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 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. $M \in \mathbb{N}_0^n$ 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

$$N^+ q(k) \preceq M(k)$$

is fulfilled. The set of enabled transitions $\mathcal{E}_t$ 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 $\sigma$ is the sum of all firing vectors $q(i), \ i = 0, 1, \ldots, k-1$, thus $q = \sum_{i=0}^{k-1} q(i)$. Activating the firing sequence $\sigma$ under the initial marking $M_0$ leads to the $M(k)$, described by $M(k) = M_0 + Nq$. 

Copyright lies with the authors 2125
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

$$\sigma_t = (t_{q(0)}, \tau_0) \cdot (t_{q(1)}, \tau_1) \ldots (t_{q(k-1)}, \tau_{k-1}) \quad (2)$$

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

$$(\sigma_t, M_0) = M_0 [(t_{q(0)}, \tau_0)] M(1) \ldots M(k-1) [(t_{q(k-1)}, \tau_{k-1})] M(k) \quad (3)$$

or in short $M_0 [\sigma_t] M_k$.

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 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 $\sigma_t$ 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

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

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 $Q(v)$ 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).

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. $N^- q_t \leq M^- q_t$ 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 $\sigma$ and the timed firing sequence $\sigma_t$ have the same order of transitions. Thus the firing time $\tau_v$ of $t'$ cannot be earlier than the firing time $\tau_t$ of $t$. To get the correct firing time of $t'$, not only direct reduced precursor marking $M^-$ has to be considered, but all reduced predecessor markings for which $t'$ is still enabled. Additionally the firing time has to fulfill $\tau_v \geq \tau_t$. Otherwise, firing order in $\sigma_t$ would change.

The procedure of the expansion step is shown in Algorithm 1. In line 3, the set of enabled transitions $\mathcal{F}_t$ for the leaf node marking $M(v_l)$ is determined. The while loop (lines 4-18) is executed until all $t \in \mathcal{F}_t$ 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 $\tau_t$, firing count vector $\vec{q}$ 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 $\mathcal{F}_t$ at $M^-$ can be determined. For the remaining transitions in $\mathcal{F}_t$ 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:** \( v_r = v_l, q_r = 0 \)
3. Determine enabled transition set \( \mathcal{F}_r \) of \( M(v_l) \) with (1)
4. While \( \mathcal{F}_r \neq \emptyset \) do
   5. \( \tau^+ = \tau(v_r), \overline{q}^r = \overline{q} - q(v_r), v_r = v_{\text{parent}}(v_r) \)
   6. \( M^r = \max\{M(v_r) - Nq_r, 0\} \)
   7. Determine enabled transition set \( \mathcal{F}_r \) of \( M^r \) with (1)
5. For \( t \in \mathcal{F}_r \setminus \mathcal{T}_l \) do
6. Add a child node \( v_{\text{child}} \) to \( v_l \) and \( \mathcal{Y}_{\text{child}} \)
7. \( q(v_{\text{child}}) = q_t, \overline{q}(v_{\text{child}}) = \overline{q}(v_r) + q_t \)
8. \( v_{\text{parent}}(v_{\text{child}}) = v_l, M(v_{\text{child}}) = M(v_l) + Nq_t \)
9. \( \tau(v_{\text{child}}) = \max\{\tau(v_l), \tau^+ + d_t\} \)
10. \( \mathcal{F}_r = \mathcal{F}_r \setminus t \)
11. End for \( t \in \mathcal{F}_r \)
12. End while \( \mathcal{F}_r \neq \emptyset \)
13. Output: \( \mathcal{Y}_{\text{child}}, Tree \)

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

3.3 Simulation

The neighborhood of the child nodes \( v_{\text{child}} \in \mathcal{Y}_{\text{child}} \) are examined by randomly firing transitions. This results in a random exploration of part of the reachability graph. For each \( v_{\text{child}} \in \mathcal{Y}_{\text{child}} \) a number of simulations denoted by \( sim \) are carried out. The simulation has 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 \( v_{\text{child}} \) is shown.

The for-loop (line 4-22) executes the simulation until \( depth \) transitions have fired. At first, the set of enabled transitions \( \mathcal{F}_r \) for the \( M(v_{\text{sim},i-1}) \) are determined. If \( \mathcal{F}_r \) is empty, it is checked whether \( M(v_{\text{sim},i-1}) \) is the destination marking \( M_{des} \) or a deadlock and the corresponding variable is set to one (lines 6-9). If \( \mathcal{F}_r \) is not empty, a transition is randomly selected from \( \mathcal{F}_r \) (\( q_t \) 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 \( v_{\text{sim}} \), the variables \( converge_{sim} \) and \( \text{deadlock} \).

