OPTIMISATION OF SCHEDULING PROBLEMS BASED ON TIMED PETRI NETS

Thomas Löscher[1], Gašper Mušič[2], Felix Breitenecker[1]

[1]Vienna University of Technology, Institute for Analysis and Scientific Computing
1040 Vienna, Wiedner Hauptstrasse 8-10, Austria
[2]University of Ljubljana, Faculty of Electrical Engineering
1000 Ljubljana, Tržaška 25, Slovenia

tloescher@osiris.tuwien.ac.at(Thomas Löscher)

Abstract

This paper deals with modelling and simulation of scheduling and sequencing problems based on Petri Nets. In particular, Timed, Coloured, and Stochastic Petri Nets are used to model and implement specific scheduling problems in the field of production processes and other discrete event systems. The Petri Net models are simulated over the time domain and a simulation-based optimisation is implemented to optimise the input sequences. In this work a new conflict resolution is implemented and a sophisticated way of defining firing sequences is developed. This new approach offers the possibility to model queuing, sequencing or scheduling problems being independent of the appearance of any conflicts. The optimisation of sequencing and scheduling problems works by automated changing and evaluating of the used sequences and parameter specifications. This kind of optimisation problem is too complex to be solved to optimality. A promising alternative is to use heuristics, like genetic algorithms, simulated annealing or threshold accepting. 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. In case of stochastic processes the comparison of alternative system configurations is a highly sophisticated problem. In this work a sequential paired t-test and variance reduction techniques are used and implemented to solve the stochastic optimisation for sequencing and scheduling problems. All the implemented features, functionalities and capabilities are compared and tested in two case studies including the modelling, simulation and optimisation of a production cell and the well-known travelling salesman problem.

Keywords: Petri Nets, Optimisation, Manufacturing, Scheduling

Presenting Author's Biography

Thomas Löscher was born in Horn, Austria and went to the Vienna University of Technology, where he studied technical mathematics and obtained his degrees in 2004 and 2007. He worked for the ARC Seibersdorf research company in the field of discrete event simulation. He is member of the ARGESIM group and his PhD thesis deals with the simulation-based optimisation of scheduling problems based on timed Petri Nets. His e-mail address is: tloescher@osiris.tuwien.ac.at.

ISBN 978-3-901608-32-2 *Copyright © 2007 EUROSIM / SLOSIM*

1 Introduction

Petri Nets basically model processes on a very low level. Up to now a lot of extensions of Petri Nets exist. The introduction of time delays makes it possible to simulate over the time domain. The use of colours simplifies the graphical description and due to this fact models of greater complexity can be defined and used. In this work another interesting and important extension is developed to get the capability of modelling scheduling problems. This means that it is possible to build up Petri Nets based on the description of scheduling problems. All conditions and constraints are realised by the basic properties of Petri Nets. The input parameters are characterised by different firing sequences of the selected Petri Net corresponding to sequences of the present scheduling problem. On this account system configurations can be easily changed by the use of different input sequences. All properties of scheduling problems are now described by the use of Petri Nets and the evaluation of a chosen schedule is reduced to a single simulation run of the Petri Net based on the used input sequences.

Starting from this development an automated optimisation can be implemented based on several heuristic methods. The scheduling problem is reduced to a simple combinatorial problem. The objective function is defined by the evaluation of the Petri Net simulation over the time domain. Another aspect of this work is given by the use of stochastic time delays. In this case the stochastic optimisation results in a highly sophisticated problem.

All these extensions and functionalities are developed and implemented in the so called MATLAB PetriSimM toolbox where the user can model, simulate and optimise scheduling problems based on Petri Nets.

2 Petri Nets and Scheduling

Scheduling deals with the allocation of resources to activities over time, by respecting precedence, duration, capacity and incompatibility constraints, in order to achieve the optimal use of resources or the optimal accomplishment of tasks. Scheduling involves the arrangement, coordination, and planning of the utilisation of resources to achieve an objective. Of the resources available, time is becoming an important commodity. Time is the resource most often planned and is present in all scheduling. In its simplest form, the overall time cycle required for production or completion is the most usual scheduling situation [1]. Scheduling problems arise in domains like manufacturing, transportation, computer processing and production planning. Many well defined problems like job-shop, flow-shop problem or scheduling used in flexible assembly systems can be found in literature [2, 3].
There exists special classifications of scheduling problems [4, 5, 6] which leads to different approaches of modelling and algorithms. This work provides a general framework for modelling, simulation, analysis and optimisation of scheduling problems based on Timed Petri Nets. No special definitions are given for the scheduling problem besides of some restrictions for the underlying Timed Petri Net. The sequence orders are the only parameters used. All other conditions and specifications are determined by the basic properties of Timed Petri Nets. Simulation is an appropriate tool for evaluating and analysing scheduling problems based on Timed Petri Nets to get the needed performance measures [7]. With respect to the optimtisation problem this work is focused on the overall cycle time as main interest of the performance measures. But using Petri Nets offers a lot of possibilities to get other interesting parameters of the system. For example, utilisation and allocation of the resources can be easily measured and shown by Gantt Charts.

2.1 Definition

The class of scheduling problems which can be modelled as Timed Petri Nets is defined by the following criteria:

- **same initial marking** - Each scheduling problem has to start with the same initial marking. This means that different system configurations can only realised by changing the sequence parameters.
- **sequences assigned to transitions** - The sequence parameters are assigned to the transitions and they correspond to the firing sequences of the transitions.
- **conflicts solved** - All conflicts have to be resolved and determined at the beginning.
- **bounded** - The TPN has to be bounded.
- **not reversible** - The TPN must not be reversible. In this case reversibility is extended with the defined firing sequences. This means that it is not allowed to come back to a home state including the current state and position of sequence parameters.
- **all transitions not live** - No transition is allowed to be live.
- **terminating simulation** - Because of the previous criterion the simulation of the Timed Petri Net is terminating.

