Preprints of the 21st IFAC World Congress (Virtual)
Berlin, Germany, July 12-17, 2020

A Monte-Carlo Tree Search based Tracking Control Approach for Timed Petri Nets

Raphael Fritz *, **Nico Krebs** *, **and Ping Zhang** *

* *Institute of Automatic Control, University of Kaiserslautern, Germany*
(e-mail: fritzr@eit.uni-kl.de; krebs@rhrk.uni-kl.de; pzhang@eit.uni-kl.de)

Abstract: In this paper, an approach for the tracking control problem for discrete event systems modeled by timed Petri nets (TPN) is proposed. The approach applies the Monte-Carlo Tree Search to the tracking control problem for TPN to find a firing sequence from an initial marking to the desired destination marking that minimizes the required duration. The proposed tracking control method randomly searches a small part of the reachability graph and incrementally constructs a search tree to find the optimal solution. This reduces the computational effort and allows the approach to solve the tracking control problem for larger systems. The approach has capabilities for deadlock avoidance and can be applied to a wide range of control problems like reachability analysis, fault-tolerant control and scheduling problems.

Keywords: Timed Petri Nets, Tracking Control, Scheduling, Manufacturing Systems, Discrete Event Systems, Monte-Carlo Tree Search, Reachability Analysis

1. INTRODUCTION

Discrete event systems (DES) are a suitable tool to model and control modern manufacturing systems and transportation systems (Cassandras and Lafortune (2008)). For larger systems with concurrent operations, DES described by Petri nets (PN) are often applied. The focus in this paper is on DES described by timed Petri nets (TPN) and the control problem of finding the time optimal firing sequence that reaches the desired state. The inclusion of temporal specification introduces additional complexity into the tracking control problem since not only the sequence of the transitions but also their firing times have to be decided. This problem is called tracking control problem and has applications like reachability analysis (Lefebvre (2018)) and scheduling (Lee and DiCesare (1994); Fritz et al. (2019)).

A method to determine optimal firing sequences for TPN based on integer linear programming (ILP) problem is presented in Mbaye et al. (2018). Lee and DiCesare (1994) apply the A* algorithm to the scheduling problem for TPN. This approach is further developed in Jeng and Chen (1998) with a heuristic function based on the state equation, in Zhang et al. (2005) with dynamic programming, in Pan et al. (2013) with a temporal analysis and Wang and Wang (2012) use a dynamic search window with best-first algorithm. Branch and Bound search is used by Jung et al. (2013) for scheduling in TPN. A Beam search based approach has been proposed by Cherif et al. (2019). Xing et al. (2012) apply a genetic algorithm and Lei et al. (2014) use a heuristic function based on firing count vectors to find an appropriate timed firing sequence. Lefebvre (2018) proposed an approach based on model predictive control.

A core part of the approach proposed in this paper for the tracking control of TPN is a Monte-Carlo method called Monte-Carlo Tree Search (MCTS). It combines tree search methods with random sampling of the search space (Browne et al. (2012)). In each iteration of the MCTS, a search tree is incrementally built, where the nodes are evaluated based on simulations. Google Deepmind's AlphaGo demonstrated that MCTS can solve problems with large search spaces (Silver et al. (2016)).

The first idea of applying MCTS to tracking control for untimed PN has been proposed in Fritz et al. (2019). In this paper, the approach is further developed to handle timed Petri nets and find an optimal or near-optimal timed firing sequence. It is shown how the MCTS is adapted to handle the timing components of TPN. The resulting MCTS based approach efficiently solves the tracking control problem for TPN while avoiding deadlocks. The main challenge is to find an appropriate cost function to evaluate a marking during the search. A new method to evaluate the cost for a marking is proposed. The method is based on the combination of the evaluation functions used in Fritz et al. (2019) which results in a more accurate cost estimation and a faster convergence to the destination marking.

2. PRELIMINARIES

2.1 Petri net

This section provides the necessary definitions based on David and Alla (1992) that will be used later. A Petri net can be represented by the four-tuple $PN = (P, T, N^+, N^-)$, with a set of m places $P = \{p_1, p_2, \ldots, p_m\}$ and a set of n transitions $T = \{t_1, t_2, \ldots, t_n\}$. The post-incidence matrix N^+ (pre-incidence matrix N^-) specifies the arcs and their weights from transitions to places (from places to transitions). $N = N^+ - N^-$ is the incidence matrix. $M \in \mathbb{N}_0^m$ is a marking and (PN, M_0) is a PN with initial marking M_0. Transition t_j is represented by the n-dimensional firing vector $q(k) = (q_1(k)\ q_2(k)\ \cdots\ q_n(k))^T$, whose j-th entry $q_j(k)$ is 1, while all other entries are 0. The new marking $M(k+1)$, resulting from firing $t \in T$ at time instant k is determined by the PN state equation $M(k+1) = M(k) + Nq(k)$. A transition $t \in T$ is enabled at marking $M(k)$, only if

$$N^- q(k) \leq M(k) \tag{1}$$

is fulfilled. The set of enabled transitions $\mathcal{T}_e$ contains all $t \in T$ that fulfill (1) for a specific $M(k)$. Assume that $\sigma = t_{q(0)} t_{q(1)} \cdots t_{q(k-1)}$ is a firing sequence of transitions with the length $|\sigma| = k$. The firing count vector q corresponding to the firing sequence σ is the sum of all firing vectors $q(i)$, $i = 0, 1, \ldots, k-1$, thus $\overline{q} = \sum_{i=0}^{k-1} q(i)$. Activating the firing sequence σ under the initial marking M_0 leads to the $M(k)$, described by $M(k) = M_0 + N\overline{q}$.

