In order to be able to apply simulation-optimization technique, transition sequence must be defined, which will be manipulated during optimization. In the presented case this is not trivial, since the set of transitions participating in the schedule is not fixed. E.g. the first operation of job J_1 can be performed either using resource M_1 or M_3, which implies firing either t_{111} or t_{112}. The optimization should therefore not only permute the transition order in the given sequence but also change the elements of the sequence. Since this is difficult to achieve in a systematic way, the model is changed in a way which permits to apply standard combinatorial optimization techniques and a part

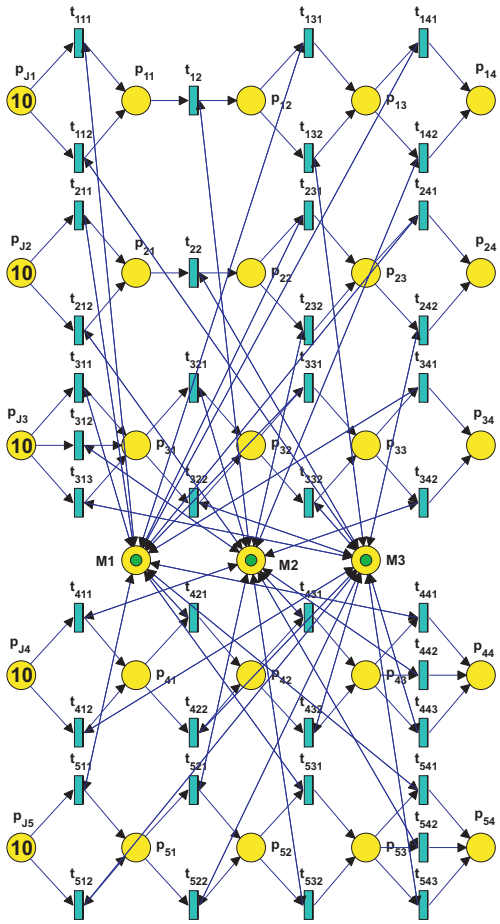

Figure 4: PN model of the second scheduling example

Table 1: Comparison of results - Examples 1-3

Example	Dispatching rules	Heuristic search	Simulation-optimization
example 1	134-138	134	134 (SA)
example 2	370-410	392	420 (SA)
example 3	254-281	258	313 (SA)

of it is shown in Figure 5.

A transition-place pair is inserted before every operation triggering transition set. This newly inserted transitions always participate in the operation sequence, no matter which resource is chosen to perform the operation. A sequence of these transitions may be optimized in order to determine the best operation sequence. It must be noted, however, that such a sequence does not fully determine a possible operation schedule and therefore a certain amount of randomness must be permitted in order to cover all possible schedules. The optimization results are therefore not as good as with the other techniques, which can be observed from Table 1. The results shown in the table were obtained by simulated annealing optimization strategy, while results of threshold accepting and genetic algorithms were similar and are not shown in the table.

The third example (example 4 in Lee and DiCesare (1994)) deals with ten jobs, each with varied number of operations in a sequence, and sharing five machines and

