Electronic Communications of the EASST
Volume 51 (2012)

Proceedings of the 5th International Workshop on Petri Nets, Graph Transformation and other Concurrency Formalisms (PNGT 2012)

Optimization in Graph Transformation Systems with Time Using Petri Net Based Techniques

Szilvia Varró-Gyapay

12 pages

Guest Editors: Julia Padberg, Kathrin Hoffmann
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Optimization in Graph Transformation Systems with Time Using Petri Net Based Techniques

Szilvia Varró-Gyapay

gyapay@mit.bme.hu
Budapest University of Technology and Economics
Department of Measurement and Information Systems

Abstract: Extra–functional properties of IT systems have to be analyzed and subsequently optimized carefully during the design phase in order to assure a proper quality of service and decrease operational costs. Several verification and validation methods are known to check the correctness of the system services, while optimization may serve to reach boundaries thus minimizing costs or duration of operating the system. However, the combination of the best practices of the two fields according to the purpose of the analysis is a challenging question. In a previous paper, we showed how such a problem can be formalized and solved when the evolution of the system is captured by graph transformation systems (GTS) with cost parameters attached to each graph transformation rule denoting the cost of firing the rule.

This technique is adapted in the current paper to deliver a time–optimal trajectory in a GTS with time. While the *cost of a GT rule sequence* always equals to the sum of the cost of the involved GT rules, the concurrent application of GT rules may reduce the *minimal duration of a GT rule sequence*, which is a major conceptual difference concerning optimization.

Keywords: graph transformation with time, Petri nets, optimization

1 Introduction

The analysis of the qualitative and quantitative requirements of an IT system identifies the errors in the model and improves the efficiency of the system in order to assure a proper quality of service and decrease the operational costs and time. Qualitative requirements can be for instance the guarantee of safety of a system, while quantitative requirements are the optimality and the reliability of the system. Combined optimization and validation, verification problems can be typically expressed as reachability problems with quantitative or qualitative measurements called *optimal trajectory problem*.

The current paper proposes a method *to deliver a trajectory with minimal duration in graph transformation systems (GTSs) with time according to a reachability criteria*. At first, (i) the GTS is transformed into a cardinality P/T net where places are to count the elements of an edge or node type and transitions simulates the effect of a rule application in the number of the elements of a certain type abstracting from the structure of the graph. Then (ii) an optimal solution is delivered for the cardinality P/T net, and (ii) this solution is used as a hint to perform a guided exploration at the level of GTS.

Related work. Related works can be categorized into two groups: that aim at the optimization of the modeled system (usually in a quite specialized case or systems with specific structure) in the field of GTSs, e.g. [FL76, TÖ9] or Petri nets, e.g. [TL04, LH11], and that aim at the analysis of the GTS, e.g. the reachability analysis of timed GTSs based on the time automata approach [HSE10] or the (guided) exploration of the search space, e.g. [HHRV11, Ren03]. The current approach is novel w.r.t. previous works that it combines the GTSs with time based modeling of the optimization problems, and the optimization of the transformed P/T net by solving integer linear programming problems.

Previous contributions. In [GSV04] GTSs with *time* were modeled and optimized in the SPIN model checker. The advantage of the proposed method is that the SPIN model checker retrieves directly the optimal trajectory by searching the state space. However, the drawback of the approach is that all the elements of the graph that can be reached during the evolution of the graph transformation system have to be known a priori.

In [VV06] an optimal sequence of graph transformation rules in GTSs with *cost* between two graphs was delivered: (i) at first, the GTS was abstracted into a cardinality Petri net with cost parameters, and (ii) at second, an optimal trajectory of the Petri net abstraction was delivered. (iii) Then the corresponding GT rule sequences were checked in the GTS with cost: if there were no appropriate solution, the next optimal trajectory of the Petri net was developed and checked in the GTS level again, and so on.

Since the possible concurrent firing of rules may reduce the *minimal duration of a GT rule sequence* in GTSs with time this approach cannot be adapted directly to the timed case. Therefore, the underlying optimization problem of the abstraction P/T net is modified such that it models the concurrent firing of rules (transitions).

The novelty of the current approach compared with the previous ones is that it *handles time* in the GTS such that also the *creation of new graph elements is allowed.* On the other hand, this concept shows a promising way how to extend the calculation of an optimal path for other GTS based approaches. The paper proposes a *solution to compute time–optimal trajectories in GTSs with time*, where the evolution of the system is captured by (i) graph transformation rules, and (ii) timestamps of the graph elements. This approach is illustrated in the paper in a special case of GTS with time.

2 Basic Definitions

2.1 Graph Transformations with Time

In [GHV02] we introduced graph transformation with time as typed graph transformation with time attribute. The time attribute is a logical clock (typically) with non-negative integer values that can be attached to nodes. The time attribute has a distinguished role to define how the time progresses in discrete steps in a consistent way.