A timed Petri net is defined as $TPN = (PN, M_0, D)$ where D is a vector containing the firing durations d_i of the transitions $t_i \in T$. A transition t_i can fire at the earliest d_i time units (TU) after it is enabled (see (1)). A timed firing sequence is denoted as

$$\sigma_\tau = (t_{q(0)}, \tau_0)\ (t_{q(1)}, \tau_1)\ \dots\ (t_{q(k-1)}, \tau_{k-1}) \tag{2}$$

where $\tau_0, \dots, \tau_{k-1}$ denote the firing times of the transitions. The duration of the σ_τ is $d(\sigma_\tau) = \tau_{k-1}$ and the length is $|\sigma_\tau| = k$. Activating the timed firing sequence σ_τ under the initial marking M_0 leads to marking $M(k)$, i.e.

$$(\sigma_\tau, M_0) = M_0[(t_{q(0)}, \tau_0)\rangle M(1) \cdots M(k-1)[(t_{q(k-1)}, \tau_{k-1})\rangle M(k) \tag{3}$$

or in short $M_0[\sigma_\tau\rangle M_k$.

2.2 Monte-Carlo Tree Search

The Monte-Carlo Tree Search method combines tree search with random sampling of the search space. The method builds incrementally a tree and in each iteration the four basic steps of MCTS are executed (Browne et al. (2012)): 1) Starting from the root node, a leaf node is selected, 2) child nodes are added to the selected leaf node in the expansion, 3) simulations are carried out from the child nodes and the result of the simulation is evaluated, 4) the evaluation results from the simulations are backpropagated through the tree and the visited nodes are updated. The procedure is shown in Fig.1. These four steps are repeated until a stopping criterion is fulfilled.

2.3 Problem formulation

The problem discussed in this paper is the tracking of a destination marking M_{des} in a TPN by finding an appropriate timed firing sequence, i.e. $M_0[\sigma_\tau\rangle M_{des}$ with $\sigma_\tau = (t_{q(0)}, \tau_0)\ (t_{q(1)}, \tau_1)\ \dots\ (t_{q(k)}, \tau_k)$. The timed trajectory (σ_τ, M_0) should have minimal duration, i.e. no other timed firing sequence σ_τ' exists that reaches the destination marking M_{des} and has a duration $d(\sigma_\tau') < d(\sigma_\tau)$.

3. MONTE-CARLO TREE SEARCH FOR TIMED PN

This section demonstrates how the basic concept of the four MCTS steps can be applied to the tracking control problem for TPN. The tracking control approach of Fritz et al. (2019) can only handle untimed PN, therefore the different MCTS steps have to be adapted to fit the new control goal of finding the timed firing sequence σ_τ with the minimal duration.

The following notations are used. The root node of a tree is v_0, v_{parent} is a parent node with the set of child nodes $\mathcal{V}_{child}$. A single child node is denoted as v_{child}. A leaf node is node without any child nodes and is denoted as v_l. All the nodes are collected in the search tree denoted by $Tree$. Each node

$$v = \{M(v), q(v), \overline{q}(v), v_{parent}(v), \tau(v), cost(v)\} \tag{4}$$

in the tree contains the corresponding marking denoted by $M(v)$, the firing vector $q(v)$ corresponding to the transition that has fired from the previous node to the current node, the firing count vector $\overline{q}(v)$ containing the transitions fired from the root node to the current node and the parent node $v_{parent}(v)$ of node v. The accurate duration of the firing sequence from the root node to the current node is denoted by $\tau(v)$ which is also the firing time of the transition corresponding to $q(v)$. The cost $cost(v)$ denotes the duration estimation for the whole timed firing sequence, i.e. the sum of the accurate cost $\tau(v)$ and the estimated cost from $M(v)$ to the destination marking M_{des}.

Finding an appropriate method to calculate $cost(v)$ is the main challenge. $cost(v)$ is used in the selection process and guides the search. A good calculation method for $cost(v)$ leads to a fast convergence to the destination marking and a time optimal firing sequence. Since $cost(v)$ is an estimation of the total firing sequence duration from M_0 to M_{des}, the determination of $cost(v)$ is executed in multiple steps in the MCTS. This requires significant changes in the expansion step, the simulation step and the evaluation function to handle the timed tracking control problem compared to the untimed approach in Fritz et al. (2019).

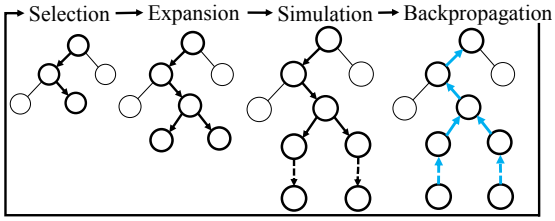

Fig. 1. Monte-Carlo Tree Search

3.1 Selection

In each iteration of the MCTS, the selection starts in the root node and selects the node used for the expansion step. From the root node v_0, the tree is traversed until a leaf node v_l is reached. The selection strategy used in this paper selects for each node, starting from the root node v_0, the child node v_{child} with the lowest cost $cost(v_{child})$ until a leaf node v_l is reached.