2.2 Priority

The resolution of all conflicts is an important criterion of scheduling problems modelled as Timed Petri Nets. For this purpose a priority ranking can be established for the firing of the transitions. A priority value can be set to each transition. This value has to be greater or equal than one and defines the priority of the transition. The highest priority is given to 1. If there is a conflict between at least two transitions the transition with the highest priority will always fire. If two or more transitions have the highest priority value it will be randomly decided which transition fires. Figure 1 shows a

 Copyright © 2007 EUROSIM / SLOSIM

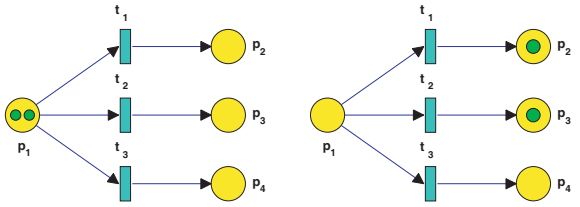

Fig. 1 Petri Net with priority

small Petri Net with defined priorities before and after firing. In this example transition t_1 has the highest priority equal to 1 and transitions t_2 and t_3 have both the priority equal to 2. Place P_1 contains two tokens and therefore a conflict occurs because all three transitions are enabled but only two of them can fire. Transition t_1 fires in any case. Transitions t_2 and t_3 have the same priority and the firing is randomly decided. In this case transition t_2 fires but it is also possible that transition t_3 fires instead of t_2 after restarting the simulation.

2.3 Sequence

Next to the definition of a priority Petri Nets need another important extension to get the capability of modelling scheduling problems. The main input parameters of scheduling problems are represented by the sequences of different tasks. A change of the input parameters leads to a different system configuration and to different results. These sequences can be directly implemented and defined to the transitions of the Petri Net. Disjoint groups of transitions can be selected and to each group a firing rule can be assigned. The transitions are numbered within the group starting from 1 to the selected group length. Now a firing list can be defined for the group of transitions. This list consists of the assigned numbers of the transitions. The values of the list correspond to the firing of the transitions. All transitions of the group are deactivated instead of the transition represented by the first number of the firing list. This means that all deactivated transitions are not able to be enabled by any tokens of the input places. After the firing of the selected transition the next value of the list is taken. If the end of the firing list is reached all transitions are activated again and they behave like normal transitions not controlled by a firing list. Figure 2

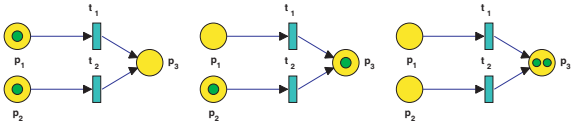

Fig. 2 Petri Net with sequence

shows the firing of a Petri Net with a defined sequence. In this example the sequence $(1, 2)$ is assigned to the transitions t_1 and t_2. In the left Petri Net transition t_1 is enabled and transition t_2 is deactivated due to the firing list. The middle Petri Net shows the marking after firing of transition t_1. Transition t_2 is activated again and now enabled whereas transition t_1 is deactivated in this step. The final marking is shown in the right Petri Net of figure 2 after firing of transition t_2.

On the one hand transitions can be selected to assign priorities and on the other hand transitions can be added to groups to define firing lists. These two possibilities are disjoint. This means that either a transition can have a priority or a transition can be member of a sequence group. 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. The following ranking and hierarchy is used for the interacting of the different kinds of transitions:

1. **Sequence:** Transitions which are members of a sequence group always have the highest priority. Conflicts inside sequence transitions are determined by the sequence priority.

2. **Priority:** Transitions with priority get the second highest priority.

3. **Normal:** Transitions without any priorities or sequences are defined as normal transitions. Conflicts of normal transitions are solved through the strategies mentioned in the previous section.

2.4 Deadlock

The termination of the simulation is an important and an essential criterion for modelling scheduling problems based on Timed Petri Nets. Depending on the selected input sequences each system configuration leads to different simulation results. The end of a simulation corresponds to a final marking of the reachability graph. The final markings are reached during the simulation through a desired natural deadlock. Because of the sequence extensions of Timed Petri Nets a new kind of deadlock can occur during the simulation. If the transition activated due to the firing list is not able to be enabled and no other transition is enabled in the current simulation step a sequence deadlock happens. This sequence deadlock represents an invalid system configuration and it can be separately detected.

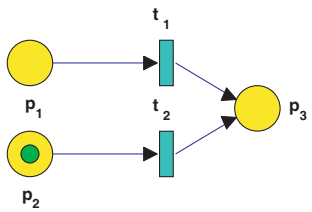

Fig. 3 Sequence deadlock

Figure 3 shows this new kind of deadlock. Transition t_1 and t_2 are added to a sequence group containing the firing list $(1, 2)$. Transition t_1 is activated but not enabled and therefore a sequence deadlock occurs and the simulation stops.

ISBN 978-3-901608-32-2 Copyright © 2007 EUROSIM / SLOSIM

3 Optimisation

3.1 Introduction

Scheduling problems basically belong to the field of combinatorics. A set of tasks should be ordered to build an optimum corresponding to certain constraints. An optimum can be both a maximum and a minimum. Therefore the optimisation of a scheduling problem can be a minimising or maximising problem. In this work all constraints and specifications of scheduling problems are determined by the basic properties of Timed Petri Nets. The sequence orders are the only parameters used. All possible permutations of these sequences build the solution space of the scheduling problem. In general a so called objective or fitness function is defined which assigns a certain value to each solution of the solution space. This fitness value is used to define the quality of the selected solution. In this case the objective function is determined through the underlying Timed Petri Net. The evaluation of the fitness value is performed by simulation resulting in the overall cycle time of the system. For maximising problems this value should be as great as possible and respectively for minimising problems it should be as small as possible. But both problem types are equivalent. The multiplication of the objective value of a minimising problem by -1 results in a maximising problem and vice versa. Thus, in the following only minimising problems are considered.

