AutoCAT: Automated Product-Form Solution of Stochastic Models

Giuliano Casale and Peter G. Harrison

Abstract We propose algorithms to automatically generate exact and approximate product-form solutions for large Markov processes that cannot be solved by direct numerical methods. Focusing on models that can be described as cooperating Markov processes, which include queueing networks and stochastic Petri nets as special cases, it is shown that finding a global optimum for a non-convex quadratic program is sufficient for determining a product-form solution. Such problems are notoriously hard to solve due to the inherent difficulty of searching over non-convex sets. Using a potential theory for Markov processes, convexification techniques and a class of linear constraints that follow from stochastic characterization of product-form solutions, we obtain a family of linear programming relaxations that can be solved efficiently. A sequence of these linear programs is solved under increasingly tighter constraints to determine the exact product-form solution of a model when one exists. This approach is then extended to obtain approximate solutions for non-product-form models. Finally, our new techniques are validated with examples and increasingly complex case studies, which show the effectiveness of the method on both conventional and novel performance models.

1 Introduction

Performance modeling often involves the abstraction of the various components of a system under study and their mutual interactions as a Markov process. Although there exist several high-level formalisms for specifying particular classes of Markov

Giuliano Casale
Imperial College London, Department of Computing, 180 Queen's Gate, SW7 2AZ London, e-mail: g.casale@imperial.ac.uk

Peter G. Harrison
Imperial College London, Department of Computing, 180 Queen's Gate, SW7 2AZ London, e-mail: pgh@doc.ic.ac.uk

processes, such as queueing networks or stochastic Petri nets, the state space explosion problem typically limits our ability to compute metrics related to the long-term behavior of the system. A notable exception is the class of product-form models, in which the equilibrium probability of a state is a scaled product of the marginal state probabilities of the Markov processes that represent the individual components of the system. Foremost examples of models enjoying a product-form include open and closed queueing networks with single and multiple service classes [6,29], possibly supporting various forms of blocking [5] and different arrival types [17, 19, 20], stochastic Petri nets [3], Markovian process algebras [26] and stochastic automata networks [18].

We introduce AUTOCAT, an optimization-based technique that automatically *constructs* exact or approximate product-forms for a very large class of performance models. We consider models that may be described as a cooperation (*i.e.*, synchronization) of Markov processes over a given set of named actions [41]. This class of processes includes as special cases queueing networks, stochastic Petri nets, stochastic automata and several other model types that are popular in performance evaluation [27]. While certain Markov processes enjoy a number of useful properties for determining a product-form solution, such as reversibility [31], quasi-reversibility [30, 37] and local balance [38], cooperating Markov processes additionally benefit from their compositional structure, which is conducive to recursive analysis and the reversed compound agent theorem (RCAT) in particular [24, 25]. As we discuss in Section 2, RCAT defines a set of sufficient conditions for cooperating Markov processes to enjoy a product-form solution. To the best of our knowledge, RCAT is the most general formalism available to construct product-forms by means of simple conditions that do not require direct solution of the joint probability distribution of the model at equilibrium. We leverage on this result and show that RCAT product-form conditions are equivalent to a non-linear optimization problem with non-convex quadratic constraints. Non-convex global optimization is $\mathcal{NP}$-hard in general [11], thus we derive efficient linear programming (LP) relaxations that are solved sequentially to find the exact product-form of a model when one exists. Since the length of such a sequence depends on the tightness of the LP relaxations, we define a hierarchy of increasingly tighter linear programs, based on a potential theory for Markov processes [12], convexification techniques [36] and a set of linear constraints that we derive from the RCAT product-form conditions.

Most importantly, this procedure is extended to the *approximate* analysis of non-product-form Markov processes – as arise in the vast majority of practical systems. It is applied first in 'toy' examples, to illustrate the method, and then in case studies to validate its main features and to assess its numerical tractability and accuracy. Among these models, it is shown that such approximations may be useful to investigate closed queueing network models with phase-type distributed service times [8]. Recently, it has been shown in [14, 15] that such models may be approximated quite accurately by approximate product-form solutions. We here provide an example illustrating that the AUTOCAT approximation may provide improved accuracy with respect to the methods of [14, 15].

Our method provides one of the first available, non-application-specific algorithms for product-form analysis; moreover, at the same time, it constructs automatically workable approximations for the equilibrium probabilities of interacting Markov processes without product-form. A preliminary constructive tool of this type was proposed by Argent-Katwala in [2], where a symbolic solver was proposed for product-forms based on the sufficient conditions of RCAT. This tool constructed product-forms for Markov processes composed from others for which a product-form was already known, but it was not able to *detect* whether a given Markov process admits a product-form solution. Moreover, the cost of symbolic linear algebra inevitably makes the technique applicable only to simple processes. In [9, 10], Buchholz defines the first general-purpose automatic technique for identifying exact and approximate product-form solutions in stochastic models. The methodology minimizes a residual error norm using an optimization technique based on efficient quadratic programming. Using the stochastic automata network (SAN) formalism, Buchholz's method uses a Kronecker representation of the cooperations to avoid generating the joint state space. Then, an iterative technique searches for a local optimum that is used to compute an approximate product-form. In [4, 34], Marin *et al.* propose INAP, a fixed-point method to estimate reversed rates in RCAT product-forms. The main benefit of the INAP algorithm is computational efficiency, which enables the analysis of large models.

Summarizing, our main contributions are the following:

- We present an algorithm to automatically decide whether a given Markov model has a product-form solution and, if so, to compute it without solving the underlying global balance equations. Specifically, in Section 3, we introduce a formulation of the problem in the form of a quadratically constrained optimization and we obtain efficient linear relaxations in Section 4.
- In Section 5, we show that this algorithm guarantees, within the boundaries of the numerical tolerance of the optimizer, to find a product-form solution if it exists.
- Next, approximation techniques stemming from our methodology are developed in Section 6. Such approximations can be applied to a wide class of performance models that are represented as cooperations of Markov processes.

Our methodology is validated on small examples and case studies in Section 7, which testify to the effectiveness of the approach on performance models of practical interest. These include stochastic Petri nets and closed queueing networks with phase-type distributed service times.

