

Optimal Aggregation of Components for the Solution of Markov Regenerative Processes

Elvio Gilberto Amparore^(✉) and Susanna Donatelli

University of Torino, Corso Svizzera 187, Torino, Italy
{amparore,susi}@di.unito.it

Abstract. The solution of non-ergodic Markov Renewal Processes may be reduced to the solution of multiple smaller sub-processes (components), as proposed in [4]. This technique exhibits a good saving in time in many practical cases, since components solution may reduce to the transient solution of a Markov chain. Indeed the choice of the components might significantly influence the solution time, and this choice is demanded in [4] to a greedy algorithm. This paper presents a computation of an optimal set of components through a translation into an *integer linear programming* problem (ILP). A comparison of the optimal method with the greedy one is then presented.

1 Introduction

A *Markov Regenerative Process* (MRP) is a stochastic process defined by a sequence of time instants called *renewal times* in which the process loses its memory, i.e. the age of non-exponential (general) events is 0. The behaviour between these points is then described by a time-limited stochastic process. MRPs have been studied extensively in the past [13, 16], and many solid analysis techniques exist. MRPs are considered the richest class of stochastic processes for which it is still possible to compute an exact numerical solution, and have therefore attracted a significant interest in the performance and performability community.

This paper considers the subclass of MRP in which the time limited stochastic process is a CTMC, general events are restricted to deterministic ones, and at most one deterministic event is enabled in each state. This type of MRPs arise for example in the solution of Deterministic Stochastic Petri nets (DSPN), in the model-checking of a one-clock CSL^{TA} formula [12] and in Phased-Mission Systems (PMS) as in [8, 15].

The steady-state solution of an MRP involves the computation and the solution of its discrete time *embedded Markov chain*, of probability matrix \mathbf{P} . The construction of \mathbf{P} is expensive, both in time and memory, because this matrix is usually dense even if the MRP is not. The work in [13] introduces an alternative *matrix-free* technique (actually \mathbf{P} -free), based on the idea that \mathbf{P} can be substituted by a function of the basic (sparse) matrices of the MRP.

When the MRP is non-ergodic it is possible to distinguish transient and recurrent states, and specialized solution methods can be devised. The work in [2, 4] introduces an efficient steady-state solution for non-ergodic MRPs,

in matrix-free form, called *Component Method*. To the best of our knowledge, this is the best available technique for non-ergodic DSPN and for CSL^{TA}, as well as for non-ergodic MRPs in general.

The work in [2, 4] and its application to CSL^{TA} in [5] identify a need for aggregating components into bigger ones, and observe that the performance of the algorithm may depend on the number, size, and solution complexity of the components. The aggregation is defined through a set of rules, to decide which components can be aggregated together, and through a greedy-heuristic algorithm that performs aggregations as much as it can. In this paper we observe that the greedy algorithm of [4] may actually find a number of components that is not minimal. The greedy solution seems to work quite well on the reported example, but the lack of optimality makes it hard to determine if it is convenient.

This paper formalizes the optimality criteria used in [4] and defines an ILP for the computation of an optimal set of components: to do so, the component identification problem is first mapped into a graph problem.

The paper develops as follows: Sect. 2 defines the necessary background. Section 3 defines the component identification problem in terms of the MRP graph. Section 4 defines the ILP that computes the optimal set of components. Section 5 discusses the performance of the ILP method and how it compares to the greedy method and concludes the paper.

2 Background and Previous Work

We assume that the reader has familiarity with MRPs. We use the definitions of [13]. Let $\{\langle Y_n, T_n \rangle \mid n \in \mathbb{N}\}$ be the *Markov renewal sequence* (MRS), with *regeneration points* $Y_n \in \mathcal{S}$ on the state space \mathcal{S} encountered at *renewal time instants* T_n . An MRP can be represented as a discrete event system (like in [11]) where in each state a general event g is taken from a set G . As the time flows, the age of g being enabled is kept, until either g fires (Δ event), or a Markovian transition, concurrent with g , fires. Markovian events may actually disable g (preemptive event, or \bar{Q} event), clearing its age, or keep g running with its accumulated age (non-preemptive event, or Q event).

Definition 1 (MRP Representation). *A representation of an MRP is a tuple $\mathcal{R} = \langle \mathcal{S}, G, \delta_g, \Gamma, Q, \bar{Q}, \Delta \rangle$ where \mathcal{S} is a finite set of states, $G = \{g_1 \dots g_n\}$ is a set of general events, δ_g is the duration of event g , $\Gamma : \mathcal{S} \rightarrow G \cup \{E\}$ is a function that assigns to each state a general event enabled in that state, or the symbol E if no general event is enabled, $Q : \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}_{\geq 0}$ is the non-preemptive transition rates function (rates of non-preemptive Markovian events), $\bar{Q} : \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}_{\geq 0}$ is the preemptive transition rates function (rates of preemptive Markovian events), $\Delta : \mathcal{S} \times \mathcal{S} \rightarrow \mathbb{R}_{\geq 0}$ is the branching probability distribution (probability of states reached after the firing of the general event enabled in the source state). Let α be the initial distribution vector of \mathcal{R} .*

Given a subset of states $A \in \mathcal{S}$, let $\Gamma(A) = \{\Gamma(s) \mid s \in A\}$ be the set of events enabled in A . Let the *augmented set* \hat{A} be defined as set of states A plus

the states of $\mathcal{S} \setminus A$ that can be reached from A with one or more non-preemptive Markovian events (**Q** events). To formulate MRP matrices, we use the *matrix filter* notation of [13]. Let \mathbf{I}^g be the matrix derived from the identity matrix of size $|\mathcal{S}|$ where each row corresponding to a state s with $\Gamma(s) \neq \{g\}$ is set to zero. Let \mathbf{I}^E be the same for $\Gamma(s) \neq \{E\}$.

By assuming $\{Y_n, T_n\}$ to be time-homogeneous, it is possible to define the *embedded Markov chain* (EMC) of the MRP. The EMC is a matrix \mathbf{P} of size $|\mathcal{S}| \times |\mathcal{S}|$ defined on the MRS as $\mathbf{P}_{i,j} \stackrel{\text{def}}{=} \Pr\{Y_n = j \mid Y_{n-1} = i\}$. A full discussion on the EMC matrix can be found in [13, Chap. 12]. Matrix \mathbf{P} is usually dense and slow to compute. To avoid this drawback, a matrix-free approach [14] is commonly followed. We now recall briefly the matrix-free method for non-ergodic MRP in reducible normal form.

