A general algorithm to compute the steady-state solution of product-form cooperating Markov chains

Andrea Marin *, Samuel Rota Bulò *
*Dipartimento di Informatica
Università Ca' Foscari di Venezia
Venice, Italy 30172
Email: {marin,srotabul}@dsi.unive.it

***Abstract*—In the last few years several new results about product-form solutions of stochastic models have been formulated. In particular, the Reversed Compound Agent Theorem (RCAT) and its extensions play a pivotal role in the characterization of cooperating stochastic models in product-form. Although these results have been used to prove several well-known theorems (e.g., Jackson queueing network and G-network solutions) as well as novel ones, to the best of our knowledge, an automatic tool to derive the product-form solution (if present) of a generic cooperation among a set of stochastic processes, is not yet developed. In this paper we address the problem of solving the non-linear system of equations that arises from the application of RCAT. We present an iterative algorithm that is the base of a software tool currently under development. We illustrate the algorithm, discuss the convergence and the complexity, compare it with previous algorithms defined for the analysis of the Jackson networks and the G-networks. Several tests have been conducted involving the solutions of a (arbitrary) large number of cooperating processes in product-form by RCAT.**

I. Introduction

Stochastic models play a central role in the field of performance evaluation of computer software and hardware architectures. In this paper we focus on those models whose underlying stochastic processes are continuous-time Markov chains (CTMCs). In these cases, several performance indices can be derived by the steady-state analysis, i.e., by computing the state probabilities in the long run (when time $t \to \infty$). If the CTMC is ergodic (i.e., irreducible and all the states are positive recurrent) the limiting distribution of the state probabilities is unique and independent of the initial state distribution. The stationary distribution of an ergodic CTMC is generally computed as the normalized solution of a system of linear equations called system of global balance equations (GBEs). Such a system has a coefficient matrix whose rank is the number of states of the chain minus one.

From a modeler's point of view, it is often convenient to describe a system using a modular approach. In other words, a system S can be seen as the composition of a set of subsystems $S_1, \ldots, S_N$, that are modeled separately and cooperate according to a formal definition. Several formalisms allow one to use this approach very naturally (e.g., the performance evaluation process algebra (PEPA) presented in [1] or queueing networks), i.e., they introduce an intuitive way to combine the sub-models and formally specify the CTMC underlying the combined model. The main drawback of this modeling technique is known as the state-space explosion. Roughly speaking, the state-space explosion occurs when the number of states of the composed model is so high it makes the automatic derivation of the steady-state solution unfeasible, even if the number of states of each sub-model, in isolation, is reasonably small.

In order to overcome this problem, product-form solutions are investigated. The first results on product-form models appeared in queueing theory [2], [3] and afterwards they have been extended to other formalisms such as stochastic Petri nets [4], [5]. Informally, we say that a stochastic model is in product-form if it is possible to express its stationary distribution as product of functions whose argument depends only on the state of a single sub-model. In practice, in case of a product-form model, one has to find a correct parameterization of each sub-model and then derive its stationary distribution as if it was in isolation. Then, the stationary distribution of the former model is given by the normalized product of these distributions. In general, product-form models lead to a computationally efficient solution, because they do not require the GBEs for the CTMC of the composed model to be solved. Obviously, now the analysis is shifted on deciding if a model admits a product-form solution and how the sub-models should be parameterized. Jackson and BCMP results state sufficient conditions for a queueing network to be in product-form, and they require the solution of a linear system of equations, called traffic equations, whose order is linear in the number of queues in the network. However, the introduction of G-networks [6] has shown important examples of product-form models whose traffic equations form a non-linear system. In order to efficiently compute the solution of product-form G-networks some iterative algorithms have been defined [7], [8].

Some novel results for deciding and computing the product-form solution of interacting CTMCs are the reversed compound agent theorem (RCAT) and other theorems derived from it [9], [10], [11]. Although these results are based on a definition of the processes (and their composition) inherited from PEPA, their application is not limited to process algebra. Indeed, Jackson, BCMP and G-network product-form solutions have been proved using these theorems [9], [12], [13].

In general, RCAT based theorems require the solution of a non-linear set of traffic equations which may often become a

difficult task. In practice, as far as we know, there are no tools which, given the description of the interacting models, decide if the product-form exists and, if this is the case, numerically compute the stationary distribution.

In this paper we use the multi-agent RCAT (MARCAT) theorem redefined assuming a set of structural conditions, and we address the problem of providing a simple and efficient iterative algorithm that solves the non-linear traffic equations that arise from its application and computes the stationary state distribution. In [11] the authors define an iteration scheme whose fixed point, which exists by Brouwer's theorem, is the solution of the traffic equations of MARCAT. However, from a practical point of view, the convergence of the iterations from any starting point is not guaranteed. A slightly different approach, named Meercat, has been proposed in [14]. The author introduces a language (PEPAinf) to define PEPA models with an infinite number of states and a regular structure (e.g., queueing networks). Meercat derives the traffic equations which are exported in an ASCII format and then solved by an extern symbolic tool such as Mathematica. It is worth noting that the symbolic solution of the non-linear system of equation is not guaranteed and in general is an inefficient process. Moreover, the algorithmic construction of the traffic equations is itself a non-trivial task which would require a deeper investigation.

As opposed to the two aforementioned approaches, the algorithm we propose in this paper does not make directly use of the traffic equation. Indeed, it is based on the simple idea of solving iteratively the CTMCs underlying the set of interacting models with an initial random parameterization. At each iteration we adjust the parameterization of the models according to a set of simple rules. When the iterations converge we can decide if we have a product-form solution and its unnormalized distribution. Although we have no proof of the convergence of our algorithm for the general case, we encountered no cases where this did not happen. Moreover, we think that our solution has some strengths with respect to the iteration scheme on the traffic equations. First of all, it is really easy to implement because we only need to compute the solutions of linear systems. Additionally, experimental evidence shows that the convergence is very fast (we always converged within 20 iterations) even for very large models (e.g., joint processes with 10^{20} states). Note that the algorithm is very efficient in case of product-form compositions of many models with a relatively small number of states, that is a quite reasonable practical requirement.

The paper is structured as follows. In the first part, we illustrate the theoretical framework underlying the algorithm we propose. Then, in Section III, we discuss a simplified version of the algorithm with the aim of explaining its fundamental peculiarities. Section IV illustrates its application to two well-known cases, i.e., Jackson networks and G-networks. In the former case we prove that our algorithm is exactly the Jacobi iteration scheme on the system of traffic equations and always converges, while in the latter one we show that it is exactly the algorithm defined in [15]. Moreover, in this section we show the application of the algorithm to a simple problem in order to clarify its dynamic. Section V addresses the problem of optimizing the algorithm and gives its complete definition. Finally, Section VI discusses some final remarks.

