EFFICIENT EXPLORATION OF COLOURED PETRI NET BASED SCHEDULING PROBLEM SOLUTIONS

Gašper Mušič[(a)]

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

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

ABSTRACT

The paper deals with simulation-optimization of schedules that are modelled by simple Coloured Petri nets (CPNs). CPN modelling is combined by predefined transition sequence conflict resolution strategy to enable generation of neighbouring solutions that are always feasible. This way standard local search optimization algorithms can be effectively applied to CPN models. Modelling approach and neighbourhood construction procedure are explained in detail. Some preliminary results of tests on standard job shop benchmark problems are provided.

Keywords: Petri nets, simulation, optimization, scheduling, local search

1. INTRODUCTION

Among modelling formalisms suitable for description of systems with highly parallel and cooperating activities, Petri nets are perhaps the most widely used one. With Petri nets, production systems' specific properties, such as conflicts, deadlocks, limited buffer sizes, and finite resource constraints can be easily represented in the model (Tuncel and Bayhan 2007).

Optimization of planning and scheduling problems has been investigated within the production control community for several years. This is one of the fields where information technology has an immediate and considerable impact on the efficiency and quality of production control and related manufacturing processes. The simplicity of model building, the possibility of realistic problem formulation as well as the ability of capturing functional, temporal and resource constraints within a single formalism motivated the investigation of Petri net based optimization of planning and scheduling problems.

In our previous work a simulation based optimization approach applying Petri nets was intensively studied, as well as other, more classical approaches, such as dispatching rules and reachability tree based heuristic search (Gradišar and Mušič 2007, Löscher, Mušič and Breitenecker 2007, Mušič, Löscher and Breitenecker 2008, Mušič 2008).

In the reported works, Petri nets based scheduling methods were compared and a certain level of experience was gained about the behaviour of the methods in relation to different scales of problems. Among others, reachability tree based heuristic search methods (Lee and DiCesare 1994, Yu, Reyes, Cang and Lloyd 2003, Mujica, Piera and Narciso 2010) were of particular interest, since the rich structural analysis framework of Petri nets seemed a promising way to derive a suitable heuristic function that would significantly improve the efficiency of the search and obtained results.

The obtained results meet the expectations for small or moderate size problems. Unfortunately, the results for complex problems, such as standard job-shop benchmark problems (Taillard 1993), are not satisfactorily, even with the advanced heuristic functions.

Local search methods, on the other hand, are widely used in operational research (OR) community (Blazewicz, Domschke and Pesch 1996, Vaessens, Aarts and Lenstra 1996). Their performance is not significantly decreased for larger problems. The optimality of obtained solutions is not guaranteed (Pinedo 2008) but they can be obtained in computational time that is significantly shorter compared to other methods. In particular, the Tabu search algorithms (Dell'Amico and Trubian 1993, Taillard 1994, Nowicki and Smutnicki 1996) are claimed to represent the state-of-the-art by a comfortable margin over the closest competition (Watson, Whitley and Howe 2005).

This motivated the investigations on combination of Petri net modelling approach and local search methods (Löscher, Mušič and Breitenecker 2007). In particular, a combination of efficient generation of feasible neighbouring solutions from Coloured Petri net representation of the problem and local search is tested in this paper.

The paper shows how a Coloured Petri net model of a scheduling problem can be used in conjunction with state-of-the-art local search algorithms provided a special type of parameterized conflict resolution strategy and neighbouring solution generation procedure are adopted. Parameters in a form of sequence vectors are adjoined to shared resources in the system. The makespan of a feasible schedule is calculated through CPN model simulation, which is supervised by the sequence vectors. Constrained permutations on these vectors are used to generate neighboring schedule solutions that are always feasible, which improves the effectiveness of CPN based exploration of solutions compared to previous works.

2. JOB-SHOP SCHEDULING AND LOCAL SEARCH

A job-shop scheduling problem is a well known problem in the field of operations research. It is defined as the determination of the order in which a set of jobs (tasks) $\{J_i | i = 1, \ldots, n\}$ is to be processed through a set of machines (resources) $\{M_k | k = 1, \ldots, m\}$.

Job i is specified by a set of operations $\{o_j | j = 1, \ldots, m_j\}$ representing the processing requirements on various machines. Processing times are assigned to individual operations. Job shop problem assumes that all jobs have to be processed on all machines while the operations processing order (routing) is fixed but not identical for individual jobs. The objective of a job shop scheduling problem is most often to determine a job sequence schedule for every machine, which will minimize the total processing time, i.e., makespan.

This objective can be reached with various strategies including heuristic based dispatching rules, simulation-optimization and local search methods.

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 $N(s) = \{F(s) \,|\, F \in AM\}$.

A general local search method may be described as follows. Each iteration starts with a solution $s \in S$ and then a solution $s' \in N(s)$ or a modification $F \in AM$ which provides $s' = F(s)$ is chosen. Based on the values of the objective function $f : S \rightarrow \mathbb{R}$, $f(s)$ and $f(s')$, the new solution is adopted or discarded. The next iteration starts with either the old or the new solution. Different methods of choice of solution for the next iteration lead to different local search techniques, e.g. simulated annealing, tabu search, and genetic algorithms (Brucker 2001).

