Petri Net Based Trajectory Optimization

Ákos Hajdu[1], Róbert Német[1], Szilvia Varró–Gyapay[2], and András Vörös[1]*

[1] Department of Measurement and Information Systems
Budapest University of Technology and Economics, Budapest, Hungary
[2] Department of Computer Science and Systems Technology
University of Pannonia, Veszprém, Hungary

Abstract. Optimization problems are becoming more prevalent in the design of complex systems. Petri nets are widely used for the modeling of such systems. An optimization problem can be translated to find an optimal trajectory where a cost is assigned to each step. The reachability problem of Petri nets answers whether a given state is reachable from the initial state. However, reachability analysis is a computationally hard problem, especially in the case of asynchronous or infinite state systems. In this paper we examine a recently published algorithm that solves reachability using abstraction methods and we extend this approach to be able to handle optimal trajectory problems.

1 Introduction

Nowadays, information systems are becoming more and more complex where modeling and automatic analysis techniques gain an increasing task. Petri nets are widely used for the description of such systems due to their expressive power and simplicity. In addition cost parameters can be assigned to the transitions of the Petri net that enables the modeling of optimization problems in discrete event systems. Such optimization problems can be formulated as optimal trajectory problems when an optimal path is searched for from a starting state to a target state.

The so-called reachability problem is to answer the question whether a given state is reachable from an initial state in the system. There are many algorithms that solve or approximate the reachability problem of Petri nets. One of the most efficient is the so-called "counterexample guided abstraction refinement" (CEGAR) algorithm, which takes the state equation of the Petri net as the initial abstraction and refines it with constraints gained from the state space exploration. As the reachability problem is at least EXPSPACE-hard, the algorithm sacrifices the completeness for efficiency.

In this paper we introduce a new approach – based on the CEGAR algorithm – to solve the optimal trajectory problem. Our algorithm traverses the state

* This publication has been supported by the European Union and Hungary and co-financed by the European Social Fund through the project TÁMOP-4.2.2.C-11/1/KONV-2012-0004 - National Research Center for Development and Market Introduction of Advanced Information and Communication Technologies.

space with advanced exploration methods and discovers the necessary invariants to find the trajectory. The algorithm is extended to handle cost functions and the optimization problem drives the trajectory selection to find the optimal solution.

2 Background

In this section we introduce the background of our work. First, we present Petri nets (Section 2.1) as the modeling formalism used in our work based on [7]. Then we introduce the reachability problem, which serves as a basis for optimal trajectory search (Section 2.2). At the end of the section we present linear programming briefly (Section 2.3).

2.1 Petri nets

Petri nets are graphical models for concurrent and asynchronous systems, providing both structural and dynamical analysis. A discrete Petri net is a tuple $PN = (P, T, E, W)$, where P is the set of *places*, T is the set of *transitions*, with $P \neq \emptyset \neq T$ and $P \cap T = \emptyset$, $E \subseteq (P \times T) \cup (T \times P)$ is the set of *arcs* and $W : E \to \mathbb{Z}^+$ is the weight function assigning weights $w^-(p_j, t_i)$ to the edge $(p_j, t_i) \in E$ and $w^+(p_j, t_i)$ to the edge $(t_i, p_j) \in E$.

A *marking* of a Petri net is a mapping $m : P \to \mathbb{N}$. If a place p contains k tokens in a marking m then $m(p) = k$. The initial marking is denoted by m_0.

Dynamic behavior. A transition $t_i \in T$ is *enabled* in a marking m, if $m(p_j) \geq w^-(p_j, t_i)$ holds for each $p_j \in P$ with $(p_j, t_i) \in E$. An enabled transition t_i can *fire*, consuming $w^-(p_j, t_i)$ tokens from places $p_j \in P$ with $(p_j, t_i) \in E$ and producing $w^+(p_j, t_i)$ tokens on places $p_j \in P$ with $(t_i, p_j) \in E$. The firing of a transition t_i in a marking m is denoted by $m[t_i\rangle m'$ where m' is the marking after firing t_i.

A word $\sigma = t_1 t_2 \dots t_n \in T^*$ is a *firing sequence*. A firing sequence is *realizable* in a marking m and leads to m' (denoted by $m[\sigma\rangle m'$), if $m[t_1\rangle \dots [t_n\rangle m'$. The *Parikh image* of a firing sequence σ is a vector $\wp(\sigma) : T \to \mathbb{N}$, where $\wp(\sigma)(t_i)$ is the number of the occurrences of t_i in σ.