II. Theoretical framework

The algorithm we are introducing in the following sections is based on a set of results presented to study stochastic models in product-form. In particular, we focus on the Multi-agent Reversed Compound Agent Theorem (MARCAT) presented in [11] with some limitations. The result is based on the PEPA algebra [1]. Let P_i and P_j represent two PEPA agents. Let us assume that the derivation graphs of P_i and P_j have ergodic underlying CTMCs. The cooperation between P_i and P_j is $P_i \underset{\mathcal{L}}{\bowtie} P_j$, where $\mathcal{L}$ is the set of action types on which the agents synchronize. Let $a \in \mathcal{L}$, then we limit our analysis to cooperations that satisfy these conditions: 1) $a \in \mathcal{L}$ must be either passive at every instance of P_i (P_j) or active at every instance of P_i (P_j), and 2) if $a \in \mathcal{L}$ is active in P_i then it must be passive in P_j and vice versa. Given the set $\mathcal{L}$, let $\mathcal{P}_i(\mathcal{L})$ be the set of action types that are passive with respect to P_i and $\mathcal{A}_i(\mathcal{L})$ be the set of action types that are active with respect to P_i. For the sake of simple notation we write $\mathcal{P}_i$ for $\mathcal{P}_i(\mathcal{L})$, and similarly for the set of active action types.

We can generalize the notion of pairwise cooperation to multiple agents as illustrated in [11]. We write:

$$\underset{\mathcal{L}}{\overset{N}{\bowtie}}_{k=1} P_k\,, \qquad k \geq 2\,,$$

where $\mathcal{L}$ is the set of synchronizing action types and $a \in \mathcal{L}$ is active exactly in one agent P_i and passive in one agent P_j, with $1 \leq i, j \leq N$ and $i \neq j$. Let $\mathcal{L}_k = \mathcal{P}_k \cup \mathcal{A}_k$, then we write:

$$\underset{\mathcal{L}}{\overset{N}{\bowtie}}_{k=1} P_k \overset{\text{def}}{=} (\ldots((P_1 \underset{\mathcal{M}_2}{\bowtie} P_2) \underset{\mathcal{M}_3}{\bowtie} P_3) \underset{\mathcal{M}_4}{\bowtie} \cdots) \underset{\mathcal{M}_N}{\bowtie} P_N,$$

where $\mathcal{M}_k = \mathcal{L}_k \cap \left(\cup_{j=1}^{k-1} \mathcal{L}_j\right)$. From now ahead we assume that the underlying CTMC of $\underset{\mathcal{L}}{\bowtie}_{k=1}^{N} P_k$ has an irreducible subgraph.

In order to keep this paper self-contained we introduce MARCAT. With respect to the theorem proved in [11] the following formulation is less general because it introduces some structural conditions similar to those required by original RCAT [9]. We discuss the impact of MARCAT in its very general formulation on our algorithm in the conclusions. Note that the notation $P_k\{(a,p) \leftarrow (a,q)\}$ stands for a syntactical renaming of the pair (a,p) with the pair (a,q) in the specification of P_k, as formally defined in [9].

Theorem 1 (MARCAT): Suppose that the cooperation of agents $\underset{\mathcal{L}}{\bowtie}_{k=1}^{N} P_k$, denoting stationary Markov processes, has a derivation graph with an irreducible subgraph G. Let $a \in \mathcal{A}_k \cap \mathcal{P}_j$ with $1 \leq j, k \leq N$, $j \neq k$ and assume that the following conditions are satisfied:

1) a is enabled in every derivative of P_j,

2) every derivative of P_k is reachable through an action with type a,
3) in the reversed process $\overline{P_k}$ of P_k, every occurrence of the active action type a has the same rate $\overline{p_a}$.

Then the reversed agent $\overline{\bowtie_{\mathcal{L}}^{N}_{k=1} P_k}$ with derivation graph containing the reversed subgraph $\overline{G}$ is

$$\bowtie_{\mathcal{L}}^{N}_{k=1} \overline{R_k}\{(\overline{a}, \overline{p_a}) \leftarrow (\overline{a}, \top) | a \in \mathcal{A}_k\},$$

where $R_k = P_k\{\top_a \leftarrow x_a | a \in \mathcal{P}_k\}$, with $k = 1, \ldots, N$ and $\{x_a\}$ are the unique solutions of the rate equations:

$$\{\top_a = \overline{p_a} | a \in \mathcal{A}_k,\ 1 \leq k \leq N\} \tag{1}$$

where $\overline{p_a}$ is the symbolic rate of action type $\overline{a}$ in $\overline{P_k}$.

Note that providing an intuitive explanation of the MARCAT conditions in terms of requirements of the modeled systems is out of the scope of this paper. We suggest to refer to the original work [9] for a deeper analysis and interpretation of these conditions.

III. Solution algorithm

According to Theorem 1 it is straightforward to derive the product-form solution of $\bowtie_{\mathcal{L}}^{N}_{k=1} P_k$ once the reversed rates $\overline{p_a}$ are known. However, the computation of the reversed rates as the solution of the system of equations (1) can be a non-trivial task. In fact, although these equations form a linear system for some models (e.g., for Jackson networks [9]), this is not true in general (e.g., for G-networks [13]). Nevertheless, the existence and uniqueness of the solution of (1) has been shown in [11], but as far as we know, there are still no algorithms to derive it in the most general case, which have been proved to converge.

In this section we introduce an algorithm that allows us to compute the steady-state solution of product-form cooperating processes for which the MARCAT theorem applies, but before discussing the main idea, we introduce some notations and definitions that will be used in the sequel.

A. Notations and definitions

In this framework we represent an agent P_k as a 4-tuple:

$$P_k = (\mathcal{S}_k, \mathcal{A}_k, \mathcal{P}_k, \lambda_k),$$

where $\mathcal{S}_k$ is the set of derivatives (states) of agent P_k, $\mathcal{A}_k$ and $\mathcal{P}_k$ are the sets of active and passive action types, respectively, involved in the cooperation ($\mathcal{L}_k = \mathcal{A}_k \cup \mathcal{P}_k$), and $\lambda_k : (\mathcal{L}_k \cup \{\varepsilon\}) \times \mathcal{S}_k \times \mathcal{S}_k \to \mathbb{R}_{\geq 0}$ is a function such that $\lambda_k(a, \alpha, \beta)$ is the rate of the transition labeled a from state α to state β. If $\lambda_k(a, \alpha, \beta) = 0$ then we implicitly assume that there are no transitions in agent P_k from state α to β with action type a. Note that if $a \in \mathcal{P}_k$ and there exists a passive action of type a from state α to β then $\lambda_k(a, \alpha, \beta) = x_a$ where x_a is a variable denoting an unknown positive rate for passive action type a. The label ε is such that $\varepsilon \notin \mathcal{L}_k$ and it is the type of all the transitions that do not participate to the cooperation.

Assuming the rates of the passive action types known, the rate transition matrix $\mathbf{Q}_k = [q_k(\alpha, \beta)]$, with $\alpha, \beta \in \mathcal{S}_k$ associated with P_k can be derived as follows

$$q_k(\alpha, \beta) = \begin{cases} \sum\limits_{a \in \mathcal{L}_k \cup \{\varepsilon\}} \lambda_k(a, \alpha, \beta) & \text{if } \alpha \neq \beta \\ -\sum\limits_{\delta \in \mathcal{S}_k \setminus \{\alpha\}} q_k(\alpha, \delta) & \text{if } \alpha = \beta\,. \end{cases} \tag{2}$$

Note that, in this framework, a cooperation is well defined if:

- for each $a \in \mathcal{L}$ there exists a unique pair (k, l) such that $k \neq l$ and $a \in \mathcal{A}_k \cap \mathcal{P}_l$;
- for each $k = 1, \ldots, N$, we have $\mathcal{P}_k \subseteq \mathcal{L}$, i.e., all the passive actions cooperate with a corresponding active action.