Graph transformation (GT) provides a rule-based manipulation of graph models. A *graph transformation system* $GTS = (R, TG)$ consists of a type graph TG and a finite set of graph transformation rules typed over TG. Formally, a graph $G = (N, E, src, trg)$ is a 4-tuple with a set N of nodes, a set E of edges, a source and a target function $src, trg : E \to N$. A type graph TG is an ordinary graph. An instance graph G is typed over TG by a typing morphism $type : G \to TG$.

A *graph transformation (GT) rule* typed over a type graph TG is given by $r = (L \xleftarrow{l} K \xrightarrow{r} R)$ where L (left-hand side or LHS), K (context) and R (right-hand side or RHS) graphs are typed over TG and graph morphisms l, r are injective. The negative application conditions (NAC) of a GT rule is a (potentially empty) set of pairs (N, n) with N being a graph also typed over TG and $n : L \to N$ an injective graph morphism. In addition, a rule may have several NACs.

The *application* of a rule $r = (L \xleftarrow{l} K \xrightarrow{r} R)$ to a *host graph* G alters the model graph by replacing the pattern defined by L with the pattern of R. This is performed by (i) *finding a match* of the L pattern in model G; (ii) *checking the negative application conditions* N, i.e. the rule may be applied only if there is no match of N in L prohibiting the presence of certain elements; (iii) *removing* a part of the model G that can be mapped to the L pattern but not the R pattern yielding an intermediate graph D; (iv) *adding* new elements to the intermediate graph D which exist in R but not in L yielding the derived graph H.

A *graph transformation sequence (GT sequence)* is a sequence of rule applications: $G_0 \overset{r_1}{\Longrightarrow} G_1 \overset{r_2}{\Longrightarrow} G_2 \Longrightarrow \ldots$ We say that a graph G is reachable from G_0 if there is a transformation sequence from G_0 to G.

Typed graph transformation with time. To incorporate time into typed graph transformation, *time data type* is introduced as domain for time-valued attributes. In [GHV02] we followed the approach of *time environment–relationship* (TER) nets to define the semantics of graph transformation with time. TER nets are high–level Petri nets where tokens are environments such that time is a distinguished environment that assigns values to variables [GMMP91].

A *type graph with time* TG is a type graph with attribute *time*. An instance graph with time over TG for the data type $T = \langle D_{time}, +, 0, \geq \rangle$ is an instance graph $< G, type : G \to TG >$ over TG such that the data type sort *time* is interpreted by D_{time}, that is, $D_{time} = \{x \in G_N | type(x) = time\}$. Ordinary nodes are connected to the time attribute nodes by *chronos* edges representing the time attribute declaration, i.e. $chronos \in TG_E : src(chronos) = TG_N \setminus \{T\}, trg(chronos) = T$.

Graph transformation system with time. A graph transformation system with time consists of a type graph such that the above defined time attribute is included in it, and the rules manipulate the time attributes in the graphs such that

- **Condition 1. Local monotonicity:** for all vertices $x \in L$ and $y \in R$, the timestamp of x is smaller or equal to the timestamp of y, and
- **Condition 2. Uniform timestamps:** for all vertices $x, y \in R$ the timestamp of x equals the timestamp of y.

These conditions ensure a behaviour of time such that (i) an operation or transaction specified by a rule cannot take negative time, i.e., it cannot decrease the timestamps of the nodes it is applied to, and (ii) each rule application is atomic, that is, all effects specified in the RHS are observed at the same time, called the firing time of the rule.

Duration of a transformation sequence. In analogy with TER nets [GMMP91], for each transformation sequence s using only rules that satisfy the above two conditions, there exists an equivalent sequence s' such that s' is *time-ordered*, that is, timestamps are monotonically non-decreasing as the sequence advances. The *duration of a transformation sequence* p is the firing time of the last rule in a corresponding time–ordered path denoted by $d(p)$.

Example 1 *An example GTS with time is illustrated by a storage testing and reconfiguration line. The testing of an untested storage needs a non–reserved test cell. After a storage is tested, it is ready to ship to the customers. Both untested and tested storages can be reconfigured into another untested or tested storages with the other type. The model elements are shown in the type graph in Fig. 1 on the left as a UML class diagram, while the rules are depicted in Fig. 2.*

An instance graph is shown in Fig. 1 on the right described as a UML object diagram: there is an untested storage with type1 *that is under test (connected by a* test1 *edge to a reserved test cell), a tested storage with* type1*, and an untested storage with* type2*.*