Definition 2 (RNF). *The reducible normal form of an EMC \mathbf{P} is obtained by rearranging the states s.t. \mathbf{P} is in upper-triangular form:*

$$\mathbf{P} = \left[\begin{array}{ccc} \mathbf{T}_1 & \boxed{\mathbf{F}_1} & \\ & \ddots & \ddots \\ & & \mathbf{T}_k & \boxed{\mathbf{F}_k} \\ & & & \mathbf{R}_{k+1} & \\ & & & & \ddots \\ & & & & & \mathbf{R}_m \end{array} \right] \left. \begin{array}{l} \left. \vphantom{\begin{matrix} \mathbf{T}_1 \\ \vdots \\ \mathbf{T}_k \end{matrix}} \right\} k \geq 0 \text{ transient subsets.} \\ \left. \vphantom{\begin{matrix} \mathbf{R}_{k+1} \\ \vdots \\ \mathbf{R}_m \end{matrix}} \right\} (m-k) \text{ recurrent subsets,} \\ \text{with } m > k. \end{array} \right\} \quad (1)$$

The RNF of \mathbf{P} induces a directed acyclic graph, where each node is a subset of states \mathcal{S}_i (called component i). Let \mathbf{I}_i be the filtering identity matrix, which is the identity matrix where rows of states not in \mathcal{S}_i are zeroed.

When \mathbf{P} is in RNF, the steady-state probability distribution can be computed using the *outgoing probability* vectors $\boldsymbol{\mu}_i$. The vector $\boldsymbol{\mu}_i$ gives for each state $s \in (\mathcal{S} \setminus \mathcal{S}_i)$ the probability of reaching s in one jump while leaving \mathcal{S}_i :

$$\boldsymbol{\mu}_i = \left(\mathbf{I}_i \cdot \boldsymbol{\alpha} + \sum_{j < i} (\mathbf{I}_i \cdot \boldsymbol{\mu}_j) \right) \cdot (\mathbf{I} - \mathbf{T}_i)^{-1} \cdot \mathbf{F}_i, \quad i \leq k \quad (2)$$

Since matrix inversion is usually expensive, a product of a generic vector \mathbf{u} with $(\mathbf{I} - \mathbf{T}_i)^{-1}$ can be reformulated as a linear equations system $\mathbf{x} \cdot (\mathbf{I} - \mathbf{T}_i) = \mathbf{u}$. This system can be computed iteratively using vector \times matrix products with \mathbf{uT}_i . The steady state probability of the i -th recurrent subset is given by:

$$\boldsymbol{\pi}_i = \left(\mathbf{I}_i \cdot \boldsymbol{\alpha} + \sum_{j=1}^k (\mathbf{I}_i \cdot \boldsymbol{\mu}_j) \right) \cdot \lim_{n \rightarrow \infty} (\mathbf{R}_i)^n, \quad k < i \leq m \quad (3)$$

The Component Method computes first Eq. (2) for all transient components, taken in an order that respects the condition $j < i$ of the formula, and then computes the probability for the recurrent subsets based on Eq. (3).

Since the construction of \mathbf{P} is not always feasible, a matrix-free method has been devised in [4] for the computations of \mathbf{uT}_i and \mathbf{uF}_i . This generalisation

provides: (1) a derivation of the m subsets \mathcal{S}_i which is based only on \mathbf{Q} , $\bar{\mathbf{Q}}$ and Δ ; (2) the matrix-free form of the sub-terms \mathbf{T}_i , \mathbf{F}_i and \mathbf{R}_i , to be used in Eqs. (2) and (3). Observing that solution costs may differ depending on the structure of the subterms, it is convenient to distinguish three different matrix-free formulations.

[**Class** C_E] Condition: $\Gamma(\mathcal{S}_i) = \{E\}$. No general event is enabled in the \mathcal{S}_i states. The matrix-free products are defined as $\mathbf{uT}_i = \mathbf{I}_i \cdot \mathbf{a}_i(\mathbf{u})$ and $\mathbf{uF}_i = (\mathbf{I} - \mathbf{I}_i) \cdot \mathbf{a}_i(\mathbf{u})$, with the term $\mathbf{a}_i(\mathbf{u})$ defined as follow, given $\mathbf{I}_i^E = \mathbf{I}_i \cdot \mathbf{I}^E$ and $\mathbf{Q}_i^E = \mathbf{I}_i^E \cdot \mathbf{Q}$:

$$\mathbf{a}_i(\mathbf{u}) = \mathbf{u} \cdot (\mathbf{I}_i^E - \text{diag}^{-1}(\mathbf{Q}_i^E) \mathbf{Q}_i^E)$$

Time cost of a product with \mathbf{T}_i or \mathbf{F}_i is of $O(|\mathbf{Q}_i^E|)$.

[**Class** C_M] Either $|\Gamma(\mathcal{S}_i)| > 1$ or $\Gamma(\mathcal{S}_i) = \{g\} \wedge (\bar{\mathbf{Q}}_i \cdot \mathbf{I}_i \neq \mathbf{0} \vee \Delta_i \cdot \mathbf{I}_i \neq \mathbf{0})$. Let $\mathbf{b}_i(\mathbf{u})$ be defined as:

$$\mathbf{b}_i(\mathbf{u}) = \left(\mathbf{u} \cdot \sum_{g \in G} \mathbf{I}_i^g \cdot e^{\mathbf{Q}_i \delta^g} \right) \cdot \Delta + \left(\mathbf{u} \cdot \sum_{g \in G} \mathbf{I}_i^g \cdot \int_0^{\delta^g} e^{\mathbf{Q}_i x} dx \right) \cdot \bar{\mathbf{Q}}$$

The term $\mathbf{b}_i(\mathbf{u})$ gives the probability distribution of the next regeneration state reached with the firing of the general event (Δ event) or with the preemption of the general event enabled ($\bar{\mathbf{Q}}$ event). Note that the computation of $\mathbf{b}_i(\mathbf{u})$ on a subset \mathcal{S}_i of states has to consider all the states in the augmented set $\hat{\mathcal{S}}_i$, since we have to consider all states, also outside of the component, in which the system can be found at the next regeneration state. The products with \mathbf{T}_i and \mathbf{F}_i are defined as:

$$\mathbf{uT}_i = \mathbf{I}_i \cdot (\mathbf{a}_i(\mathbf{u}) + \mathbf{b}_{\hat{i}}(\mathbf{u})), \quad \mathbf{uF}_i = (\mathbf{I} - \mathbf{I}_i) \cdot (\mathbf{a}_i(\mathbf{u}) + \mathbf{b}_{\hat{i}}(\mathbf{u}))$$

The term $(\mathbf{I} - \mathbf{T}_i)^{-1}$ in Eq. (2) requires a fixed-point iteration to be evaluated. The time cost of $\mathbf{b}_i(\mathbf{u})$ is that of the uniformization, which is roughly $O(|\mathbf{Q}_i| \times R^g)$, with R^g the right truncation point [14, Chap. 5] of a Poisson process of rate $\delta_g \cdot \max_{s \in \mathcal{S}_i} (-\mathbf{Q}(s, s))$.