2 Preliminaries

We consider a collection of M Markov processes that cooperate over a set of A actions. Each cooperating process might represent, for example, a queue, a stochastic automata, a Petri net or an agent in a stochastic process algebra. Process k is

defined on a set of $N_k \geq 1$ states, such that the joint state space of the Markov process comprising the cooperation has up to $N_{prod} = \prod_k N_k$ states. Process indices are $k, m = 1, \ldots, M$, $m \neq k$, action indices are $a, b, c = 1, \ldots, A$ and (marginal) state indices for process k are $n_k, n'_k = 1, \ldots, N_k$. An action a labels a synchronizing transition in a pairwise cooperation between two processes k and $m \neq k$, which can only take place in both processes simultaneously. We follow the convention of defining *active* and *passive* roles for each action a in the pair of processes it synchronizes. The set of active (respectively passive) actions for process k is denoted by $\mathcal{A}_k$ (respectively $\mathcal{P}_k$). Consider an action a such that $a \in \mathcal{A}_k$ and $a \in \mathcal{P}_m$, *i.e.* which is active in k and passive in $m \neq k$. Further, assume that when action a is enabled, it triggers with rate μ_a state transitions $n_k \to n'_k$ and $n_m \to n'_m$ in processes k and m respectively. We summarize this information in *rate matrices* $\boldsymbol{A}_a$ and $\boldsymbol{P}_a$ of orders N_k and N_m respectively. That is, we set the values $\boldsymbol{A}_a[n_k, n'_k] = \mu_a$ and $\boldsymbol{P}_a[n_m, n'_m] = p_m$, for each pair (n_k, n'_k) and (n_m, n'_m) where a is enabled, where p_m is the probability of the transition $n_m \to n'_m$ in the passive process when action a takes place, and $\boldsymbol{M}[i, j]$ stands for the element at row i and column j of matrix $\boldsymbol{M}$. Note that the rate of the passive action is unspecified, this later being assigned according to the equilibrium behavior of the active process (*i.e.*, process k here) [24]. Observe also that the rates of transitions $n_k \to n_k$ lie on the diagonal of $\boldsymbol{A}_a$. Such rates define *hidden transitions* which do not alter the local state of the active process but can affect the local state of the passive process m. Finally, we account for local state-jumps that are not due to cooperations, which we call *local transitions*. The rates of all local transitions for process k are stored in the $N_k \times N_k$ matrix $\boldsymbol{L}_k$.

2.1 Product-Form Solutions

We assume the joint process underlying the cooperation to be ergodic and the goal of our analysis is to determine a product-form expression for the model's joint state probability function at equilibrium. Unless otherwise stated, we always refer to the RCAT product-form defined in [22]; note that this is a superset of product-forms that can be obtained by quasi-reversibility [35]. Our goal is to find marginal probability vectors $\pi_k(n_k)$ for each cooperating process k such that the equilibrium solution of the model enjoys the product-form expression

$$\alpha(n_1, \ldots, n_k, \ldots, n_M) = G^{-1} \pi_1(n_1) \pi_2(n_2) \cdots \pi_M(n_M) \tag{1}$$

where G is a normalising constant and $\alpha(n_1, \ldots, n_k, \ldots, n_M)$ is the joint state probability function. Under the RCAT methodology, finding a product-form solution such as (1) requires one to analyze each process k in 'isolation', *i.e.*, to study its transitions over the marginal state space $\mathcal{S}_k = \{n_k \,|\, 0 \leq n_k < N_k\}$. If process k cooperates passively on one or more actions $b \in \mathcal{P}_k$, their (passive) rates of occurrence in isolation are undefined. Thus, they cannot be solved for the marginal

probabilities $\pi_k(n_k)$ before such rates are assigned. This is because the rate of a passive action in process k may depend on the state of the cooperating process m, as we elaborate below. The RCAT theorem introduced in [24, 25] establishes that, if we can define a generator matrix $\boldsymbol{Q}_k$ on $\mathcal{S}_k$ satisfying the conditions RC1, RC2 and RC3 stated at the end of this section, then the equilibrium vectors $\boldsymbol{\pi}_k$ satisfying $\boldsymbol{\pi}_k\boldsymbol{Q}_k = 0$, $\boldsymbol{\pi}_k\mathbf{1} = 1$ for $1 \leq k \leq M$, provide a product-form solution (1). Specifically, if the three sufficient conditions of RCAT are met, a certain outgoing rate x_b, called a *reversed rate*, is associated with each passive action $b \in \mathcal{P}_k$ in each state n_k where b is enabled. We point to [24] for a probabilistic interpretation of x_b as a rate in a time-reversed Markov process. The RCAT conditions together with the reversed rates then allow the generators $\boldsymbol{Q}_k$ of each Markov component-process to be defined uniquely as follows:

$$\boldsymbol{Q}_k \equiv \boldsymbol{Q}_k(\boldsymbol{x}) = \boldsymbol{L}_k + \sum_{a\in\mathcal{A}_k} \boldsymbol{A}_a + \sum_{b\in\mathcal{P}_k} x_b\boldsymbol{P}_b - \boldsymbol{\Delta}_k(\boldsymbol{x}), \tag{2}$$

where $\boldsymbol{x} = (x_1, \ldots, x_a, \ldots, x_A)^T > \mathbf{0}$ is the vector of reversed rates and $\boldsymbol{\Delta}_k(\boldsymbol{x})$ is the diagonal matrix ensuring that $\boldsymbol{Q}_k\mathbf{1} = \mathbf{0}$. From $\boldsymbol{Q}_k$, we can compute the product-form solution of the cooperation based on (1). This provides a major computational advantage over direct solution of the joint process.

2.1.1 RCAT Sufficient Conditions

The original formulation of RCAT was expressed using the stochastic process algebra PEPA [24, 25]. We provide here a reformulation of RCAT's conditions using matrix expressions that are simpler to integrate in optimization programs.