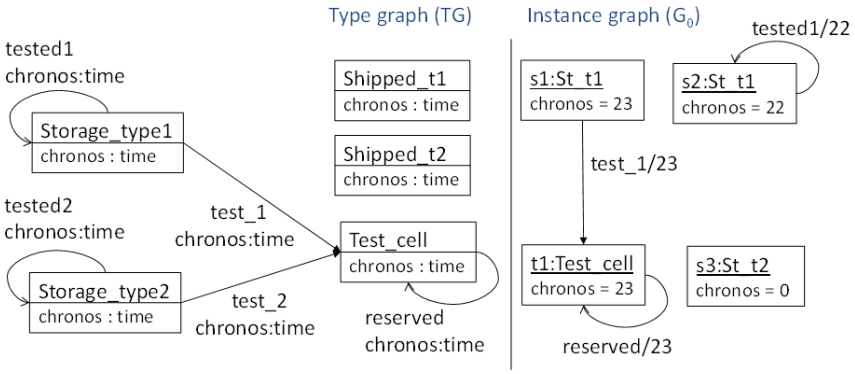

Figure 1: Type and instance graph

In Fig. 2 rules manipulating a storage with type1 *are presented. The name and the time of the rule are written over and below the blue rule arrow, respectively. The timestamp of an edge is written after the type of the edge separated by a /. Since the current example does not allow multiple edges with the same type between two nodes, the identifiers of the edges are omitted.*

The time of a rule equals the maximum of the timestamps in the LHS plus the duration of the rule, and all graph elements in the RHS get this time as its timestamp. It is easy to check that conditions Local monotonicity *and* Uniform timestamp *hold for all rules.*

For space consideration only the testing of a storage is described in details. A storage can be tested if (i) it is not tested yet, (ii) its test has not started yet, i.e. there is no test cell connected to it, and (iii) there exists a free test cell. These conditions are described by the negative application conditions NAC1-3*, respectively. If there exists such a storage* s1 *and test cell* t1 *rule* start_test_1 *is applied: a* test1 *edge is drawn from* s1 *to* t1 *representing the start of the test, and* t1 *gets an edge* reserved *denoting its allocation for* s1*. The duration of the rule is* 1 *time unit.*

The test of a storage is carried out by applying rule test_1*: the rule searches for a storage that has a* test1 *edge to a reserved test cell. After the test is carried out, a self–loop edge* tested1 *is created for the storage denoting its state and the other edges are deleted, i.e. the test cell becomes again free. The duration of the test is* 21 *time unit.*

The rules for a storage with type2 *are similar such that the durations are* $1, 23, 16, 13$*, and* 1 *for the rules* start_test2*,* test_2*,* reco_2*,* reco_2_tested*, and* ship_2*, respectively. Note that the example can be modeled in several ways using attributes, or edges to denote the type of a storage. However, the current version provides the easy construction of the reachability statement.*

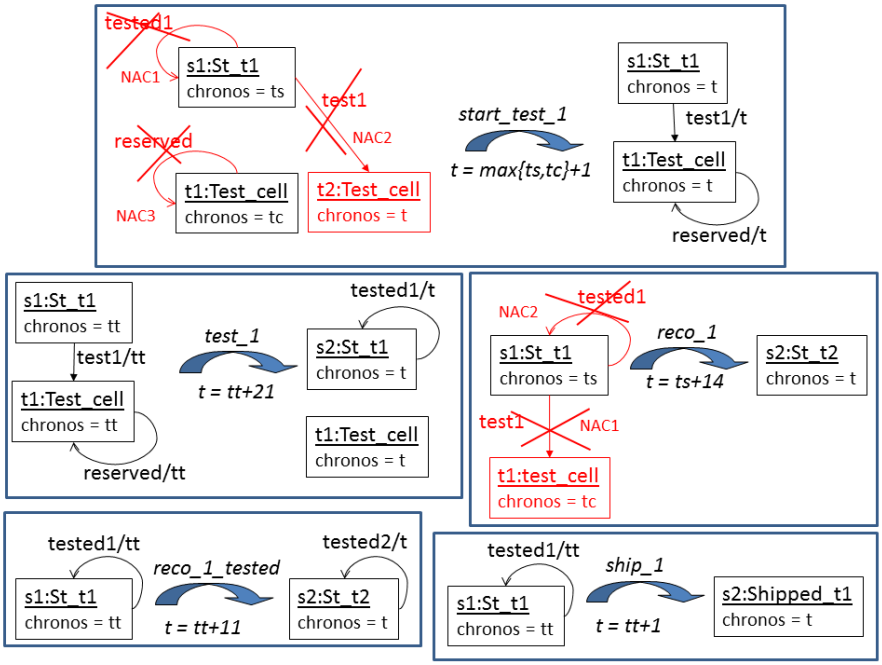

Figure 2: GT rules

2.2 Place/Transition Nets

Now we give a short introduction into the theory of Place/Transition nets based on [Mur89].