Local search algorithms are simple to implement and fast in execution, but they have the main disadvantage that they can terminate in the first local minimum, which might give an objective function that deviates substantially from the global minimum. The useful algorithms should be able to leave the local minimum by sometimes accepting transitions leading to an increase in the objective function. Simulated Annealing is an example of such an approach where cost-increasing transitions are accepted with a non-zero probability which decreases gradually as the algorithm continues its execution (van Laarhoven, Aarts and Lenstra 1992).

The previously described scheduling methods can be used in combination with various modelling formalisms. As mentioned in the introduction, a Coloured Petri net framework will be used here.

3. COLOURED PETRI NETS

Coloured Petri nets (CPNs) used for modelling o scheduling problems in this paper are defined as follows. Note that the definition is different from (Jensen 1997) in the sense that it does not alow for transition guards. Instead it closely follows one of the representations used in (Basile, Carbone and Chiacchio 2007) with an important difference: a different interpretation of transition delays is used, which is closer to that of (Jensen 1997).

A $CPN = (\mathcal{N}, M_0)$ is a Coloured Petri net system, where: $\mathcal{N} = (P, T, Pre, Post, Cl, Co)$ is a Coloured Petri net structure:

- $P = \{p_1, p_2, \ldots, p_k\}, k > 0$ is a finite set of places.
- $T = \{t_1, t_2, \ldots, t_l\}, l > 0$ is a finite set of transitions (with $P \cup T \neq \emptyset$ and $P \cap T = \emptyset$).
- Cl is a set of colours.
- $Co : P \cup T \rightarrow Cl$ is a colour function defining place marking colours and transition occurrence colours. $\forall p \in P, Co(p) = \{a_{p,1}, a_{p,2}, \ldots, a_{p,u_p}\} \subseteq Cl$ is the set of u_p possible colours of tokens in p, and $\forall t \in T, Co(t) = \{b_{t,1}, b_{t,2}, \ldots, b_{t,v_t}\} \subseteq Cl$ is the set of v_t possible occurrence colours of t.
- $Pre(p,t) : Co(t) \rightarrow Co(p)_{MS}$ is an element of the pre-incidence function and is a mapping from the set of occurrence colours of t to a multiset over the set of colours of p, $\forall p \in P, \forall t \in T$. It can be represented by a matrix whose generic element $Pre(p,t)(i,j)$ is equal to the weight of the arc from p w.r.t colour $a_{p,i}$ to t w.r.t colour $b_{t,j}$. When there is no arc with respect to the given pair of nodes and colours, the element is 0.
- $Post(p,t) : Co(t) \rightarrow Co(p)_{MS}$ is an element of the post-incidence function, which defines weights of arcs from transitions to places with respect to colours.

$M(p) : Co(p) \rightarrow \mathbb{N}$ is the marking of place $p \in P$ and defines the number of tokens of a specified colour in the place for each possible token colour in p. Place marking can be represented as a multiset $M(p) \in Co(p)_{MS}$ and the net marking M can be represented as a $k \times 1$ vector of multisets $M(p)$. M_0 is the initial marking of a Colured Petri net.

3.1. Timed models

As described in (Bowden 2000), there are three basic ways of representing time in Petri nets: firing durations (FD), holding durations (HD) and enabling durations (ED). The FD 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. When using HD principle, a firing has no duration but 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. With ED principle, the firing of the transitions has no duration while the time delays are represented by forcing transitions that are enabled to stay so for a specified period of time before they can fire.

The ED concept is more general than HD. Furthermore, in (Lakos and Petrucci 2007) an even more general concept is used, which assigns delays to individual arcs, either inputs or outputs of a transition. This way both ED and HD concepts are covered, and the enabling delay may even depend on the source of transition triggering while holding delay may differ among different activities started by the same transition.

When modelling several performance optimization problems, e.g. scheduling problems, such a general framework is not needed. It is natural to use HD 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. HD principle is also used in timed version of CPNs defined by Jensen, although the unavailability of the tokens is only defined implicitly through the corresponding time stamps. While CPNs allow the assignment of delays both to transition and to output arcs, we further simplify this by allowing time delay inscriptions to transitions only. This is sufficient for the type of examples investigated here, and can be generalized if necessary.

To include a time attribute of the marking tokens, which implicitly defines their availability and unavailability, the notation of (Jensen 1997) will be adopted. Colours are adjoined to token number by ‘c notation and coloured tokens are accompanied with a timestamp, which is written next to the token number and colour and separated from the colour by @. E.g., two c-coloured tokens with time stamp 10 are denoted 2‘c@10. A collection of tokens with different colours and/or time stamps is defined as a multiset, and written as a sum (union) of sets of timestamped coloured tokens. E.g., two c-coloured tokens with time stamp 10 and three d-coloured tokens with timestamp 12 are written as 2‘c@10+3‘d@12. The timestamp of a token defines the time from which the token is available.