3.2 Expansion

During the expansion steps, the search tree is expanded by finding the direct successor nodes, i.e. child nodes, of the selected leaf node and adding them to the search tree. The child nodes of the selected leaf node v_l can be determined based on the enabled transitions. Different from the MCTS for untimed PN in Fritz et al. (2019), the parallel firing of multiple transitions should be considered, since the goal is to find the time optimal firing sequence. This requires the calculation of the earliest possible firing times of the transitions fired during the expansion.

Suppose that after extracting the tokens from a marking M required for firing a transition $t \in T$, another transition $t' \in T$ is still enabled, i.e. $N^- q_{t'} \leq M^-$ with the reduced predecessor marking $M^- = M - N^- q_t$ and q_t' as the firing vector corresponding to t'. Therefore, the firing of t' can already commence before t finishes firing, reducing the total time required for firing both transitions. To find the earliest possible firing time of t', we adopt the assumption from Lefebvre (2018) that the untimed firing sequence σ and the timed firing sequence σ_τ have the same order of transitions. Thus the firing time $\tau_{t'}$ of t' cannot be earlier than the firing time τ_t of t. To get the correct firing time of t', not only direct reduced predecessor marking M^- has to be considered, but all reduced predecessor markings for which t' is still enabled. Additionally the firing time has to fulfill $\tau_{t'} \geq \tau_t$. Otherwise, firing order in σ_τ would change.

The procedure of the expansion step is shown in Algorithm 1. In line 3, the set of enabled transitions $\mathcal{T}_r$ for the leaf node marking $M(v_l)$ is determined. The while loop (lines 4-18) is executed until all $t \in \mathcal{T}_r$ are fired and their correct firing time is determined. Lines 5-7 determines the information of the next predecessor node, by first determining the firing time τ^+, firing count vector $\overline{q}^-$ associated with the transitions fired from the current predecessor node to v_l and next predecessor node v^-. Based on this information, the reduced predecessor marking M^- and the enabled transitions $\mathcal{T}_e$ at M^- can be determined. For the remaining transitions in $\mathcal{T}_r$ that that are not enabled at M^-, a child node v_{child} is added to the tree (lines 8-12), where q_t is

Algorithm 1: Expansion of a leaf node v_l

1 **Input:** PN, v_l, $Tree$
2 **Init:** $v^- = v_l, \overline{q}^- = 0$
3 Determine enabled transition set $\mathscr{T}_r$ of $M(v_l)$ with (1)
4 **while** $\mathscr{T}_r \neq \emptyset$ **do**
5 $\quad \tau^+ = \tau(v^-), \overline{q}^- = \overline{q}^- + q(v^-), v^- = v_{parent}(v^-)$
6 $\quad M^- = \max\{M(v^-) - N^- q^-, 0\}$
7 $\quad$ Determine enabled transition set $\mathscr{T}_e$ of M^- with (1)
8 $\quad$ **for** $t \in \mathscr{T}_r \setminus \mathscr{T}_e$ **do**
9 $\quad\quad$ Add a child node v_{child} to v_l and $\mathscr{V}_{child}$
10 $\quad\quad q(v_{child}) = q_t, \overline{q}(v_{child}) = \overline{q}(v_l) + q_t,$
$\quad\quad\quad v_{parent}(v_{child}) = v_l, M(v_{child}) = M(v_l) + Nq_t$
11 $\quad\quad \tau(v_{child}) = \max\{\tau(v_l), \tau^+ + d_t\}$
12 $\quad\quad \mathscr{T}_r = \mathscr{T}_r \setminus t$
13 $\quad$ **for** $t \in \mathscr{T}_r$ **do**
14 $\quad\quad$ **if** $\tau(v^-) + d_t < \tau(v_l) \vee v^- = v_0$ **then**
15 $\quad\quad\quad$ Add a child node v_{child} to v_l and $\mathscr{V}_{child}$
16 $\quad\quad\quad q(v_{child}) = q_t, \overline{q}(v_{child}) = \overline{q}(v_l + q_t,$
$\quad\quad\quad\quad v_{parent}(v_{child}) = v_l, M(v_{child}) = M(v_l) + Nq_t$
17 $\quad\quad\quad \tau(v_{child}) = \max\{\tau(v_l),\ d_t\}$
18 $\quad\quad\quad \mathscr{T}_r = \mathscr{T}_r \setminus t$
19 **Output:** $\mathscr{V}_{child}$, $Tree$

the firing vector of transition t and the parent node is set to v_l. The corresponding firing time $\tau(v_{child})$ is calculated in line 11 and the transition is removed from from $\mathscr{T}_r$. For the remaining transitions in $\mathscr{T}_r$ (transitions that are still enabled at M^-), it is checked whether an earlier firing time would change the order of transitions in the timed firing sequence σ_τ or the root node is reached (line 14). If that is the case, the node corresponding to the transitions is added, the firing time is set to the firing time of $\tau(v_l)$ or d_t in case the root node is reached and t is removed from $\mathscr{T}_r$. This is repeated until all transitions fire, either because they are not enabled in the predecessor state, the firing order of transitions in σ_τ or the root node is reached.

