Schedule Optimization based on Coloured Petri Nets and Local Search

Gašper Mušič

University of Ljubljana, Faculty of Electrical Engineering, Tržaška 25, 1000 Ljubljana, Slovenia;
gasper.music@fe.uni-lj.si

Simulation Notes Europe SNE 23(1), 2013, 9 - 16
DOI: 10.11128/sne.23.tn.10163
Received: Oct. 9, 2012 (Selected MATHMOD 2012 Postconf. Publ.); Revised Accepted: March 10, 2013;

Abstract. The contribution deals with simulation-optimization of schedules that are modelled by simple Coloured Petri Nets (CPNs). CPN modelling of standard classes of scheduling problems is addressed and compact CPN representations of scheduling problems are proposed. It is shown how a combination of CPN representations with predefined transition sequence conflict resolution strategy can be used to optimize schedules by standard local search optimization algorithms. Possible neighbourhood construction procedures for various problem classes are proposed with the emphasis on solutions feasibility.

Introduction

Petri nets compose a general modelling formalism suitable for description of systems with highly parallel and cooperating activities. Among others, they are increasingly used for modelling and analysis in the field of manufacturing systems. An important advantage of Petri nets is that production systems' specific properties, such as conflicts, deadlocks, limited buffer sizes, and finite resource constraints can be easily represented within a single formal model (Tuncel and Bayhan, 2007).

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 manufacturing 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 et al., 2007; Mušič, 2008).

In particular, the investigations focused on combinations of Petri net modelling approach and local search methods (Löscher et al., 2007; Mušič, 2009).

This paper focuses on Coloured Petri Net (CPN) models of various classes of scheduling problems. General modelling guidelines for standard problems, such as open shop, flow shop and job shop problems are proposed. The paper also explores the possibilities of use of the CPN models 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. Constrained permutations on transition firing vectors are used to generate neighbouring schedule solutions that are always feasible, which improves the effectiveness of CPN based exploration of solutions compared to previous works.

1 CPN Representations of Scheduling Problems

Literature on deterministic scheduling classifies the manufacturing scheduling problems according to the machine environment structure, processing characteristics and constraints, and objectives (Pinedo, 2008). Standard machine environment structures lead to standard scheduling problems, e.g., open shop, flow shop and job shop problems, which are commonly studied. All three problem classes address a problem of sequencing n jobs (tasks) through a set of m machines (resources) where every job has to be processed once on every machine and every such job operation requires a specified processing time. The problems differ in restrictions on the job routings.

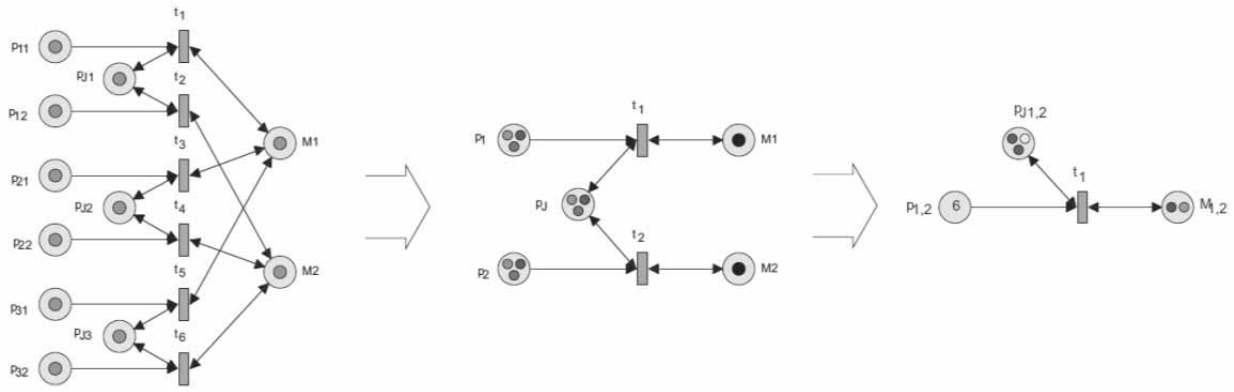

Figure 1: PN and CPN models of an open shop scheduling problem.