Time stamps are elements of a time set TS, which is defined as a set of numeric values. In many software implementations the time values are integer, i.e. $TS = \mathbb{N}$, but will be here admitted to take any positive real value including 0, i.e. $TS = \mathbb{R}_0^+$. Timed markings are represented as collections of time stamps and are multisets over TS: TS_{MS}. By using HD principle the formal representation of a Coloured Timed Petri net is defined as follows.

$CTPN = (\mathcal{N}, M_0)$ is a Coloured Timed Petri net system, where:

- $\mathcal{N} = (P, T, Pre, Post, Cl, Co, f)$ is a Coloured Time Petri net structure with $(P, T, Pre, Post, Cl, Co)$ as defined above.
- $f : Co(t) \rightarrow TS$ is the time function that assigns a non-negative deterministic time delay to every occurrence colour of transition $t \in T$.
- $M(p) : Co(p) \rightarrow TS_{MS}$ is the timed marking, M_0 is the initial marking of a timed Petri net.

3.2. Firing rule

Functions Pre and $Post$ define the weights of directed arcs, which are represented by arc inscriptions in the matrix form. In the case when the all the weights in the matrix are 0, the arc is omitted. Let ${}^\bullet t_b \subseteq P \times Cl$ denote the set of places and colours which are inputs to occurrence colour $b \in Co(t)$ of transition $t \in T$, i.e., there exists an arc from every $(p, a) \in {}^\bullet t$ to t with respect to colours $a \in Co(p)$ and $b \in Co(t)$.

To determine the availability and unavailability of tokens, two functions on the set of markings are defined. The set of markings is denoted by $\mathbb{M}$. Given a marking and model time, $m : P \times \mathbb{M} \times TS \rightarrow Co(p)_{MS}$ defines the number of available coloured tokens, and $n : P \times \mathbb{M} \times TS \rightarrow Co(p)_{MS}$ the number of unavailable coloured tokens for each place of a TPN at a given model time $\tau_k \in TS$.

Two timed markings can be added (denoted $+_\tau$) in a similar way as multisets, i.e. by making a union of the corresponding multisets. The definition of subtraction is somewhat more problematic. To start with, a comparison operator is defined. Let M_1 and M_2 be markings of a place $p \in P$. By definition, $M_1 \geq_\tau M_2$ iff $m(p, M_1, \tau_k) \geq m(p, M_2, \tau_k), \forall \tau_k \in TS, \forall a \in Co(p)$.

Similarly, the subtraction is defined by the number of available tokens, and the subtrahend should not contain any unavailable tokens. Let M_1, M_2 and M_3 be markings of a place $p \in P$, $M_1 \geq_\tau M_2$, and $m(p, M_1, \tau_k)$, $m(p, M_2, \tau_k)$, and $m(p, M_3, \tau_k)$, be the corresponding numbers of available tokens at time τ_k, and $n(p, M_2, \tau_k) = 0$. The difference $M_3 = M_1 -_\tau M_2$ is then defined as any $M_3 \in \mathbb{M}$ having $m(p, M_3, \tau_k) = m(p, M_1, \tau_k) - m(p, M_2, \tau_k)$.

Using the above definitions, the firing rule of a CTPN can be defined. Given a marked $CTPN = (\mathcal{N}, M)$, a transition t is time enabled at time τ_k w.r.t occurrence colour $b \in Co(t)$, denoted $M[t_b\rangle_{\tau_k}$ iff $m(p, M, \tau_k) \geq Pre(p, t)(b)$, $\forall p \in {}^\bullet t$. An enabled occurrence transition can fire, and as a result removes tokens from input places and creates tokens in output places. If transition t fires w.r.t occurrence colour b, then the new marking is given by $M'(p) = M(p) -_\tau Pre(p, t)(b)@\tau_k +_\tau Post(p, t)(b)@(\tau_k + f(t, b)), \forall p \in P$. Here the subtraction operation is implemented in such a way that in case of several choices, the token with the oldest timestamp is always removed first. If marking M_2 is reached from M_1 by firing t_b at time τ_k, this is denoted by $M_1[t_b\rangle_{\tau_k} M_2$. The set of markings of TPN $\mathcal{N}$ reachable from M is denoted by $R(\mathcal{N}, M)$.

4. COLOURED PETRI NET MODELLING OF SCHEDULING PROBLEMS

An important concept in PNs is that of conflict. Two transition firings 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 1. 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 opera-

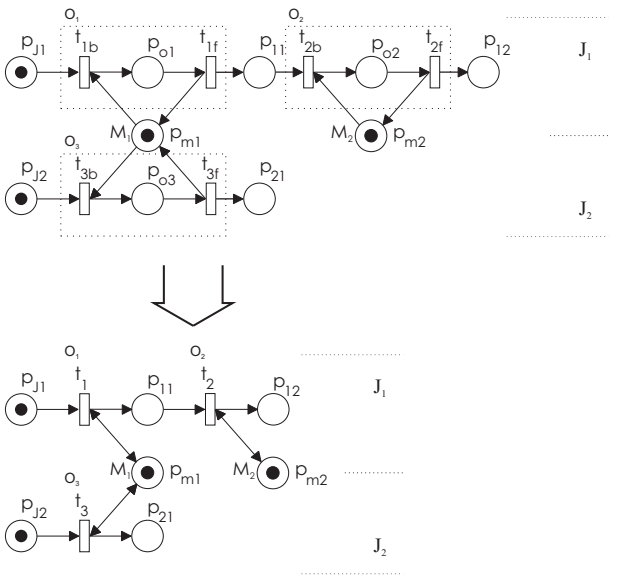

Figure 1: A PN model of a simple scheduling problem

tions, 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.