2.2 Reachability problem.

A marking m' is *reachable* from m if a realizable firing sequence $\sigma \in T^*$ exists, for which $m[\sigma\rangle m'$ holds. The set of all reachable markings from the initial marking m_0 of a Petri net PN is denoted by $R(PN, m_0)$. The aim of the *reachability problem* is to check if $m' \in R(PN, m_0)$ holds for a given marking m'. The reachability problem is decidable [6], but it is at least EXPSPACE-hard [5].

State Equation. The *incidence matrix* of a Petri net is a matrix $C_{|P|\times|T|}$, where $C(i,j) = w^+(p_i,t_j) - w^-(p_i,t_j)$. Let m and m' be markings of the Petri net, then the *state equation* takes the form $m + Cx = m'$. Any vector $x \in \mathbb{N}^{|T|}$ fulfilling the state equation is called a *solution*. Note that for any realizable firing sequence σ leading from m to m', the Parikh image of the firing sequence fulfills the equation $m + C\wp(\sigma) = m'$. On the other hand, not all solutions of the state equation are Parikh images of a realizable firing sequence. Therefore, the existence of a solution for the state equation is a necessary but not sufficient criterion for the reachability. A solution x is called *realizable* if a realizable firing sequence σ exists with $\wp(\sigma) = x$.

T-invariants. A vector $x \in \mathbb{N}^{|T|}$ is called a *T-invariant* if $Cx = 0$ holds. A realizable T-invariant represents the possibility of a cyclic behavior in the modeled system, since its complete occurrence does not change the marking. However, during firing the transitions of the T-invariant, some intermediate markings can be interesting for us.

Solution space. Each solution x of the state equation $m + Cx = m'$, can be written as the sum of a *base vector* and the linear combination of T-invariants [8], which can formally be written as $x = b + \sum_i n_i y_i$, where $b \in \mathbb{N}^{|T|}$ is the base vector and $n_i \in \mathbb{N}$ is the coefficient of the T-invariant $y_i \in \mathbb{N}^{|T|}$.

2.3 Linear programming (LP)

Linear programming is a mathematical approach for finding an optimal solution in a given mathematical model and requirements [2]. A linear programming problem is formalized as follows:

$$\begin{aligned} &\text{minimize} \quad c^T x, \\ &\text{subject to } Ax \leq b \text{ and } x \geq 0, \end{aligned}$$

where x is the vector of variables, b, c are vectors and A is a matrix of coefficients. The linear programming problem can be solved in polynomial time. When all the variables of x are integers, the problem is called *integer linear programming* (ILP) problem, which is NP-hard.

3 CEGAR

In this section we present the concept of abstraction (Section 3.1) and a recently published algorithm that applies the CEGAR approach on the reachability problem of Petri nets (Section 3.2). Furthermore, we present some of our previous improvements at the end of the section that optimize and extend the algorithm.

3.1 Abstraction

Abstraction is a general mathematical approach for solving hard problems. The abstract model has a less detailed state space representation by hiding the irrelevant details. However, due to the abstraction, some action of the abstract model may not be realizable in the original model. In this case, the abstraction has to be refined. This approach is called the "counterexample guided abstraction refinement" (CEGAR).

3.2 CEGAR approach on Petri nets

Recently, a new algorithm was published, which applies the CEGAR approach on the reachability problem of Petri nets [8]. Figure 1 shows an overview of the algorithm and each step is detailed later in this section.

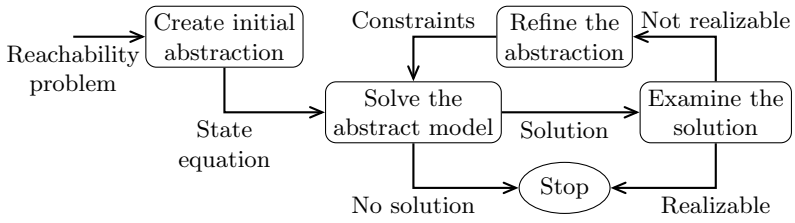


Fig. 1. Petri net CEGAR algorithm flowchart

Initial abstraction. The input of the algorithm is a reachability problem $m' \in R(PN, m_0)$, which is transformed into the initial abstraction, namely the state equation of the form $m_0 + Cx = m'$.

Solving the abstract model. Solving the abstract model (i.e. the state equation) is an integer linear programming problem. The ILP solver yields a minimal solution with respect to the cost function. In the original algorithm [8] the sum of the firing count of transitions is minimized in order to obtain trajectories with the shortest length.

The feasibility of the state equation is a necessary, but not sufficient condition for reachability, therefore if no solution exists, the target marking is not reachable. Otherwise, the obtained solution must be checked whether it is realizable in the original model (i.e. in the Petri net PN).