A Place/Transition net (or shortly P/T net) is a 5-tuple $PN = (P,T,E,w,M_0)$ where P is a set of places, T is a set of transitions, $E \subseteq (P \times T) \cup (T \times P)$ is the set of arcs , $M_0 : P \to \mathbb{N}$ is the initial marking mapping places to nonnegative integers, while $w : E \to \mathbb{N}^+$ maps arcs to positive integers. Furthermore $\bullet t = \{p \mid (p,t) \in E\}$ denotes the input places, while $t\bullet = \{p \mid (t,p) \in E\}$ denotes the output places of transition t. Finally, $\bullet p = \{t \mid (t,p) \in E\}$ are the incoming transitions while $p\bullet = \{t \mid (p,t) \in E\}$ are the outgoing transitions of place p.

A transition t is enabled if each of its input places contains at least as many tokens as is specified by the weight function, formally, $\forall p \in \bullet t : M(p) \geq w(p,t)$. The firing of an enabled transition t removes a $w(p,t)$ amount of tokens from the input places (p), and $w(t,p)$ tokens are produced for the output places, i.e. $\forall p \in P : M'(p) = M(p) - w(p,t) + w(t,p)$.

The incidence matrix W of the net describes the net token flow (of the P/T net) when firing a transition. Mathematically, W is a $|P| \times |T|$–dimensional matrix of integers $\mathbb{N}$ such that $W_{ij} = w(t_i,p_j) - w(p_j,t_i)$, where $1 \leq i \leq |P|, 1 \leq j \leq |T|$.

A *transition firing sequence* (or shortly firing sequence) $s = \langle t_{i_1}, t_{i_2}, \ldots, t_{i_k} \rangle$ is a sequence of transition firings between states M_0 and M_k such that $\langle M_0, t_{i_1}, M_1, t_{i_2}, \ldots, t_{i_k}, M_k \rangle$ where for all $1 \leq j \leq k$ t_{i_j} is enabled in M_{j-1} and M_j is yielded by the firing of t_{i_j} in M_{j-1}. The *transition occurrence vector* or *Parikh–vector* σ of a trajectory $s = t_{i_1}, \ldots, t_{i_k}$ counts the occurrence number of the individual transitions in the firing sequence, i.e. $\sigma(t_j) = |\{i_l | i_l = j, l = 1..k\}|$.

A marking M is *reachable* from a state M_0 (denoted by $M_0[s > M$) if there is a transition firing sequence s from M_0 to M. Then the so–called state equation holds: $M = M_0 + W \cdot \sigma$, where σ is the transition occurrence vector of s. A marking $M_{partial}$ is *partially reachable* from M_0 if there is a transition firing sequence s from M_0 to a marking M such that $M_{partial} \leq M$. Then the following inequality holds: $M_{partial} \leq M_0 + W \cdot \sigma$, where σ is the *transition occurrence vector* of s.

The duration parameter of a transition in the abstraction P/T net is defined as in [Ram74], as follows. A *Petri net with duration parameters* is a $PN_d = \langle PN, d \rangle$, where the duration function $d : T \to \mathbb{N}$ assigns duration to the firing of the individual transitions, such that each firing of a transition in a trajectory $s =< t_{i_1}, \ldots, t_{i_k} >, t_{i_r} \in T$ has a beginning and an end time. *The duration of the firing sequence* can be interpreted as the end time of all the fired transitions in the sequence. Once a transition is enabled and selected to fire, the tokens are removed at the beginning time of the firing from the input places and the duration of the transition starts to be counted down. After the duration time elapsed, the required tokens are put into the output places. This way, conflicting transitions cannot disable each other by stealing tokens *during* the firing of a transition.

Now we provide a detailed description how to carry out optimization of GTS with time using a state space traversal strategy guided by optimal solutions of the P/T net abstraction of the GTS.

3 Optimization of GTS with Time by Guided State Space Traversal

3.1 Optimal Trajectory Problem for GTS

Combined reachability and optimization problems can be defined as follows: (i) let decide, whether a particular state of the system is reachable from the initial state using the available resources, and (ii) if the state is reachable, then an optimal trajectory has to be computed. Frequently, in engineering problems only a subset of nodes and edges is relevant from practical point of view. Therefore, in case of graph transformation systems *partial reachability* means that we should reach a graph G that *covers* the desired (partial) graph $G_{partial}$, i.e. there is a subgraph of graph G that is isomorphic to $G_{partial}$ (denoted by $G \supseteq G_{partial}$). We also say that $G_{partial}$ is partially reachable from G_0.

Then the optimal trajectory problem can be described as follows. Given a graph transformation system GTS together with an initial instance graph G_0, and a graph $G_{partial}$, find a trajectory (path) tr from graph G_0 to graph G ($G_0 \stackrel{p}{\Longrightarrow} G$) such that $G \supseteq G_{partial}$, and it is optimal, i.e. $\forall tr' : G_0 \stackrel{tr'}{\Longrightarrow} G', G' \supseteq G_{partial} : d(tr) \leq d(tr')$. We denote this problem as $OT = ((GTS, G_0), G_{partial})$.