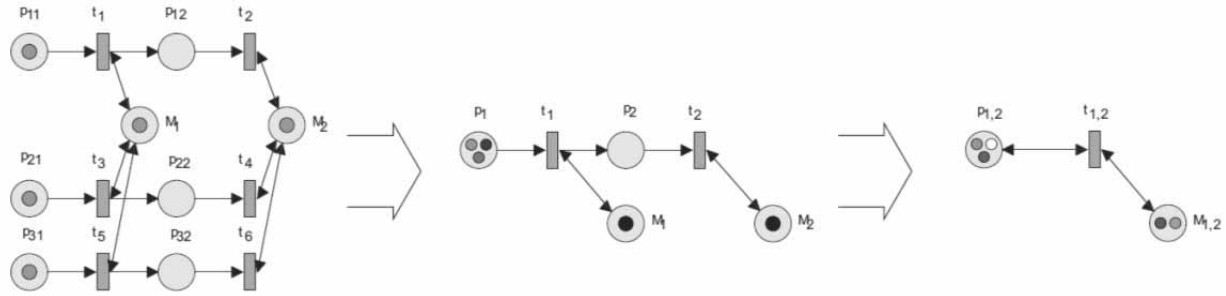

Figure 2: PN and CPN models of a flow shop scheduling problem.

Petri nets can be used to effectively model all three standard problem classes. In particular, Coloured Petri nets enable to develop compact models with a chosen level of abstraction, which are functionally equivalent to basic Place/Transition model. To illustrate this, Figure 1 shows a simple open-shop problem of three jobs and two machines.

The open shop problem structure states that the processing order of job operations is arbitrary, i.e., there are no restrictions with regard to the routing of each job through machine environment (Pinedo, 2008). Therefore the operations are represented as sequentially independent PN transitions that share a set of machines on one hand, and a set of jobs on the other. The shared resources are linked to transitions by self loops, which are for simplicity of drawing represented by double arrowed arcs.

Timed Petri nets are used where a token is declared unavailable for a specified time period after every transition firing. This way only a single operation of a job can be processed at a time and also only a single operation can be performed on a machine at a time. A single operation occurrence in a sequence is enforced by additional places which restrict the transition firing.

The model is shown in three levels of abstraction, with common Place/Transition Petri net (PN) structure used in the leftmost model and CPNs used in the other two models.

In the Place/Transition PN model the machines are denoted M_i and jobs by p_{J_i}. Start of every operation is modelled by transition firing. Since the transitions related to the same machine are in conflict, only one of the corresponding transitions can fire at a time. Conflict resolution strategy determines a schedule of machine assignments. Additional places p_{ij} correspond to j-th operation of job i and restrict the operations to be triggered only once in a sequence.

In the first CPN model (central model in Figure 1) all jobs are represented by the same set of places and transitions. The distinction among different jobs is achieved by introduction of token colours and transition occurrence colours. Every token colour corresponds to a job, whereas every occurrence colour corresponds to triggering of operation within the job. This way various operation durations can be represented by a single transition. In the next level of abstraction also the machines are folded into one place.

Consequently also the transitions representing operation triggering and places enforcing single operation triggering are folded together into only one place-transition pair. All the interrelations among different place tokens and transition firings are now encoded into complex colour based enablement conditions of different transition occurrence colours. E.g., for the model of Figure 1, marking of the final CPN is determined by 6 token colours in $p_{1,2}$, 3 token colours in $p_{J1,2}$ and 2 token colours in $M_{1,2}$, while t_1 has 6 occurrence colours. Every occurrence of t_1 defines a set of colour dependent weights of the incoming and outgoing arcs. The colour dependent weights related to a single arc can be represented by a vector, and the weights of this arc related to all transition occurrences form a matrix. In the CPN models the problem structure is therefore folded within arc weights that become matrices.