Examining the solution. The solution of the state equation is a vector $x \in \mathbb{N}^{|T|}$, where $x(t)$ denotes the number of times a transition $t \in T$ has to fire in order to reach m' from m_0. However, x does not include any information

about the order of the transition firings and whether they are enabled. Thus, the algorithm must explore the state space of the Petri net with the limitation that each transition t can fire at most $x(t)$ times. If the target marking m' can be reached with this limit (i.e. x is realizable), it is a sufficient proof for reachability. Otherwise, x is a counterexample and the abstraction has to be refined.

Refining the abstraction. If a solution x is not realizable, the ILP solver has to be forced to generate a different solution. This can be done by adding additional *constraints* (i.e. linear inequalities over transitions) to the state equation. The following two types of constraints were defined in [8].

- *Jump constraints* have the form $|t_i| < n$, where $n \in \mathbb{N}$, $t_i \in T$ and $|t_i|$ represents the firing count of the transition t_i. Jump constraints can be used to obtain different base vectors, exploiting their pairwise incomparability.
- *Increment constraints* have the form $\sum_{i=1}^{k} n_i|t_i| \geq n$, where $n_i \in \mathbb{Z}$, $n \in \mathbb{N}$, and $t_i \in T$. Increment constraints can be used to reach non-base solutions. This means that a new solution $x + y$ is obtained, where y is a T-invariant.

After adding the new constraint, the state equation may become infeasible, or a new solution is obtained. Figure 2 presents the solution space. The bottom dots represent base solutions, while the cones represent the linear space formed by the T-invariants. The upper dots correspond to non-base solutions. Jumps are denoted by dashed arrows and increments by continuous arrows.

At each non-realizable solution multiple jump and/or increment constraints can be applied. The algorithm traverses the solution space using depth-first search until a realizable solution is found, or the state equation becomes infeasible and there are no more solutions to backtrack to.

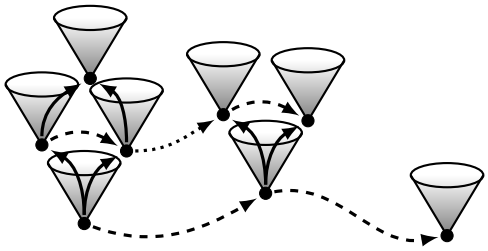


Fig. 2. Solution space of the state equation

Extensions. In our previous work [3] we proved by a counterexample that the original algorithm [8] is incorrect and we suggested a solution to overcome the problem. We also presented several examples where the algorithm could not

decide reachability. We extended the set of decidable problems, but the algorithm still lacks completeness. Furthermore, we introduced some new optimization methods in order to improve the efficiency of the algorithm [3].

4 Trajectory optimization

In this section we introduce our new approach that solves the optimal trajectory problem based on the Petri net CEGAR algorithm (Section 4.1). We formulate our method using a pseudo code (Section 4.2) and we also present some measurement results (Section 4.3).

4.1 Trajectory optimization using CEGAR

The core abstraction of the CEGAR algorithm is the state equation extended with further constraints, which forms an ILP problem. The ILP solver yields a solution minimizing a cost function on the variables. In the CEGAR approach variables correspond to firing counts of transitions. Originally, the algorithm assigned every transition an equal cost in order to produce trajectories with the shortest length [8].

This behavior makes the CEGAR algorithm suitable for trajectory optimization purposes. In our new approach we assign an arbitrary cost to transitions. Therefore, the ILP solver now minimizes the total cost of the trajectory (i.e. the sum of the costs of transitions) instead of its length.

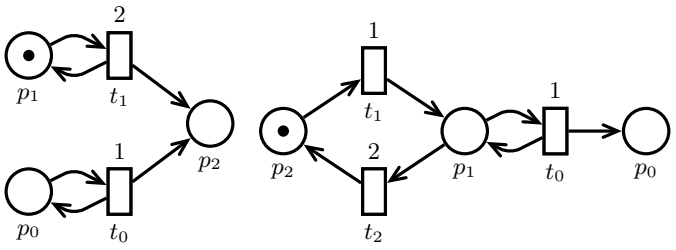
In order to fit this strategy into the CEGAR approach, the solution space traversal has to be modified slightly. The original algorithm explores the solution space using DFS search for a fast convergence to a realizable solution. However, we want an optimal solution regarding the cost function. Thus, we focus the search on solutions with minimal total costs. This is achieved by storing the not yet examined solutions in a sorted set, from which we always continue with the minimal one.