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 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 \( M(v_{\text{sim}}) \) to \( M_{des} \) with the weights as the firing durations \( d_t \) 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:

\[
\tau(M_{M_{des}}) = \min_{\tau(M_{M_{sim}})} D \overline{q}(M_{M_{des}})
\]

subject to

\[
N \overline{q}(M_{M_{des}}) = M_{M_{des}} - M
\]

\[
\overline{q}(M_{M_{des}}) \in \mathbb{N}_0, j = 1, \ldots, n
\]

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 \( M \), i.e.

\[
\tau(M_{M_{des}}) = D \overline{q}(M_0, M_{des}) - \overline{q}(M_0, M)
\]

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 \( \overline{\tau}(M_0, M_{des}) \) calculated for evaluation (6). If an element of \( \overline{\tau}(M_0, M_{des}) \) is negative, the firing count vector \( \overline{\tau}(M_0, M) \) represents a different firing sequence than \( \overline{\tau}(M_0, M_{des}) \). 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. \( \overline{\tau}(M_0, M_{des}) \)) to the firing count vector from \( M_0 \) to \( M \) (i.e. \( \overline{\tau}(M_0, M) \)), a new firing count vector \( \overline{\tau} \) 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 \( \overline{\tau} \in Q_{des} \) 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 \( Q_{des} \) is empty during the execution of Algorithm 2, the current marking is a deadlock. The evaluation is modified into \( \tau(M_0, M_{des}) = \rho_{dl} \) with \( \rho_{dl} \) as the penalty cost of a deadlock. Since \( \rho_{dl} \) influences the evaluation of \( v_{child} \) and through the backpropagation all nodes upstream, \( \rho_{dl} \) 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 \( \rho_{dl} \). 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 \( \overline{\tau}(v_{sim}) \) matches a firing count vector in \( Q_{des} \). If this is the case, the smallest duration estimation is determined by comparing \( \overline{\tau}(v_{sim}) \) with every firing count vector in \( Q_{des} \). If \( \overline{\tau}(v_{sim}) \) 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 \( \overline{\tau} \) is added to \( Q_{des} \). The estimation of the duration of the complete firing sequence \( \tau(v_{sim}) \) 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).

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 \( v_{child} \in V_{child} \) 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.
   \[
   cost(v_{child}) = \frac{\sum_{i=1}^{sim} cost_i(v_{sim})}{sim} \quad (7)
   \]

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

Algorithm 3: Evaluation

1. Input: \( PN, D, v_{sim}, Q_{des}, converge_{sim}, deadlock \)
2. if \( converge_{sim} = 1 \) then \( \tau(M_{sim}, M_{des}) = 0 \)
3. else if \( deadlock = 1 \) then \( \tau(M_{sim}, M_{des}) = \rho_{dl} \)
4. else
   \[
   \tau(M_{sim}, M_{des}) = \infty
   \]
   5. for \( j = 1 \) to \( |Q_{des}| \) do
   6. if \( Q_{des}(j) > \overline{\tau}(v_{sim}) \) then
   7. \( \tau = D(Q_{des}(j)) - \overline{\tau}(v_{sim}) \)
   8. if \( \tau < \tau(M_{sim}, M_{des}) \) then
   9. \( \tau(M_{sim}, M_{des}) = \tau \)
10. end if
11. end if
12. end for
13. \( Q_{des} = Q_{des} \cup \overline{\tau} \)
14. if \( \tau(M_{sim}, M_{des}) = \infty \) then
15. Calc. \( \tau(M_{sim}, M_{des}) \) and \( \overline{\tau}(M_{sim}, M_{des}) \) with (5)
16. else
17. \( \tau = \tau(M_{sim}, M_{des}) + \overline{\tau}(v_{sim}) \)
18. end if
19. end if
20. end if
21. end if
22. end if
23. Output: \( cost(v_{sim}) \)

\[
\text{cost}(v_{child}) = \min_{i=1}^{sim} \text{cost}_i(v_{sim}) \quad (8)
\]

\( cost(v_{sim}) \) 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
   \[
   \text{cost}(v_{parent}) = \frac{\sum_{v_{child} \in V_{child}} \text{cost}(v_{child})}{|V_{child}|} \quad (9)
   \]