[**Class** C_g] Condition: $\Gamma(\mathcal{S}_i) = \{g\} \wedge \bar{\mathbf{Q}}_i \cdot \mathbf{I}_i = \mathbf{0} \wedge \Delta_i \cdot \mathbf{I}_i = \mathbf{0}$. A single general event g is enabled in \mathcal{S}_i , and all the Δ and $\bar{\mathbf{Q}}$ transitions exits from \mathcal{S}_i in one step. The matrix-free products with \mathbf{T}_i and \mathbf{F}_i are then:

$$\mathbf{uT}_i = \mathbf{0}, \quad \mathbf{uF}_i = (\mathbf{I} - \mathbf{I}_i) \cdot \mathbf{b}_{\hat{i}}(\mathbf{u})$$

which means that the term $(\mathbf{I} - \mathbf{T}_i)^{-1}$ in (2) reduces to the identity.

3 Identification of an Optimal Set of Components

As observed in [2, 4], the performance of the Component Method may vary significantly depending on the number, size and class of the considered components. There are two main factors to consider. The first one is that the complexity of the computation of the outgoing probability vector μ_i in Eq. (2) depends on the

class of component \mathcal{S}_i , and a desirable goal is to use the most convenient method for each component. The second one is that the presence of many small components, possibly with overlapping augmented sets, increases the solution time, as observed in [4], where it was also experimentally observed that the number of SCCs of a non-ergodic MRP can be very high (tens of thousands is not uncommon) also in non artificial MRPs. Therefore multiple components should be joined into a single one, as far as this does not lead to solving components of a higher complexity class.

In [4] a greedy method was proposed that aggregates components to reduce their number, while keeping the component classes separated. The identification of the components starts from the observation that the finest partition of the states that produces an acyclic set of components are the strongly connected components (SCC), where the bottom SCCs (BSCC) represent the recurrent components of the MRP. The greedy algorithm aggregates components when *feasible and convenient*. Two components can be aggregated if acyclicity is preserved (*feasibility*), thus ensuring that the MRP has a reducible normal form, and if the resulting component has a solution complexity which is not greater than that of the two components (*convenience*). The objective is then to *find the feasible and convenient aggregation with the least number of components*. The greedy algorithm works as follows:

1. Let Z be the set of SCCs of \mathcal{S} , and let $F_Z \subseteq Z$ be the frontier of Z , i.e. the set of SCC with *in-degree* of 0 (no incoming edges).
2. Take an SCC s from F_Z and remove it from Z .
3. Aggregate s with as many SCCs from F_Z as possible, ensuring that the class of the aggregate remains the same of the class of s .
4. Repeat the aggregation until Z is empty.

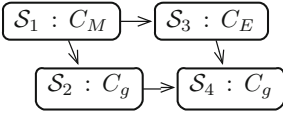


Fig. 1. Counter-example.

The main limitation of this method is that it depends on the visit order, since the aggregation of step 3 only visits the frontier. This limitation is necessary to ensure the acyclicity, but it may lead to sub-optimal aggregations. Indeed Fig. 1 shows the SCCs of an MRP, along with their classes, where the greedy algorithm may fail to provide the minimal aggregation. If the visit order is $\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_4$, at the time of visiting \mathcal{S}_2 the *in-degree* of \mathcal{S}_4 will still be 1, since \mathcal{S}_3 is yet to visit. Therefore the method will not merge \mathcal{S}_2 with \mathcal{S}_4 , resulting in a sub-optimal aggregation. Viceversa, the visit order $\mathcal{S}_1, \mathcal{S}_3, \mathcal{S}_2, \mathcal{S}_4$ allows the greedy algorithm to aggregate \mathcal{S}_2 and \mathcal{S}_4 together.

The goal of this paper is indeed to propose a method that identifies the optimal set of valid partitions (feasible and convenient).

Definition 3 (MRP Valid Partition). *A set of components of an MRP state space is a valid partition iff (1) the components are acyclic; and (2) each component, which belongs to one of the three classes (C_E , C_g and C_M), should not be decomposable into an acyclic group of sub-components of different classes.*

Acyclicity ensures that the partition is feasible and can be used for the Component Method. Condition (2) ensures convenience, i.e. by aggregating we do not increase the complexity of the solution method required for the component.

Definition 4 (MRP Component Optimization Problem). *The MRP component optimization problem consists in finding a valid partition of the MRP with the smallest number of components.*

It should be clear that this problem does not necessarily result in the fastest numerical solution of the MRP, since other factors, like rates of the components and numerical stability, may come into play: as usual the optimization is only as good as the optimality criteria defined, but results reported in [4] show that the component method is always equal or better, usually much better, than the best MRP solution method that considers the whole MRP. We transform the component optimization into a graph optimization problem for graphs with two types of edges: joinable (for pair of vertices that can stay in the same component) and non-joinable (for pair of vertices that have to be in different components).

3.1 Reformulation as a Graph Problem

We use standard notation for graphs. Let $G = \langle V, E \rangle$ be a directed graph, with V the set of vertices and $E \subseteq V \times V$ the set of edges. Notation $v \rightsquigarrow w$ indicates that vertex w is reachable from vertex v .

Definition 5 (DAG-LJ). *A labelled directed acyclic graph with joinable edges is a graph $\mathcal{G} = \langle V, \Sigma, \text{Lab}, E, E_N \rangle$, where:*

- $\langle V, E \rangle$ is an acyclic (direct) graph;
- Σ is a finite set of labels and $\text{Lab} : V \rightarrow \Sigma$ is a vertex labelling function;
- $E_N \subseteq E$ is the set of non-joinable edges; For ease of reference we also define $E_J = E \setminus E_N$ as the set of joinable edges;
- $\forall v, v' \in V, \langle v, v' \rangle \in E_J \Rightarrow \text{Lab}(v) = \text{Lab}(v')$;

Notations $v \xrightarrow{J} v'$ and $v \xrightarrow{N} v'$ are shorthands for a joinable and a non-joinable edge from v to v' , respectively. Given a label $l \in \Sigma$, the *section* D_l of \mathcal{G} is the set of vertices of equal label: $\{v \in V \mid \text{Lab}(v) = l\}$. Let $\mathcal{D} = \{D_l \mid l \in \Sigma\}$ be the *set of sections* of \mathcal{G} . Let $\text{sect}(v)$ be the section of vertex v .

We now define the concept of valid and optimal partition of a DAG-LJ, to later how how an optimal valid partition of \mathcal{G} induces a set of optimal components of the MRP for the component method.

Definition 6. *A valid partition of the vertices V of DAG-LJ \mathcal{G} is a partitioning $\mathcal{P} = \{P_1, \dots, P_m\}$ of the set of vertices V such that:*

1. $\forall P \in \mathcal{P}$ and $\forall v, v' \in P : \text{Lab}(v) = \text{Lab}(v')$;
2. $\forall P \in \mathcal{P} : E_N \cap (P \times P) = \emptyset$;
3. *Partition elements P are in acyclic relation.*

and we indicate with $\text{Parts}(\mathcal{G})$ the set of all valid partitions of \mathcal{G} .