3.2 P/T net Abstraction

Let an optimal trajectory problem of GTS with time $OT = ((GTS, G_0), G_{partial})$ be given where (i) time is discrete, i.e. the time data type is defined by the natural numbers, (ii) each graph element has a chronos value, (iii) and the time of the rule application is defined as the maximum of the timestamps in the LHS plus a constant, denoted as the duration of the rule application $d(r)$. Note that a GTS satisfying these criteria conforms to the time–conditions of Section 2.1.

The essence of the abstraction technique is to derive a cardinality P/T net with duration pa-

rameters which simulates the original GTS by abstracting from the structure of instance graphs and only counting the number of elements (nodes or edges) of a certain type by placing tokens to a corresponding place. These tokens are circulated by transitions derived from each GT rule which simulate the effect of the rule on the number of elements of certain types by adding and removing tokens from corresponding places.

Timestamp. The timestamp of a graph element x in an instance graph G is denoted as $time(x)$.

Cardinality. Let $card(G,x)$ denote the cardinality (i.e. the number of graph objects) of type $x \in TG$ in graph G. Formally, $card(G,x) = |\{n \mid n \in N \cup E \wedge type(n) = x\}|$.

Mapping. Then the mapping $\mathscr{F}()$ of a graph transformation system with time $GTS_t = (Rule, TG), G_0, G_{partial}$ into a cardinality P/T net with duration $PN_d := \langle (P,T,E,w,M_0), d \rangle$ is defined as follows. Note, that in the definition a node or edge of the graph are ordinary nodes and edges (not time attribute nodes or chronos edges).

- *Types into places.* For each node and edge $y \in N_{TG} \cup E_{TG}$ in the type graph TG, a corresponding place $p_y = \mathscr{F}(y)$ is defined in the cardinality P/T net.
- *Instances into tokens.* For each node and edge $x \in N_G \cup E_G$ in an instance graph G with type $y = type(x)$, a token is generated in the place $\mathscr{F}(y)$. A corresponding marking m_G of the instance graph G is an assignment of natural numbers to places: $m_G(p_y) = |\{x | type(x) = y \wedge x \in G\}|$.
- *Rules into transitions.* For each rule $r \in Rule$, $r = (L \xleftarrow{l} K \xrightarrow{r} R)$, $time(r) = max(\{time(x) | x \in L\}) + d(r), d(r) \in \mathbb{N}$, in the graph transformation system GTS, a transition $t_r = \mathscr{F}(r)$ is generated in the cardinality P/T graph such that

  Left-hand side: If there is a graph object x of type $y = type(x)$ in L, then an incoming arc (p_y, t_r) is generated in the net where $p_y = \mathscr{F}(y)$. This way the required number of tokens in the preset of transition t_r is equal to the number of graph objects in L of the type of the corresponding place. Formally, $\forall x,y : x \in L \wedge y = type(x) \wedge \mathscr{F}(y) = p_y \implies (p_y, t_r) \in E \wedge w(p_y, t_r) = card(L,y)$.

  Right-hand side: If there is a graph object x of type $y = type(x)$ in R, then an outgoing arc (t_r, p_y) is generated in the net where $p_y = \mathscr{F}(y)$. In other words, as many tokens are produced to place p_y by firing the transition as the number of graph objects of the same type y in the right–hand side of the rule. Formally, if $\forall x,y : x \in R \wedge y = type(x) \wedge \mathscr{F}(y) = p_y \implies (t_r, p_y) \in E \wedge w(t_r, p_y) = card(R,y)$.

  Duration: The duration of the transition is $d(\mathscr{F}(r)) = d(r)$.

Example 2 *Let us revisit our ongoing example. In Fig. 3, a part of its cardinality P/T net is shown: the places of the P/T net and the abstraction transition of rule* test_1 *(see Fig. 2). The tokens in the places represent the instance graph in Fig. 1 on the right.*

The LHS of the rule contains two nodes with types St_t1 *and* Test_cell*, and two edges with types* test1 *and* reserved*. Thus the corresponding transition has four incoming arcs from the corresponding places. Since the RHS consists of two nodes with types* St_t1*, and* Test_cell*, and one edge with type* tested1*, the transition has three outgoing arcs to the corresponding places. The duration of the transition is equal to* 21*.*