2. Cost of parent node as the minimal cost of child nodes
   \[
   \text{cost}(v_{parent}) = \min_{v_{child} \in V_{child}} \text{cost}(v_{child}) \quad (10)
   \]

The backpropagation traverses the tree from leaf node \( v_l \) to \( v_0 \) and updates \( \text{cost}(v) \) 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 \( \Delta_{max} \) is exceeded. At the beginning \( converge_{sim} \) is set to 0, the root node of the tree is added and \( Q_{des} \) is initialized with \( \overline{\tau}(M_0, M_{des}) \). 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 \( \sigma_{T} \) 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 \( \Delta_{max} \) 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 \( \Delta_{max} \) 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

1. **Input:** PN, M₀, Mdes, depth, sim, ∆max
2. **Init:** create root node v₀ with marking M₀, converge = 0
3. Calculate \( \overline{\tau}^* (M₀, M_{des}) \) with (5), \( Q_{des} = \overline{\tau}^* (M₀, M_{des}) \)
4. **while** (converge = 0) \& (∆ < ∆max) **do**
   5. **Execute Selection** starting from v₀ resulting in leaf node v₁ (Section 3.1)
   6. **Execute Expansion** with Algorithm 1
   7. **if** Mdes \( \in \{M(v_{child}) | v_{child} \in \mathcal{Y}_{child}\} \) **then**
      8. **converge = 1**, \( \tau_{des} = \{v \in \mathcal{Y}_{child} | M(v_{child}) = M_{des}\} \)
   9. **else**
      10. **for** vchild \( \in \mathcal{Y}_{child} \) **do**
          11. **for** i = 1 to sim **do**
              12. **Execute Simulation** with Algorithm 2
              13. **Execute Evaluation** with Algorithm 3
              14. **Calculate** cost(vchild) with (7) or (8)
              15. **Delete** simulation nodes from Tree
      16. **Execute Backpropagation** starting from v₁ (Section 3.5).
   17. **if** converge = 1 **then**
      18. **Determine** timed firing sequence \( \sigma_t \) from v₀ to vdes
      19. **Output:** \( \sigma_t \)
   20. **else Output:** Tracking not successful

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 \( D = [1 \ 2 \ 1 \ 2 \ 1 \ 1 \ 1 \ 3 \ 3 \ 3 \ 3 \ 3 \ 3 \ 1 \ 1] \).

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 \( \rho_{dl} = D \overline{\tau}^* (M₀, M_{des}) \) (firing vector \( \overline{\tau}^* (M₀, M_{des}) \) 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 % 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 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 \( \Delta_{max} \) 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

<table>
<thead>
<tr>
<th>Mdes</th>
<th>στ</th>
<th>Success in %</th>
<th>Mean time in (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCTS</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>depth=2</td>
<td>sim=4</td>
<td>100/83.1</td>
<td>100/92.3</td>
</tr>
<tr>
<td>depth=7</td>
<td>sim=14</td>
<td>24/100/0.9</td>
<td>0/100/1</td>
</tr>
<tr>
<td>BFS</td>
<td></td>
<td>100/1</td>
<td>100/19</td>
</tr>
<tr>
<td>MPC</td>
<td>H=2</td>
<td>100/3.4</td>
<td>100/100/20.4</td>
</tr>
<tr>
<td>H=4</td>
<td>100/1.6</td>
<td>100/20.4</td>
<td>0/0/0</td>
</tr>
<tr>
<td>H=10</td>
<td>100/1</td>
<td>100/20.4</td>
<td>0/0/0</td>
</tr>
</tbody>
</table>

Table 4. Performance comparison of approaches

<table>
<thead>
<tr>
<th>Mdes</th>
<th>Opt. sequence found in %</th>
<th>Mean error in TU</th>
</tr>
</thead>
<tbody>
<tr>
<td>MCTS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>depth=2</td>
<td>sim=4</td>
<td>100/1</td>
</tr>
<tr>
<td>depth=7</td>
<td>sim=14</td>
<td>24/100/0.9</td>
</tr>
<tr>
<td>BFS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>100/1</td>
<td>100/19</td>
<td>0/100/6.2</td>
</tr>
<tr>
<td>MPC</td>
<td>H=2</td>
<td>100/3.4</td>
</tr>
<tr>
<td>H=4</td>
<td>100/1.6</td>
<td>100/20.4</td>
</tr>
<tr>
<td>H=10</td>
<td>100/1</td>
<td>100/20.4</td>
</tr>
</tbody>
</table>

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 |στ| 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, Hc = 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 adaptation 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