3.3 Simulation

The neighborhood of the child nodes $v_{child} \in \mathscr{V}_{child}$ are examined by randomly firing transitions. This results in a random exploration of part of the reachability graph. For each $v_{child} \in \mathscr{V}_{child}$, a number of simulations denoted by *sim* are carried out. The simulation have a predetermined depth $depth$, denoting the number of transitions that should fire in sequence. In Algorithm 2, the execution of a single simulation for a v_{child} is shown.

The for-loop (line 4-22) executes the simulation until $depth$ transitions have fired. At first, the set of enabled transitions $\mathscr{T}_e$ for the $M(v_{sim,i-1})$ are determined. If $\mathscr{T}_e$ is empty, it is checked whether $M(v_{sim,i-1})$ is the destination marking M_{des} or a deadlock and the corresponding variable is set to one (lines 6-9). If $\mathscr{T}_e$ is not empty, a transition is randomly selected from $\mathscr{T}_e$ (q_t is the firing vector of t) and a corresponding simulation node is generated (lines 10-12). Lines 13-21 contain the determination of the earliest firing time of the fired transition. The approach is similar to Algorithm 1. The result is the final simulation node v_{sim}, the variables $converge_{sim}$ and $deadlock$.

3.4 Evaluation

An important part of the MCTS is the selection of the evaluation function. The goal of the tracking control approach in this paper is to find the timed firing sequence with the minimum duration that forces the TPN from the initial marking to the destination marking. In Fritz et al. (2019), two different

Algorithm 2: Simulation

1 **Input:** PN, v_{child}, $Tree$, $depth$
2 **Init:** $v^- = v_{child}, \overline{q}^- = 0, v_{sim,0} = v_{child},$
3 $\quad converge_{sim} = 0, deadlock = 0$
4 **for** $i = 1$ *to* $depth$ **do**
5 $\quad$ Determine enabled transition set $\mathscr{T}_e$ of marking
$\quad\quad M(v_{sim,i-1})$ with (1)
6 $\quad$ **if** $\mathscr{T}_e = \emptyset$ **then**
7 $\quad\quad$ **if** $M(v_{sim,i-1}) = M_{des}$ **then** $converge_{sim} = 1$
8 $\quad\quad$ **else** $deadlock = 1$
9 $\quad\quad$ **break**
10 $\quad$ Randomly select a $t \in \mathscr{T}_e$
11 $\quad$ Add simulation node $v_{sim,i}$
12 $\quad q(v_{sim,i}) = q_t, \overline{q}(v_{sim,i}) = \overline{q}(v_{sim,i-1}) + q_t,$
$\quad\quad v_{parent}(v_{sim,i}) = v_{sim,i-1}, M(v_{sim,i}) = M(v_{sim,i-1}) + Nq_t$
13 $\quad$ **while** *True* **do**
14 $\quad\quad \tau^+ = \tau(v^-), \overline{q}^- = \overline{q}^- + q(v^-), v^- = v_{parent}(v^-)$
15 $\quad\quad M^- = \max\{M(v^-) - N^- \overline{q}^-, 0\}$
16 $\quad\quad$ **if** $N^- q_t > M^-$ **then**
17 $\quad\quad\quad \tau(v_{sim,i}) = \max\{\tau(v_{sim,i-1}), \tau^+ + d_t\}$
18 $\quad\quad\quad$ **break**
19 $\quad\quad$ **else if** $\tau(v^-) + d_t < \tau(v_{sim,i-1}) \vee v^- = v_0$ **then**
20 $\quad\quad\quad \tau(v_{sim,i}) = \max\{\tau(v_{sim,i-1}),\ d_t\}$
21 $\quad\quad\quad$ **break**
22 $\quad v^- = v_{sim,i}$
23 $v_{sim} = v_{sim,depth}$
24 **Output:** v_{sim}, $converge_{sim}$, $deadlock$

evaluation functions are proposed for the tracking control of untimed PN. These approaches are adapted to handle TPN. Furthermore a combination of the two approaches is proposed to exploit their advantages and reduce the disadvantages. The idea for both approaches is to estimate the weighted number of required transitions from $M(v_{sim})$ to M_{des} with the weights as the firing durations d_i in vector D.

ILP-based approach: The first approach is based on an integer linear programming (ILP) problem to determine solution candidates of the firing count vector. The firing count vector is weighted by the firing durations resulting in the following ILP:

$$\begin{aligned} \tau(M, M_{des}) = \min_{\overline{q}(M,M_{des})} \quad & D\overline{q}(M, M_{des}) \\ \text{subject to} \quad & N\overline{q}(M, M_{des}) = M_{des} - M \\ & \overline{q}_j(M, M_{des}) \in \mathbb{N}_0,\ j = 1, \ldots, n \end{aligned} \tag{5}$$

We denote with $\overline{q}^*(M, M_{des})$ the optimal firing count vector found by (5). $\tau(M, M_{des})$ is the estimation of the duration of the firing sequence from M to M_{des}. It is important to note that $\tau(M, M_{des})$ is an upper bound of the duration of the firing sequence represented by $\overline{q}^*(M, M_{des})$, since $D\overline{q}(M, M_{des})$ in (5) is the sum of the firing durations which does not consider the parallel firing of transitions.