In general, an open shop problem with n jobs and m machines requires n token colours in job place, $n \cdot m$ token colours in operation place and m token colours in machine place. The number of transition occurrence colours equals the number of distinct operations, i.e. $n \cdot m$. The self loop arcs between job place and the transition are weighted by $n \times nm$ matrices, the arc between operation place and the transition by $nm \times nm$ matrix and self loop arcs between machine place and the transition are weighted by $m \times nm$ matrices.

Note that while the shown model is complete, it may be advantageous to add additional n colours to job place in order to represent the resulting schedule more easily. The final marking of the model in Figure 1 is obtained when all the operation tokens are consumed while all job and machine tokens are back in place and available. The schedule has to be reconstructed by observing unavailable tokens in operation and machine places. This is easier if tokens are deposited in operation place also during the last operation, although these are not employed to enable subsequent transition firing.

Similarly, any flow shop problem can be represented by a PN structure similar to Figure 2. Again the model is shown in three forms with increasing level of abstraction. Compared to open shop problems, the flow shop problem states that all jobs have the same routing, i.e. every job is processed on all machines in exactly the same machine order.

In Place/Transition PN model this is achieved by enforcing a directed token flow through a set of places representing operations of a single job. Places are linked in a chain with operation triggering transitions in between a pair of places. Transitions representing the same machine operation within all jobs are linked to a common place that represents a machine resource. No additional places are needed to ensure proper operation triggering as this is now achieved by sequential ordering of job operation places and transitions.

As before, the CPN model can be obtained by merging together the job operation chains, which are folded into a single sequence of places and transitions as shown in the center of Figure 2. The final CPN model is obtained by folding together the machine places, which consequently folds together also operation places and transitions. The required token flow is achieved by introduction of token colours, transition occurrence colours and arc weights in matrix form as described above. For a flow shop problem with n jobs and m machines, $n \cdot m$ token colours are required in job operation place and m token colours in machine place. The number of transition occurrence colours equals the number of distinct operations, i.e. $n \cdot m$. The arcs between job place and the transition are weighted by $nm \times nm$ matrices and arcs between machine place and the transition are weighted by $m \times nm$ matrices. The final marking of the model is obtained when all the job operation tokens are consumed while all machine tokens are back in place and available. Similarly as above, additional n colours can be added to job place to reconstruct the resulting schedule more easily.

In general, the model of Figure 2 represents a problem, where jobs are not processed in the same order on every machine. E.g., a job can pass another job while waiting for processing on a particular machine. Often the same job processing order is required on all machines, which forms a *permutation* flow shop problem. Within the optimization approach described in the next section, this can be easily achieved if the machine sequences are mutually dependent.

A job shop scheduling problem is modelled very much in the same way as the flow shop. The difference is only in machine assignments, which are not the same for every job. Instead, each jobs follows its own route through the set of machines, while it still visits every machine only once.

The different machine assignments reflect in diverse connections of operation transitions to machine places in P/T PN model. The final CPN model has the same graphical representation as in Figure 2; the machine assignments are hidden within arc weights in matrix form.

1.1 Simple Coloured Petri nets

Building on the two above examples a formal definition of Coloured Petri nets is given 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 et al. (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:

- $P = \{p_1, p_2, \dots, p_k\}$, $k > 0$ is a finite set of places. $T = \{t_1, t_2, \dots, t_l\}$, $l > 0$ 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. $\forall p \in P, Co(p) = \{a_{p,1}, a_{p,2}, \dots, a_{p,u_p}\} \subseteq Cl$ is the set of up possible colours of tokens in p, and $\forall t \in T, Co(t) = \{b_{t,1}, b_{t,2}, \dots, b_{t,v_t}\} \subseteq Cl$ 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 Coloured Petri net.

1.2 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). When using FD principle the transition firing has a duration. In contrast, 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, which has the same effect. 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.

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 operations are considered non-preemptive. HD principle is also used in the timed version of CPNs defined by Jensen (1997), where the unavailability of the tokens is 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 time attributes of the marking tokens, every coloured token is accompanied with a time stamp where 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.