The steady-state solution $\boldsymbol{\pi}_k$ of agent P_k, where $k = 1, \ldots, N$, is the non-trivial solution of the system of GBEs:

$$\boldsymbol{\pi}_k \mathbf{Q}_k = \mathbf{0}\,, \qquad \sum_{\alpha \in \mathcal{S}_k} \pi_k(\alpha) > 0, \tag{3}$$

where $\pi_k(\alpha)$ denotes the α-th component of the row vector $\boldsymbol{\pi}_k$.

B. Main idea

The main idea of our algorithm consists in using MARCAT for the computation of the transition rates with passive action types, i.e., the unknown rates. Note that if these values were known, by (3) the computation of the steady-state solution of each agent would be clearly trivial. We will now see how this can be accomplished. Let us consider the simple case of two agents cooperating on the action type set $\mathcal{L}$ and assume that conditions 1 and 2 of MARCAT are satisfied. Moreover, suppose that every $a \in \mathcal{L}$ is active in P_1 and passive is P_2. In this case we can simply derive the stationary distribution $\boldsymbol{\pi}_1$ of P_1, which does not depend on the solution $\boldsymbol{\pi}_2$ of P_2. Let α and β be a pair of distinct states in the CTMC of P_1 such that the transition rate $q_1(\alpha, \beta)$ is positive. Then, we can calculate the rate $\overline{q_1(\beta, \alpha)}$ in the CTMC of the reversed agent $\overline{P_1}$ as [16]:

$$\overline{q_1(\beta, \alpha)} = \frac{\pi_1(\alpha)}{\pi_1(\beta)} q_1(\alpha, \beta)\,. \tag{4}$$

Suppose that several actions take agent P_1 from α to β and one of these has type $a \in \mathcal{L}$ and rate $\lambda_1(a, \alpha, \beta)$. Then the reversed rate $\overline{\lambda_1(a, \beta, \alpha)}$ of action type $\overline{a}$ from state β to state α is given by [11]:

$$\overline{\lambda_1(a, \beta, \alpha)} = \frac{\lambda_1(a, \alpha, \beta)}{q_1(\alpha, \beta)} \overline{q_1(\beta, \alpha)}. \tag{5}$$

Using this equation we can verify condition 3, i.e., $\overline{\lambda_1(a, \beta, \alpha)} = \overline{p_a}$ independently of states α and β. Note that we have to consider the case of self-loops on active actions, i.e., a transition with label a from state γ to itself with rate $\lambda_1(a, \gamma, \gamma)$. In this case $\overline{\lambda_1(a, \gamma, \gamma)} = \lambda_1(a, \gamma, \gamma)$ by definition [9] and this must also be equal to $\overline{p_a}$. Now we can replace every unknown rate x_a of passive actions $a \in \mathcal{L}$ of P_2 with

$\overline{p_a}$ and derive the stationary solution $\boldsymbol{\pi}_2$ for P_2. By MARCAT we have that the joint solution is $\boldsymbol{\pi}_{1,2} \propto \boldsymbol{\pi}_1 \boldsymbol{\pi}_2$.

Unfortunately, it is not always true that all the actions in $\mathcal{L}$ are active with respect to the same agent and therefore this trivial solution cannot be applied. In order to deal with the general case, we develop an iteration scheme, which starts with randomly generated stationary distributions for the agents and, at each step, it updates the unknown rates of the transitions with passive action types using the steady-state distributions from the previous iteration. The obtained new rates are then used to compute the new steady-state solutions of the agents for the next iteration. Note that if an agent has more than one transition with the same active action type a we know that the reverse rate of each of them is expected to be constant by MARCAT condition 3 in the solution however, during this iterative process, they may indeed differ, so that we may have more than one candidate to choose from when updating x_a. In this case, we decide to update x_a with the mean of those values, although other selection strategies may be employed (e.g., weighted mean).

The resulting iterative algorithm is illustrated by Algorithm 1.

Algorithm 1: Simplified algorithm.

Input: agents $P_1, \ldots, P_N$; precision ϵ; maximum number of iterations M

Output: unnormalized stationary distribution $\boldsymbol{\pi}$ of $\bowtie_{\mathcal{L}}^{k=1,N} P_k$

Randomly initialize $\boldsymbol{\pi}_k$ for all $k = 1, \ldots, N$
$n = 0$
repeat
  for $k = 1, \ldots, N$ **do**
    $\boldsymbol{\pi}_k^{prev} \leftarrow \boldsymbol{\pi}_k$
  for $j, k = 1, \ldots, N$ **do**
    foreach $a \in (\mathcal{P}_j \cap \mathcal{A}_k)$ **do**
      /* Λ is the set of the reversed rates of a */
      $\Lambda \leftarrow \left\{ \lambda_k(a, \alpha, \beta) \frac{\pi_k(\alpha)}{\pi_k(\beta)} \,:\, \alpha, \beta \in \mathcal{S}_k \right\} \setminus \{0\}$
      foreach $\alpha, \beta \in \mathcal{S}_j \,:\, \lambda_j(a, \alpha, \beta) \neq 0$ **do**
        /* set the rates of the passive actions */
        $\lambda_j(a, \alpha, \beta) \leftarrow \text{mean}(\Lambda)$
  Compute $\boldsymbol{\pi}_k$ for all $k = 1, \ldots, N$
  $n \leftarrow n + 1$
until $n > M$ ***or*** $\forall k = 1, \ldots, N. \|\boldsymbol{\pi}_k - \boldsymbol{\pi}_k^{prev}\| < \epsilon$;
/* Check if the reversed rates are constant */
for $k = 1, \ldots, N$ **do**
  foreach $a \in \mathcal{A}_k$ **do**
    $\Lambda \leftarrow \{\lambda_k(a, \alpha, \beta) \,:\, \alpha, \beta \in \mathcal{S}_k\} \setminus \{0\}$
    if $\Lambda \neq \emptyset$ ***and*** $\max(\Lambda) - \min(\Lambda) > \epsilon$ **then**
      fail: MARCAT product-form not identified
return $\{\boldsymbol{\pi}_k\}_{k=1,\ldots,N}$

We terminate the iteration in two cases:

- the maximum number of iterations M is reached;
- the distance between two consecutive steady-state distributions is less than a user-defined precision ϵ, for all agents.

At the end, the solution is considered valid if the reversed rates of all the active action types for each process are constant within the precision ϵ or, in other words, the difference between the maximum and the minimum reversed rates corresponding to every active action type must be less than ϵ.

C. Convergence, termination and complexity

As we said in the introduction we do not have a proof of convergence of the algorithm in the general case yet (although in the following section we show that it converges for some special cases). As far as we know, this lack of proof is shared with the iteration scheme presented in [11] and with some algorithms defined for the iterative solution of a class of extended G-networks, e.g., [17].