Place p_{m1} is a resource place. It models the machine M_1 and is linked to t_{1b} and t_{3b}, which start two distinct operations. Clearly, the conflict between t_{1b} and t_{3b} 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.

The transitions that model finishing of operation (t_{if} in Figure 1) are not relevant for scheduling and can be removed. Same holds for intermediate buffer places since the holding duration interpretation of transition delays guarantees that a subsequent transition can not fire before the precedent transition delay expires. The model can be therefore simplified as shown in the lower part of Figure 1. The occupation of a shared resource M_i during the evolution of the system is marked by a presence of unavailable token in the corresponding place p_{mi}.

With the introduction of token and occurrence colours, the resource sharing as described above can be represented in even much more compact model. Several jobs that go through a similar operation sequence can be folded together and represented by a single place/transition sequence with different token colours. The transition occurrence colours enable to distinguish different jobs both in terms of operation durations as well as in terms of their dependence on shared resources. The model from Figure 1 therefore maps to the model in Figure 2. The two jobs are represented by two token colours while a third colour is added to model resource availability. Both remaining transitions appear with two occurrence colours to model different durations where the absence of the second operation in

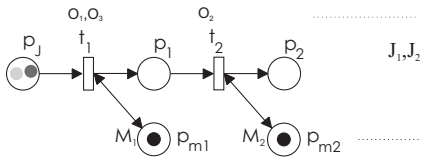

Figure 2: A CTPN model of a simple scheduling problem

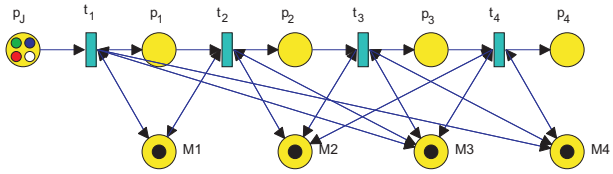

Figure 3: A simple job shop problem

Table 1: Operation durations for a simple job shop problem

Operation\Job	J_1	J_2	J_3	J_4
o_1	54	9	38	95
o_2	34	15	19	34
o_3	61	89	28	7
o_4	2	70	87	29

Table 2: machine requirements for a simple job shop problem

Operation\Job	J_1	J_2	J_3	J_4
o_1	3	4	1	1
o_2	1	1	2	3
o_3	4	2	3	2
o_4	2	3	4	4

job J_2 is simply modelled by setting the duration to zero.

A more elaborated example is shown in Figure 3. The model is based on a test example from Taillard (1993). It consists of four jobs and four machines. Every job includes four operations. Operation durations are shown in Table 1 and resource requirements in Table 2. Note that arc weights are not shown in the figure, they will be shown in the sequel. Nevertheless, only the arcs with at least one nonzero weight for any occurrence colour are shown.

Further compaction can be achieved by folding the operation places. Job sequences, operation durations and resource requirements are coded by different sets of colours and corresponding transition guards and expressions (Mujica, Piera and Narciso 2010). Since the transition guards and expressions are not supported by the type of CTPNs used in this paper, this representation can not be used here. The proposed representation is therefore not the most compact one but has the advantage of a very efficient coding in a general mathematical analysis software, e.g. Matlab.

In Matlab, the flow matrices of a CPN can be represented as cell matrices of size $|P| \times |T|$, where each element is a cell containing weight matrix of size $|Co(p)| \times |Co(t)|$. E.g. for example in Figure 3 the corresponding

pre-incidence matrix is

$$Pre = \begin{bmatrix} I_4 & 0 & \cdots & 0 \\ 0 & I_4 & 0 & 0 \\ 0 & 0 & I_4 & 0 \\ 0 & \cdots & 0 & I_4 \\ 0 & & \cdots & 0 \\ R_{11} & R_{12} & R_{13} & R_{14} \\ R_{21} & R_{22} & R_{23} & R_{24} \\ R_{31} & R_{32} & R_{33} & R_{34} \\ R_{41} & R_{42} & R_{43} & R_{44} \end{bmatrix} \quad (1)$$

where I_4 stands for 4×4 identity matrix and zeros should be interpreted as 4×4 zero matrices. R_{ij} define i-th resource requirements of j-th operation within jobs:

$$\begin{array}{ll} R_{11} = \begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix} & R_{12} = \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix} \\ R_{21} = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix} & R_{22} = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \\ R_{31} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} & R_{32} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \\ R_{41} = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} & R_{42} = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix} \\ R_{13} = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix} & R_{14} = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix} \\ R_{23} = \begin{bmatrix} 0 & 1 & 0 & 1 \end{bmatrix} & R_{24} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \\ R_{33} = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} & R_{34} = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \\ R_{43} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} & R_{44} = \begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix} \end{array} \quad (2)$$

Furthermore, the cell matrix can be any time converted to an incidence matrix of the corresponding unfolded P/T net and reverse, the P/T net can be folded back. The only information necessary consists of the place and transition colour sets of individual nodes in the CPN.