Note that the presence of a non-joinable edge $v \xrightarrow{N} v'$ implies that v and v' cannot stay in the same partition element, in any valid partition. A joinable edge $v \xrightarrow{J} v'$ means that v and v' are allowed to be in the same partition element (and they are, unless other constraints are violated). From a valid partition we can build a graph which is a condensation graph, the standard construction in which all the vertices belonging to the same partition are replaced with a single vertex, from which we can easily check acyclicity.

An optimal partition (not necessarily unique) is then defined as:

Definition 7 (Optimal Partition of \mathcal{G}). *A valid partition $\mathcal{P}^* \in \text{Parts}(\mathcal{G})$ is optimal if the number of partition elements m is minimal over $\text{Parts}(\mathcal{G})$.*

3.2 Partitioning an MRP

MRPs have a natural representation as directed graphs: MRP states are mapped onto vertices and non-zero elements in \mathbf{Q} , $\bar{\mathbf{Q}}$, and Δ are mapped onto edges. Figure 2, upper part shows the graph of an MRP \mathcal{R} of 10 states, s_1 to s_{10} , and one general event g_1 . For each state we list the $\Gamma(s_i)$, which is either g_1 or E , if no general event is enabled. Transition rates are omitted. The mapping to DAG-LJ cannot be done at the MRP state level, since this results in general in a non-acyclic directed graph. Since our objective is to find an *acyclic* set of components we can map SCC of the MRP (instead of MRP states) to vertices and connection among SCCs to edges, since SCCs are the finest partition that satisfies acyclicity. When mapping to DAG-LJ, labels are used to account for the class of the SCCs, and non-joinable edges are used to identify connections that violates the convenience of component aggregation.

Definition 8. *Given an MRP $\mathcal{R} = \langle S, G, \delta_g, \Gamma, \mathbf{Q}, \bar{\mathbf{Q}}, \Delta \rangle$, its corresponding DAG-LJ $\mathcal{G}(\mathcal{R}) = \langle V, \Sigma, \text{Lab}, E, E_N \rangle$ is defined as:*

- $V = \text{SCC}(S)$. Each vertex is a strongly connected component of MRP states. Let $\text{states}(v)$ be the set of states in the strongly connected component $v \in V$.
- The set Σ of labels is $\{C_E, C_M\} \cup \{C_g \mid g \in G\}$ and $\text{Lab}(v)$ is defined as:
 - $\text{Lab}(v) = C_E$ iff $\Gamma(\text{states}(v)) = E$;
 - $\text{Lab}(v) = C_g$ with $g \in G$ iff $\Gamma(\text{states}(v)) = \{g\}$ and $\forall s, s' \in \text{states}(v) : \bar{\mathbf{Q}}(s, s') = 0 \wedge \Delta(s, s') = 0$; (g is enabled continuously, no firing that disables and immediately re-enables g is allowed)
 - otherwise $\text{Lab}(v) = C_M$.
- $E = \{\langle v, v' \rangle : \exists s \in \text{states}(v) \text{ and } s' \in \text{states}(v') \text{ such that } \mathbf{Q}(s, s') \neq 0 \text{ or } \bar{\mathbf{Q}}(s, s') \neq 0 \text{ or } \Delta(s, s') \neq 0\}$.
- Edge $\langle v, v' \rangle$ is a joinable edge iff $\text{Lab}(v) = \text{Lab}(v')$ and: (1) either $\text{Lab}(v) = M$ or (2) all MRP transitions from the states of v to the states of v' are \mathbf{Q} transitions. All other edges are non-joinable. Note that if there is a joinable and a non-joinable edge between v and v' , the former is ignored, since E_J is defined as $E \setminus E_N$.

$\mathcal{G}(\mathcal{R})$ has $|G| + 2$ distinct labels that induce $|G| + 2$ distinct sections: (D_E) if the SCC is of class C_E ; (D_g) if the SCC is of class C_g , for the general event $g \in G$; (D_M) if the SCC is of class C_M .

Example 1. The MRP of Fig. 2 (upper part) has only two SCCs with more than one state: $\{s_2, s_3, s_4\}$ and $\{s_6, s_7\}$. The bottom-left part shows the DAG-LJ \mathcal{G} built from the SCCs of \mathcal{R} . The DAG has three sections: D_E for SCCs of class C_E (all the states of the SCC enables only exponential transitions), D_{g_1} for SCCs in which all states enable g_1 and D_M for the remaining ones. Vertices v_3 and v_5 are connected by a joinable edge, since only \mathbf{Q} transitions connect states of v_3 with states of v_5 , while the edge $\langle v_3, v_4 \rangle$ is non-joinable because $\Delta(s_5, s_8) \neq 0$. the condensation graph of a valid partition of the DAG-LJ is shown on the right of Fig. 2. The partition satisfies the requirements of Definition 6: all vertices in the same partition element have the same label, and it is not possible to go from one vertex to another vertex in the same partition elements through a non-joinable edge. Since the condensation graph is acyclic this is a valid partition. \square

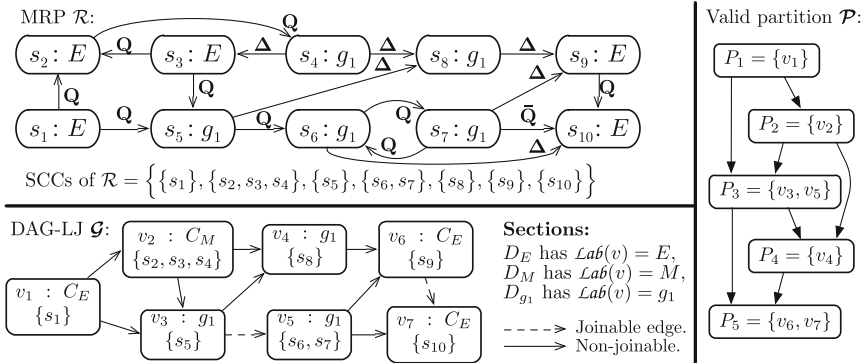


Fig. 2. Example of an MRP, its DAG-LJ, and a valid partition.

We now prove that an optimal partitioning of the DAG-LJ generated from an MRP is a solution of the MRP component optimization problem.

Property 1. If $\mathcal{G}(\mathcal{R})$ is the DAG-LJ of an MRP \mathcal{R} , and $\mathcal{P}^* = \{P_1, \dots, P_m\}$ is an optimal partition of \mathcal{G} , then $\mathcal{P}_{\mathcal{R}}^* = \{\mathcal{S}_1, \dots, \mathcal{S}_m\}$, with $\mathcal{S}_i = \bigcup_{v \in P_i} (\text{states}(v))$ is a solution of the MRP component optimization problem of \mathcal{R} according to Definition 4.