In principle there are two possibilities to solve such optimisation problems. On the one hand exact solutions can be computed forming an exact optimum of the selected problem. Scheduling problems belong to the class of NP-hard problems [4] and therefore exponential run-time would be needed to compute an exact solution. On the other hand it is possible to apply approximation algorithms like heuristic algorithms [4, 8]. These algorithms have polynomial run-time and produce solutions that are guaranteed to be within a fixed percentage of the actual optimum. Any approach without formal guarantee of performance can be considered a heuristic. Such approaches are useful in practical situations if no better methods are available [4].

3.2 Local Search

Local search is an iterative procedure which moves from one solution in the search space S to another as long as necessary. In order to systematically search through S, the possible moves from a solution s to the next solution should be restricted in some way. To describe such restrictions a neighbourhood structure $N : S \rightarrow 2^S$ is introduced on S. For each $s \in S$, $N(s)$ describes the subset of solutions which can be reached in one step by moving from s. The set $N(s)$ is called the neighbourhood of s. Usually it is not possible to calculate the neighbourhood structure $N(s)$ beforehand because S has an exponential size. To overcome this difficulty, a set AM of allowed modifications $F : S \rightarrow S$ is introduced. For a given solution s, the neighbourhood of s can be defined by $N(s) = \{F(s) \,|\, F \in AM\}$.

Using these definitions, a local search method may be described as follows. Each iteration starts with a solution $s \in S$ and choose a solution $s' \in N(s)$ or a modification $F \in AM$ which provides $s' = F(s)$. Based on the values of the objective function $f : S \rightarrow \mathbb{R}$, $f(s)$ and $f(s')$, the starting solution of the next iteration is chosen. According to different criteria used for the choice of the starting solution of the next iteration different types of local search techniques are arisen [4]. The iterative improvement algorithm takes the solution with the smallest value of the objective function and can be formulated as follows:

Algorithm Iterative Improvement:
begin
  choose initial solution $s \in S$
  repeat
    generate neighbour solution $s' \in N(s)$
    if $f(s') \leq f(s)$ **then**
      $s := s'$
  until $f(s') \leq f(s), \forall\, s' \in N(s)$
end

3.3 Simulated Annealing

Local search algorithms are simple to implement and quick to execute, but they have the main disadvantage that they terminate in the first local minimum which might give an objective function that deviates substantially from the global minimum. The reason why a local search algorithm terminates in the first local minimum it encounters is that only transitions corresponding to a decrease in the objective function are accepted by the algorithm. Alternatively, an algorithm should be considered which attempts to avoid becoming trapped in a local minimum by sometimes accepting transitions corresponding to an increase in the objective function. Simulated Annealing is an example of the latter approach where in addition to cost-decreasing transitions, cost-increasing transitions are accepted with a non-zero probability which decreases gradually as the algorithm continues its execution [9].

Simulated Annealing (SA) exploits an analogy between the way in which a metal cools and freezes into a minimum energy crystalline structure (the annealing process) and the search for a minimum in a more general system. The algorithm is based upon that of Metropolis et al. [10], which was originally proposed as a means of finding the equilibrium configuration of a collection of atoms at a given temperature. Kirkpatrick et al. [11] proposed in 1983 that the Metropolis algorithm forms the basis of an optimisation technique for combinatorial and other problems.

The pseudo-code of the local search algorithm is extended for the Simulated Annealing algorithm in the following way:

Algorithm Simulated Annealing:
begin
  generate initial solution $s \in S$
  $i := 0$
  repeat

 Copyright © 2007 EUROSIM / SLOSIM

generate neighbour solution $s' \in N(s)$
if $f(s') \leq f(s)$ **then**
$s := s'$
else if $min(1, exp(\frac{f(s)-f(s')}{c_i})) > rnd([0,1])$ **then**
$s := s'$
$c_{i+1} := g(c_i)$
$i := i + 1$
until termination criterion
end

Simulated Annealing is a method which seeks to avoid being trapped in a local minimum. It is a randomised method because:

- s' is chosen randomly from $N(s)$
- in the $i - th$ step s' is accepted with probability

$$min(1, exp(\frac{f(s) - f(s')}{c_i}))$$

where (c_i) is a sequence of positive control parameters with $\lim_{i\to\infty} c_i = 0$.

The interpretation of this probability is as follows. If $f(s') \leq f(s)$, then s is replaced by s' with probability one. If, on the other hand, $f(s') > f(s)$, then s is replaced by s' with some probability. This probability decreases with increasing i. In other words, a local minimum can be left, but the probability for doing so will be low after a large number of steps. In the Simulated Annealing algorithm $rnd[0,1]$ denotes a function which yields a uniformly distributed random value between 0 and 1. Furthermore, the sequence (c_i) is created by a function g, i.e. $c_{i+1} = g(c_i)\ \forall\, i$ [4].

3.4 Threshold Accepting

The Threshold Accepting algorithm (TA) is one of the youngest heuristic algorithms. Dueck and Scheuer [12] proposed Threshold Accepting as a variance of Simulated Annealing in 1990. The essential difference between SA and TA consists of the different acceptance rules. TA accepts every new configuration which is not much worse than the old one whereas SA accepts worse solutions only with rather small probabilities. An apparent advantage of TA is its greater simplicity. It is not necessary to compute probabilities or to make random decisions. The pseudo-code for the Threshold Accepting algorithms can be defined in the following way:

Algorithm Threshold Accepting:
begin
generate initial solution $s \in S$
$i := 0$
repeat
generate neighbour solution $s' \in N(s)$
if $f(s') \leq f(s)$ **then**
$s := s'$
else if $f(s') - f(s) < t_i$ **then**
$s := s'$
$t_{i+1} := g(t_i)$
$i := i + 1$
until termination criterion
end

A newly generated solution $s' \in N(s)$ is now accepted if the difference $f(s') - f(s)$ is smaller than some non-negative threshold t. The threshold t is a positive control parameter which is gradually reduced in an analogue way to the temperature of the Simulated Annealing algorithm. The neighbourhood functions defined in the previous section can also be used for the TA algorithm. Further, the cooling strategies for the temperature can be used for the threshold. The TA algorithm also has the same termination criteria as the SA algorithm.