As mentioned before, in the CPN models the problem structure is folded in arc weights represented by matrices. This is convenient for automatic generation of the models as matrices can be easily constructed algorithmically and the graphical representation of CPNs is rather simple compared to graphical layout of P/T Petri nets with high number of places, transitions and arcs. CPNs can be therefore effectively applied as a modelling framework in conjunction with automatic model generation and optimization based on production management data.

2 Neighbourhood Solution Generation Strategy

In our previous work (Löscher et al., 2007; Mušič et al., 2008) different ways of solution space exploration were studied. Extensive testing of the reachability tree search based approaches has been performed and also several experiments with local search techniques have been made.

2.1 Prescribed transition firing sequences

In Löscher et al. (2007) the approach is presented, which extends the Petri net representation by sequences and priorities. When using sequences, disjoint groups of transitions are selected and mapped to sequence vectors. 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 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 sequence-supervised 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.

To illustrate this, consider a simple job shop example of four jobs and four machines. Operation durations are shown in Table 1 and resource requirements in Table 2.

The problem is modelled by a CPN similar to Figure 2. The model can then be simulated by applying SPT (Shortest Processing Time first) rule (Haupt, 1989) as a default conflict resolution mechanism. The resulting sequence represents a possible schedule, shown in Figure 3.

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	80	28	7
o_4	2	79	87	29

Table 1: Operation durations 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

Table 2: Machine requirements for a simple job shop problem.

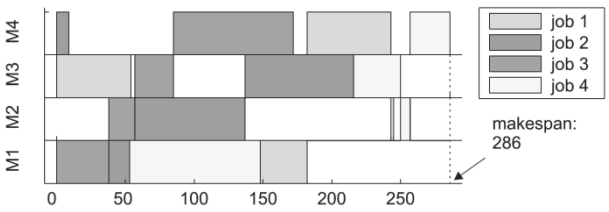

Figure 3: A possible solution of the given job-shop problem.

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 $M4$ modelled by four token colours in place M_{1-4}. Related sets of transition occurrence colours are:

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

where $t_{i,cj}$denotes the occurrence colour of t_{1-4}; $c_j$corresponds to job $J_j$and i corresponds to the operation number within the job.

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 $i, 1 \leq i \leq 4$:

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

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. The problem of this approach is that by perturbing sequence index vectors the resulting transition firing sequence may easily become infeasible, which results in a deadlock during simulation (a sequence imposed deadlock). Such an infeasible solution can be ignored and a new perturbation can be 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.

2.2 Generation of neighbourhood solutions

In the Operation Research (OR) literature several neighborhood generation operators have been proposed (Blazewicz et al., 1996; Jain et al., 2000).

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. 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:

$$CP = \{O_i : \rho_i = \rho_{i-1} + \tau_{i-1}, i = 2 \ldots n\} \quad (3)$$

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.

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}:

$$C_{max} = \sum_{O_i \epsilon CP} \tau_i \quad (4)$$

Figure 4 shows a redrawn gantt chart from Figure 3 with indication of the critical path (grey) and the sequence of critical operations. The shown critical path consists of 5 blocks.

Critical operations in Figure 4 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 et al., 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 et al., 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. If these transitions are linked to predefined firing sequences parameterized by index vectors, a move operator corresponds to a permutation of an index vector.

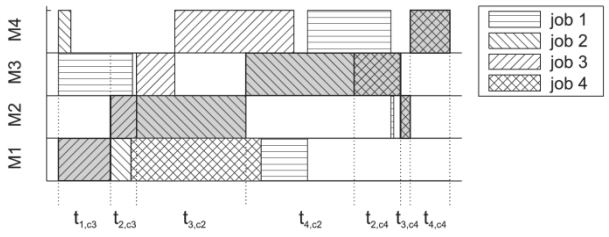

Figure 4: A critical path within a schedule and critical transitions.