For example, the Matlab command

```
>> PTPre=cell2mat(Pre)
```

unfolds the pre-incidence matrix from the previous example into a pre-incidence matrix of an equivalent P/T Petri net and the command

```
>> Pre=mat2cell(PTPre, ncolP, ncolT)
```

reproduces back the original cell matrix, provided that vectors $ncolP$ and $ncolT$ contain information about numbers of token and occurrence colours for all $p \in P$ and $t \in T$. E.g., for the above example $ncolP = \begin{bmatrix} 4 & 4 & 4 & 4 & 4 & 1 & 1 & 1 & 1 \end{bmatrix}^T$ and $ncolP = \begin{bmatrix} 4 & 4 & 4 & 4 \end{bmatrix}^T$.

This way the CPN framework can be used to efficiently encode various scheduling problems into a compact representation. Later the CPN representation can be analyzed directly, or can be translated into an equivalent P/T Petri net, which enables the application of standard PN analysis methods as well as PN based scheduling techniques.

4.1. Derivation of optimal or sub-optimal schedules

A derived Coloured 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** (Haupt 1989), such as Shortest Processing Time (SPT) or Longest Processing Time (LPT), can be introduced when solving the conflicting situations. 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. The schedule of process operations can be determined by observing the marking evolution of the net. Depending on the given scheduling problem a convenient rule should be chosen. Usually, different rules are needed to improve different predefined production objectives (makespan, throughput, production rates, and other temporal quantities).

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.

Recent reports in scheduling literature show an increased interest in the use of **meta-heuristics**, such as genetic algorithms (GA), simulated annealing (SA), and tabu search (TS). Meta-heuristics have also been combined with Petri net modelling framework to solve complex scheduling problems (Tuncel and Bayhan 2007). With such an approach, the modelling power of Petri nets can be employed, and relatively good solutions of scheduling problems can be found with a reasonable computational effort. Compared to reachability tree based search methods, meta-heuristics require less memory.

5. COLOURED PETRI NET SIMULATION BASED EXPLORATION OF THE SOLUTION SPACE

In our previous work (Löscher, Mušič and Breitenecker 2007, Mušič, Löscher and Breitenecker 2008) different ways of solution space exploration were studied. Extensive testing of the reachability tree search based approaches has been performed. The approach is very general, as it can be applied to any kind of scheduling problem that can be represented as a Petri net. Unfortunately, the approach does not perform very well on the standard job shop benchmarks (Mušič 2008). This motivated the exploration of alternative approaches, including local search based techniques.

In (Löscher, Mušič and Breitenecker 2007) the approach is presented, which extends the Petri net representation by sequences and priorities. Priorities are used as a way of parametrizing the conflict resolution strategy. For this purpose a priority ranking is assigned to transitions. If there is a conflict between a pair of transitions the transition with higher priority will fire.

Another way of parametrization is to select disjoint groups of transitions and map them to sequences. A firing list is defined by ordering transitions within the group. During the model evolution a set of sequence counters is maintained and all transitions belonging to sequences are disabled except of transitions corresponding to the current

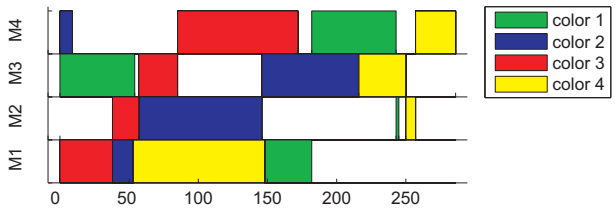

Figure 4: A possible solution of the given job-shop problem

state of the sequence counters. After firing such a transition the corresponding sequence counter is incremented.

This way the transition firing sequence can be parameterized. If the model represents a scheduling problem, the sequence obtained by a simulation run of the Petri net model from the prescribed initial to the prescribed final state is a possible solution to the problem, i.e. it represents a feasible schedule.

E.g., the model from Figure 3 can be simulated by applying SPT rule (Haupt 1989) as a default conflict resolution mechanism. The resulting sequence represents a possible schedule, shown in Figure 4.

The same schedule can be obtained by fixing the sequential order of transitions in conflicts related to shared resources in the system. E.g. in the above example the shared resources are machines M1 to M4. Related sets of transitions are:

$$\begin{aligned} S_{M1} &= \{t_{1,c3}, t_{1,c4}, t_{2,c1}, t_{2,c2}\} \\ S_{M2} &= \{t_{2,c3}, t_{3,c2}, t_{3,c4}, t_{4,c1}\} \\ S_{M3} &= \{t_{1,c1}, t_{2,c4}, t_{3,c3}, t_{4,c2}\} \\ S_{M4} &= \{t_{1,c2}, t_{3,c1}, t_{4,c3}, t_{4,c4}\} \end{aligned} \quad (3)$$

where $t_{i,cj}$ denotes cj occurrence colour of t_i and colour cj corresponds to job J_j.