3.5 Genetic Algorithms

Genetic algorithms (GA) are numerical optimisation algorithms inspired by both natural selection and natural genetics. The method is a general one, capable of being applied to an extremely wide range of problems. Genetic algorithms were developed by John Holland [13] in the late sixties. They combine survival of the fittest among string structures with a structured yet randomised information exchange to form a search algorithm with some of the innovative flair of human search. In the original definition the different solutions of the search space are represented and encoded in binary strings but also other encodings are possible. Especially for combinatorial and scheduling problems the solutions can be encoded directly. This means that the sequence of tasks can be directly used in the algorithm. In the GA lingo all used terms and elements have special names. A solution of the problem is called individual or phenotype and its representation is called genome, chromosome or genotype. In case of scheduling problems those two definitions are coincided. The search space of the selected problem is called fitness landscape and the objective function is called fitness function. Compared to Simulated Annealing or Threshold Accepting Genetic algorithms are initialised with a population of individuals which are usually random and be spread throughout the search space. A typical algorithm then uses three operators, selection, crossover, and mutation to direct the population over a series of time steps or generations towards convergence at the global optimum [14, 15]. The pseudo-code for a GA can be defined in the following way:

Algorithm GA:
begin
$i := 0$
generate initial population $P(i)$
evaluate individuals in $P(i)$
repeat
$i := i + 1$
select $P(i)$ from $P(i-1)$
recombine individuals in $P(t)$
evaluate individuals in $P(t)$
until termination criterion
end

Genetic algorithms are suitable for problems with an unknown search space and for cases where no other methods can be used. The algorithm and its operators

 Copyright © 2007 EUROSIM / SLOSIM

do not need any problem specific information. Therefore scheduling problems can be solved and optimised by the use of Genetic algorithms [16, 17, 18].

3.6 Due-date Scheduling

In many scheduling problems, e.g. production systems, the finishing times of certain tasks play an important role concerning the optimisation of the overall cycle time. In the deterministic case each simulation run of a certain system configuration yields exactly the same result. Hence in this work it is possible to extend deterministic scheduling problems with the capability of due dates for certain and interesting tasks because the excess of the due dates is significant for each different system configuration. If the defined due dates are exceeded during the simulation a penalty time will be added to the overall cycle time at the end of the simulation. Now the objective or fitness function of the scheduling problem represents the overall cycle time plus the cumulative penalty time of the due dates. The new value of the fitness function can be used for the direct optimisation of the scheduling problems and the simultaneous indirect optimisation of the due dates.

3.7 Stochastic Optimisation

The analysis and the comparison of the output data of terminating simulation models using random numbers and stochastic distribution functions is a highly sophisticated problem. The difficulty is that the simulation output data are stochastic, so comparing two different system configurations on the basis of only a single run of each is a very unreliable approach. The results of one simulation run are not significant and in general a certain number of replications should be done to avoid making serious errors leading to fallacious conclusions and poor decisions [19].

3.7.1 Sequential paired t-test

In this work the comparison of different system configurations is reduced to a pairwise comparison. For stochastic simulation models it is difficult to get exact comparable and significant results for many different system configurations. Therefore only two different system configurations are taken into account to decide which one of the two alternatives is better. On this account the stochastic optimisation is only implemented for Simulated Annealing and Threshold Accepting. The comparison is effected by forming a sequential confidence interval for the difference in the two expectations to see whether the observed difference is significantly different from zero. If the confidence interval misses or contains zero the test for the difference is accepted or rejected, respectively [19].

3.7.2 Variance Reduction

Variance reduction techniques try to reduce the variance of the sample mean. Depending on the selected problem there exist many different variance reduction techniques. In this case Common Random Numbers (CRN) are used [19, 20]. This technique is applied when two or more alternative system configurations are compared.

4 PetriSimM toolbox

4.1 Introduction

MATLAB, the classical engineering tool, does not really offer tools to handle discrete event systems based on Petri Nets. One commercial tool can be obtained by the MATLAB Connections Program [21, 22]. There are few Petri Net tools based on MATLAB which are free but these tools are rudimentary and can handle only the basics of Petri Nets. A lot of other tools exist for analysis, modelling and simulation of discrete event systems using the advantages of Petri Nets [23, 24] but all these tools cannot handle the optimisation of scheduling problems.
In this work an open source MATLAB toolbox for Petri Nets is presented and introduced. The so called MATLAB PetriSimM toolbox is based on a basic toolbox [25] which deals with analysis, supervisory control synthesis, and non-timed simulation. This basic toolbox is programmed in MATLAB version 5.3 and is therefore adapted to MATLAB version 7.2 (R2006a) to form the MATLAB PetriSimM toolbox. The toolbox is embedded in the MATLAB environment and its usage requires version 7.0 or higher. Furthermore the toolbox is extended with the capability of Timed Petri Nets and timed simulation using the holding durations principle [26, 27, 28]. In another step Coloured Petri Nets are developed for the use in the MATLAB PetriSimM toolbox. The enabling duration principle is added as a second approach of implementing time into Petri Nets. A new way of defining firing sequences is found to be able to model scheduling problems being independent of the occurrence of any conflicts [29, 30]. Finally the toolbox is extended with the optimisation of scheduling problems containing heuristic algorithms like Simulated Annealing, Threshold Accepting and Genetic Algorithms. In case of stochastic processes a sequential paired t-test and variance reduction techniques are used and implemented to solve the stochastic optimisation for sequencing and scheduling problems. To sum up, the sophisticated MATLAB PetriSimM toolbox offers the capabilities of analysis, modelling and simulation of Petri Nets. Furthermore it is possible to optimise scheduling problems based on Timed, Coloured, and Stochastic Petri Nets. The open source MATLAB PetriSimM toolbox can be used for education in a graduate level and for modelling and simulating real life processes of discrete event systems in equal measure.