- RCAT *Condition 1 (RC1).* Passive actions are always enabled, *i.e.*, $\boldsymbol{P}_a\mathbf{1} \geq \mathbf{1}$, for all $a \in \mathcal{P}_k, 1 \leq k \leq M$.
- RCAT *Condition 2 (RC2).* For $a \in \mathcal{A}_k$, each state in process k has an incoming transition due to active action a, *i.e.*, $\boldsymbol{A}_a^T\mathbf{1} > \mathbf{0}$, for all $a \in \mathcal{A}_k, 1 \leq k \leq M$.
- RCAT *Condition 3 (RC3).* There exists a vector of reversed rates

  $$\boldsymbol{x} = (x_1, \ldots, x_a, \ldots, x_A)^T > \mathbf{0},$$

  such that the generators $\boldsymbol{Q}_k$ have equilibrium vectors $\boldsymbol{\pi}_k$ that satisfy the following *rate equations* $\boldsymbol{\pi}_k\boldsymbol{A}_a = x_a\boldsymbol{\pi}_k$, for all $a \in \mathcal{A}_k, 1 \leq k \leq M$. (Note that we use the generalized expression introduced in [35] in place of the original condition in [24], although the two forms are equivalent.)

If the above conditions are met, then the vectors $\boldsymbol{\pi}_k$ immediately define a product-form solution (1). We stress, however, that verifying RC3 is much more challenging than RC1 and RC2 as it is necessary in practice to *find* a reversed rate vector $\boldsymbol{x}$ that satisfies the rate equations.

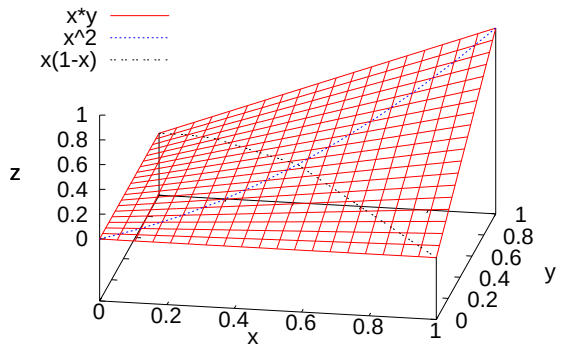


Fig. 1 The bilinear surface $z = xy$ is non-convex since it includes both convex (*e.g.* $z = x^2$) and concave (*e.g.* $z = x(1-x)$) functions.

3 Does a Product-Form Exist?

Let us now turn to the problem of finding an algorithm that automatically constructs a RCAT product-form if one exists. We assume initially that every component-process has a finite state space ($N_k < \infty \ \forall k$), the generalization to countably infinite state spaces being simple, as discussed in Appendix 9. According to RC3, to construct a RCAT product-form we need to find a solution $\boldsymbol{x} > 0$ of the exact non-linear system of equations

$$\begin{aligned} \text{ENS}: \qquad \boldsymbol{\pi}_k \boldsymbol{A}_a &= x_a \boldsymbol{\pi}_k, & a \in \mathcal{A}_k, 1 \le k \le M \\ \boldsymbol{\pi}_k \boldsymbol{Q}_k(\boldsymbol{x}) &= \boldsymbol{0}, & 1 \le k \le M \\ \boldsymbol{\pi}_k \boldsymbol{1} &= 1, & 1 \le k \le M \end{aligned}$$

This defines a non-convex feasible region due to the bilinear products $x_a \boldsymbol{\pi}_k$ and $x_b \boldsymbol{\pi}_k$ in the first two sets of constraints. Such a region is illustrated in Figure 1. We now provide the following characterization.

Proposition 1. *Consider vectors* $\boldsymbol{x}^{L,0} = (x_1^{L,0}, x_2^{L,0}, \ldots, x_A^{L,0})^T$ *and* $\boldsymbol{x}^{U,0} = (x_1^{U,0}, x_2^{U,0}, \ldots, x_A^{U,0})^T$ *defined by the values*

$$x_a^{L,0} = \min_{i \in \mathcal{I}^+} \textstyle\sum_j \boldsymbol{A}_a[i,j], x_a^{U,0} = \max_i \textstyle\sum_j \boldsymbol{A}_a[i,j]$$

where $\mathcal{I}^+ = \{i \mid \sum_j \boldsymbol{A}_a[i,j] > 0\}$. *Then any feasible solution of* ENS *satisfies the necessary condition* $\boldsymbol{x}^{L,0} \le \boldsymbol{x} \le \boldsymbol{x}^{U,0}$.

Proof. The statement follows directly from RC3 since $x_a = x_a \boldsymbol{\pi}_k \boldsymbol{1} = \boldsymbol{\pi}_k \boldsymbol{A}_a \boldsymbol{1}$. Thus, $x_a = \boldsymbol{\pi}_k \boldsymbol{A}_a \boldsymbol{1} \ge \boldsymbol{\pi}_k (x_a^{L,0} \boldsymbol{1}) = x_a^{L,0}$ and $x_a = \boldsymbol{\pi}_k \boldsymbol{A}_a \boldsymbol{1} \le \boldsymbol{\pi}_k (x_a^{U,0} \boldsymbol{1}) = x_a^{U,0}$. □

Let us also note that $\boldsymbol{x}$ satisfies ENS if and only if it is a global minimum for the quadratically constrained program

$$\begin{aligned}
\text{QCP}: \quad f_{qcp} &= \min \textstyle\sum_a (\boldsymbol{s}_a^+ + \boldsymbol{s}_a^-) & \\
\boldsymbol{\pi}_k \boldsymbol{A}_a - x_a \boldsymbol{\pi}_k &= \boldsymbol{s}_a^+ - \boldsymbol{s}_a^- & a \in \mathcal{A}_k, 1 \leq k \leq M \\
\boldsymbol{\pi}_k \boldsymbol{Q}_k(\boldsymbol{x}) &= \boldsymbol{0} & 1 \leq k \leq M \\
\boldsymbol{\pi}_k \boldsymbol{1} &= 1 & 1 \leq k \leq M \\
\boldsymbol{s}_a^+ \geq \boldsymbol{0},\ \boldsymbol{s}_a^- &\geq \boldsymbol{0} & a \in \mathcal{A}_k \\
\boldsymbol{x}^{L,0} \leq \boldsymbol{x} &\leq \boldsymbol{x}^{U,0} &
\end{aligned}$$