If these sets are mapped to four independent sequences, and a set of index vectors

$$V = \{V_1, V_2, V_3, V_4\}$$

is adjoined, where V_i is a corresponding permutation of integer values i, $1 \leq i \leq 4$:

$$\begin{aligned} V_1 &= \{1, 4, 2, 3\} \\ V_2 &= \{1, 2, 4, 3\} \\ V_3 &= \{1, 3, 4, 2\} \\ V_4 &= \{1, 3, 2, 4\} \end{aligned} \quad (4)$$

a supervised simulation run, which forces the prescribed sequential order of conflicting transitions, results in the same schedule as above.

The sequence supervised simulation is implemented by a simple modification of the regular CTPN simulation algorithm. After the enabled transitions are determined in each simulation step, the compliance of the set of enabled transitions to the state of the sequence counters is checked. Transitions that take part in defined sequences but are not pointed to by a counter are disabled.

The exploration of the solution space and the related search for the optimal schedule can then be driven by modifications of sequence index vectors. Such a modification leads to a neighbourhood solution of a given solution and the related modification is usually defined through a neighbourhood function.

In the work of (Löscher, Mušič and Breitenecker 2007) several neighbourhood functions as well as different local search strategies were implemented in the PetriSimM toolbox for Matlab and some results are shown in (Löscher, Mušič and Breitenecker 2007, Mušič, Löscher and Breitenecker 2008).

5.1. Generation of feasible neighbourhood solutions from a CTPN model

The problem in the previously described approach is that by perturbing sequence index vectors the resulting transition firing sequence may easily become infeasible, which results in a deadlock during simulation. The search procedures implemented in PetriSimM were designed so that such an infeasible solution is ignored and a new perturbation is tried instead. While this works for many problems, in some cases the number of feasible sequences is rather low and such an algorithm can easily be trapped in an almost isolated point in the solution space.

The job shop scheduling approaches reported in the OR literature started to address the issue of efficient neighbourhood generation quite a while ago. With the wide acceptance of the Tabu search algorithm as the most promising methods for schedule optimisation the design of efficient neighborhood generation operator become the central issue and several such operators have been proposed (Blazewicz, Domschke and Pesch 1996, Jain, Rangaswamy and Meeran 2000, Watson, Whitley and Howe 2005).

The question is how to link these operators and related effective schedule optimization algorithms with Coloured Petri net representation of scheduling problems. As mentioned above the Petri net scheduling methods have advantages in unified representation of different aspect of underlying manufacturing process in a well defined framework. Unfortunately, the related optimization methods are not as effective as some methods developed in the OR field. The link of two research areas could be helpful in bridging the gap between highly effective algorithms developed for solving academic scheduling benchmarks and complex real-life examples where even the development of a formal model can be difficult (Gradišar and Mušič 2007).

A possible way of such a link is the establishment of a correspondence of a critical path and the sequence index vectors described previously.

In a given schedule the critical path CP is the path between the starting and finishing time composed of consequent operations with no time gaps:

$$CP = \{O_i : \rho_i = \rho_{i-1} + \tau_{i-1}, i = 2 \ldots n\} \quad (5)$$

where O_i are operations composing the path, ρ_i is the release (starting) time of operation O_i, and τ_i is the duration of O_i.

The operations O_i on the path are critical operations. Critical operations do not have to belong to the same machine (resource) but they are linked by starting/ending times.

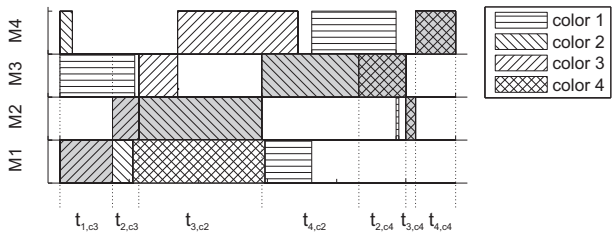

Figure 5: A critical path within a schedule and critical transitions

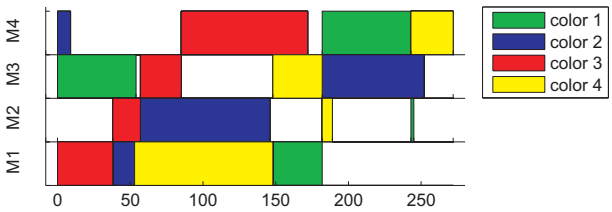

Figure 6: An optimized solution of the given job-shop problem

Critical path can be decomposed in a number of blocks. A block is the longest sequence of adjacent critical operations that occupy the same resource.

The length of the path equals the sum of durations of critical operations and defines the makespan C_{max}:

$$C_{max} = \sum_{O_i \in CP} \tau_i \tag{6}$$

Figure 5 shows a redrawn gantt chart from Figure 4 with indication of the critical path and the sequence of critical operations. The shown critical path consists of 5 blocks.

Critical operations in Figure 5 are denoted by transition labels that trigger the start of a critical operation when fired. A transition that triggers a critical operation will be called a critical transition.