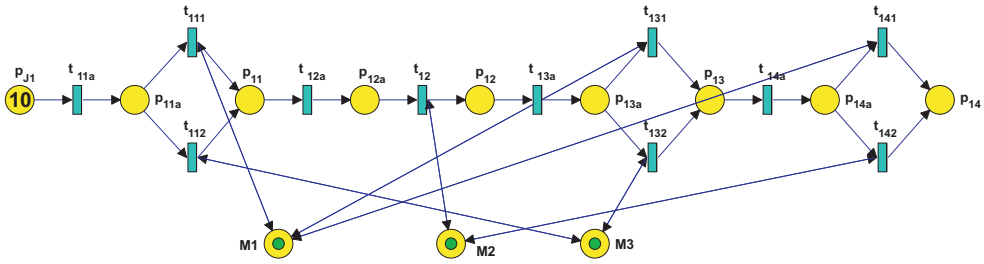

Figure 5: Part of the modified PN model of the second scheduling example

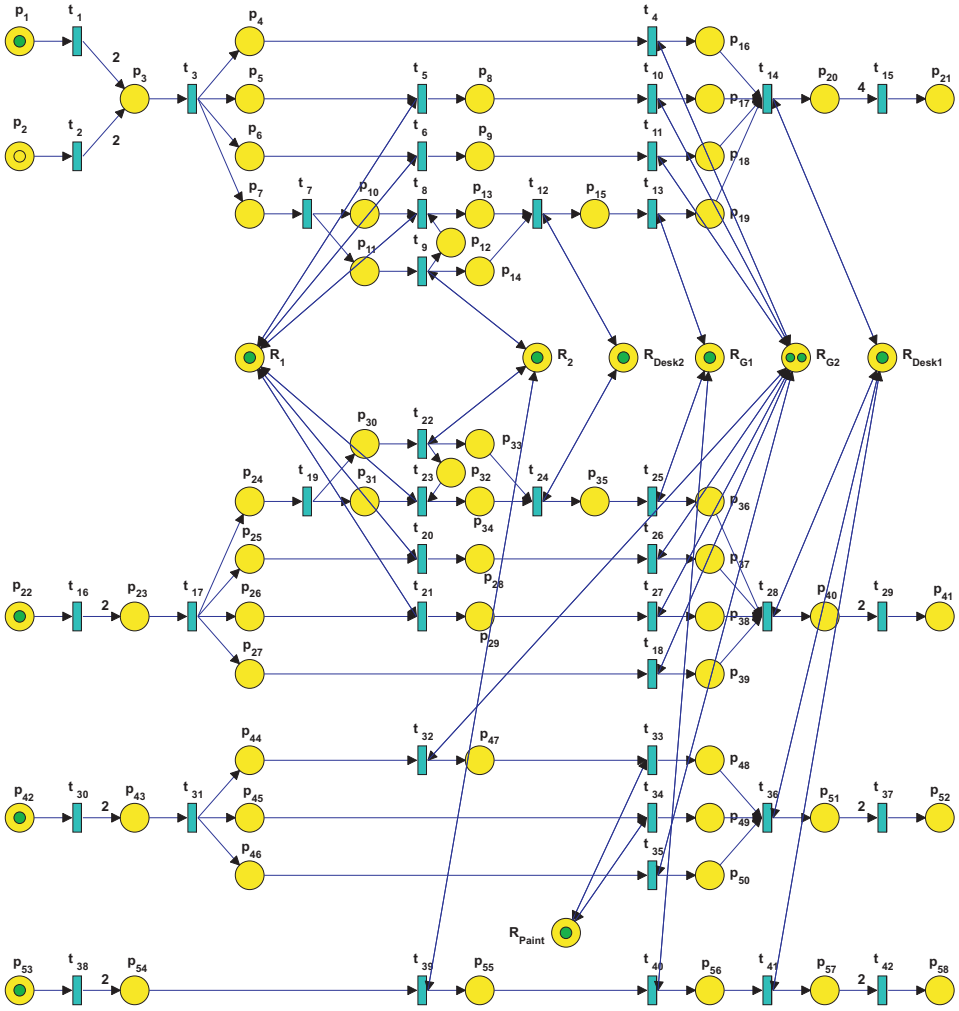

Figure 6: PN model of the furniture fittings production

three robots as resources. The example is similar to the previous one with the main difference in the varying length of jobs and corresponding lot sizes. The corresponding results are also shown in Table 1.

Finally, two case studies based examples are presented. Example 4 originates from the furniture fittings production case study presented in Gradišar and Mušič (2007). The case study deals with the production of furniture fittings described by work orders and product routing tables, and sharing resources such as assembly desks, paint and galvanization facilities etc. A model building algorithm was derived, which builds a Petri net model for a specific set of work orders. In Gradišar and Mušič (2007) such a work order set is defined and an example of the generated model is presented. An improved version of the model building algorithm is applied here, which removes redundant places and transitions and results in the simplified model shown in Figure 6.

Example 5 deals with scheduling of operations in a special type of Production cell. A number of different products is processed in the cell using special fixtures and moving platforms. The process flow in the cell consists of several steps such as setup of fixtures, setup of products, processing, dismounting of products and dismounting of fixtures. The number of platforms is limited and there is a single resource for mounting and dismounting and another

Table 2: Comparison of results - Examples 4 and 5

Example	Dispatching rules	Heuristic search	Simulation-optimization
example 4	220-230	220	220 (SA)
example 5	10160	9943	9976 (SA) 9949 (GA)

one for processing. The main scheduling problem is to determine the optimal processing order given the number of products, platforms and fixtures. The cell is described in detail in Löscher, Mušič and Breitenecker (2007).

Results of comparison for the last two examples are shown in Table 2. Again only the results obtained by simulated annealing are shown in the last column. Example 4 is rather simple from the scheduling point of view and so all techniques result in the same makespan of 220 time units. Nevertheless, the example is based on the real production in a furniture fittings production company and indicates that many real-life production problems may be adequately solved by simple dispatching rules. In such cases, the main issue is how to build appropriate model for scheduling while the quality of the optimization algorithm is of less importance.