At each iteration, Algorithm 1 computes a set of stationary distributions $\boldsymbol{\pi}_k$ with $k = 1, \ldots, N$ that can be easily bounded in a compact space (e.g. by normalizing the probabilities), i.e., for each $\alpha \in \mathcal{S}_k$ we have $0 < \pi_k(\alpha) < 1$. Hence, the algorithm cannot diverge, but still it may exhibit an undesired cyclic behavior. For this reason we have introduced the constant M with the aim of terminating the execution after a user-defined number of iterations. Even if the execution, for a given input and an initial random distribution of $\boldsymbol{\pi}_k$, converges, it may be the case that $\boldsymbol{\pi}_k$ is a fixed point but not the expected steady-state solution. Indeed, even if there exists a fixed point satisfying the MARCAT theorem, in theory there could also exist other fixed points yielding non-constant reverse rates in correspondence with transitions with the same active action type, thus violating MARCAT. Note that this may happen only if we have to compute the mean of the reversed rates of the active action types and this is never the case, for instance, in simple models such as Jackson networks and G-networks. Nevertheless, we applied our algorithm on randomly generated product-form cooperating agents, in which the computation of the mean of the reversed rates was required, and we never observed such a behaviour, but we always obtained the expected solution.

Assuming that all the agents $P_1, \ldots, P_N$ have the same number of states $|\mathcal{S}_k| = r$, the computational complexity of the algorithm is $\mathcal{O}(Nr^3)$ for each iteration. In our experiments, the number of iterations required for convergence (with tolerance less than $\epsilon = 10^{-5}$) was always less than 20, therefore we consider it as a constant factor.

IV. Examples

In this section we compare our approach with those previously defined for the solution of Jackson networks [2] and G-networks [15]. We show that in the former case Algorithm 1 becomes the Jacobi iteration scheme for the solution of the traffic equation system of the queueing network, while in the

latter it corresponds to the iteration scheme presented in [15] by Gelenbe et al. .

A. Jackson networks

Jackson queueing networks (JQNs) [2] consist of a set of exponential queues $\{1, \ldots, N\}$. Customers arrive to station k from the outside according to a Poisson process with rate γ_k and then they move among the queues according to a routing matrix $\mathbf{P} = [p_{kl}]$, where p_{kl} represents the probability that a customer enters station l after being served by station k with $1 \leq k, l \leq N$, while p_{k0} is the probability that the customer exits the network after being served by station k. Obviously $\sum_{l=0}^{M} p_{kl} = 1$. The service rate of station k is μ_k. The steady-state solution of a JQN is derived from the solution of the system of traffic equation in e_k, i.e.:

$$\begin{cases} e_1 = \gamma_1 + \sum_{l=1}^{N} e_l p_{l1} \\ \quad\vdots \\ e_N = \gamma_N + \sum_{l=1}^{N} e_l p_{lN} \end{cases} \tag{6}$$

where e_k is interpreted as the total arrival rate to station k for $k = 1, \ldots, N$. The load factor of a station is defined as $\rho_k = e_k/\mu_k$. Then:

$$\boldsymbol{\pi}(n_1, \ldots, n_N) = \prod_{k=1}^{N} \boldsymbol{\pi}_k(n_k) \propto \prod_{k=1}^{N} \left(\frac{e_k}{\mu_k}\right)^{n_k} .$$

In this case we model the Jackson network as explained in [9], i.e., every station k is a process with an infinite number of states. State $i \geq 0$ represents the number of customers in station k. Process P_k corresponding to the k-th queue is: $\mathcal{S}_k = \mathbb{N}$, $\mathcal{A}_k = \{a_{kl} : p_{kl} > 0, 1 \leq k, l \leq N\}$, $\mathcal{P}_k = \{a_{lk} : p_{lk} > 0, 1 \leq k, l \leq N\}$, and function λ_k is defined as follows (we write x_{lk} instead of $x_{a_{lk}}$):

$$\lambda_k(a, \alpha, \beta) = \begin{cases} \gamma_k & \text{if } \gamma_k > 0, a = \varepsilon, \beta = \alpha + 1 \\ x_{lk} & \text{if } p_{lk} > 0, a = a_{lk}, \beta = \alpha + 1 \\ \mu_k p_{k0} & \text{if } p_{k0} > 0, a = \varepsilon, \beta = \alpha - 1 \\ \mu_k p_{kl} & \text{if } p_{kl} > 0, a = a_{kl}, \beta = \alpha - 1 \\ 0 & \text{otherwise} \end{cases} .$$

Note that we exclude the case of self-loops in the station because, in this case, Theorem 1 cannot be straightforwardly applied, therefore from now ahead we assume $p_{kk} = 0$ for all $k = 1, \ldots, N$.

Example 1: Let us consider the JQN of Figure 1. We can model the behavior of this JQN with agents P_1, P_2, P_3 as depicted by Figure 2 where, according to the PEPA notation, the $\top$ symbol denotes the rates of the passive actions. Basically, the arrivals to a queue coming from another queue are modeled using passive actions, while the departures to other queues are modeled by active actions. We immediately observe that the structural conditions of MARCAT are satisfied and that the reversed rates of the active action type are constant. In [9] the author proves these properties for a general JQN and obtains the Jackson theorem using RCAT. Note that processes P_k, $k = 1, \ldots, 3$, can be completely described using just states 0 and 1 by exploiting their intrinsic regularity, hence we truncate the agent representation in order to consider just the first two states.

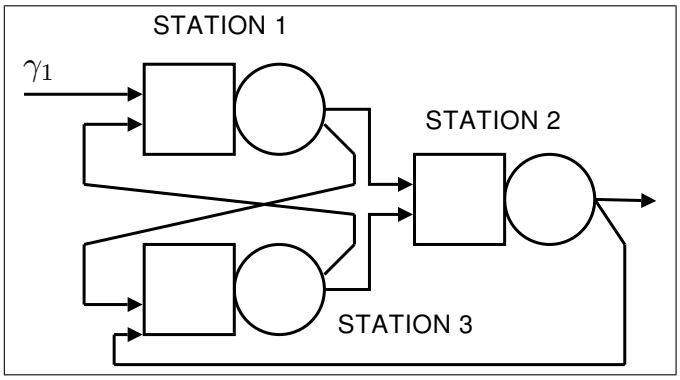

Fig. 1. Example of JQN.

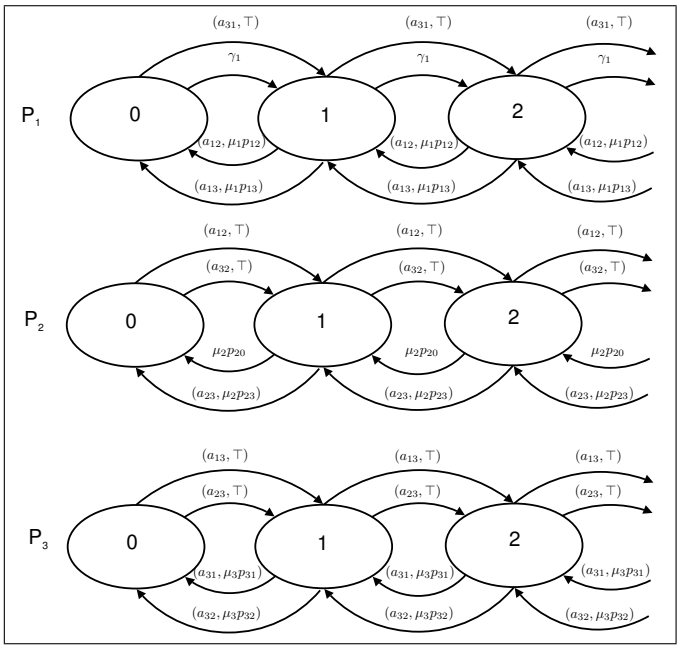

Fig. 2. Agents of the JQN of Figure 1, where the PEPA symbol $\top$ denotes the unknown rate of the passive action types.