Difference-based approach: The idea is to executed (5) once at the beginning of the MCTS to get $\overline{q}^*(M_0, M_{des})$. Afterwards the evaluation of a marking M is done based on the weighted absolute difference between firing count vector $\overline{q}(M_0, M)$ of the firing sequence connecting M_0 to M and $\overline{q}^*(M_0, M_{des})$, i.e.

$$\tau(M, M_{des}) = D|\overline{q}^*(M_0, M_{des}) - \overline{q}(M_0, M)| \tag{6}$$

The main advantage of (6) is the lower computational effort required for the evaluation. The ILP-based evaluation with (5) give a more accurate estimation of the still required duration, since it updates the duration based on the current marking M. Due to the execution of (5) for each simulation, the computational

burden is much higher. Since the evaluation is required for every simulation, a faster evaluation method is preferred.

Combination approach: In the following, an approach to combine those two evaluation methods is proposed. It can be easily determined if the firing count vector of a firing sequence matches the firing count vector $\overline{q}^*(M_0,M_{des})$ calculated for evaluation (6). If an element of $\overline{q}^*(M_0,M_{des})-\overline{q}(M_0,M)$ is negative, the firing count vector $\overline{q}(M_0,M)$ represents a different firing sequence than $\overline{q}^*(M_0,M_{des})$. The basic idea is to execute the ILP-based evaluation (5) in these cases, resulting in a more accurate estimation of the duration $\tau(M,M_{des})$. By adding the firing count vector found by the ILP-based evaluation (i.e. $\overline{q}^*(M,M_{des})$) to the firing count vector from M_0 to M (i.e. $\overline{q}(M_0,M)$), a new firing count vector $\overline{q}$ from M_0 to M_{des} that goes through M can be generated. This vector is then added to the set of possible firing count vectors Q_{des}. For the evaluation of the next marking, evaluation (6) is executed for every $\overline{q}\in Q_{des}$ and the shortest duration is used for the estimation.

Deadlock avoidance: To incorporate deadlock avoidance capabilities into the proposed tracking control method, the penalty approach from Fritz et al. (2019) is used. If $\mathscr{T}_e$ is empty during the execution of Algorithm 2, the current marking is a deadlock. The evaluation is modified into $\tau(M,M_{des})=\rho_{dl}$ with ρ_{dl} as the penalty cost of a deadlock. Since ρ_{dl} influences the evaluation of v_{child} and through the backpropagation all nodes upstream, ρ_{dl} should be selected to balance deadlock avoidance and exploration.

Evaluation of a simulation: The evaluation procedure for a single simulation combining the new evaluation method with the deadlock penalty is shown in Algorithm 3. At the beginning, it is checked whether the simulation converges to the destination marking or a deadlock has occurred (lines 2-3). In case of convergence, the remaining duration is set to zero. For deadlocks, the estimated duration is set to the penalty cost ρ_{dl}. If neither a deadlock occurs nor the destination marking is reached, the simulation marking $M_{sim}=M(v_{sim})$ is evaluated based on the approach combining (5) and (6) (lines 5-16). In lines 5-10, it is checked whether $\overline{q}(v_{sim})$ matches a firing count vector in Q_{des}. If this is the case, the smallest duration estimation is determined by comparing $\overline{q}(v_{sim})$ with every firing count vector in Q_{des}. If $\overline{q}(v_{sim})$ does not match any firing count vector in Q_{des}, the estimation is done with ILP-based evaluation (5) (lines 11-16). If no solution is found, penalty cost are assigned. In case a solution is found, the resulting firing count vector is combined with the firing count vector from M_0 to M_{sim} and a new firing count vector $\overline{q}$ is added to Q_{des}. The estimation of the duration of the complete firing sequence $cost(v_{sim})$ is done by adding the accurate duration $\tau(v_{sim})$ of already fired transitions to the duration estimation $\tau(M_{sim},M_{des})$ of the remaining firing sequence (line 17).

Algorithm 3: Evaluation

```
1  Input: PN, D, v_sim, Q_des, converge_sim, deadlock
2  if converge_sim = 1 then τ(M_sim, M_des) = 0
3  else if deadlock = 1 then τ(M_sim, M_des) = ρ_dl
4  else
5      τ(M_sim, M_des) = ∞
6      for j = 1 to |Q_des| do
7          if Q_des(j) − q̄(v_sim) > 0 then
8              τ = D(Q_des(j) − q̄(v_sim))
9              if τ < τ(M_sim, M_des) then
10                 τ(M_sim, M_des) = τ
11     if τ(M_sim, M_des) = ∞ then
12         Calc. τ(M_sim, M_des) and q̄*(M_sim, M_des) with (5)
13         if no solution for (5) then τ(M_sim, M_des) = ρ_dl
14         else
15             q̄ = q̄*(M_sim, M_des) + q̄(v_sim)
16             Q_des = Q_des ∪ q̄
17 cost(v_sim) = τ(v_sim) + τ(M_sim, M_des)
18 Output: cost(v_sim)
```

3.5 Backpropagation

The final step of a MCTS iteration is backpropagating the information gained by the simulation upstream. At first, the cost of the child nodes $v_{child}\in\mathscr{V}_{child}$ are updated. Two different approaches for the calculation of the cost of a child node v_{child} are considered in this paper.

(1) Mean cost of all simulations, i.e.

$$cost(v_{child})=\frac{\sum_{i=1}^{sim}cost_i(v_{sim})}{sim} \tag{7}$$

(2) Minimal cost of all simulations, i.e.

$$cost(v_{child})=\min_{i=1,\dots,sim}(cost_i(v_{sim})) \tag{8}$$

$cost_i(v_{sim})$ denotes the cost of the i-th simulation calculated by Algorithm 3.