The production cell example, on the other hand, shows the advantage of more elaborate optimization techniques. It must be noted that only the best results obtained in a number of optimization runs are shown in the table. E.g., the actual makespan interval obtained by the SPT rule was 10160-10678, while the results of the other two techniques strongly depend on parameter settings, but still gave better results in most cases.

5. CONCLUSIONS

The used methodology enables a straightforward Petri net modelling of typical scheduling problems and derivation of related schedules. It turns out, however, that not all described approaches are suitable for every type of scheduling problems. A simulation-optimization based approach, for example, is very flexible and enables incorporation of specific details in the model, but the solution of all the conflicts in the model must be parametrized in order to achieve a deterministic solution of a simulation run. This involves an extra modelling effort and is even not feasible for specific types of problems. Heuristic search based methods are easier to implement but the results are strongly dependent on the chosen heuristic function. In some cases, rather poor results are obtained if search initially starts in a non-optimal direction. The dispatching rules based methods are the easiest to implement and attractive for practice, but the results may be far from the optimum.

REFERENCES

Arakawa, M., Fuyuki, M. and Inoue, I., 2002. A simulation-based production scheduling method for minimizing the due-date deviation, *Internation Transactions in Operational Research*, 9, 153–167.

Bowden, F. D. J., 2000. A brief survey and synthesis of the roles of time in petri nets, *Mathematical & Computer Modelling*, 31, 55–68.

Cassandras, C. G. and Lafortune, S., 1999. *Introduction to Discrete Event Systems*, Kluwer Academic Publishers, Dordrecht.

Dueck, G. and Scheuer, T., 1990. Threshold Accepting: A General Purpose Optimisation Algorithm Appearing Superior to Simulated Annealing, *Journal of Computational Physics*, 90, 161–175.

Goldberg, D. E., 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addision-Wesley Publishing Comp., Inc.

Gradišar, D. and Mušič, G., 2007. Production-process modelling based on production-management data: a Petri-net approach, *International Journal of Computer Integrated Manufacturing*, 20 (8), 794–810.

Jain, A. S. and Meeran, S., 1999. Deterministic job-shop scheduling: Past, present and future, *European Journal of Operational Research*, 113 (2), 390–434.

Lee, D. Y. and DiCesare, F., 1994. Scheduling flexible manufacturing systems using Petri nets and heuristic search, *IEEE Transactions on robotics and automation*, 10 (2), 123–132.

Löscher, T., Mušič, G. and Breitenecker, F., 2007. Optimisation of scheduling problems based on timed petri nets, *Proc. EUROSIM 2007*, Vol. II, Ljubljana, Slovenia.

Murata, T., 1989. Petri nets: Properties, analysis and applications, *Proc. IEEE*, 77, 541–580.

Mušič, G., Löscher, T. and Gradišar, D., 2006. An Open Petri Net Modelling and Analysis Environment in Matlab, *Proc. IMM 2006 Conf., Barcelona, Spain, 123-128*.

Semini, M. and Fauske, H., 2006. Applications of discrete-event simulation to support manufacturing logistics decision-making: A survey, *Proceedings of the 2006 Winter Simulation Conference*, pp. 1946–1953.

Silva, M. and Teruel, E., 1997. Petri nets for the design and operation of manufacturing systems, *European Journal of Control*, 3 (3), 182–199.

Vidal, R. V. V. (ed.), 1993. *Applied Simulated Annealing*, Springer-Verlag Berlin Heidelberg.

Weigert, G., Horn, S. and Werner, S., 2006. Optimization of manufacturing processes by distributed simulation, *International Journal of Production Research*, 44 (18-19), 3677–3692.

Xiong, H. H. and Zhou, M. C., 1998. Scheduling of semiconductor test facility via Petri nets and hybrid heuristic search, *IEEE Transactions on semiconductor manufacturing*, 11 (3), 384–393.

Yu, H., Reyes, A., Cang, S. and Lloyd, S., 2003a. Combined Petri net modelling and AI based heuristic hybrid search for flexible manufacturing systems-part I: Petri net modelling and heuristic search, *Computers and Industrial Engineering*, 44 (4), 527–543.

Yu, H., Reyes, A., Cang, S. and Lloyd, S., 2003b. Combined Petri net modelling and AI based heuristic hybrid search for flexible manufacturing systems-part II: Heuristic hybrid search, *Computers and Industrial Engineering*, 44 (4), 545–566.

Zuberek, W. M., 1991. Timed petri nets: definitions, properties and applications, *Microelectronics and Reliability*, 31 (4), 627–644.