In a JQN every agent can be modeled using just two states 0 and 1, then if we fix $\pi_k(0) = 1$ we have that $\pi_k(1) = \rho_k$, i.e., the load factor of queue k. In this case we have that for each $k = 1, \ldots, N$ the n-th iteration computes:

$$\pi_k(1)^{(n+1)} = \frac{\sum_{l=1}^{N} x_{lk}^{(n)} + \gamma_k}{\mu_k} , \tag{7}$$

where:

$$x_{lk}^{(n)} = \pi_l(1)^{(n)} \mu_l p_{lk} ,$$

is the reversed rate of active action type a_{lk} in agent P_l, with $1 \leq l, k \leq N$. Note that a fixed point for this iteration scheme is the solution of system (6) by setting $e_k = \pi_k(1)\mu_k$. Indeed, under this renaming, (7) turns out to be the Jacobi's iterative

method for the solution of the linear system (6) and since the matrix of coefficients deriving from it, namely $\mathbf{B} = [b_{kl}]$ with:

$$b_{kl} = \begin{cases} \mu_k & \text{if } k = l\,, \\ -p_{kl}\mu_k & \text{otherwise}\ , \end{cases}$$

is irreducibly (column) diagonally dominant, i.e., $|b_{kk}| \geq \sum_{l \neq k} |b_{lk}|$ for all k except at least one column where the inequality is strict (i.e., customers can leave the network after being served by at least one of the queues), the iterations always converge to the fixed point.

B. G-networks

G-networks with product-form solutions have been introduced in [6]. They consist of a set of N nodes (G-queues), and two types of customers, i.e., positive and negative. Positive customers behave exactly in the same way as customers in a normal JQN, i.e., they arrive from the outside to G-queue k according to an independent Poisson process with rate γ_k^+. The service time at G-queue k is exponentially distributed with rate μ_k. After being served by G-queue k, positive customers may leave the node as positive customers and join node l with probability p_{kl}^+, or leave the network with probability p_{k0}^+ or, finally join node l as negative customer with probability p_{kl}^-. In the latter case, if G-queue l has $n > 0$ customers, then the negative customer deletes a queued customer taking the node population to $n-1$, while if the queue is empty, then the negative customer does not affect the node state. Finally, negative customers can arrive from the outside to a node k according to an independent Poisson process with rate γ_k^-. Matrices $\mathbf{P}^+ = [p_{kl}^+]$, with $1 \leq k \leq N$ and $0 \leq l \leq N$ and $\mathbf{P}^- = [p_{mn}^-]$, with $1 \leq m, n \leq N$ model the positive and negative routing of the customers. In [6] the author proves, by verifying the network GBEs, that such a model has a product-form solution that depends on the solution of a non-linear system of traffic equations. Moreover, he gives an iteration scheme for computing the unnormalized stationary distribution.

In [13] the author proves that G-networks can be studied using theorem 1 and system (1) is non-linear. In our framework every G-queue corresponds to an agent P_k, with $k = 1, \ldots, N$ with an infinite number of states. Process P_k is: $\mathcal{S}_k = \mathbb{N}$, $\mathcal{A}_k = \{a_{kl}^+ : p_{kl}^+ > 0, 1 \leq k, l \leq N\} \cup \{a_{kl}^- : p_{kl}^- > 0, 1 \leq k, l \leq N\}$, $\mathcal{P}_k = \{a_{lk}^+ : p_{lk} > 0, 1 \leq k, l \leq N\} \cup \{a_{lk}^- : p_{lk} > 0, 1 \leq k, l \leq N\}$, and function λ_k is defined as follows (we write x_{lk}^+ and x_{lk}^- instead of $x_{a_{lk}^+}$ and $x_{a_{lk}^-}$, respectively):

$$\lambda_k(a, \alpha, \beta) = \begin{cases} \gamma_k^+ & \text{if } \gamma_k^+ > 0, a = \varepsilon, \beta = \alpha + 1 \\ \gamma_k^- & \text{if } \gamma_k^- > 0, a = \varepsilon, \beta = \alpha - 1 \\ x_{lk}^+ & \text{if } p_{lk}^+ > 0, a = a_{lk}^+, \beta = \alpha + 1 \\ \mu_k p_{k0}^+ & \text{if } p_{k0}^+ > 0, a = \varepsilon, \beta = \alpha - 1 \\ \mu_k p_{kl}^+ & \text{if } p_{kl}^+ > 0, a = a_{kl}^+, \beta = \alpha - 1 \\ \mu_k p_{kl}^- & \text{if } p_{kl}^- > 0, a = a_{kl}^-, \beta = \alpha - 1 \\ x_{lk}^- & \text{if } p_{lk}^- > 0, a = a_{lk}^-, \beta = \alpha - 1 \\ x_{lk}^+ & \text{if } p_{lk}^- > 0, a = a_{lk}^-, \beta = \alpha = 0 \\ 0 & \text{otherwise} \end{cases}.$$

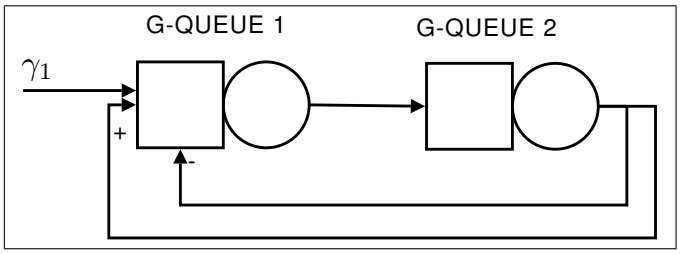

Fig. 3. Example of G-network.

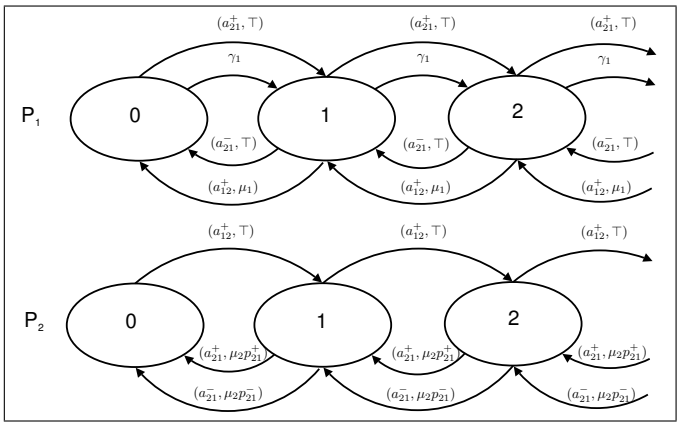

Fig. 4. Agents of the G-network of Figure 3, where the PEPA symbol $\top$ denotes the unknown rates of passive action types.