which has $O(A + N_{sum})$ variables and $O(AMN_{max})$ constraints, where $N_{sum} = \sum_k N_k$ and $N_{max} = \max_k N_k$. Here $\boldsymbol{s}_a^+$ and $\boldsymbol{s}_a^-$ are slack variables that guarantee feasibility of all constraints in the early stages of the non-linear optimization where the solver may be unable to determine a feasible assignment of $\boldsymbol{x}$ in ENS. By construction, $f_{qcp} \geq 0$. Furthermore, RC3 holds if and only if $f_{qcp} = 0$. Since all other quantities are bounded, we can also find upper and lower bounds on $\boldsymbol{s}_a^+$ and $\boldsymbol{s}_a^-$. As such, QCP is a quadratically constrained program with box constraints, a class of problems that is known to be $\mathcal{NP}$-hard [11]. The difficulty in direct solution of ENS or QCP is clear even in 'toy problems', such as identifying product-forms in Jackson networks, *i.e.* queueing networks with exponential servers. For example, searching for a product-form in a Jackson queueing network with 2 feedback queues one often finds that MATLAB's fmincon function fails to identify in ENS the search direction due to the small magnitudes of the gradients. QCP has better numerical properties than ENS, but it can take up to 5-10 minutes on commonly available hardware to find the reversed rates $\boldsymbol{x}$ needed to construct the product-form (1). Thus QCP quickly becomes intractable on models with several queues. This shows that constructing product-form solutions by numerical optimization methods is, in general, a difficult problem. Moreover, it motivates an investigation of convex relaxations of ENS and QCP to derive efficient techniques for automatically constructing product-forms. Indeed, automatic product-form analysis is fundamental to generating approximations for non-product-form models, as we show in Section 6.

4 Linearization Methodology

We now seek to obtain efficient linear programming (LP) relaxations of ENS which overcome the difficulties of solving the non-linear system directly. To obtain an effective linearization, we first apply, in Section 4.1, an established convexification technique [36]. A tighter linear relaxation specific to RCAT is then developed in Section 4.2 and is shown to dramatically improve the quality of the relaxation. Finally, Section 4.3 obtains a tighter formulation based on a potential theory for Markov processes.

4.1 Convex Envelopes

In order to obtain a linearization of ENS, we first rewrite the generator matrix of process k as $\boldsymbol{Q}_k(\boldsymbol{x}) = \widetilde{\boldsymbol{T}}_k + \sum_{b\in\mathcal{P}_k} x_b \widetilde{\boldsymbol{P}}_b$, where $\widetilde{\boldsymbol{P}}_b$ is the sum of the $\boldsymbol{P}_b$ matrices and of the component of $\boldsymbol{\Delta}_k(\boldsymbol{x})$ that multiplies x_b. Then we condense the non-linear components of ENS into variables $\boldsymbol{z}_{c,k} = x_c\boldsymbol{\pi}_k$ such that we replace the bilinear terms $x_c\boldsymbol{\pi}_k$ in $\boldsymbol{\pi}_k\boldsymbol{Q}_k(\boldsymbol{x})$ by $\boldsymbol{z}_{c,k}$. Consequently, the only non-linear constraints left are $\boldsymbol{z}_{c,k} = x_c\boldsymbol{\pi}_k$ which we linearize to obtain a LP relaxation of ENS. We first observe that all variables involved in the bilinear product $x_c\boldsymbol{\pi}_k$ are bounded, as a consequence of Proposition 1 and the fact that probabilities are bounded. We can thereby always write bounds $\boldsymbol{x}^{L,0} \leq \boldsymbol{x} \leq \boldsymbol{x}^{U,0}$ and $\boldsymbol{\pi}_k^L \leq \boldsymbol{\pi}_k \leq \boldsymbol{\pi}_k^U$. Under these assumptions, for all $c \in \mathcal{A}_k$ and $1 \leq k \leq M$, $\boldsymbol{z}_{c,k}$ is always enclosed in the convex envelope proposed by McCormick in [36], which is known to be the tightest linear relaxation for bounded bilinear variables. Adding the constraint $\boldsymbol{z}_{c,k}\mathbf{1} = x_c$, this yields a linear programming relaxation of ENS:

$$
\begin{aligned}
\mathsf{LPR}(n): f^n_{lpr} = \min f(\boldsymbol{x},\boldsymbol{\pi}_k,\boldsymbol{z}_{c,k}) \qquad & \text{s.t.} \\
\boldsymbol{\pi}_k\boldsymbol{A}_a - \boldsymbol{z}_{a,k} = \mathbf{0} \qquad & 1\leq k\leq M, a\in\mathcal{A}_k \\
\boldsymbol{\pi}_k\widetilde{\boldsymbol{T}}_k + \textstyle\sum_b \boldsymbol{z}_{b,k}\widetilde{\boldsymbol{P}}_b = \mathbf{0} \qquad & 1\leq k\leq M \\
\boldsymbol{z}_{c,k} \geq x_c^{L,n}\boldsymbol{\pi}_k + x_c\boldsymbol{\pi}_k^{L,n} - x_c^{L,n}\boldsymbol{\pi}_k^{L,n} \qquad & 1\leq k\leq M, c\in\mathcal{A}_k\cup\mathcal{P}_k \\
\boldsymbol{z}_{c,k} \leq x_c^{L,n}\boldsymbol{\pi}_k + x_c\boldsymbol{\pi}_k^{U,n} - x_c^{L,n}\boldsymbol{\pi}_k^{U,n} \qquad & 1\leq k\leq M, c\in\mathcal{A}_k\cup\mathcal{P}_k \\
\boldsymbol{z}_{c,k} \leq x_c^{U,n}\boldsymbol{\pi}_k + x_c\boldsymbol{\pi}_k^{L,n} - x_c^{U,n}\boldsymbol{\pi}_k^{L,n} \qquad & 1\leq k\leq M, c\in\mathcal{A}_k\cup\mathcal{P}_k \\
\boldsymbol{z}_{c,k} \geq x_c^{U,n}\boldsymbol{\pi}_k + x_c\boldsymbol{\pi}_k^{U,n} - x_c^{U,n}\boldsymbol{\pi}_k^{U,n} \qquad & 1\leq k\leq M, c\in\mathcal{A}_k\cup\mathcal{P}_k \\
\boldsymbol{z}_{c,k}\mathbf{1} = x_c \qquad & 1\leq k\leq M, c\in\mathcal{A}_k\cup\mathcal{P}_k \\
\boldsymbol{\pi}_k\mathbf{1} = 1 \qquad & 1\leq k\leq M \\
\boldsymbol{\pi}_k^{L,n} \leq \boldsymbol{\pi}_k \leq \boldsymbol{\pi}_k^{U,n} \qquad & 1\leq k\leq M \\
\boldsymbol{x}^{L,n} \leq \boldsymbol{x} \leq \boldsymbol{x}^{U,n} &
\end{aligned}
$$