4.2 GUI

Figure 4 shows a screenshot of the graphical user interface (GUI) of the PetriSimM toolbox. The GUI is divided into a menu bar, a button bar and an axes area. In the menu bar different modes can be chosen. The user can switch between analysis, non-timed simulation, timed simulation based on the holding durations principle, and timed simulation based on the enabling durations principle. Furthermore models can be saved, loaded, exported and printed. For each type of simulation several parameters can be set. Another important part of the menu bar is the options menu where the priority wizard and Gantt chart wizard can be started. Next

 Copyright © 2007 EUROSIM / SLOSIM

to the options menu the optimisation menu is placed. There several parameters and modes for the optimisation can be changed and selected. The button bar contains several buttons for building, changing, zooming, simulating and analysing the Petri Net models. In the axes area the Petri Net models can be created, simulated and the so called token game can be shown. Figure 4 also contains the Petri Net model of a production cell which is later used for the otpimisation.

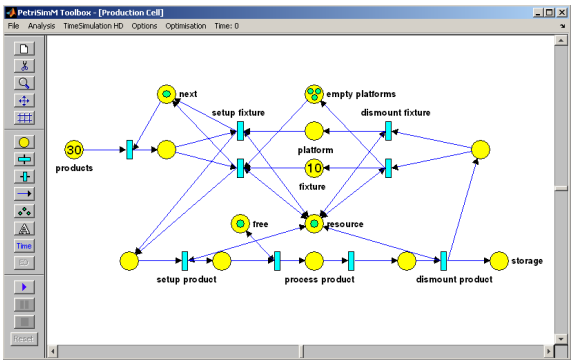

Fig. 4 Graphical User Interface

4.3 Analysis

The PetriSimM toolbox offers several features for the analysis of the modelled Petri Net. In the analysis section interesting properties like P-invariants, T-invariants, reachable sets, cover-ability tree, marking bounds and dead markings can be derived. Furthermore supervisory control methods can be used. All these analysis and control features are developed in the previous version of the toolbox [25] and are not further treated in this context. In this work they are only mentioned for the sake of completeness.

4.4 Simulation

The simulation is a main part of the PetriSimM toolbox. It is separated into non-timed simulation, timed simulation using holding durations and timed simulation using enabling durations. For all three simulation modes an animation of the token game can be visualised. But this feature is only used for educational purposes because through the animation the simulation speed is highly increased. It is possible to change the animation speed in the parameter section and for the optimisation the animation is deactivated. Another interesting parameter for non-timed simulation is the firing probability. This parameter controls the firing of enabled transitions. This means that it is randomly decided if an enabled transition can fire or not depending on the defined firing probability. For timed simulation time delays can be assigned to the transitions which can be deterministic or stochastic. This means that any probability distribution function can be defined to each transition. For this purpose, any MATLAB m-file can be written resulting to a single positive value, or existing MATLAB probability distribution functions, which can be used to model stochastic time delays. Only the positive part of the used function is taken. If the result of the used stochastic function is negative the time delay is set to zero and a warning is displayed.

5 Case Studies

In this work two case studies are processed to test and compare the implemented optimisation algorithms. The first case study contains the modelling, simulation, and optimisation of a special type of production cell [31, 32]. The Petri Net model of this problem consists of many conflicts and therefore the holding durations principle is used to model the production cell. All used time delays are deterministic and due dates and arrival times are therefore assigned for selected products. This is a second reason for choosing holding durations because the arrival times can be easily realised by the use of initial unavailable tokens. The modelling, simulation, and optimisation of the well-known travelling salesman problem [33] is done in the second case study. Maybe Petri Nets are not really the best solution for modelling this problem but it is very useful to show the capabilities of the MATLAB PetriSimM toolbox. The conflicts of the underlying Petri Net are disabled by the definition of the sequence list and therefore no real conflicts have to be considered. For this reason the enabling durations are used to model this problem to reduce the computational duration of the optimisation. In this case the time delays of the transitions are modelled by stochastic distribution functions. Variance reduction is used to speed up the stochastic optimisation and selected benchmarks are done to show the advantage of the implemented technique.

5.1 Production Cell

In this production cell different types of products can be processed. Each product type requires a special fixture and for each type exist only one fixture. This fixture is used to fix the product into the suitable position for processing. Platforms are used to move and shift the products inside the production cell. The process flow is

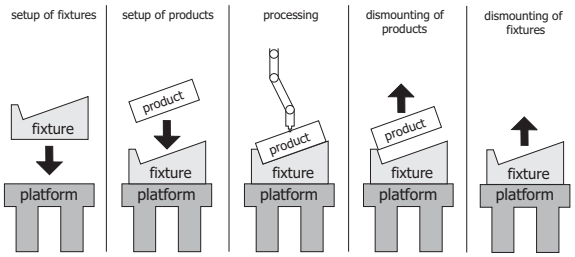

Fig. 5 Process Flow of Production Cell

separated into five different production steps (figure 5):
Setup of fixtures: If the fixture for the next product is not mounted onto a platform it has to be mounted onto an empty one. This procedure takes the setup-time for the fixture. Otherwise the platform with the fixture for the next product is taken without loosing time.
Setup of products: Products are mounted onto the fixtures. For this step the setup-time for the product is needed.

 Copyright © 2007 EUROSIM / SLOSIM

Processing: In this step the product is processed automatically and computer controlled. This procedure takes the processing time for the product and the processing is characterised by long process times.
Dismounting of products: After the processing step the products are dismounted from the fixture. For this process part the dismounting time for the product is needed.
Dismounting of fixtures: Fixtures are dismounted depending on the sequence order of the products. If the fixture is dismounted it takes the dismounting time for the fixture. Otherwise the platform with the fixture on it is waiting for setup the product.