Proof. Recall that each partition element \mathcal{S}_i is a set of SCCs of \mathcal{R} and each SCC belongs to one of the three classes (C_E , C_g and C_M). We need to prove that $\mathcal{P}_{\mathcal{R}}^*$ is a solution of the component optimization problem of Definition 4, which requires to prove that $\mathcal{P}_{\mathcal{R}}^*$ is an MRP valid partition and that m is minimal.

A valid MRP partition is characterized by (1) acyclicity and (2) each component should not be decomposable into an acyclic group of sub-components of

different classes. Acyclicity of the set of \mathcal{S}_i trivially descends from the acyclicity of \mathcal{P}^* . For point (2) we can observe that all SCCs in the same partition element, by Definition 6 condition 1, have the same label and therefore have the same complexity class. Therefore point (2) can be proven by showing that it is never the case that the union of two SCCs of the same class results in a component of a different class if the two SCCs are in the same partition element. If the two SCCs are classified as C_E , then all states are exponential, and the union is still in C_E . If the two SCCs are classified as C_g , then we know that there is no \mathbf{Q} nor Δ transitions inside the single SCC, so that the classification of their union into an higher class (C_M) can only be originated by an arc between the states of the two SCCs, but the presence of such an arc, by definition of the non-joinable edges in $\mathcal{G}(\mathcal{R})$, produces a non-joinable arc between the DAG-LJ vertices of the two SCCs, and this violates point 2 of Definition 6 (there can be no non-joinable edges between vertices of the same partition element). If the two SCCs are classified as C_M , then all arcs between them, if any, are joinable, and the two SCCs can end up in the same partition element, which is also of class C_M .

Optimality of $\mathcal{P}_{\mathcal{R}}^* = \{\mathcal{S}_1, \dots, \mathcal{S}_m\}$ trivially descends from optimality of $\mathcal{P}^* = \{P_1, \dots, P_m\}$, as it is never the case that two SCCs that can be joined together result in a pair of vertices with a non-joinable edge between them, which is true by definition of $\mathcal{G}(\mathcal{R})$. \square

4 Formulation of the ILP

This section defines an optimization problem with integer variables whose solution allows to build \mathcal{P}^* . For each vertex $v \in V$ the ILP considers $|\mathcal{D}|$ integer variables : a variable x_v and one variable y_v^D for each section $D \in \mathcal{D} \setminus \text{sect}(v)$ (each section excluded that of v). We shall refer to these two types of variables simply as x and y variables. The optimal partition of \mathcal{G} is then built as: $\mathcal{P}^*(\mathcal{G}) = \bigcup_{D \in \mathcal{D}} \left(\bigcup_{i=1}^{N_D} (P_i^D) \right)$ where $P_i^D = \{v \mid v \in D \wedge x_v = i\}$, and N_D is the number of partition elements of section D (optimization target).

Definition 9. *The optimization problem is:*

Minimize $\sum_{D \in \mathcal{D}} N_D$ subject to:

Rule 1. $\forall v \in V: x_v \geq 1$ and $\forall D \neq \text{sect}(v): y_v^D \geq 0$

Rule 2. $\forall v \in V: x_v \leq N_D$

Rule 3. $\forall v, v' \in V$ with $\text{sect}(v) = \text{sect}(v')$ and $v \xrightarrow{-J} v'$: $x_v \leq x_{v'}$

Rule 4. $\forall v, v' \in V$ with $\text{sect}(v) = \text{sect}(v')$ and $v \xrightarrow{N} v'$: $x_v < x_{v'}$

Rule 5. $\forall v \in D, v' \notin D$ if $v \xrightarrow{N} v'$ then: $x_v \leq y_{v'}^D$

Rule 6. $\forall v \in D, v' \notin D$ if $v' \xrightarrow{N} v$ then: $y_{v'}^D < x_v$

Rule 7. if $v \xrightarrow{-J} v'$ or $v \xrightarrow{N} v'$ then $\forall D \notin \{\text{sect}(v), \text{sect}(v')\}$ add: $y_v^D \leq y_{v'}^D$

Rule 8. $\forall v, w \in D$ such that $\neg(v \rightsquigarrow w) \wedge \neg(w \rightsquigarrow v)$ add¹ the constraint: $x_v \leq x_w \Rightarrow \forall D' \neq D : y_v^{D'} \leq y_w^{D'}$.

Rule 1 sets a minimum value for the x and y variables. Rule 2 defines the N_D value as the maximum of all x variables of the same section. This value is part of the ILP goal function. Rules 3 and 4 constrains the x variables of the same section: if there is a non-joinable edge the order must be strict. Note that the relative order of the x variables follows the arc sense. No constraint is present if there is no direct edge between v and v' .

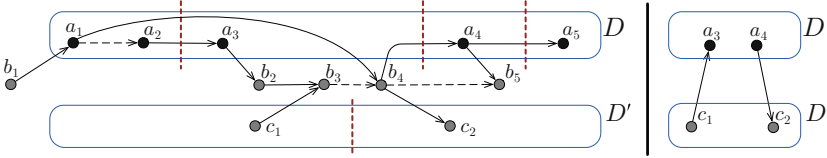


Fig. 3. The use of y variables to respect acyclicity

The remaining constraints take into account the requirement of acyclicity. Observe the portion of DAG-LJ reported in Fig. 3, left. a_i vertices are in section D , c_i are in section D' and b_i are in some other unspecified section(s). Since there is no arc between a_3 and a_4 the first 4 rules do not generate any constraint between the x variables of the two vertices, but if a_3 and a_4 end up in the same partition element acyclicity will be violated. The y variables are then defined as:

$$y_v^D = \max(0, x_w \mid w \in D \wedge w \rightsquigarrow v) \quad (4)$$

For each vertex v , variables y_v^D is the maximum over the x values of the vertices in D that can reach v . The value of y_v^D is used for the definition of the x variables of those vertices $w \in D$ that can be reached from v , if any. If there is an edge $v \rightarrow w$, then x_w has to be strictly greater than y_v^D . Back to Fig. 3, left, $y_{b_4}^D$ stores the maximum value among x_{a_1} and x_{a_3} , therefore $y_{b_4}^D = x_{a_3}$, while $y_{b_4}^{D'}$ has the same value of x_{c_1} . Indeed Rules 5 to 7 of the ILP ensure that the optimal solution of the ILP assigns to each y the correct value, as we shall prove in Theorem 1. In the example Rules 5 to 7 insert the following constraints: $x_{a_3} \leq y_{b_2}^D \leq y_{b_3}^D \leq y_{b_4}^D < x_{a_4}$, therefore $x_{a_3} \neq x_{a_4}$, so x_{a_3} and x_{a_4} end up in different elements of the partition and acyclicity is preserved.