Finally, starting from the leaf node, the new cost is backpropagated through the tree until the root node is reached. Similar to the child node cost, two approaches are considered here.

(1) Cost of parent node as the mean cost of child nodes

$$cost(v_{parent})=\frac{\sum_{v_{child}\in\mathscr{V}_{child}}cost(v_{child})}{|\mathscr{V}_{child}|} \tag{9}$$

(2) Cost of parent node as the minimal cost of child nodes

$$cost(v_{parent})=\min_{v_{child}\in\mathscr{V}_{child}}cost(v_{child}) \tag{10}$$

The backpropagation traverses the tree from leaf node v_l to v_0 and updates $cost(v)$ of each visited node with either (9) or (10).

4. TIMED TRACKING CONTROL WITH MCTS

The MCTS steps developed in Section 3 can be combined to get the tracking control algorithm as shown in Algorithm 4. The stopping criteria is that the algorithm converges to the destination marking M_{des} or that a pre-specified time limit Δ_{max} is exceeded.

At the beginning *converge* is set to 0, the root node of the tree is added and Q_{des} is initialized with $\overline{q}^*(M_0,M_{des})$. In each MCTS iteration, the MCTS steps are executed. After the execution of the expansion step, it is checked if a marking in the set of child nodes is the destination marking (line 7). If this condition is fulfilled, the MCTS is stopped and the timed firing sequence σ_τ that forces the initial marking M_0 to the destination marking M_{des} is determined in line 18. By traversing the search tree from node v_{des} to the root node v_0, the transitions and corresponding firing times can be taken from the visited nodes. If the MCTS algorithm did not converge in the current iteration, the simulation and backpropagation steps are executed (lines 10-16). In case that the pre-specified time limit Δ_{max} is reached before the algorithm converges, the search is terminated and the tracking is not successful.

The computational effort of Algorithm 4 scales linearly with the simulation parameters *sim* and *depth*. These parameters also influence the convergence rate of the algorithm. If a finite time limit Δ_{max} is pre-specified, Algorithm 4 may stop before it converges to the destination marking M_{des}, due to the random

Algorithm 4: Monte-Carlo Tree Search

Input: PN, M_0, M_{des}, $depth$, sim, Δ_{max}
Init: create root node v_0 with marking M_0, $converge = 0$
Calculate $\overline{q}^*(M_0, M_{des})$ with (5), $Q_{des} = \overline{q}^*(M_0, M_{des})$
while $(converge = 0) \wedge (\Delta < \Delta_{max})$ **do**
  Execute **Selection** starting from v_0 resulting in leaf node v_l (Section 3.1)
  Execute **Expansion** with Algorithm 1
  if $M_{des} \in \{M(v_{child}) | v_{child} \in \mathcal{V}_{child}\}$ **then**
    $converge = 1$, $v_{des} = \{v \in \mathcal{V}_{child} | M(v_{child}) = M_{des})\}$
  else
    for $v_{child} \in \mathcal{V}_{child}$ **do**
      for $i = 1$ *to* sim **do**
        Execute **Simulation** with Algorithm 2
        Execute **Evaluation** with Algorithm 3
      Calculate $cost(v_{child})$ with (7) or (8)
      Delete simulation nodes from $Tree$
    Execute **Backpropagation** starting from v_l (Section 3.5).
if $converge = 1$ **then**
  Determine timed firing sequence σ_τ from v_0 to v_{des}
  Output: σ_τ
else Output: Tracking not successful

exploration of the reachability graph in the simulation step. Algorithm 4 cannot guarantee to find the timed firing sequence with the minimal duration, since the random exploration of the reachability graph may lead to suboptimal selection of the transitions. The globally optimal solution can often not be found for larger systems in finite time, therefore a fast convergence to a near optimal solution as provided by the proposed method is often preferred.

5. EXAMPLE

In order to illustrate the timed tracking control approach based on MCTS, a manufacturing system with two types of products is considered. The corresponding PN model is shown in Fig. 2. The system is a slightly modified model of the system used in Lefebvre (2018) (places $p20$ and $p21$ and transitions $t15$ and $t16$ are added to the model to separate input and output storage). The firing duration vector for the transitions is $D = [1\ 1\ 2\ 1\ 2\ 1\ 1\ 1\ 3\ 3\ 3\ 3\ 3\ 3\ 1\ 1]^T$.

The goal is to find a time optimal firing sequence that allows the production of the two product types. The production of the products is finished when the tokens from the initial places $p1$ and $p8$ are transported to the destination places $p20$ and $p21$. The proposed timed tracking control algorithm is tested with the initial marking as shown in Fig. 2 and different destination marking M_{des}. The destination markings M_{des} in Tables 1-4 indicate how many tokens from $p1$ and $p8$ are transported to $p20$ and $p21$. For example $[1\ 1]$ indicates that one token is transported from $p1$ to $p20$ and one from $p8$ to $p21$.