The processing of the products is automated and is independent of any restrictions. By contrast, one resource is shared by the four other process steps of the process flow. This means that the process steps have to be operated one after another. Therefore the following prioritisation is used and implemented to optimise the process flow:

1. dismounting of fixtures
2. dismounting of products
3. setup of products
4. setup of fixtures

5.1.1 Due Dates

In case of due dates selected products have given arrival and finishing times. After the products have passed the processing part of the production cell they are finished. If the desired finishing times are exceeded a penalty time will be added to the overall cycle time of the production.

5.1.2 Implementation

The Petri Net model of the production cell is automatically generated by the use of a programmed template. The function *productioncell()* creates all needed places, transitions, tokens and colours whereas the following input parameters can be used: names of the products, the number of platforms, the initial sequence order of the products, and the time delay matrix. The model

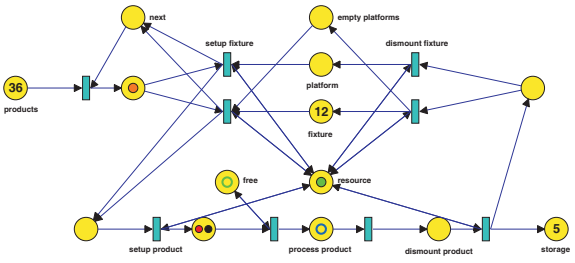

Fig. 6 Model of Production Cell

(figure 6) is separated into the five parts of the process flow. The sharing of the resource is implemented as a conflict of the place ”resource” and its connected transitions (setup fixture, dismount fixture, setup product and dismount product). The different colours represent the different product types and fixtures. In this case an additional colour is used to model the empty platforms, the availability of the resource and all other constraints. The determination of the optimal dismounting sequence for the fixtures is a highly sophisticated problem. Usually there are less platforms than fixtures available in the production cell. On this account the fixtures have to be dismounted on time. On the one hand, a deadlock can occur if the fixtures are not dismounted because then all platforms are occupied and no new fixture can be set up. On the other hand, needless time consuming steps are done if the fixtures are dismounted too early. A user defined sequence function called *platformsequence()* is used and implemented to solve this problem. The dismounting sequence of the fixtures is calculated depending on the current sequence of the products and the number of used platforms.

5.1.3 Results

In this case study a production cell is considered where 15 different products can be processed. The batch size for each product is defined from 2 to 4 resulting in 45 processed products and a Petri Net model consisting of 224×135 sized input and output matrices.
150 iterations are performed for the Simulated Annealing and Threshold Accepting algorithm to optimise the production sequence. The implemented cooling strategies and neighbourhood functions are compared. For the Genetic algorithm three different population sizes are considered and 20 generations are derived. Selected parameter specifications are tested and the different implemented crossover functions are compared. Finally, the results for all algorithms are compared and shown. Figure 7 shows the Gantt chart for the initial sequence

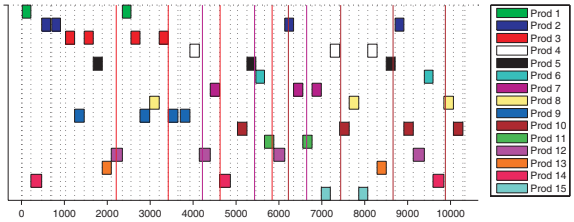

Fig. 7 Gantt Chart of Initial Solution

of the products representing the initial solution of the optimisation problem. The coloured vertical lines stand for the desired finishing times of the selected products. In the initial sequence not all products are available on time and therefore additional gaps are created in the present Gantt chart.

Comparison

Figure 8 shows the results of the comparison for Simulated Annealing, Threshold Accepting and the Genetic Algorithm. All three heuristic algorithms lead to very good and similar results. Table 1 contains the needed computational times for each optimisation algorithm and the calculated duration of the production. In this example the Genetic algorithm produced the best result but 2000 simulation runs are needed for 20 generations

 Copyright © 2007 EUROSIM / SLOSIM

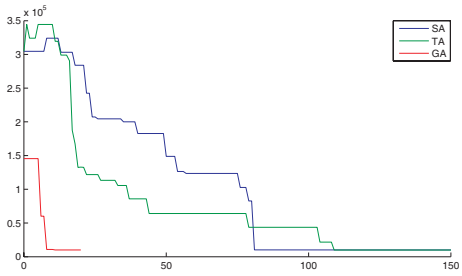

Fig. 8 Comparison of SA, TA and GA

using a population of 100 individuals. Therefore the computational effort for GA amounts to about fourteen times as much as that for 150 iterations of the two other algorithms. The TA algorithm performs the optimisation in the fastest way. The Gantt chart of the optimised

	SA	TA	GA
Comp. Times	133.9	129.4	1775.4
Results	9976	9984	9949

Tab. 1 Comparison of Duration - SA, TA and GA

sequence representing the best solution of the optimisation problem is shown in figure 9. Now all products are available on time and therefore no additional gaps are created in the present Gantt chart. Furthermore, all products are finished on time and therefore no penalty times are added to the overall cycle time of the production. Table 2 shows the additional penalty values for the

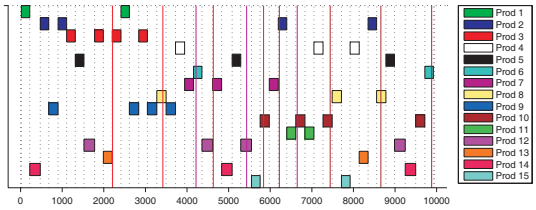

Fig. 9 Gantt Chart of Best Solution

initial and best solution. If the defined due dates are exceeded the difference of the real and desired finishing times is multiplied by the factor 100. Hence, the cumulative penalty time for the duration of the production amounts 2942 time units. The second results shown in the table 2 are the so called setup times for the initial and best solution. In this example the setup time means the time when no product is in the processing part of the production cell. The optimised solution brings in about 89 per cent improvement for the setup time.