for an arbitrary *linear* objective function $f^n_{lpr} = f(\cdot)$, where n is an integer, used in Section 5 to parameterize a sequence of upper and lower bounding vectors on $\boldsymbol{x}$ and $\boldsymbol{\pi}_k$. The above optimization program is a LP that can be solved in polynomial time using interior-point methods. The number of variables and constraints in ENS grows asymptotically as $O(A + N_{sum})$ and $O(AMN_{max})$, respectively. In the above linearized version LPR, the number of variables increases as $O(AN_{sum})$ and the number of constraints remains at $O(AMN_{max})$ asymptotically.

Rejecting Existence of Product-Forms. When LPR is infeasible we can conclude that a RCAT product-form does not exist for the model under study. To see this, it is sufficient to observe that LPR is a relaxation of ENS. Thus, all solutions of ENR are feasible points of LPR, but there exist points in LPR which do not solve ENS. Thus, since the feasible region of LPR is larger than that of ENR, we conclude that

if LPR is infeasible, then so is ENS. This provides an interesting innovation over existing techniques for determining product-form solutions since none is currently able to exclude the existence of a product-form when one cannot be found. (Note that, although we can then conclude that there is no RCAT product-form, we cannot exclude the possibility that there is a non-RCAT product-form, were such to exist.)

4.2 Tightening the Linear Relaxation

We now define our first method for obtaining tighter linearizations of ENS, based on specific properties of the RCAT theorem. This is useful because McCormick's bounds are known to be wide in many cases [1]. Applying recursively the rate equations in RC3 v times we may write $\boldsymbol{\pi}_k \boldsymbol{A}_a^{v+1} = x_a^{v+1} \boldsymbol{\pi}_k$, for all $a \in \mathcal{A}_k$, $1 \leq k \leq M$, since RC3 implies that we can exchange scaling by x_a with right multiplication by $\boldsymbol{A}_a$. Summing over all $v \geq 0$, we get $\boldsymbol{\pi}_k \boldsymbol{A}_a \boldsymbol{H}_a = x_a (1 - x_a)^{-1} \boldsymbol{\pi}_k$, where $\boldsymbol{H}_a = (\boldsymbol{I} - \boldsymbol{A}_a)^{-1}$ and we have assumed, without loss of generality, that the units of measure of the rates are scaled such that $x_a \leq x_a^U < 1$ and $\rho(\boldsymbol{A}_a) < 1$, where $\rho(\boldsymbol{M})$ denotes the spectral radius of a matrix $\boldsymbol{M}$. Rearranging terms and using $\boldsymbol{z}_{a,k} = x_a \boldsymbol{\pi}_k$, we get the new linear constraint

$$\boldsymbol{\pi}_k \boldsymbol{A}_a \boldsymbol{H}_a = \boldsymbol{z}_{a,k} (\boldsymbol{I} + \boldsymbol{A}_a \boldsymbol{H}_a) \tag{3}$$

for all active actions $a \in \mathcal{A}_k$ and processes $k,\ 1 \leq k \leq M$. This provides an extra set of constraints that can be added to the linear relaxation of ENS to refine (reduce) the feasible region. Note that since $\boldsymbol{A}_a$ is a constant matrix, (3) is a linear equation in $\boldsymbol{\pi}_k$ and $\boldsymbol{z}_{a,k}$.

The advantages of the method outlined above become even more apparent when we consider the generator constraint in LPR. For example, if we left-multiply by x_a, we get

$$x_a \boldsymbol{Q}_k = \boldsymbol{z}_{a,k} \widetilde{\boldsymbol{T}}_k + \textstyle\sum_b \boldsymbol{z}_{b,k} \boldsymbol{A}_a \widetilde{\boldsymbol{P}}_b = \boldsymbol{0} \tag{4}$$

where we use the fact that the exchange rule holds for $\boldsymbol{z}_{b,k}$ too, since $x_a \boldsymbol{z}_{b,k} = x_a x_b \boldsymbol{\pi}_k = x_b \boldsymbol{\pi}_k \boldsymbol{A}_a = \boldsymbol{z}_{b,k} \boldsymbol{A}_a$, for all $a \in \mathcal{A}_k$, $1 \leq k \leq M$. Equation (4) creates a direct linear relationship between terms $\boldsymbol{z}_{a,k}$ and $\boldsymbol{z}_{b,k}$ for active and passive actions that cannot be inferred directly from LPR since it is based on exact knowledge of the bilinear relation $\boldsymbol{z}_{b,k} = x_b \boldsymbol{\pi}_k$. As we show in an illustrative example at the end of this subsection, the additional constraints (3)-(4) greatly improve the LP approximation of ENS. Furthermore, following a similar argument, we can generate a hierarchy of linear constraints for $v = 0, 1, \ldots$

$$\boldsymbol{z}_{a,k} \boldsymbol{A}_a^v \widetilde{\boldsymbol{T}}_k + \textstyle\sum_b \boldsymbol{z}_{b,k} \boldsymbol{A}_a^{v+1} \widetilde{\boldsymbol{P}}_b = \boldsymbol{0} \tag{5}$$

together with the condition obtained by summing over $v \geq 0$:

$$\boldsymbol{z}_{a,k} \boldsymbol{H}_a \widetilde{\boldsymbol{T}}_k + \textstyle\sum_b \boldsymbol{z}_{b,k} \boldsymbol{A}_a \boldsymbol{H}_a \widetilde{\boldsymbol{P}}_b = \boldsymbol{0}. \tag{6}$$

In summary, we have refined the linearization into the hierarchy of tight linear programming relaxations TLPR(n, V) which extends LPR by including the constraints (5) for $v = 1, \ldots, V$ and (6).