The MCTS based timed tracking control approach is tested by selecting different parameters. The penalty cost is set to $\rho_{dl} = D\overline{q}^*(M_0, M_{des})$ (firing vector $\overline{q}^*(M_0, M_{des})$ is calculated with (5)). The algorithm is implemented on a PC with 3.4 GHz CPU and 8 GB RAM. The test results are compared based on four performance indices: 1) The Success rate in in % that shows the number of times the algorithm converges to M_{des}, 2) the mean computation time to solve the tracking control problem, 3) the number of times the algorithm has found the time optimal firing sequence in %. Since the globally optimal solution can often not be found in finite time, the best result found with all

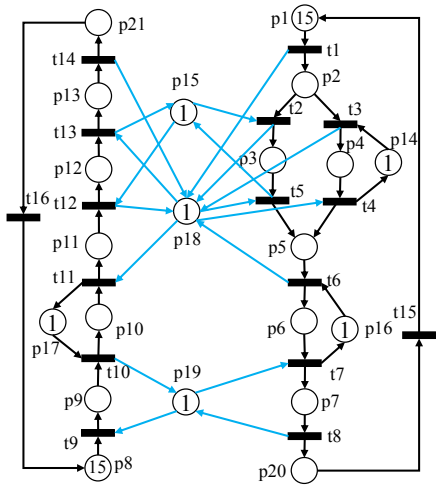

Fig. 2. Petri net model of a manufacturing system

Table 1. Comparison of backpropagation strategies

M_{des}	Success in %/ Mean time in (s)		Opt. sequence found in %/ Mean error in TU	
	Mean (7) & Mean (9)	Min (8) & Min (10)	Mean (7) & Mean (9)	Min (8) & Min (10)
[1 1]	100/0.9	100/0.13	83/0.17	48/0.7
[2 2]	0/x	100/5.52	0/∞	66/0.56

Table 2. Comparison of evaluation methods

M_{des}	Success in %/ Mean time in (s)			Opt. sequence found in %/ Mean error in TU		
	ILP (5)	Diff. (6)	Algo. 3	ILP (5)	Diff. (6)	Algo. 3
[1 1]	100 /6.41	100/0.04	100/0.1	70/0.3	40/0.6	48/0.7
[2 2]	100/19.1	100/0.04	100/5.5	29/0.7	0/3.5	22/0.5
[3 3]	85/43.7	100/0.09	100/6.2	6/0.5	0/6.2	2/0.6

algorithms is regarded as the time optimal firing sequence. And 4) the mean error from the duration of the time optimal firing sequence in time units. Each test is repeated 100 times and the mean computation times and mean error of the durations only include the successful runs. The time limit Δ_{max} is set to 120s.

5.1 Selection of optimal parameters for the MCTS

At first, we compare the influence of the different backpropagation strategies proposed in Section 3.5, i.e. selection of child cost and parent cost calculation method. For both, either the mean cost ((7) and (9)) or the minimal cost ((8) and (10)) are used for the calculation. Due to space limitation only the results of two combinations are shown in Table 1. The MCTS parameter are $depth = 1$, $sim = 4$. Using the minimal child cost and the minimal parent cost leads to a faster convergence to M_{des}. The main reason is that the mean values lead to a lower gradient in the cost function and thus a slower convergence.

Next, the influence of the three different evaluation methods presented in Section 3.4 is compared. It can be seen in Table 2 that the evaluation based on (6) results in the fastest convergence, but also introduces the highest mean error. The results from applying the ILP (5) and the proposed evaluation method in Algorithm 3 results in comparable durations for the firing sequence, but the mean computation time using Algorithm 3 is much shorter. The proposed algorithm combines the low computational effort of (6) with the accuracy of (5).

Based on the results in Tables 1 and 2, the best parameter combination for the MCTS is evaluation of the simulation results with Algorithm 3, child cost and parent cost estimation as the minimal cost, i.e. (8) and (10).

Table 3. Success rate/ solution time of approaches

M_{des}	$\lvert\sigma_\tau\rvert$	Success in %/ Mean time in (s)					
		MCTS $depth$=2 sim=4	MCTS $depth$=7 sim=14	DFS	BFS	MPC H=2 H_τ=5	MPC H=4 H_τ=10
[1 1]	12	100/3.1	100/2.1	100/47.2	100/0.3	100/0.6	100/8.7
[2 2]	24	100/3.6	100/0.9	0/x	100/1	100/1.6	100/34.7
[5 5]	60	100/1.5	100/1.9	0/x	100/6.2	100/6.2	0/x
[10 10]	120	100/3.4	100/4.2	0/x	100/25.8	100/20.4	0/x
[15 15]	180	100/4.8	100/7	0/x	100/62.4	100/46.2	0/x

Table 4. Performance comparison of approaches

M_{des}	Opt. sequence found in %/ Mean error in TU					
	MCTS $depth$=2 sim=4	MCTS $depth$=7 sim=14	DFS	BFS	MPC H=2 H_τ=5	MPC H=4 H_τ=10
[1 1]	18/5	28/2.8	0/7	24/0.8	21/4.4	15/4.1
[2 2]	2/1.5	0/1.2	0/∞	0/2.4	0/5.7	0/6.9
[5 5]	3/3.4	10/2.6	0/∞	0/3.4	0/8.8	0/∞
[10 10]	2/4.5	1/4	0/∞	0/4.7	0/14.1	0/∞
[15 15]	2/5	2/3.8	0/∞	0/5.7	0/22	0/∞

5.2 Comparison with other tracking control methods

To evaluate the performance of the tracking control approach based on MCTS for TPN, the proposed approach is compared with three other approaches, Depth-first search (DFS) (Cormen et al. (2001)), Best-first search (BFS) (Wang and Wang (2012)) and Model predictive control (MPC) (Lefebvre (2018)).