The scheduling literature describes several neighborhoods based on manipulations (moves) of critical operations (Blazewicz, Domschke and Pesch 1996). One of the classical neighborhoods is obtained by moves that reverse the processing order of an adjacent pair of critical operations belonging to the same block (van Laarhoven, Aarts and Lenstra 1992). Other neighbourhoods further restrict the number of possible moves on the critical path, e.g. (Nowicki and Smutnicki 1996).

Clearly every critical transition participates in one of the conflicts related to shared resources, e.g. sets (3) for the given case. If these transitions are linked to predefined firing sequences parameterized by index vectors V_i (4), a move operator corresponds to a permutation of an index vector.

For example, in the schedule shown in Figure 5 a move can be chosen, which swaps the two operations in the third block on the critical path. This corresponds to the swap of transitions $t_{4,c2}$ and $t_{2,c4}$ in the sequence S_{M3}, which is implemented by the exchange of third and fourth element within V_3 index vector:

$$move(V_3) : \{1, 3, 4, 2\} \mapsto \{1, 3, 2, 4\}$$

A new schedule obtained by simulation with modified V_3 is shown in Figure 6.

When the move is limited to swap of a pair of the adjacent operations in a block on the critical path this narrows down the set of allowed permutations. The most important feature of such a narrowed set of permutation on the index vector is that every permutation from this set will result in a feasible firing sequence, i.e. a feasible schedule. Therefore no deadlock solutions can be generated, which are often encountered when unrestricted permutations on the index vectors are used.

Based on this observation a set of neighbourhood functions can be defined which limit the permutations of the index vectors in a way that will produce feasible firing sequences only. Several widely used move operators can be implemented. Such a neighbourhood function permits the optimisation of schedules represented as PN or CTPN models by a wide variety of local search optimization techniques.

It is also important to note that such a neighbourhood function is comparable to exploring the reachability tree in an event driven manner. It is possible that certain feasible firing sequence imposes one or more intervals of idle time between transitions, i.e. some transitions are enabled but can not fire due to sequence restrictions. This is different from the exploration in a time driven manner when a transition has to be fired whenever at least one transition is enabled. The difference is important in cases when the optimal solution can be missed unless some idle time is included in the schedule as shown in (Piera and Mušič 2011).

The described neighbourhood generation procedure was coded in Matlab and used in combination with a simple Simulated annealing (SA) search algorithm. Comparison of the minimum makespan for the above job shop problem calculated by the proposed algorithm and some other standard algorithms is shown in Table 3. SA-SPT denotes the combined algorithm with the Simulated Annealing and the SPT rule (Mušič 2009), RT-search stands for a reachability tree based heuristic search (Lee and DiCesare 1994, Yu, Reyes, Cang and Lloyd 2003), and SA-CPN-N1 denotes Simulated Annealing and a CPN-based neigbourhood function as proposed in this paper.

Clearly, the reachability tree based search and SA-CPN-N1 outperform other algorithms with regard to the result. It must be noted, however, that the computational effort in the case of reachability tree based search is much higher.

Table 3: Calculated makespan for a simple job shop problem

Algorithm	Makespan
SPT	286
LPT	341
RT-search	272
SA-SPT	286
SA-CPN-N1	272

Table 4: Calculated makespan for a set of 15 jobs/15 machines problems

Algorithm	Makespan				
	ta01	ta02	ta03	ta04	ta05
SPT*	1462	1429	1452	1668	1618
LPT*	1701	1674	1655	1751	1828
RT-search	1592	1465	1637	1590	1568
SA-SPT	1359	1358	1352	1362	1352
SA-CPN-N1	1299	1326	1357	1353	1344
optimum	1231	1244	1218	1175	1224

* min out of 100 runs

Results in Table 3 were obtained by implementation of simple neighbourhood based on N1 move operator of (van Laarhoven et al. 1992) which was also used in (Taillard 1993) - the notation N1 is taken from (Blazewicz, Domschke and Pesch 1996). Only a single critical path was considered. Other neighbourhoods can be easily implemented and these as well as some other extensions of neighbourhood generation algorithm are currently being tested.

The computational complexity drawback of reachability tree based search is much more obvious with complex problems. Table 4 shows the preliminary results of a set of standard benchmark problems with 15 jobs and 15 machines (Taillard 1993). For reference also the optimal values are listed (source: *http://mistic.heig-vd.ch/taillard/*). The reachability tree based search has to be limited to predefined maximum tree size in order to complete in a reasonable time.

In contrast to that the SA-SPT and proposed SA-CPN-N1 algorithms are able to improve the initial SPT solutions with a moderate effort. A prototype implementation of tabu search algorithm (TS-CPN-N1) has also been tested and the obtained results are comparable to the SA based search. It is expected that the tests with other neighbourhood operators would further improve the obtained results, which is one of the tasks for the future work.

6. CONCLUSIONS

The presented results indicate that the proposed combination of CPN models, sequence based conflict resolution and local search performs relatively well with a moderate computational effort. The approach may be interesting for practice, in particular because of the ability to use various existing PN or CPN models of different problems. In general, any scheduling problem can be optimized that can be represented by a timed Petri net in such a way that relations among jobs and shared resources are fixed and a shared resource always participates in the given job's operation sequence, regardless or the determined schedule.