We remark that, even though the above formulation is much more detailed than LPR, it inevitably requires an increased number of constraints, which now grows as $O(VAMN_{max})$, while the complexity in terms of number of variables is the same as LPR. Thus, increased accuracy is obtained at a cost of additional computational complexity.

4.3 Potential-Theory Constraints

Potential theory is often applied in sensitivity and transient analyses of Markov processes and in Markov decision process theory [12]. We use it to derive tighter linearizations of ENS; to the best of our knowledge, this is the first time that potentials are applied to product-form theory.

Consider a process with generator matrix $\boldsymbol{Q}_k(\boldsymbol{x})$ and equilibrium probability vector $\boldsymbol{\pi}_k(\boldsymbol{x})$, and define the vector $\boldsymbol{f}(n_k) = (f_1, \ldots, f_i, \ldots, f_{N_k})^T$ where $f_i = 1$ if $i = n_k$ and $f_i = 0$ otherwise. The linear metric $\eta_k(\boldsymbol{x}) = \boldsymbol{\pi}_k(\boldsymbol{x})\boldsymbol{f}(n_k) = \pi_k(n_k)$ is then the marginal probability of state n_k in process k, given $\boldsymbol{x}$. Potential theory provides compact formulas for studying the changes in the values of linear functions such as $\eta_k(\boldsymbol{x})$ under arbitrarily large perturbations of the generator matrix $\boldsymbol{Q}_k(\boldsymbol{x})$. Let $\boldsymbol{x}_0 > 0$ be an arbitrary reference point for $\boldsymbol{x}$ such that $\boldsymbol{Q}_k(\boldsymbol{x}_0)$ is a valid generator matrix with equilibrium vector $\boldsymbol{\pi}_k^0$ and $\eta_k(\boldsymbol{x}_0) = \pi_k^0(n_k)$. Then it is straightforward to show that the difference between $\eta_k(\boldsymbol{x})$ and $\eta_k(\boldsymbol{x}_0)$ is (as in [12]):

$$\eta_k(\boldsymbol{x}) - \eta_k(\boldsymbol{x}_0) = \boldsymbol{\pi}_k(\boldsymbol{x})(\boldsymbol{Q}_k(\boldsymbol{x}) - \boldsymbol{Q}_k(\boldsymbol{x}_0))\boldsymbol{g}(\boldsymbol{x}_0, n_k) \tag{7}$$

where $\boldsymbol{g}(\boldsymbol{x}, n_k) = (-\boldsymbol{Q}_k(\boldsymbol{x}) + \mathbf{1}\boldsymbol{\pi}_k(\boldsymbol{x}))^{-1}\boldsymbol{f}(n_k)$ is the so-called *zero-potential* of the function $\eta_k(\boldsymbol{x})$. (Notice that $\boldsymbol{\pi}_k(\boldsymbol{x}) = \boldsymbol{\pi}_k(\boldsymbol{x})(-\boldsymbol{Q}_k(\boldsymbol{x}) + \mathbf{1}\boldsymbol{\pi}_k(\boldsymbol{x}))$.) For the system under study, we can use (2) to rewrite (7) as

$$\pi_k(n_k) - \pi_k^0(n_k) = \textstyle\sum_b (\boldsymbol{z}_{b,k}(\boldsymbol{x}) - x_b^0 \boldsymbol{\pi}_k(\boldsymbol{x}))\boldsymbol{P}_b \boldsymbol{g}(\boldsymbol{x}_0, n_k)$$

By defining the *potential matrix* $\boldsymbol{G}(\boldsymbol{x}_0) = [\boldsymbol{g}(\boldsymbol{x}_0, n_1) \quad \boldsymbol{g}(\boldsymbol{x}_0, n_2) \quad \ldots \quad \boldsymbol{g}(\boldsymbol{x}_0, N_k)]$ we obtain a new set of linear constraints

$$\boldsymbol{\pi}_k - \boldsymbol{\pi}_k^0 = \textstyle\sum_b (\boldsymbol{z}_{b,k}(\boldsymbol{x}) - x_b^0 \boldsymbol{\pi}_k(\boldsymbol{x}))\boldsymbol{P}_b \boldsymbol{G}(\boldsymbol{x}_0) \tag{8}$$

This provides a further tightening of the linear relaxation of ENS.

Note that zero-potentials can be memory consuming to evaluate because the matrix inverse $(-\boldsymbol{Q}_k(\boldsymbol{x}) + \mathbf{1}\boldsymbol{\pi}_k(\boldsymbol{x}))^{-1}$ does not preserve the sparsity of $\boldsymbol{Q}_k$. Furthermore, the rank-1 update $\mathbf{1}\boldsymbol{\pi}_k$ cannot be performed efficiently since $\boldsymbol{Q}_k$ is a singular matrix and updating techniques such as the Sherman-Morrison formula do not apply [39]. We address this computational issue by the algorithm shown in Figure 2,

which modifies the classical Jacobi iteration [40] to take advantage of the rank-1 structure of the term $\mathbf{1}\boldsymbol{\pi}_k$. That is, at each iteration, we isolate a vector $\boldsymbol{h}$ from the residual matrix in such a way that the matrix $\mathbf{1}\boldsymbol{\pi}_k$ is never explicitly computed. Thus, only vectors of the same order of $\boldsymbol{\pi}_k$ are stored in memory and also $\boldsymbol{Q}_k$ remains in sparse form. In this way, each potential can always be computed efficiently with respect to storage requirements and in the worst case has asymptotic computational cost $O(JN_k^2)$, J being the number of Jacobi iterations. The potential matrix $\boldsymbol{G}$ is therefore computed in $O(JN_k^3)$ steps and, for the fixed reference point $\boldsymbol{x}_0$, needs to be evaluated only once, requiring a computation time that is usually small compared to the time required to solve the linear optimization programs. Moreover, even for the largest models, it is always possible to consider constraints arising from a subset of the columns of $\boldsymbol{G}$, again posing a trade-off between computational costs and accuracy. Finally, using the exchange rule discussed in Section 4.2, we again