Example 2: Let us consider the simple G-network depicted in Figure 3. For this simple G-network, agents P_1 and P_2 are shown in Figure 4. Note that, since the structures of the agents are very regular and the underlying processes are a birth and death processes with constant rates, we can apply our algorithm on agents that consist only of states 0 and 1. Let P_k be an agent modeling a G-queue, and let $\mathcal{S}_k = \{0, 1\}$ (see Example 2). Then we set $\pi_k(0) = 1$ and we derive the following iteration scheme:

$$\pi_k(1)^{(n+1)} = \frac{\gamma_k^+ + \sum_{l=1}^N x_{lk}^{+(n)}}{\mu_k + \gamma_k^- + \sum_{l=1}^N x_{lk}^{-(n)}},$$

where $x_{lk}^{+(n)}$ and $x_{lk}^{-(n)}$ are the reversed rate of actions a_{lk}^+ and a_{lk}^- at the n-th iteration, respectively:

$$\begin{aligned} x_{lk}^{+(n)} &= \pi_l(1)^{(n)} \mu_l p_{lk}^+ \\ x_{lk}^{-(n)} &= \pi_l(1)^{(n)} \mu_l p_{lk}^-. \end{aligned}$$

We observe that this iteration scheme is the same given in [18] to compute the unnormalized stationary distribution of G-networks.

C. Randomly generated instances

In order to consider more general cases, we have written a random generator of agents that ensures that their composition is in product-form by MARCAT. We can control the following parameters of the generator:

- the number of active and passive action types

label	from	to	λ_1(label, from, to)
ε	2	3	3.78295
ε	3	1	3.08947
ε	3	2	2.74250
a	2	3	0.21705
a	3	1	2.91053
a	3	2	2.25750
b	1	2	x_b
b	2	2	x_b
b	3	1	x_b

TABLE I
RATES OF PROCESS P_1 OF FIGURE 5.

label	from	to	λ_2(label, from, to)
ε	1	2	8.00000
ε	1	4	3.00000
ε	2	1	6.15150
ε	2	3	8.28395
ε	2	4	7.67033
ε	3	1	15.0000
ε	3	2	9.70455
ε	4	1	16.0000
b	2	1	5.64850
b	2	3	0.51605
b	2	4	2.12967
b	3	2	6.99545
a	1	2	x_a
a	2	2	x_a
a	3	1	x_a
a	4	1	x_a

TABLE II
RATES OF PROCESS P_2 OF FIGURE 5.

- the number of the active action transitions of a given type and the presence of self-loops
- the number of transitions with type ε
- the number of states

Using this tool we generated several instances of cooperations, and then we verified that the computed solutions were correct. In this way it was possible to study models consisting of hundred of cooperating agents with a variable number of states. Since the solution of the GBEs for a single agent is not the main topic of this research, we focus on problems with a large number of relatively small agents rather than dealing with agents having very large state-spaces.

As an example of generated instance, we show the iterations of the algorithm for the simple cooperation of the agents depicted in Figure 5, where the processes are drawn isolating the transitions with label ε, a and b. Label a is active in P_1 and passive in P_2 and b is active in P_2 and passive in P_1.

The transition rates are given in Tables I and II and we have $\mathcal{A}_1 = \{a\}$, $\mathcal{P}_1 = \{b\}$, $\mathcal{A}_2 = \{b\}$ and $\mathcal{P}_2 = \{a\}$. Since the structural conditions of Theorem 1 are satisfied we can run Algorithm 1 and we obtain the iterations of Table III. It is possible to see that the convergence is really fast even if it is not very significant for this small example.

V. OPTIMIZATIONS

The computation of Algorithm 1 can be improved by taking into account the reciprocal dependencies of the systems of GBEs (3). Indeed, the rate transition matrix of any agent is in general a function of the steady-state solutions of some other agents. These relations can be modeled in terms of a *dependency graph*, i.e., a directed graph $D = (V, E)$, where V is the set of agents and E is the set of dependencies among them. Specifically, if $\mathbf{Q}_i$ depends on $\boldsymbol{\pi}_j$ (i.e. $\mathcal{P}_i \cap \mathcal{A}_j \neq \emptyset$) then $(i, j) \in E$. A *strongly connected component* (SCC) of $D = (V, E)$ is a maximal subset of vertices $\mathcal{C} \subseteq V$ such that there is a path from each vertex in $\mathcal{C}$ to every other vertex in $\mathcal{C}$. Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two SCCs of D, then we say that $\mathcal{C}_1$ *depends on* $\mathcal{C}_2$ if there exists $(i, j) \in \mathcal{C}_1 \times \mathcal{C}_2$ such that there is a path in D from i to j. Note that by the maximality of a SCC, we can either have that $\mathcal{C}_1$ depends on $\mathcal{C}_2$ or vice versa. Moreover, the set of SCCs $\{\mathcal{C}_1, \ldots, \mathcal{C}_T\}$ of a dependency graph forms a partition of its vertex set, and by contracting each SCC to a single vertex the resulting dependency graph is acyclic. This implies that we can always order the set $\{\mathcal{C}_1, \ldots, \mathcal{C}_T\}$ of SCCs of D in a way as to have that for every $1 \leq i < j \leq T$, $\mathcal{C}_i$ does not depend on $\mathcal{C}_j$. Indeed, it suffices to apply the Tarjan's algorithm [19] on D.

Another optimization is related to the computation of the reversed rate of the active action types. Indeed, we have observed in Section III that the reversed rate of a transition from a state to itself labeled with an active action type (self-loop), is the equal to the forward rate, independently of the stationary distribution. As a consequence all the self-loops with the same active type must have the same rate or MARCAT conditions are trivially not satisfied. Moreover, it is possible to set the correct rate of the passive action types corresponding with an active action type with at least one self-loop.

These first two optimizations are implemented in Algorithm 2.

Finally, we observe that the algorithm that we propose can be easily implemented in a parallel version. In fact, the computation of the stationary probabilities of each process can be done in parallel at each iteration.

Example 3: As an instance, we consider the cooperation of 10 agents whose dependency graph is shown in Figure 6. Recall that, the dependency graph is defined such that every agent is a vertex and, given a pair of agents P_k and P_l, there is an edge from P_k to P_l if $\mathcal{P}_k \cap \mathcal{A}_l \neq \emptyset$, i.e., the solution of P_k requires the reversed rates of P_l. For the example of Figure 6, Algorithm 2 computes the following sets $\mathcal{C}_i$: $\mathcal{C}_1 = \{1\}$, $\mathcal{C}_2 = \{7\}$, $\mathcal{C}_3 = \{2\}$, $\mathcal{C}_4 = \{9, 10\}$, $\mathcal{C}_5 = \{3, 4, 5, 6, 8\}$ and converges to the solution within 7 iterations (needed to solve the set $\mathcal{C}_5$).