Two examples can be seen in Figure 3 where the costs are written above the transitions. Consider the example in Figure 3(a) where we want to produce a token in p_2. The minimal solution for the state equation is to fire t_0 once and t_1 zero times (i.e. $x_0 = (1, 0)$). However, this solution is not realizable since t_0 is not enabled. By applying the jump constraint $|t_0| < 1$, we obtain the solution $x_1 = (0, 1)$ (i.e. firing t_1 once), which is realizable.

Consider now the example in Figure 3(b) where we want to produce a token in p_0. The minimal solution for the state equation is to fire t_0 once (i.e. $x_0 = (1, 0, 0)$). However this solution is not realizable since t_0 is not enabled. By applying the increment constraint $|t_1| \geq 1$, we can obtain the solution $x_1 = (1, 1, 1)$, where the T-invariant $\{t_1, t_2\}$ "borrows" a token in p_1 to enable t_0.

4.2 Pseudo code

Algorithm 1 presents our new approach using pseudo code. The input of the algorithm is the reachability problem $m' \in R(PN, m_0)$ and the cost $z : T \to \mathbb{Z}$



(a) Jump constraint example

(b) Increment constraint example

Fig. 3. Example nets for jump and increment constraints

assigned to transitions. The algorithm stores the solutions in a list Q. At first it tries to solve the ILP problem with no constraints (the fifth parameter being $\emptyset$). While there is a solution in the list that was not examined, the algorithm takes out the solution x that has minimal total cost (zx). If the solution is realizable by some firing sequence σ, the algorithm terminates. Otherwise it searches for jump and increment constraints. An ILP problem is solved for each constraint c' by adding c' to the previous constraints of x. If new solutions are found they are put in the list Q. If all solutions were examined and no realizable is found, the answer is "Not reachable".

Algorithm 1: Trajectory optimizing CEGAR algorithm

Input : Reachability problem $m' \in R(PN, m_0)$ and transition costs z
Output : Trajectory σ or "Not reachable"

$C \leftarrow$ incidence matrix of PN;
$Q \leftarrow \emptyset$; // List of solutions
$Q \leftarrow \text{SolveILP}(m_0, m', C, z, \emptyset)$;
while $Q \neq \emptyset$ **do**
  $x \leftarrow \{x \mid x \in Q, zx \text{ is minimal}\}$;
  if x *is realizable* **then stop** and output σ for x;
  else
    foreach *Jump and increment constraint* c' **do**
      $Q \leftarrow \text{SolveILP}(m_0, m', C, z, \{\text{constraints of } x\} \cup \{c'\})$;
    end
  end
end
Output "Not reachable";

4.3 Results

We implemented the CEGAR algorithm with our new approach as a plug-in for the *PetriDotNet* framework [1]. We evaluated our implementation on the *traveling salesman* problem. The results can be seen in Table 1.

Table 1. Measurement results for the traveling salesman problem

Number of nodes	Runtime (s)
4	0,04
6	0,14
8	0,66
9	0,90
10	1,95
11	9,49
12	24,57
13	1067

Traveling salesman is a graph traversal optimization problem, which belongs to the class of NP-complete problems [4]. The parameter of the problem is the number of nodes in the graph.

5 Conclusion

In our paper we presented a promising new approach for the optimal trajectory problem of discrete event systems. We translated this problem to the reachability analysis of Petri nets by assigning a cost to the transitions. We solved the reachability problem using the recently published and very efficient CEGAR approach. We extended the algorithm to be able to handle transition costs and we modified the search strategy in order to reach the optimal solution. We implemented our new approach and demonstrated its efficiency with measurements. A possible future research direction is the optimization of continuous systems.

References

1. Homepage of the *PetriDotNet* framework., `http://petridotnet.inf.mit.bme.hu/`, [Online; accessed 04-Nov-2014]
2. Dantzig, G.B., Thapa, M.N.: Linear programming 1: introduction. Springer-Verlag New York, Inc., Secaucus, NJ, USA (1997)
3. Hajdu, Á., Vörös, A., Tamás, B., Mártonka, Z.: Extensions to the CEGAR approach on Petri Nets. Acta Cybernetica 21(3), 401–417 (2014)
4. Karp, R.M.: Reducibility among combinatorial problems. Springer (1972)
5. Lipton, R.: The Reachability Problem Requires Exponential Space. Research report, Yale University, Dept. of Computer Science (1976)

6. Mayr, E.W.: An algorithm for the general Petri net reachability problem. In: Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing. pp. 238–246. STOC '81, ACM, New York, NY, USA (1981)
7. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE 77(4), 541–580 (April 1989)
8. Wimmel, H., Wolf, K.: Applying CEGAR to the Petri net state equation. In: Tools and Algorithms for the Construction and Analysis of Systems, 17th International Conference, TACAS 2010 Proceedings. vol. 6605, pp. 224–238. Springer (2011)