Input: $\boldsymbol{Q}(\boldsymbol{x}_0)$, $\boldsymbol{\pi}(\boldsymbol{x}_0)$, $\boldsymbol{f}(n_k)$, ϵ_{tol}; **Output**: $\boldsymbol{g}(\boldsymbol{x}_0, n_k)$
$k = 0$, $\boldsymbol{g}^{(0)} = \boldsymbol{\pi}(\boldsymbol{x}_0)$, $\boldsymbol{R} = diag(-\boldsymbol{Q}(\boldsymbol{x}_0) + \mathbf{1}\boldsymbol{\pi}(\boldsymbol{x}_0)) + \boldsymbol{Q}(\boldsymbol{x}_0)$
do $k = k + 1$
$\boldsymbol{g}^{(k)} = (diag(-\boldsymbol{Q}(\boldsymbol{x}_0) + \mathbf{1}\boldsymbol{\pi}(\boldsymbol{x}_0)))^{-1}(\boldsymbol{R}\boldsymbol{g}^{(k-1)} - \mathbf{1}\boldsymbol{\pi}_k(\boldsymbol{x}_0)\boldsymbol{g}^{(k-1)}) + \boldsymbol{f}(n_k))$
while $||\boldsymbol{g}^{(k)}(\boldsymbol{x}_0, n_k) - \boldsymbol{g}^{(k-1)}(\boldsymbol{x}_0, n_k)||_2 > \epsilon_{tol}||\boldsymbol{g}^{(k)}(\boldsymbol{x}_0, n_k)||_2$
return $\boldsymbol{g}^{(k)}(\boldsymbol{x}_0, n_k)$

Fig. 2 Memory-efficient computation of potentials; $diag(\boldsymbol{M})$ defines a diagonal matrix from the diagonal of $\boldsymbol{M}$.

get the hierarchy of constraints

$$\boldsymbol{\pi}_k \boldsymbol{A}_a^v - x_a^v \boldsymbol{\pi}_k^0 = \textstyle\sum_b (\boldsymbol{z}_{b,k}(\boldsymbol{x}) \boldsymbol{A}_a^v - x_b^0 \boldsymbol{\pi}_k(\boldsymbol{x}) \boldsymbol{A}_a^v) \boldsymbol{P}_b \boldsymbol{G}(\boldsymbol{x}_0) \tag{9}$$

and the asymptotic condition after simple algebra becomes

$$\boldsymbol{\pi}_k \boldsymbol{A}_a' \boldsymbol{H}_a - x_a \boldsymbol{\pi}_k^0 = \textstyle\sum_b (\boldsymbol{z}_{b,k}(\boldsymbol{x}) - x_b^0 \boldsymbol{\pi}_k(\boldsymbol{x})) \boldsymbol{A}_a' \boldsymbol{H}_a \boldsymbol{P}_b \boldsymbol{G}(\boldsymbol{x}_0) \tag{10}$$

where $\boldsymbol{A}_a' \stackrel{def}{=} \boldsymbol{A}_a - \boldsymbol{A}_a^2$. Summarizing, we can add (8), (9), and (10) to LPR to generate tighter relaxations. We denote the resulting zero-potential relaxation as ZPR(n, V). Note that ZPR(n, V) has the same asymptotic complexity of TLPR(n, V).

5 Exact Product-Form Construction

We now consider a technique for finding the solution of ENS that solves a sequence of the linear relaxations defined in the previous section, *i.e.* LPR(n), TLPR(n), or ZPR(n). Since the approach is identical for all relaxations, we limit the discussion to LPR(n).

The iterative algorithm defines a sequence of progressively tighter bounds $\boldsymbol{x}^{L,n}$ and $\boldsymbol{x}^{U,n}$ on the reversed rates $\boldsymbol{x}$, such that for sufficiently large n, LPR(n) de-

termines a feasible solution $\boldsymbol{x}$ of ENS if one exists. Initial conditions are $\boldsymbol{x}^{L,0} \stackrel{def}{=} \boldsymbol{x}^{L}$, $\boldsymbol{x}^{U,0} \stackrel{def}{=} \boldsymbol{x}^{U}$, where $\boldsymbol{x}^{L}$ and $\boldsymbol{x}^{U}$ are the bounds defined in Proposition 1. We have the following result.

Proposition 2. *For each $n = 1, 2, \ldots$, consider a sequence of $2A$ linear relaxations of* ENS*, the first A with objective function $f_{lpr}^{n,c} = \max x_c$ and the remaining A with objective function $g_{lpr}^{n,c} = \min x_c$, for action c, $1 \leq c \leq A$ and bounds $\boldsymbol{x}^{L,n}$, $\boldsymbol{x}^{U,n}$, as above.*

- *If $f_{lpr}^{n,c} = x_c^{U,n}$ or $g_{lpr}^{n,c} = x_c^{L,n}$ then the c^{th} component of the linear relaxation solution $\boldsymbol{x}$ satisfies the bilinear constraint $\boldsymbol{z}_{c,k} = x_c \boldsymbol{\pi}_k$ that is necessary for a solution of* ENS*.*
- *Otherwise, $f_{lpr}^{n,c} < x_c^{U,n}$ and the bounds at iteration $n+1$ may be refined to*

$$x_c^{U,n+1} \stackrel{def}{=} f_{lpr}^{n,c}, \qquad x_c^{L,n+1} \stackrel{def}{=} g_{lpr}^{n,c}$$

  for all actions c, $1 \leq c \leq A$, which define a feasible region for LPR$(n+1)$ *that is strictly tighter than for* LPR(n)*.*

Proof. Consider the case $g_{lpr}^{n} = x_c^{L,n}$, so that LPR(n) makes the assignment $x_c = x_c^{L,n}$. Then the first two McCormick constraints become

$$\begin{aligned} \boldsymbol{z}_{c,k} &\geq x_c^{L,n} \boldsymbol{\pi}_k + x_c^{L,n} \boldsymbol{\pi}_k^{L,n} - x_c^{L,n} \boldsymbol{\pi}_k^{L,n} \\ \boldsymbol{z}_{c,k} &\leq x_c^{L,n} \boldsymbol{\pi}_k + x_c^{L,n} \boldsymbol{\pi}_k^{U,n} - x_c^{L,n} \boldsymbol{\pi}_k^{U,n} \end{aligned}$$

which readily imply the bilinear relation $\boldsymbol{z}_{c,k} = x_c^{L,n} \boldsymbol{\pi}_k = x_c \boldsymbol{\pi}_k$. A similar proof holds for $f_{lpr}^{n} = x_c^{U,n}$.