VI. CONCLUSIONS

In this paper we present an algorithm to compute the stationary distributions of product-form models that satisfy MARCAT with a set of additional structural conditions. In this framework, we model the interactions between a pair of

Input: agents $P_1, \ldots, P_N$; precision ϵ; maximum number of iterations M

Output: unnormalized stationary distribution $\boldsymbol{\pi}$ of $\bigbowtie_{\mathcal{L}}^{N}{}_{k=1} P_k$

```
/* Build and solve the dependency graph                                    */
```
$V \leftarrow \{1, \ldots, N\}$

foreach $(i, j) \in V \times V$ **do**

  if $\mathcal{P}_i \cap \mathcal{A}_j \neq \emptyset$ **then**

    $E \leftarrow E \cup \{(i, j)\}$

$[\mathcal{C}_1, \ldots, \mathcal{C}_T] \leftarrow$ Tarjan(V, E)

```
/* Solve every SCC identified                                              */
```
for $i = 1 \ldots T$ **do**

  ```
  /* Look for active actions in self-loop                                  */
  ```
  $\mathcal{F} \leftarrow \emptyset$

  foreach $j, k \in \mathcal{C}_i$ **do**

    foreach $a \in \mathcal{P}_j \cap \mathcal{A}_k$ **do**

      ```
      /* Λ is the set of the rates of self-loops of a                      */
      ```
      $\Lambda \leftarrow \{\lambda_k(a, \alpha, \alpha) : \alpha \in \mathcal{S}_k\} \setminus \{0\}$

      if $\Lambda \neq \emptyset$ **then**

        if $\max(\Lambda) - \min(\Lambda) < \epsilon$ **then**

          foreach $\alpha, \beta \in \mathcal{S}_j : \lambda_j(a, \alpha, \beta) \neq 0$ **do**

            ```
            /* set the corresponding passive rates                         */
            ```
            $\lambda_j(a, \alpha, \beta) \leftarrow$ mean(Λ)

          ```
          /* action a is resolved                                          */
          ```
          $\mathcal{F} \leftarrow \mathcal{F} \cup \{a\}$

        else

          fail: MARCAT product-form does not exists

  Randomly initialize $\boldsymbol{\pi}_k$ for all $k \in \mathcal{C}_i$

  $n \leftarrow 0$

  repeat

    foreach $k \in \mathcal{C}_i$ **do**

      $\boldsymbol{\pi}_k^{prev} \leftarrow \boldsymbol{\pi}_k$

    foreach $j, k \in \mathcal{C}_i$ **do**

      foreach $a \in (\mathcal{P}_j \cap \mathcal{A}_k) \setminus \mathcal{F}$ **do**

        ```
        /* Λ is the set of the reversed rates of a                         */
        ```
        $\Lambda \leftarrow \left\{\lambda_k(a, \alpha, \beta)\frac{\pi_k(\alpha)}{\pi_k(\beta)} : \alpha, \beta \in \mathcal{S}_k\right\} \setminus \{0\}$

        foreach $\alpha, \beta \in \mathcal{S}_j : \lambda_j(a, \alpha, \beta) \neq 0$ **do**

          ```
          /* set the rates of the passive actions                          */
          ```
          $\lambda_j(a, \alpha, \beta) \leftarrow$ mean(Λ)

    Compute $\boldsymbol{\pi}_k$ for all $k \in \mathcal{C}_i$

    $n \leftarrow n + 1$

  until $n > M$ ***or*** $\forall k \in \mathcal{C}_i. \|\boldsymbol{\pi}_k - \boldsymbol{\pi}_k^{prev}\| < \epsilon$;

  ```
  /* Check if the reversed rates are constant                              */
  ```
  foreach $k \in \mathcal{C}_i$ **do**

    foreach $a \in \mathcal{A}_k$ **do**

      $\Lambda \leftarrow \{\lambda_k(a, \alpha, \beta) : \alpha, \beta \in \mathcal{S}_k\} \setminus \{0\}$

      if $\Lambda \neq \emptyset$ ***and*** $\max(\Lambda) - \min(\Lambda) > \epsilon$ **then**

        fail: MARCAT product-form not identified

return $\{\boldsymbol{\pi}_k\}_{k \in V}$

Algorithm 2. Complete algorithm for the computation of the stationary distribution of product-form interacting processes.

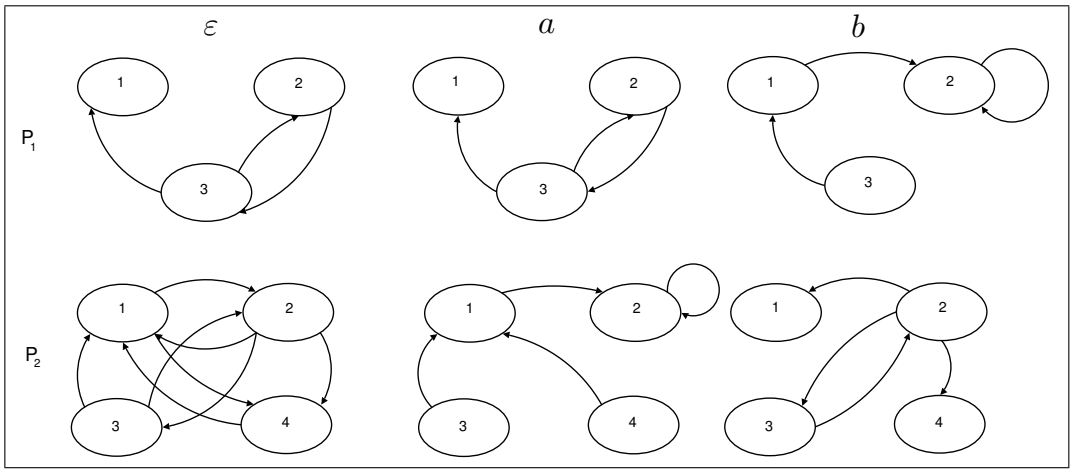

Fig. 5. Cooperation between two agents, P_1 and P_2, with 3 and 4 states, respectively. The cooperation is defined on labels a and b that are active (passive) and passive (active) with respect to P_1 (P_2).

#	$\pi_1(1)$	$\pi_1(2)$	$\pi_1(3)$	Δx_a	$\pi_2(1)$	$\pi_2(2)$	$\pi_2(3)$	$\pi_2(4)$	Δx_b
0	.055787	.278604	.875938	45.6300	.168255	.485570	.275246	.226932	15.3907
1	.266071	.597774	.137156	.988026	.406056	.379423	.067697	.146824	4.25536
2	.357752	.505092	.127156	.502830	.550291	.190975	.051409	.207323	1.93054
3	.492525	.387580	.119896	.010167	.552210	.188650	.051014	.208125	.038720
4	.494281	.386123	.119590	.005012	.554051	.186425	.050630	.208894	$7.69963E-4$
5	.495963	.384740	.119297	$1.0048E-4$	.554070	.186402	.050626	.208902	$3.79002E-4$
6	.495981	.384723	.119293	$4.9855E-5$	.554088	.186380	.050622	.208909	$6.56724E-6$

TABLE III
THE FIRST SIX ITERATIONS OF ALGORITHM 1 FOR THE INPUT OF THE EXAMPLE PRESENTED IN SECTION IV-C.