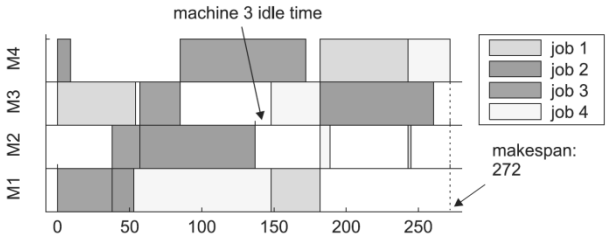

Figure 5: An optimized solution of the given job-shop problem.

For example, in the schedule shown in Figure 4 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 the third and the fourth element within V_3 index vector:

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

A new schedule obtained by simulation with modified V_3 results in a shorter makespan. Similar swap of the last two operations in the sequence S_{M2} leads to the optimal schedule (with the shortest makespan) shown in Figure 5.

When the move is limited to the 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.

The described approach is well suited to optimization of job shop problems. With minor modifications it can however also be used for optimization of other scheduling problem classes. The problem class specific constraints can be enforced as additional permutation constraints. E.g., permutation flow shop restrictions can be enforced by keeping all the sequence vectors in synchronization. In contrast, open shop problems have no restrictions on job routings. Fixing transition firing sequences for each individual machine therefore does not resolve all the conflicts in the model. The remaining conflicts can be solved on the fly during simulation in a random manner in order to cover as much as possible wide set of problem solutions.

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). In other words, the schedules generated in the proposed way belong to the class of semi-active schedules (Pinedo, 2008).

E.g., in the example of Figure 5 a small fraction of idle time is introduced before processing job 4 on machine 3. Job 2 has already been ready for processing on the same machine, which resulted in non-optimal schedule obtained by SPT rule (Figure 3). The chosen change in the firing sequence enforced the firing of $t_{2,c4}$ before $t_{4,c2}$ although $t_{4,c2}$ has been marking-enabled first. This way a schedule with shorter makespan was obtained.

3 Conclusion

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 some standard open shop, flow shop and job shop problems calculated by the proposed algorithm and some other standard algorithms has been performed. The proposed algorithm is able to improve the initial SPT solutions with a moderate effort, although it is not able to reach optimum for complex benchmarks.

Therefore a prototype implementation of tabu search algorithm has also been tested. 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.

Acknowledgments

The presented work has been partially performed within the 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

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

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

[3] Bowden, FDJ. A brief survey and synthesis of the roles of time in petri nets. *Mathematical & Computer Modelling*. 2000; 31: 55–68.

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

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

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

[7] Jensen K. *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use.* 2 edition. Berlin: Springer-Verlag; 1997. 236 p.

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

[9] Löscher T, Mušič G, Breitenecker F. (2007). Optimisation of scheduling problems based on timed petri nets. In *Proceedings EUROSIM 2007*; 2007; volume II. Ljubljana.

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

[11] Mušič, G. (2009). Petri net base scheduling approach combining dispatching rules and local search. In *21th European Modeling & Simulation Symposium*, volume 2. Puerto de La Cruz, Tenerife. P. 27–32

[12] Mušič G, Löscher T, Breitenecker F. Simulation based scheduling applying Petri nets with sequences and priorities. In *UKSIM 10th International Conference on Computer Modelling and Simulation*; 2008. Cambridge. P. 455-460

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

[14] Piera MA, Mušič G. Coloured Petri net scheduling models: Timed state space exploration shortages. *Math.Comput.Simul.*. 2001; 82: 428–441.

[15] Pinedo. ML. *Scheduling: Theory, Algorithms, and Systems.* 3rd edition. Springer Publishing Company, Incorporated; 2008.

[16] Tuncel G, Bayhan GM. Applications of Petri nets in production scheduling: a review. *International Journal of Advanced Manufacturing Technology*. 2007; 34: 762–773.

[17] van Laarhoven P, Aarts E, Lenstra J. (1992). Job shop scheduling by simulated annealing. *Operations Research*. 1992; 40: 113–125.