The above rules are effective in generating a constraints between x_v and x_w of the same section only if the two vertices are connected through a path (possibly passing through different sections). Consider the DAG-LJ of Fig. 3, right: Rules 9 to 7 produce four constraints: $x_{a_4} \leq y_{c_2}^D$, $y_{c_1}^D < x_{a_3}$, $x_{c_1} \leq y_{a_3}^{D'}$, and $y_{a_4}^{D'} < x_{c_2}$,

¹ This logic implication is not in standard ILP form. It can be transformed [10] in ILP form as follows. Let U be a constant greater than $|V|$. Add a new variable $k_{v,w}$ subject to these constraints: $0 \leq k_{v,w} \leq 1$, $Uk_{v,w} - U < x_v - x_w \leq Uk_{v,w}$ and $\forall D' \in \mathcal{D} \setminus \{D\}$ add $y_v^{D'} \leq y_w^{D'} + Uk_{v,w}$.

that allows for a ILP solution with $x_{a_3} = x_{a_4} = 1$, $x_{c_1} = x_{c_2} = 1$, $y_{c_1}^D = y_{a_4}^{D'} = 0$ and $y_{c_2}^D = y_{a_3}^{D'} = 1$. The final partition will be $\mathcal{P}^* = \{a_3, a_4\} \cup \{c_1, c_2\}$, which clearly violates acyclicity. Rule 8 accounts for these situations for pairs of unconnected (in the \leadsto sense) vertices of the same section, stating that the values of the x and y variables in the ILP solution should respect the property that $x_v \neq x_w \Rightarrow y_v^{D'} \leq y_w^{D'}$ (the \leq relation among x variables should be reflected in the order of the corresponding y variables).

Back to Fig. 3, right, four constraints are inserted by Rule 8: $x_{a_3} \leq x_{a_4} \Rightarrow y_{a_3}^{D'} \leq y_{a_4}^{D'}$, $x_{a_4} \leq x_{a_3} \Rightarrow y_{a_4}^{D'} \leq y_{a_3}^{D'}$, $x_{c_2} \leq x_{c_1} \Rightarrow y_{c_2}^D \leq y_{c_1}^D$, and $x_{c_1} \leq x_{c_2} \Rightarrow y_{c_1}^D \leq y_{c_2}^D$. And the assignment of x and y above does not satisfy the constraint. In this case a feasible solution is either $x_{a_3} > x_{a_4}$, with $x_{c_2} = x_{c_1}$ or $x_{c_2} > x_{c_1}$, with $x_{a_3} > x_{a_4}$. The final partition has then three components and is acyclic.

Rule 8 modifies the constraint on the y variables, and their definition should now be based on a different notion of reachability. Let $v \xrightarrow{*} v'$ be the *one-step extended reachability relation*, which is true if either $\langle v, v' \rangle \in E$ or $\text{sect}(v) = \text{sect}(v') \wedge x_v \leq x_{v'}$. Let $v \rightsquigarrow v'$ be the *extended reachability relation*, defined as the reachability of v' from v using the $\xrightarrow{*}$ relation. The y variables are now:

$$y_v^D = \max(0, x_w \mid w \in D \wedge w \rightsquigarrow v) \quad (5)$$

Theorem 1. *The partition \mathcal{P}^* of \mathcal{G} built on the solution of the ILP of Definition 9 for graph \mathcal{G} , is an optimal partition of \mathcal{G} according to Definition 7.*

Proof. We need to show that the ILP solution provides a partition which is valid (as in Definition 6) and which has a minimum number of elements.

Validity is articulated in three conditions, the first two are trivial, as partition elements are built from vertices of the same section (and therefore of equal label) and Rule 4 states that $x_v < x_w$ whenever there is a non-joinable edge between v and w . Acyclicity is also rather straightforward. There is a cycle among two partition elements if it exists a pair of partition elements P_i^D and $P_j^{D'}$ and vertices $v, w \in P_i^D$ and $v', w' \in P_j^{D'}$ such that $v \rightsquigarrow v'$ and $w' \rightsquigarrow w$. Obviously $x_v = x_w$ and $x_{v'} = x_{w'}$. We show that if such paths exists, then at least one constrain of the ILP is violated. We consider separately the case in which $v' \rightsquigarrow w'$ and the one in which this is not true. If $v' \rightsquigarrow w'$, then (Rules 5, 6, and 7) $x_v \leq y_{v'}^D \leq \dots \leq y_{w'}^D < x_w$, which violates the hypothesis that $x_v = x_w$. If $\neg(v' \rightsquigarrow w')$ then Rule 8 ensure that, since $x_{v'} = x_{w'}$, we must have $y_{v'}^D \leq y_{w'}^D$, moreover, by Rule 6, we have $y_{w'}^D < x_w$, which leads to $x_v \leq y_{v'}^D \leq y_{w'}^D < x_w$ which violates the hypothesis that $x_v = x_w$.

Minimality is more complicated, and is based on three observations: (1) the ILP solution builds the correct value (as per Definition 5) of the y variables of interest, (2) N_D is the number of partition elements for section D , and (3) the ILP is not over-constrained (or if v and v' could stay in the same partition element, then there is no $<$ among their x).

For point 1, let's assume that there are n vertices $w_1, \dots, w_n \in D$ such that $w_i \rightarrow v$ and $v \notin D$ (the generalisation to \rightsquigarrow is trivial due to Rule 7 that propagates the \leq constraints among y variables in presence of a direct arc). Rule

5 sets a constraint $x_{w_i} \leq y_v^D$ for each vertex w_i and at least one strict constraint $y_v^D < x_{v'}$, if there is an edge from v to $v' \in D$. Then the minimization of $x_{v'}$ assigns to y_v^D the minimum possible value, which is the minimum value that satisfies the $x_{w_i} \leq y_v^D$ constraints, which is precisely the maximum over the x_{w_i} values. The proof indicates that the y_v^D value computed by the ILP is exactly equal to the maximum only in presence of a path from v back to D , in all other cases the ILP can assign any value $y \geq \max(\dots)$. But if there is no path from v back to D , then the value of y_v^D is inessential for the definition of the x variables of D . In case instead of $w \rightsquigarrow v$, the path between w and v is made either of pairs $\langle a_k, a_h \rangle$ such that either $a_k \rightarrow a_h$ (in this case the $y_{a_k}^D$ value, set initially to x_{w_i} propagates according to Rule 7) or, by definition of \rightsquigarrow , $\text{sect}(a_k) = \text{sect}(a_h)$ and $x_{a_k} \leq x_{a_h}$. This implies (by Rule 8) that also $y_{a_k}^D \leq y_{a_h}^D$ and again the value of x_{w_i} propagates as if there were an edge between a_k and a_h . As in the previous case, if there is a path between v and a vertex in D , then the y_v^D is set precisely to the maximum among the x_{w_i} .

For point 2, we need to prove that $\forall i \in \{1..N_D\}, \exists v \in D : x_v = i$. This is true since, in the rules of the ILP, the $<$ order between x variables only involves x variables of the same section D , either directly (through Rule 4) or indirectly through y^D variables (Rule 6) which, by definition, carry the value of one of the x variables of D , as proved in point 1.