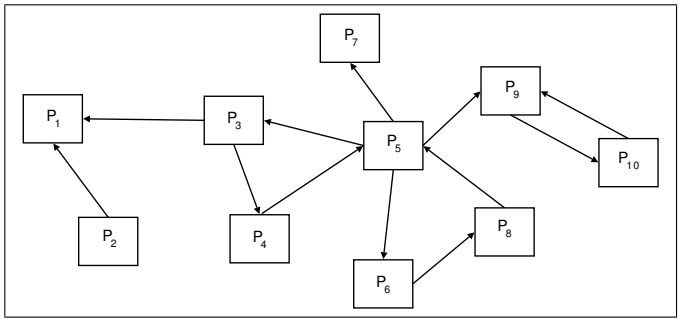

Fig. 6. Example of dependency graph of 10 agents in cooperation.

agents using active and passive action types in a PEPA-like way. The structural conditions required for the application of our algorithm can be informally summarized as follows:

1) in the cooperation, every passive action has a corresponding active action that governs its rate,
2) if an action a is passive with respect to an agent, then every state of that agent has exactly one outgoing transition of type a (possibly in self-loop)
3) if an action a is active with respect to an agent, then every state of that agent has an incoming transition of type a (possibly in self-loop)

Actually, if the latter two conditions are not satisfied, MARCAT may be nevertheless be applied if some conditions on the rates of the joint process are satisfied. In particular, in the general case, one should check for each state belonging to the ergodic sub-chain of the joint process a condition on the balance of the forward and the reversed process. Although this can be done efficiently in some cases (e.g. when the structures of the agents and of the joint process are very regular), in the general case this can have a high computational cost.

In our opinion, the main strengths of the proposed algorithm can be summarized as follows:

- it is easy to implement because the data structures that are needed are simple matrices. Moreover, the iterations are mainly based on the solution of a set of linear systems, i.e., no symbolic computation is required at all;
- the solution of the traffic equations are computed using the stationary distributions and the properties of the reversed Markov processes, instead of deriving a non-linear system from the analysis of the structures of the processes (e.g., using Kolmogorov's criteria);
- experimental evidence has shown a really fast convergence and a good numerical stability. For example, in one of our tests we have combined 10 processes with around 100 states for process, and around 15 cooperating action types, obtaining a solution in less than 10 iterations (for a computation time within a second, in a Pentium III personal computer). For this instance, the analysis of the joint process, would have been computationally very

expensive.

Of course, the drawback of this work is that we have not proved the convergence for general processes yet, although convergence is known for Jackson queueing networks and G-networks. From this point of view, as far as we know, there are no algorithms that can solve the non-linear MARCAT traffic equations in the general case and have been proved to converge. Anyway, we stressed our algorithm with several randomly generated tests and we have always obtained the correct solution.

Future works have two directions. First, some research efforts should be devoted to the definition of sufficient conditions for the convergence. Second, the algorithm may be extended in order to be able to study more complicated synchronization problems like those described in [13], [17].

REFERENCES

[1] J. Hillston, "A Compositional Approach to Performance Modelling," Ph.D. dissertation, Department of Computer Science, University of Edinburgh, 1994.

[2] J. R. Jackson, "Jobshop-like queueing systems," *Management Science*, vol. 10, pp. 131–142, 1963.

[3] F. Baskett, K. M. Chandy, R. R. Muntz, and F. G. Palacios, "Open, closed, and mixed networks of queues with different classes of customers," *J. ACM*, vol. 22, no. 2, pp. 248–260, 1975.

[4] A. A. Lazar and T. G. Robertazzi, "Markovian Petri Net Protocols with Product Form Solution." in *IEEE INFOCOM'87, The Conf. on Computer Communications, Proc., Sixth Annual Conference - Global Networks: Concept to Realization*. Washington, DC: IEEE Computer Society Press, 1987, pp. 1054–1062.

[5] W. Henderson, D. Lucic, and P. G. Taylor, "A net level performance analysis of Stochastic Petri Nets," *J. Austral. Math. Soc. Ser. B*, vol. 31, pp. 176–187, 1989.

[6] E. Gelenbe, "Product form networks with negative and positive customers," *Journal of Applied Prob.*, vol. 28, no. 3, pp. 656–663, 1991.

[7] J. M. Fourneau, "Computing the steady-state distribution of networks with positive and negative customers," in *13th IMACS World Congress on Computation and Applied Mathematics*, Dublin, 1991.

[8] E. Gelenbe and R. Suros, "G-networks with multiple classes of negative and positive customers," in *Theoret. Comput. Sci.*, 1996, pp. 141–156.

[9] P. G. Harrison, "Turning back time in Markovian process algebra," *Theoretical Computer Science*, vol. 290, no. 3, pp. 1947–1986, January 2003. [Online]. Available: http://pubs.doc.ic.ac.uk/rcat/

[10] ——, "Reversed processes, product forms and a non-product form," *Linear Algebra and Its Applications*, vol. 386, pp. 359–381, July 2004. [Online]. Available: http://pubs.doc.ic.ac.uk/ercat/

[11] P. G. Harrison and T. T. Lee, "Separable equilibrium state probabilities via time reversal in markovian process algebra," *Theoretical Computer Science*, vol. 346, no. 1, pp. 161–182, 2005.

[12] P. G. Harrison, "Reversed processes, product forms, non-product forms and a new proof of the BCMP theorem," in *Int. Conf. on the Numerical Solution of Markov Chains (NSMC 2003), Urbana IL, USA, September 2-5 2003*, September 2003, pp. 289–304. [Online]. Available: http://pubs.doc.ic.ac.uk/rcat2-nsmc/

[13] ——, "Compositional reversed Markov processes, with applications to G-networks," *Perform. Eval., Elsevier*, vol. 57, no. 3, pp. 379–408, 2004.

[14] A. Argent-Katwala, "Automated product-forms with meercat," in *SMCtools 06: workshop on Tools for solving structured Markov chains*, vol. 201, no. 10, Pisa, Italy, 2006, p. 10.

[15] S. Chabridon, E. Gelenbe, M. Hernandèz, and A. Labed, "G-networks: A survey of results, applications and solutions," in *Quantitative Methods in Parallel Systems*. Springer, 1995, pp. 114–128.

[16] F. Kelly, *Reversibility and stochastic networks*. New York: Wiley, 1979.

[17] J.-M. Fourneau and F. Quessette, "Computing the steady-state distribution of G-networks with synchronized partial flushing," in *ISCIS, 21th International Symposium*, Istanbul, Turkey, 2006, pp. 887–896.

[18] J. Fourneau, E. Gelenbe, and R. Suros, "G-networks with multiple class negative and positive customers," in *MASCOTS '94: Proc. of the Second Int. Workshop on Modeling, Analysis, and Simulation On Computer and Telecommunication Systems*. Washington, DC, USA: IEEE Computer Society, 1994, pp. 30–34.

[19] R. Tarjan, "Depth-first search and linear graph algorithms," *SIAM J. on Computing*, vol. 1, no. 2, pp. 146–160, 1972.