The Petri net scheduling methods have advantages in unified representation of different aspect of underlying manufacturing process in a well defined framework. The investigations show, however, that the related optimization methods are not as effective as some methods developed in the Operations Research field. The link of two research areas could be helpful in bridging the gap between highly effective algorithms developed for solving academic scheduling benchmarks and complex real-life examples where even the development of a formal model can be difficult.

ACKNOWLEDGMENTS

The presented work has been partially performed within Competence Centre for Advanced Control Technologies, an operation co-financed by the European Union, European Regional Development Fund (ERDF) and Republic of Slovenia, Ministry of Higher Education, Science and Technology.

REFERENCES

Basile, F., Carbone, C. and Chiacchio, P., 2007. Simulation and analysis of discrete-event control systems based on Petri nets using PNetLab, *Control Engineering Practice*, 15, 241–259.

Blazewicz, J., Domschke, W. and Pesch, E., 1996. The job shop scheduling problem: Conventional and new solution techniques, *European Journal of Operational Research*, 93, 1–33.

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

Brucker, P., 2001. *Scheduling Algorithms*, Springer-Verlag Berlin Heidelberg.

Dell'Amico, M. and Trubian, M., 1993. Applying tabu search to the job-shop scheduling problem, *Ann. Oper. Res.*, 41, 231–252.

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.

Haupt, R., 1989. A survey of priority rule-based scheduling, *OR Spectrum*, 11 (1), 3–16.

Jain, A., Rangaswamy, B. and Meeran, S., 2000. New and "stronger" job-shop neighborhoods: A focus on the method of nowicki and smutnicki(1996), *Journal of Heuristics*, 6 (4), 457–480.

Jensen, K., 1997. *Coloured Petri Nets: Basic Concepts,Analysis Methods and Practical Use*, Vol. 1, 2 edn, Springer-Verlag, Berlin.

Lakos, C. and Petrucci, L., 2007. Modular state space exploration for timed Petri nets, *International Journal on Software Tools for Technology Transfer*, 9, 393–411.

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.

Mujica, M., Piera, M. A. and Narciso, M., 2010. Revisiting state space exploration of timed coloured petri net models to optimize manufacturing system's performance, *Simulation Modelling Practice and Theory*, 18, 1225–1241.

Mušič, G., 2008. Timed Petri net simulation and related scheduling methods: a brief comparison, *The 20th European Modeling & Simulation Symposium*, Campora S. Giovanni (Amantea, CS), Italy, pp. 380–385.

Mušič, G., 2009. Petri net base scheduling approach combining dispatching rules and local search, *21th European Modeling & Simulation Symposium*, Vol. 2, Puerto de La Cruz, Tenerife, Spain, pp. 27–32.

Mušič, G., Löscher, T. and Breitenecker, F., 2008. Simulation based scheduling applying Petri nets with sequences and priorities, *UKSIM 10th International Conference on Computer Modelling and Simulation*, Cambridge, UK, pp. 455–460.

Nowicki, E. and Smutnicki, C., 1996. A fast taboo search algorithm for the job shop problem, *Management Science*, 42 (6), 797–813.

Piera, M. A. and Mušič, G., 2011. Coloured Petri net scheduling models: Timed state space exploration shortages, *Math.Comput.Simul.*, p. in press.

Pinedo, M. L., 2008. *Scheduling: Theory, Algorithms, and Systems*, 3rd edn, Springer Publishing Company, Incorporated.

Taillard, E., 1993. Benchmarks for basic scheduling problems, *European Journal of Operational Research*, 64, 278–285.

Taillard, E. D., 1994. Parallel taboo search techniques for the job shop scheduling problem, *Informs Journal on Computing*, 6, 108–117.

Tuncel, G. and Bayhan, G. M., 2007. Applications of Petri nets in production scheduling: a review, *International Journal of Advanced Manufacturing Technology*, 34, 762–773.

Vaessens, R. J. M., Aarts, E. and Lenstra, J., 1996. Job shop scheduling by local search, *INFORMS Journal on Computing*, 8, 302–317.

van Laarhoven, P., Aarts, E. and Lenstra, J., 1992. Job shop scheduling by simulated annealing, *Operations Research*, 40, 113–125.

Watson, J. P., Whitley, L. D. and Howe, A. E., 2005. Linking search space structure, run-time dynamics, and problem difficulty: A step toward demystifying tabu search, *Journal of Artificial Intelligence Research*, 24, 221–261.

Yu, H., Reyes, A., Cang, S. and Lloyd, S., 2003. 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.

AUTHOR BIOGRAPHY

GAŠPER MUŠIČ received B.Sc., M.Sc. and Ph.D. degrees in electrical engineering from the University of Ljubljana, Slovenia in 1992, 1995, and 1998, respectively. He is Associate Professor at the Faculty of Electrical Engineering, University of Ljubljana. His research interests are in discrete event and hybrid dynamical systems, supervisory control, planning, scheduling, and industrial informatics. His Web page can be found at `http://msc.fe.uni-lj.si/Staff.asp`.