The three approaches are compared to the proposed MCTS with two different search parameter cases. In the first case $depth = 2$ & $sim = 4$ and in the second case $depth = 7$ & $sim = 14$. Table 3 sums up the convergence rate, mean computation times and also shows the firing sequence lengths $|\sigma_\tau|$ of the optimal sequence and Table 4 shows the optimality indices. It can be seen in Table 3 that the MCTS approach, independent of the selected parameters, converges for every destination marking. The same is true for the BFS and the MPC with lower search depth. The DFS only finds a solution for the smallest marking, while the MPC with bigger search depth struggles to find solutions for the larger problems. BFS and MPC ($H = 2, H_\tau = 5$) have comparable mean computation times. The MCTS exhibits the best results with the lowest computation times.

The results for the optimality indices in Table 4 show that the DFS has a high mean deviation from the optimal solution. Comparing the BFS and MPC, it can be seen that the results are comparable for smaller problems, but the BFS has a lower error for larger problems. The proposed MCTS approach has the lowest mean error from the time optimal solution. It can also be seen that with the increase in simulation numbers and simulation depth, the mean error decreases. In general, all other approaches find the optimal solution only for small sizes of problems, while for larger problems only the MCTS finds the time optimal firing sequence. In summary, the proposed tracking control approach based on MCTS can reduce the computation time significantly and also finds solutions of better quality in comparison to other existing tracking control approaches for TPN.

6. CONCLUSION

An efficient approach to solve the tracking control problem for timed Petri nets based on Monte-Carlo Tree Search has been proposed. The adaption of the basic MCTS steps to the timed tracking control problem has been presented. Due to the incrementally constructed search tree and limited search in part of the reachability graph, the proposed approach can significantly reduce the computational burden while still providing optimal or near optimal solutions. Compared with the existing approaches, the proposed approach has shown advantages in computation time and optimality of solutions.

In future research, it will be investigated whether the parameters $depth$ and sim of the MCTS can be selected more systematically. An important question is also to ensure the optimality of the solution, which cannot be guaranteed currently.

REFERENCES

Browne, C., Powley, E., Whitehouse, D., Lucas, S., Cowling, P.I., Tavener, S., Perez, D., Samothrakis, S., and Colton, S. (2012). A survey of Monte Carlo tree search methods. *Trans. on Computational Intelligence and AI in Games*, 4(1), 1–43.

Cassandras, C.G. and Lafortune, S. (2008). *Introduction to Discrete Event Systems*. Springer.

Cherif, G., Leclercq, E., and Lefebvre, D. (2019). Generation filtered beam search algorithm for the scheduling of hybrid FMS using T-TPN. In *Proc. of the 18th European Control Conference*, 3225–3230.

Cormen, T.H., Leiserson, C.E., Rivest, R.L., and Stein, C. (2001). *Introduction to Algorithms*. The MIT Press.

David, R. and Alla, H. (1992). *Petri nets and Grafcet: tools for modelling discrete event systems*. Prentice-Hall, Inc.

Fritz, R., Napitupulu, J., and Zhang, P. (2019). Tracking control for Petri nets based on Monte-Carlo tree search. In *Proc. of the 18th European Control Conference*, 4180–4185.

Jeng, M.D. and Chen, S.C. (1998). A heuristic search approach using approximate solutions to Petri net state equations for scheduling flexible manufacturing systems. *International Journal of Flexible Manufacturing Systems*, 10(2), 139–162.

Jung, C., Kim, H.J., and Lee, T.E. (2013). A branch and bound algorithm for cyclic scheduling of timed Petri nets. *IEEE Trans. Autom. Sci. Eng.*, 12(1), 309–323.

Lee, D.Y. and DiCesare, F. (1994). Scheduling flexible manufacturing systems using Petri nets and heuristic search. *IEEE Trans. Robot. Autom.*, 10(2), 123–132.

Lefebvre, D. (2018). Near-optimal scheduling for Petri net models with forbidden markings. *IEEE Transactions on Automatic Control*, 63(8), 2550–2557.

Lei, H., Xing, K., Han, L., Xiong, F., and Ge, Z. (2014). Deadlock-free scheduling for flexible manufacturing systems using Petri nets and heuristic search. *Computers & Industrial Engineering*, 72, 297–305.

Mbaye, A., Lefebvre, D., and Basile, F. (2018). Control design for timed Petri nets based on lmis and structure expansion. In *Proc. of the 23rd International Conference on Emerging Technologies and Factory Automation*, 426–432.

Pan, L., Ding, Z.J., and Zhou, M.C. (2013). A configurable state class method for temporal analysis of time Petri nets. *IEEE Trans. Syst., Man, Cybern. Syst*, 44(4), 482–493.

Silver, D., Huang, A., Maddison, C., et al. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587), 484–489.

Wang, Q. and Wang, Z. (2012). Hybrid heuristic search based on Petri net for FMS scheduling. *Energy Procedia*, 17, 506–512.

Xing, K., Han, L., Zhou, M., and Wang, F. (2012). Deadlock-free genetic scheduling algorithm for automated manufacturing systems based on deadlock control policy. *IEEE Trans. Syst. Man, Cybern. B, Cybern.*, 42(3), 603–615.

Zhang, W., Freiheit, T., and Yang, H. (2005). Dynamic scheduling in flexible assembly system based on timed Petri nets model. *Robotics and Computer-Integrated Manufacturing*, 21(6), 550–558.