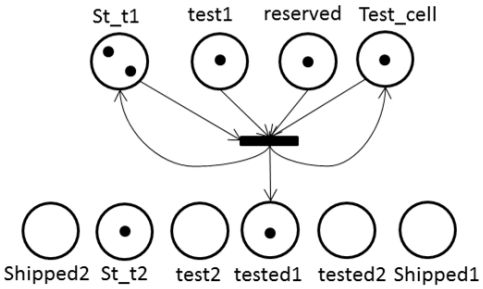

Figure 3: The cardinality P/T net of rule *start_test_1*

Simulation. The mapping $\mathscr{F}()$ is a proper abstraction of the GTS with time in the sense that the derived P/T net simulates the original GTS as shown in [VV06]. In other terms, whenever a rewriting step is executed in the GTS on an instance graph, then the corresponding transition can always be fired in the corresponding marking in the P/T net, furthermore, the result marking is an abstraction of the result graph. In this respect, for all firing sequence in the GTS there is a firing sequence in the cardinality P/T net but not the other way around. For instance, if a GT rule cannot be applied in the GT level because of violating some NACs, the corresponding transition in the P/T net may fire because no check of NACs is encoded into the transition.

The P/T net construction ensures that whenever a GT rule is applied in time instance l, the corresponding transition in the P/T net can be also fired in time instance l. Moreover, the duration of the two sequences are equal according to the equal duration of the GT rules and their abstraction transition.

As the complexity of the derived abstraction is lower than the complexity of the original GTS, the solution of the corresponding optimal trajectory problem in the P/T net can be used as a hint when exploring the state space of the original GTS.

3.3 Deriving an ILP Problem for the Optimal Trajectory Problem in the P/T Net

Since the abstraction P/T net simulates the underlying GTS, for each path (trajectory) p starting from G_0 and leading to an instance graph G which covers $G_{partial}$, there is a transition firing sequence s in the P/T net such that $M_{partial}$ is partially reachable from M_0 by s. In addition, each component of the transition occurrence vector σ of s is equal to the number of corresponding GT rule applications in trajectory p. Thus such a transition occurrence vector σ has to satisfy the state inequality $M_{partial} \leq M_0 + W \cdot \sigma$, where W is the incidence matrix of the P/T net. In other words, the search for an appropriate GT sequence can be carried out by finding a transition occurrence vector with minimal time which is followed by the check of the existence of an executable GT trajectory that is compatible with the transition occurrence vector.

Such a minimal transition occurrence vector can be derived by formulating an *integer linear programming (ILP) problem* from the state inequalities adding an objective function that minimizes the duration of a corresponding trajectory. (For an introduction into ILP problems see [Win94].) However, the core problem to establish such an ILP problem is the expression of a *linear* objective function that minimizes the firing time of the last transition in a time–ordered

sequence. In order to deliver an appropriate solution, the state inequalities are divided into state inequalities that describes the change of the net in *each time instance*.

Counting tokens per time instances. Let $token[i][l]$ denote the number of tokens at place p_i in time instance l, and let $\sigma[j][l]$ denote the number of transitions t_j that start to fire concurrently in time instance l. Then the token number at a place p_i changes in time instance l as follows: (i) some transitions starting their firing in time instance l remove tokens, while (ii) transitions that end their firing in time instance l produce tokens to this place. Mathematically, $token[i][l] = token[i][l-1] - \sum_j \sigma[j][l-1] \cdot w(p_i,t_j) + \sum_j \sigma[j][l-d(t_j)] \cdot w(t_j,p_i)$. A transition t_j may fire in time instance l if and only if there are enough tokens in its input places p_i, i.e. $token[i][l] \geq \sum_j \sigma[j][l] \cdot w(p_i,t_j)$. Note that if a transition ends its firing in time instance l, the tokens are produced into its output places only in time instance $l+1$.

Linear objective function. A linear objective function that minimizes the duration of a firing sequence can be defined by the introduction of a sink transition t_{last}, that becomes enabled if $M_{partial}$ is reached. Formally, $\bullet t_{last} = \{p_i | 0 < M_{partial}(p_i)\}, w(p_i,t_{last}) = M_{partial}(p_i)$. A Boolean variable $\sigma'[l]$ is defined for the sink transition such that it is equal to 1 for a time instance l if and only if the sink transition fires in l. Since the sink transition fires exactly ones in the trajectory, $\sum_{l=0}^{maxTime} \sigma'[l] = 1$ holds. Furthermore, the sink transition may fire in time instance l if there are enough tokens at its input places, i.e. $M_{partial}[i] \cdot \sigma'[l] \leq token[i][l] \forall p_i \in \bullet t_{last}$.

Since the sink transition fires at last, all other transitions are forbidden to fire after the sink transition. Adding a big enough number K to the problem, all $\sigma[j][l]$ components are restricted to 0 after the firing of the sink transition in time instance k by the following inequalities: $\sum_{j=1}^{|T|} \sum_{r=l}^{maxTime} \sigma[j][r] \leq K \cdot (1 - \sum_{r=0}^{l} \sigma'[r])$. Then a linear objective function is formulated as $\min \sum_{l=0}^{maxTime} (\sigma'[l] \cdot l)$ that yields k if and only if the sink transition fires in time instance k.