5.2 Travelling Salesman Problem

The well-known Travelling Salesman Problem (TSP) asks for the shortest route to visit a collection of cities and return to the starting point. In this work the symmetric version of the TSP is treated. This means that, for any two cities A and B, the distance from A to B is the same as that from B to A. The costs for each connection are derived depending on the distance and the

	Initial	Best
Penalty	294200	0
Setup time	4.39 %	0.49 %

Tab. 2 Comparison of Penalty and Setup Times

average speed of the salesman. In this case the average speed is modelled as stochastic distribution function.

5.2.1 Implementation

Many approaches and algorithms exist to model and to solve the Travelling Salesman Problem. Maybe Petri Nets are not really the best and fastest solution but the TSP can be easily modelled and optimised by the use of the MATLAB PetriSimM toolbox. Figure 10 shows the Petri Net model for the TSP which is automatically generated by the use of a programmed template. The function *tspsym()* creates all needed places, transitions and tokens whereas the following input parameters can be used: number of the cities, the initial sequence order of the visits, and the distance matrix. In this case only

Fig. 10 Model of Travelling Salesman Problem

one colour is used to model the different cities. The symmetric TSP consists of $\frac{n^2-n}{2}$ different transitions representing all possible movements for n cities. The arising conflicts of the Petri Net are deactivated by the use of a sequence list. The input sequence contains only the city numbers and therefore a user defined sequence function (*tspsequence()*) is implemented to transform the input sequence to the corresponding sequence list for the transitions.

5.2.2 Results

In the stochastic case the average speed of the salesman is modelled by the use of stochastic distribution functions. The stochastic optimisation is only possible for Simulated Annealing and Threshold Accepting algorithms because two different system configurations are always compared. Basically a certain number of simulation runs are needed to get significant results. In this example 20 cities are taken into account to realise the Travelling Salesman problem for stochastic time delays. 150 iterations are done for each optimisation algorithm. Figure 11 shows the plan of the cities and their connections representing the initial solution of the problem. Figure 12 shows the comparison of Simulated Annealing and Threshold Accepting. After 150 iterations the TA algorithm leads to better results. Table 3 contains the comparison of the duration for both algorithms and presents the time improvement achieved by the use of variance reduction. It can be seen that the variance reduction highly reduces the computational effort for the stochastic optimisation. In both cases about 70 percent of time can be saved. The decrease of the computational effort accompanies the re-

ISBN 978-3-901608-32-2 Copyright © 2007 EUROSIM / SLOSIM

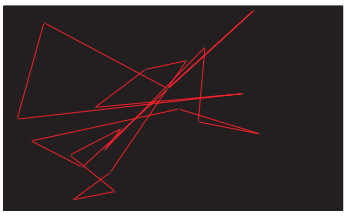

Fig. 11 Plan of Initial Solution

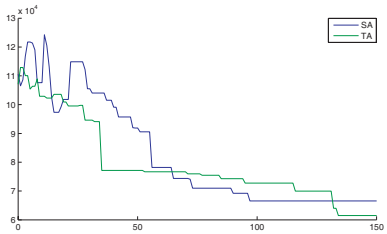

Fig. 12 Comparison of SA and TA

duction of needed simulation runs. Table 4 shows the difference for both optimisation algorithms. The use of variance reduction techniques decrease the number of needed simulation runs in an essential way. About 80 percent of runs can be saved in both cases. Figure 13 shows the plan of the cities containing the optimised route of the TSP.

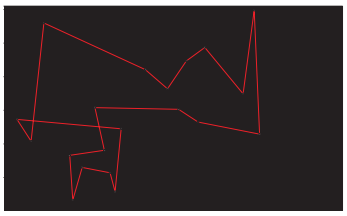

Fig. 13 Plan of Best Solution

6 Conclusion

This work provides a general framework for modelling, simulation and optimisation of scheduling problems based on Petri Nets. The basic definitions of Petri Nets are extended with the capability of time delays to enable the simulation over the time domain. Two different approaches of adding time to Petri Nets are implemented. The holding durations and enabling durations principles provide a suitable basis for modelling time dependent problems. Coloured Petri Nets offer an easier graphical description for building more complex models. Stochastic Petri Nets introduce the use of stochastic time delays realised by stochastic distribution functions. Petri Nets are extended with the capability of modelling scheduling problems whereas arbitrary firing sequences can be defined. If all conflicts are solved beforehand the scheduling problem is modelled by the use of the basic properties of Petri Nets.
All features and functionalities are developed and implemented in the open source MATLAB PetriSimM

Duration	Normal	VR	Difference	%
TA	6555.7	1416.1	5139.6	-78.40%
SA	2543.1	898.3	1644.8	-64.68%

Tab. 3 Time Improvement of Variance Reduction

Runs	Normal	VR	Difference	%
TA	138598	23834	114764	-82.80%
SA	53318	14857	38461	-72.14%

Tab. 4 Reduction of needed Simulation Runs

toolbox which is embedded in the powerful MATLAB environment. The toolbox is suitable for educational purposes as well as for modelling, simulation, and optimisation of real life processes. Several results can be shown and all produced data can be used for internal or external post-processing. Only two steps have to be done by the user of the toolbox. The first one is the development of the Petri Net model, which is building all needed conditions and constraints of the present scheduling problem. The second step is the choice of the best optimisation method and the correct and optimal specification of the needed parameters for the otpimisation algorithm. Three different heuristic methods are implemented and can be selected. Simulated Annealing, Threshold Accepting, and Genetic algorithms are forming the choice for realising the optimisation of scheduling problems. Depending on the present problem each optimisation algorithm has its advantages and disadvantages. No general proof can be done to decide which algorithm fits the best for all problems. Many optimisation studies and runs have to be processed to get significant results because in this case randomness plays a certain role. Furthermore, the search for the optimal and best suited parameters is a highly sophisticated problem and mainly depends on the present problem specification. The heuristics are implemented to form and offer a wide spread basis for the optimisation of scheduling problems. All methods can be easily extended with further functionalities and functions and new algorithms can also be implemented to the open source toolbox.
The implemented methods and functionalities are tested a case study. The optimisation leads to good and similar results for all three algorithms and no significant difference can be determined. Threshold Accepting is the fastest algorithm because of the simpler definition. No randomness is needed to check and to decide if a worse solution is accepted or not. Genetic algorithms are the most extensive method because the computational effort of each generation depends on the population size. The stochastic optimisation is time-consuming because in this case the number of needed simulation runs is modelled by a random number. If the difference of two alternative system configurations is nearby zero, basically a lot of simulation runs are needed to get a significant decision. The use of variance reduction shows a high decrease of computational time because many simulation runs can be saved for each comparison.