For point 3, we need to prove that, if w and v are in the same partition element in the optimal partitioning \mathcal{P}^* of \mathcal{G} , then they are assigned the same x value by the ILP. For simplicity, let's assume that \mathcal{P}^* is unique in $\text{Parts}(\mathcal{G})$. We prove that if $x_v \neq x_w$ then the ILP solution violates a constraint. The only way by which the ILP, given the goal of minimizing N_D , can assign a different value to x_v and x_w is the presence of $<$ among the two variables, either directly, as in Rule 4, or indirectly, through Rule 6. In the case of Rule 4, the constraint is inserted only if there is a non-joinable edge between w and v , which clearly violate the hypothesis that w and v are in the same partition element in \mathcal{P}^* . In the latter case, if it is a constraint $y_{v'}^D < x_v$ (of Rule 6) that causes x_w to be different from x_v , it means that $y_{v'}^D \geq x_w$. Definition 5 implies that there is a path between w and v that passes through vertex v' . In that case, w and v could not stay in the same partition element, otherwise acyclicity would be violated. Clearly the path between w and v could be either through \rightsquigarrow or through \rightsquigarrow^* since we have already shown that both can create a loop among partition elements. \square

We now show two small examples of DAG-LJ, whose optimal partitioning have been constructed with the ILP method. The ILPs have been solved using the `lp_solve` tool.

Example 2. Consider the DAG-LJ \mathcal{G} shown in Fig. 4, left, that has 14 vertices and 3 sections $D_{1..3}$. Each box reports in the first row the vertex and the section. The second line of each box reports the x_v number as computed by the ILP.

The minimal solution of the ILP is found with $N_{D_1} = 3, N_{D_2} = 2$ and $N_{D_3} = 2$, which leads to a partition of the vertices in 7 subsets (partition elements). Observe that v_8 is not a direct successor of v_4 , but they cannot form

a single component because it would form a loop. Since there is $v_4 \xrightarrow{N} v_6$ and $v_7 \xrightarrow{N} v_8$ with $x_{v_6} \leq x_{v_7}$, Rule 8 adds a constraint $y_4^{D_3} \leq y_8^{D_3}$, to ensure the acyclicity. Figure 4, right, shows the optimal valid partitioning \mathcal{P}^* . \square

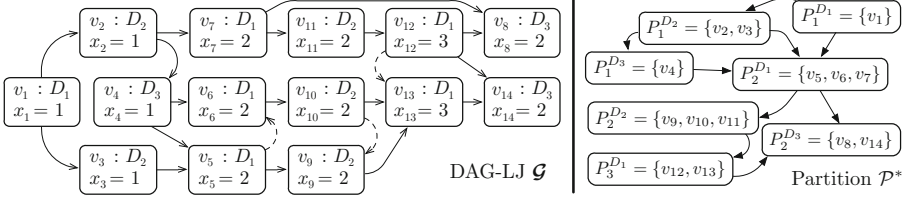


Fig. 4. Example of a DAG-LJ with the x_v values and \mathcal{P}^* .

Example 3. Figure 5 reports a rather different DAG-LJ, as there is no connection among the vertices of the same section, but if all the vertices of equal section are put in the same partition element, then acyclicity is violated.

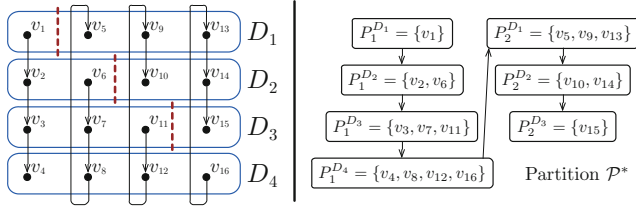


Fig. 5. An example of a DAG-LJ and \mathcal{P}^* with a complex structure.

This is a prototypical example for the need of Rule 8 in the ILP. Without that rule, all the vertices of the same sections would form a single partition element, resulting in a cyclic partitioning. The problem of determining where partition elements are separated, however, is not trivial, since there are many possible combinations. In this case, the optimization problem is crucial in finding the partition boundaries that minimize the total number of components. \square

5 Assessment and Conclusions

Since the ILP solution finds the optimal partition, the assessment of the proposed method does not address the quality of the solution, but aims at comparing the ILP solution with the greedy one of [4] (obviously on relatively small examples since ILP solution is known to be NP-hard), to identify the cases in which the greedy approach fails. Table 1 shows such a comparison.

The models used in the comparison are non-ergodic MRP created from Deterministic Stochastic Petri Nets with GreatSPN [6], and could be solved in GreatSPN using any of the implemented techniques for non-ergodic MRP (classical,

Table 1. Result of the ILP method against the greedy method.

| Model | $ \mathcal{D} $ | SCC | EJ | EN | Greedy | ILP vars | ILP | Constr.P. |
|-----------------------|-----------------|-----|----|----|--------|----------|-----|-----------|
| PhMissionA, K=1, NP=2 | 3 | 47 | 36 | 36 | 6 | 2767 | 6 | 20 |
| PhMissionB, K=3, M=2 | 3 | 52 | 47 | 36 | 6 | 2492 | 5 | 7 |
| PhMissionC, K=6, M=3 | 3 | 45 | 25 | 27 | 7 | 504 | 7 | 7 |
| Cross | 6 | 10 | 0 | 20 | 7 | 108 | 7 | 10 |
| MRP of Fig. 5 | 6 | 18 | 0 | 44 | 12 | 372 | 9 | 13 |

matrix-free, or component-based). The partition computed by the ILP (or by the greedy method) is the base for the component method, that usually is the best solver of the three available in GreatSPN. Models can be found at www.di.unito.it/~amparore/QEST16.zip: for each model the zip file includes the pdf of the net (drawn with the GreatSPN editor [1]), and a representation of their DAG-LJ and of their ILP-computed partitioning. The whole process is automatized: from the DSPN description the MRP state space, their SCCs, and the corresponding DAG-LJ is constructed, the ILP is produced and solved with `lp_solve`, the components are then computed and provided as input to the component method. A similar chain is available for the greedy method. We consider 5 models: the last two have been artificially created to investigate cases in which acyclicity is non-trivial (cases in which Rule 8 plays a significant role in constraining the solution), while the first three are variations of Phased Mission Systems (PMS). In particular they are cases of a *Scheduled Maintenance System* (SMS), inspired by [9], in which a system alternates between two phases: Work and Maintenance, and behaves differently depending on the phase (as typical in PMS). The model is studied for its transient behaviour, the stopping condition for model A is determined by the number NP of phases, while models B and C cycle over the two phases, and the stopping condition is triggered when the system reaches a failure state. K and M are the number of pieces and machines.