Time horizon. However, such an ILP problem has to be finite, i.e. the time instances have to be constrained to a time horizon. Since the duration of an optimal trajectory (if exists) is less than the sum of the durations of the included transitions, a time horizon can be estimated by solving an ILP problem that minimizes the sum of the durations of the transitions in the transition occurrence vector satisfying the state inequalities. However, it may happen, that there will be no solution within this time horizon because it is not applicable to the GTS with time. In this case a new, greater solution has to be generated to provide a new time horizon, and so on.

4 Guiding Exploration of the GT state space

A solution vector encodes not only the number of the fired transitions in this case but also the firing time of the transitions. This way, if there is a candidate solution vector, the strategy of the search for an optimal GT sequence is to *try to apply the GT rules to the instance graph in the same time instance in which the corresponding transition fires*. Then the traversal of a branch of the transition graph is suspended, i.e. the discovery of the unpromising paths is postponed,

- if a GT rule cannot be applied in the given time instance as many times as it is counted in the solution occurrence transition vector, or
- the graph yielded by applying the last rule (according to the solution transition vector) does not cover the target graph pattern. It may occur, for instance, if the rules create edges between nodes, and the target graph pattern contains edges connected to the same node: then it may happen, that the required rules are applicable to the model graph in the required order but the created edges are not connected.

If all the GT rules can be applied according to the solution transition occurrence vector and the resulted graph covers the required partial graph, then an optimal trajectory is found. If there are no branches to continue the state space traversal, the next best solution to the ILP problem is generated and is taken into account in the search strategy during the exploration of the previously unpromising paths.

Note that the simulation property of the P/T net abstraction and the strategy of the state space traversal ensures that (i) the duration of a GT sequence during the state space traversal cannot exceed the duration of the solution occurrence transition vector, and (ii) the target graph will not be covered with less duration before applying all the rules corresponding to the candidate transition occurrence vector. Otherwise there should have been another solution transition occurrence vector with less duration determining this path.

Example 3 *Let us revisit our storage example. Let an instance graph G_0 be given consisting of two storages with type* St_t1*, one storage with type* St_t2*, and two* Test_cells *in Fig. 4 such that the timestamp of all nodes is equal to* 0*. The target graph to be covered consists of one node with type* Shipped1 *and two nodes with* Shipped2*.*

At first, maxTime is calculated as described in Section 3.3. The objective value of the optimal solution is 48 *time units. Within this time horizon, the following optimal algebraic solution is delivered at first: $start\_test\_t1/0$, $start\_test\_2/0$, $start\_test\_2/1$, $start\_test\_t1/3$, $test\_1/1$, $test\_2/4$, $ship\_2/28$,$reco\_1\_tested/22$, $ship\_2/33$, $test\_1/22$, $ship\_1/43$, $t\_last/44$. The duration of this firing sequence is* 44 *defined by the firing time of the sink transition.*

Now let us start to construct a corresponding GT sequence. Since the optimal transition occurrence vector defines a time–ordered transition sequence, the search strategy is to try to apply the GT rules in the same time instance. At time 0 *two rules have to be applied:* start_test_t1 *and* start_test_t2 *both with duration* 1*. Since these rules make both test cells reserved, the rule* start_test_t2 *cannot be applied at time* 1*. This pseudo solution is delivered because the NACs of the rule that prohibit the application of the rule in this case are not encoded into the P/T net. Therefore the state space traversal is stopped, and the next best solution is generated.*

The next best solutions with durations 44 *time units fail to retrieve a valid GT sequence because of the same problem. However, the next best solution with duration* 45 *time units is already fireable thus an optimal solution is found: $start\_test\_t1/0$, $start\_test\_2/0$, $test\_1/1$, $start\_test\_t1/22$, $test\_2/1$, $ship\_2/24$, $reco\_1\_tested/22$, $ship\_2/33$, $test\_1/23$, $ship\_1/44$, $t\_last/45$. A corresponding GT sequence is depicted in Fig. 4 (where the application of the last two rules* ship_1 *and* ship_2 *are not depicted). The match of the LHS of the rules are drawn with green, and due to some space consideration independent rules are applied in one step.*