ISBN 978-3-901608-32-2 Copyright © 2007 EUROSIM / SLOSIM

7 References

[1] James J. O'Brian. *Scheduling Handbook*. McGraw-Hill Book Company, 1969.

[2] Tadeusz Sawik. *Production Planning and Scheduling in Flexible Assembly Systems*. Springer-Verlag Berlin Heidelberg, 1999.

[3] Joze Balic, Yannis A. Phillis, Nikos Tsourveloudis, and Ivo Pahole. *Flexibility in Manufacturing - Models and Measurement*. University of Maribour, Technical University of Crete, 2002.

[4] Peter Brucker. *Scheduling Algorithms*. Springer-Verlag Berlin Heidelberg, 2001.

[5] Simon French. *Sequencing and Scheduling*. Ellis Horwood Limited, 1982.

[6] B. Griffer and G. L. Thompson. Algorithms for Solving Production-Scheduling Problems. *Operations Research*, 8(4):487–503, 1960.

[7] Christian Kelling. *Simulationsverfahren für zeiterweiterte Petri-Netze*. PhD thesis, Technische Universität Berlin, 1995.

[8] Dirk C. Mattfeld. *Evolutionary Search and the Job Shop*. Physica-Verlag Heidelberg, 1996.

[9] René V. V. Vidal, editor. *Applied Simulated Annealing*. Springer-Verlag Berlin Heidelberg, 1993.

[10] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of State Calculations by Fast Computing Machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[11] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220(4598):671–680, 1983.

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

[13] John H. Holland. *Adaptation in natural and artificial systems*. The MIT Press, 1992.

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

[15] David A. Coley. *An Introduction to Genetic Algorithms for Scientists and Engineers*. World Scientific Publishing Co. Pte. Ltd., 1999.

[16] Lawrence Davis, editor. *Handbook of Genetic Algorithms*. International Thomsen Computer Press, 1996.

[17] Mituso Gen, Yasuhiro Tsujimura, and Erika Kubota. Solving Job-Shop Scheduling Problems by Genetic Algorithm. Proc. 16th International Conference on Computers and Industrial Engineering, Ashikaga, Japan, 1994, pp. 576-579.

[18] D. Goldberg and R. Lingle. Alleles, Loci and the Travelling Salesman. Proc. of the First International Conference on Genetic Algorithms and their Applications, San Matteo, Italy, 1985.

[19] Averill M. Law and W. David Kelton. *Simulation Modelling and Analysis*. McGraw-Hill, Inc., 1991.

[20] Paul Brately, Bennet L. Fox, and Linus E. Schrage. *A Guide to Simulation*. Springer-Verlag Berlin Heidelberg, 1987.

[21] The MathWorks Inc. Matlab connections – third party products and services. http://www.mathworks.com/products/connections, 2006.

[22] M.-H. Matcovschi, C. Mahulea, and O. Pastravanu. Petri Net Toolbox for MATLAB. Proc. MED 2003 Conf., Rhodes, Greece.

[23] TGI group. Petri nets world. University of Hamburg, Germany, 2006, http://www.informatik.uni-hamburg.de/TGI/PetriNets/.

[24] McLeod Institute of Simulation Sciences Hungarian Center. Cassandra 3.0. http://itm.bme.hu/mcleod/cassandra.html, 2002.

[25] Gašper Mušič, Borut Zupančič, and Drago Matko. Petri Net Based Modelling and Supervisory Control Design in Matlab. Proc. EUROCON 2003 Conf., Ljubljana, Slovenia, 362-366.

[26] Thomas Löscher, Felix Breitenecker, and Gašper Mušič. Petri Net Modelling and Simulation in Matlab – A Petri Net Toolbox. Simulation News Europe, Issue 43, July 2005, 20-21.

[27] Thomas Löscher, Felix Breitenecker, Gašper Mušič, and Dejan Gradišar. A Matlab-based Tool for Timed Petri Nets. Proc. ERK 2005 Conf., Portorož, Slovenia, 273-276.

[28] Thomas Löscher, Dejan Gradišar, Felix Breitenecker, and Gašper Mušič. Timed Petri Net Simulation in Matlab: A Production Cell Case Study. Proc. MATHMOD 2006 Conf., Vienna, Austria.

[29] Thomas Löscher and Felix Breitenecker. Petri Net Modelling and Simulation of Production Processes with PetriSimM, a MATLAB-based Toolbox. Proc. 12. ASIM - Fachtagung Simulation in Produktion und Logistik 2006, Kassel, Germany, 313 - 319.

[30] Gašper Mušič, Thomas Löscher, and Dejan Gradišar. An Open Petri Net Modelling and Analysis Environment in Matlab. Proc. IMM 2006 Conf., Barcelona, Spain, 123-128.

[31] Thomas Löscher. Simulationsbasierte optimierung zur verbesserung der produktionsabläufe in einer flexiblen fertigunszelle. Master's thesis, Technische Universität Wien, 2004.

[32] Thomas Löscher, Markus Klug, and Felix Breitenecker. Simulation-based Optimization of Production Plans for a Production Cell using Heuristic Methods - Comparison of Tabu Search, Simulated Annealing and Threshold Accepting. Proc. EUROSIM 2004 Conf., Paris, France.

[33] E. L. Lawler, Jan Karel Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. John Wiley & Sons Ltd, New York, 1985.

ISBN 978-3-901608-32-2 Copyright © 2007 EUROSIM / SLOSIM