The table reports the model name and the number of sections, SCCs, joinable and non-joinable edges. The column ‘Greedy’ indicates the number of components found by the greedy method, while the two subsequent columns reports the number of variables of the ILP and the number of components found by solving the ILP. Finally, the last column reports the number of components found by applying a constraint propagation method, i.e. by applying the ILP constraint in order to maximize the x and y variables until a fix-point is found. Constraint propagation can be seen as an approximate solution of the ILP, where the found partitioning is always valid but not necessarily optimal.

As the table shows, the greedy method performs reasonably, but it does not always found the optimal solution, although it goes very close to it (a behaviour that has been observed on other cases of “real” systems). It instead performs badly in cases created ad-hoc to experiment with Rule 8 (models Cross and MRP of Fig. 5). The constraint propagation method is consistently the worst one. The MRP size we could solve with standard computer are below a hundred

of SCCs (of course the state spaces could be much larger), which is not surprising considering that the ILP size grows rapidly with the number of SCCs (the vertices of the DAG-LJ) and that the problem is NP-hard.

Conclusions. This paper introduces a technique to find the optimal partition of a non-ergodic MRP which is the basis of the Component Method solver for MRPs. The method is both general (can be applied to any non-ergodic MRP) and optimal, as it finds the minimum number of partition elements, and therefore of components. Optimality is important not only for the solution time, but also because it provides a baseline against which to assess the greedy solution. An optimal solution is a prerequisite to compare the component method against specific ad-hoc MRP solver. A typical example are the MRPs generated from Phased Petri nets [8] for which an efficient ad-hoc solution technique was devised in [15]: this technique can be interpreted as a special case of the component method (with roughly one component per phase), moreover with the component method the class of PMS that can be efficiently solved can be enlarged to include, for example, different type of dependencies of the system behaviour from the phase description that we believe are relevant for reliability analysis and that will be investigated in our future research work. Optimality is also a prerequisite when comparing the efficiency of a CSL^{TA} model checker, as the one in [3], on verifying CSL Until formulas. The component method with optimal partitioning reduces the time complexity of the CSL^{TA} model checker to that of a CSL one (CSL model-checking algorithm as described in [7]), as already envisioned in [4].

A question that might arise is whether it is worth to define DAG-LJ, instead of deriving the ILP directly from the SCCs of the MRP. The answer is that the DAG-LJ abstraction may be used for other purposes and we are indeed currently using it in the context of model-checking of CSL^{TA} based on zone graph. The idea here is that the MRP that describes the set of accepting paths of the formula is obtained by a cross product of the Markov chain model and the zone graph (a rather trivial construction since there is a single clock) of the timed automata that describes the CSL^{TA} formula. The MRP is then solved with the component method. But this is an a-posteriori work: the MRP is first built completely, and then solved by component. The solution we are working on translates the zone graph in a DAG-LJ, computes the components of the DAG-LJ and then does the cross-product between a zone graph component and the Markov chain.

Another more than legitimate question is whether it makes sense to rely on ILP solution, in particular as it is not uncommon to have MRPs with thousands of SCCs. But luckily this is not always the case, for example the DAG-LJ of the zoned graph of the timed automata that describes a CSL^{TA} formula typically has a very limited number of SCCs, and the ILP solution can be easily found, while a similar situation arises in PMS, as typically the number of SCCs is related to the number of phases, which is usually significantly less than 10. As future work, we plan nevertheless to experiment with classical approximate ILP solvers, and to compare it with the greedy approach.

References

1. Amparore, E.G.: A new GreatSPN GUI for GSPN editing and CSL^{TA} model checking. In: Norman, G., Sanders, W. (eds.) QEST 2014. LNCS, vol. 8657, pp. 170–173. Springer, Heidelberg (2014)
2. Amparore, E.G., Donatelli, S.: A component-based solution method for non-ergodic Markov regenerative processes. In: Aldini, A., Bernardo, M., Bononi, L., Cortellessa, V. (eds.) EPEW 2010. LNCS, vol. 6342, pp. 236–251. Springer, Heidelberg (2010)
3. Amparore, E.G., Donatelli, S.: MC4CSL^{TA}: an efficient model checking tool for CSL^{TA}. In: International Conference on Quantitative Evaluation of Systems, pp. 153–154. IEEE Computer Society, Los Alamitos (2010)
4. Amparore, E.G., Donatelli, S.: A component-based solution for reducible Markov regenerative processes. *Perform. Eval.* **70**(6), 400–422 (2013)
5. Amparore, E.G., Donatelli, S.: Improving and assessing the efficiency of the MC4CSL^{TA} model checker. In: Balsamo, M.S., Knottenbelt, W.J., Marin, A. (eds.) EPEW 2013. LNCS, vol. 8168, pp. 206–220. Springer, Heidelberg (2013)
6. Baarir, S., Beccuti, M., Cerotti, D., Pierro, M.D., Donatelli, S., Franceschinis, G.: The GreatSPN tool: recent enhancements. *SIGMETRICS Perform. Eval. Rev.* **36**(4), 4–9 (2009)
7. Baier, C., Haverkort, B., Hermanns, H., Katoen, J.P.: Model-checking algorithms for continuous-time Markov chains. *IEEE Trans. Softw. Eng.* **29**(6), 524–541 (2003)
8. Bondavalli, A., Mura, I.: High-level Petri net modelling of phased mission systems. In: 10th European Workshop on Dependable Computing, pp. 91–95, Vienna (1999)
9. Bondavalli, A., Filippini, R.: Modeling and analysis of a scheduled maintenance system: a DSPN approach. *Comput. J.* **47**(6), 634–650 (2004)
10. Brown, G.G., Dell, R.F.: Formulating integer linear programs: a rogues’ gallery. *INFORMS Trans. Educ.* **7**(2), 153–159 (2007)
11. Cassandras, C.G., Lafortune, S.: Introduction to Discrete Event Systems. Springer, Secaucus (2006)
12. Donatelli, S., Haddad, S., Sproston, J.: Model checking timed and stochastic properties with CSL^{TA}. *IEEE Trans. Softw. Eng.* **35**(2), 224–240 (2009)
13. German, R.: Performance Analysis of Communication Systems with Non-Markovian Stochastic Petri Nets. Wiley, New York (2000)
14. German, R.: Iterative analysis of Markov regenerative models. *Perform. Eval.* **44**, 51–72 (2001)
15. Mura, I., Bondavalli, A.: Markov regenerative stochastic petri nets to model and evaluate phased mission systems dependability. *IEEE Trans. Comput.* **50**(12), 1337–1351 (2001)
16. Stewart, W.J.: Probability, Markov Chains, Queues, and Simulation: The Mathematical Basis of Performance Modeling. Princeton University Press, Princeton (2009)

Quantitative Evaluation of Systems

13th International Conference, QEST 2016, Quebec
City, QC, Canada, August 23-25, 2016, Proceedings

Agha, G.; Van Houdt, B. (Eds.)

2016, XVII, 382 p. 91 illus., Softcover

ISBN: 978-3-319-43424-7