The ILP problem of the example was solved by using the CPLEX opimizer *[cpl] while the*

cardinality P/T net and the GT sequence was generated by VIATRA2 *[via].*

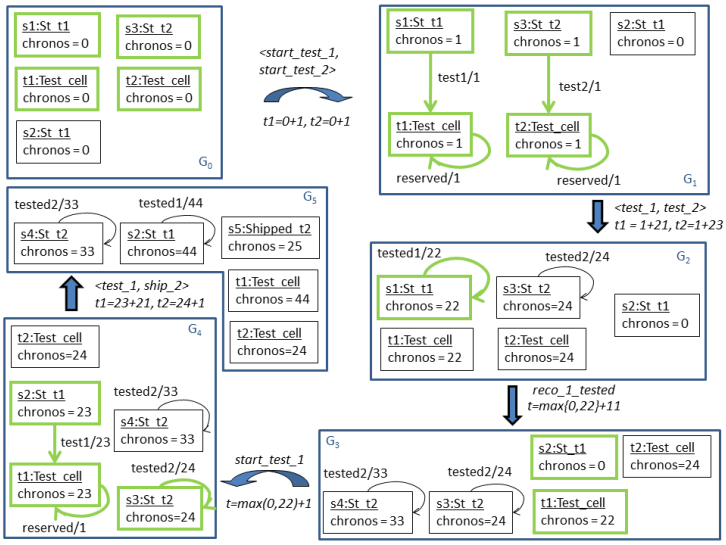

Figure 4: An optimal GT sequence

5 Conclusion and Future Work

In the current paper, we proposed a simultaneous optimization and verification for systems modeled as GTSs with time using P/T net based abstraction and optimization. In this way, we are able to find an optimal time-ordered transformation sequence leading to a desired system configuration in the original GTS with time.

Efficiency of the approach. Both the computation time of the approach and the number of backtracks to generate the next best ILP solution are subject to further investigation. In case of a large model, the computation time can be reduced by solving a reduced ILP problem that counts the tokens at the P/T net places not in each time instances, but only in e.g. each 10th time instances. Although this change needs less ILP computation time, the search for a corresponding GT sequence may results in more paths to be traversed that is subject to a further investigation. Corresponding benchmarking examples can be found at http://home.mit.bme.hu/~gyapay/.

Further extension. In the current paper, the rules of the current GTS with time were restricted to have a special structure. Since TER nets can model several other attributes in Petri nets, the optimization of general attributed graphs together with attribute conditions is targeted in the future.

Bibliography

[cpl] IBM ILOG CPLEX Optimizer. http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/.

[FL76] E. B. Fernández, T. Lang. Scheduling as a graph transformation. *IBM J. Res. Dev.* 20(6):551–559, Nov. 1976.

[GHV02] S. Gyapay, R. Heckel, D. Varró. Graph Transformation with Time: Causality and Logical Clocks. In *Proc. ICGT 2002: 1st International Conference on Graph Transformation*. LNCS, pp. 120–134. Springer-Verlag, 2002.

[GMMP91] C. Ghezzi, D. Mandrioli, S. Morasca, M. Pezzè. A Unified High-Level Petri Net Formalism for Time-Critical Systems. *IEEE Transactions on Software Engineering* 17(2):160–172, February 1991.

[GSV04] S. Gyapay, A. Schmidt, D. Varró. Joint Optimization and Reachability Analysis in Graph Transformation Systems with Time. *ENTCS* 109:137–147, 2004.

[HHRV11] Á. Hegedüs, Á. Horváth, I. Ráth, D. Varró. A model-driven framework for guided design space exploration. In *ASE*. Pp. 173–182. 2011.

[HSE10] C. Heinzemann, J. Suck, T. Eckardt. Reachability Analysis on Timed Graph Transformation Systems. In *Proceedings of the Fourth International Workshop on Graph-Based Tools (GraBaTs 2010)*. 2010.

[LH11] L. Li, C. N. Hadjicostis. Least-Cost Transition Firing Sequence Estimation in Labeled Petri Nets With Unobservable Transitions. *IEEE T. Automation Science and Engineering* 8(2):394–403, 2011.

[Mur89] T. Murata. Petri Nets: Properties, Analysis and Applications. In *Proc. IEEE*. Volume 77(4), pp. 541–580. 1989.

[Ram74] C. Ramchandani. Analysis of Asynchronous Concurrent Systems by Timed Petri Nets. Technical report, 1974.

[Ren03] A. Rensink. The GROOVE Simulator: A Tool for State Space Generation. In *AGTIVE*. Pp. 479–485. 2003.

[TÖ9] H. Tönnies. An evolutionary graph transformation system as a modelling framework for evolutionary algorithms. In *Proc. of the 32nd Annual German Conference on Advances in Artificial Intelligence*. Pp. 201–208. Springer-Verlag, 2009.

[TL04] A. Tarek, N. Lopez-Benitez. Optimal Legal Firing Sequence of Petri Nets Using Linear Programming. *Optimization and Engineering* 5(1):25–43, 2004.

[via] VIATRA2 Model Transformation Framework, An Eclipse GMT Subproject. http://www.eclipse.org/gmt/VIATRA2/.

[VV06] S. Varró-Gyapay, D. Varró. Optimization in Graph Transformation Systems Using Petri Net Based Techniques. *ECEASST* 2, 2006. Selected papers of PNGT 2006.

[Win94] W. L. Winston. *Operations Research — Applications and Algorithms*. Duxbury Press, Belmont, California, USA, 3rd edition, 1994.