Otherwise, if $f_{lpr}^{n} > x_c^{L,n}$, then the feasible region of LPR$(n+1)$ does not include any point outside LPR(n) and excludes points $x_c = x_c^{L,n}$. Hence it is strictly tighter than the feasible region for LPR(n). □

The above result guarantees that, if the sequence of linear relaxations yields feasible solutions, then the bounding box for LPR(n) defined by

$$[x_1^{L,n}, x_1^{U,n}] \times [x_2^{L,n}, x_2^{U,n}] \times \cdots \times [x_A^{L,n}, x_A^{U,n}]$$

can only decrease its volume or keep it constant as n increases. The volume must therefore converge as n increases, and for sufficiently large n, $x_c^{U,n} \approx x_c^{U,n+1}$ and $x_c^{L,n} \approx x_c^{L,n+1}$ in each dimension c. However, this implies that the outcome of iteration $n+1$ needs to be $f_{lpr}^{n+1,c} = x_c^{U,n}$ and $g_{lpr}^{n+1,c} = x_c^{U,n}$, which gives $\boldsymbol{z}_c = x_c \boldsymbol{\pi}_k$ by the first case of Proposition 2. Thus, for sufficiently large n, the border of the bounding box intersects points which are feasible for ENS, at least along one dimension c. This yields several possible outcomes for the sequence of linear relaxations:

- The constraints in the linear relaxations are infeasible. As observed earlier, this allows us to conclude that no feasible RCAT product-form exists for the model under study.

- One or more solutions $\boldsymbol{x}$ of the $2A$ linear relaxations are also feasible solutions of ENS. This allows us to construct directly a product-form solution by (1).
- No solution $\boldsymbol{x}$ of the $2A$ linear relaxations is feasible for ENS for all dimensions $c = 1, \ldots, A$. We have never encountered such a case in product-form detection for stochastic models; however, it can be resolved by a standard branch-and-bound method and re-applying the iteration on each partition of the feasible region.

Summarizing, a sequence of linear relaxations is sufficient to identify a product-form solution if one exists. No guarantee on the maximum number of linear programs to be solved can be given since the problem is $\mathcal{NP}$-hard in general, however we show in Section 7 that this is typically small. In Section 6, we further illustrate how this sequence of linear programs can be modified to identify an approximate product-form for a cooperation of Markov processes.

5.1 Practical Implementation

Pure Cooperations. If $\boldsymbol{x} = \mathbf{0}$ is a valid solution of ENS, we call the model a *pure cooperation*. This is because $\boldsymbol{x} = \mathbf{0}$ implies that $\boldsymbol{\pi}_k = \mathbf{0}$, which in turn requires all entries of $\boldsymbol{L}_k$ to be zero for all processes. Hence, the model's rates are solely those of cooperations. Pure cooperations represent a very large class of models of practical interest, e.g. closed queueing networks with exponential or hyper-exponential service, but their product-form analysis is harder due to the existence of infinite solutions $\boldsymbol{x}$.

Suppose a model is a pure cooperation and consider a graph defined by the $M \times M$ incidence matrix $\boldsymbol{G}$ such that $\boldsymbol{G}[i, j] = 1$ if and only if process j is passive in a cooperation with the active process i, 0 otherwise. Then if $r = \text{rank}(\boldsymbol{G}) < M$ the model has $M - r$ degrees of freedom in assigning the values of the $\boldsymbol{x}$ vector. Thus, for these models, there exists a continuous solution surface in ENS, rather than a single feasible solution. As we show in Section 7.3, this creates difficulties for existing product-form analysis techniques. However, we show that our method finds the correct solution $\boldsymbol{x} > 0$ under the condition that only objective functions of the type $\max x_c^U$ are used in the linear relaxations. This is because the search algorithm would otherwise converge, due to a lack of a strong lower bound, to the unreliable solution $\boldsymbol{x} = \mathbf{0}$.

Numerical properties. As the area of the bounding boxes decreases, the linear relaxations can be increasingly challenging to solve, due to the presence of many hundreds of constraints on a small area and to the numerical scale of the equilibrium probabilities, which can become very small when N_k is several tens or hundreds of states. In such conditions, and without a careful specification of the linear programs, the solver may erroneously return that the program is infeasible, whereas a feasible solution does exist. However, a number of strategies can prevent such problems. First, it is often beneficial to reformulate equality constraints as 'soft' constraints,

e.g., for a small $\epsilon_{tol} > 0$

$$\boldsymbol{\pi}_k \boldsymbol{Q}_k = \mathbf{0} \qquad \Rightarrow \qquad -\epsilon_{tol}\boldsymbol{\pi}_k \leq \boldsymbol{\pi}_k \boldsymbol{Q}_k \leq \epsilon_{tol}\boldsymbol{\pi}_k$$

which differentiates tolerances depending on the value of each individual term in $\boldsymbol{\pi}_k$. Another useful strategy consists of tuning the numerical tolerances of the LP solver. For instance, in IBM ILOG CPLEX's primal and dual simplex algorithms, this may be achieved by setting the Markowitz numerical tolerance to a large value such as 0.5. In addition, if the relaxation used is TLPR or ZPR, it is often beneficial for a numerically challenging model to revert to the LPR formulation, which is less constraining. Finally, for models where the feasible region is sufficiently small, one could solve QCP directly without much effort and with the benefit of removing the extra constraints introduced by the linear relaxations. In our implementation, such corrections are done at run-time though a set of retrial runs upon detection that a LP is infeasible.

6 Automated Approximations

Using the preceding LP-based method, a non-product-form solution may be approximated using a product-form. The particular approximation we propose differs depending on which condition out of RC1, RC2, and RC3 is violated. Two approximations are now elaborated.

6.1 Rate Approximation

RC3 becomes infeasible when the solver cannot find a single reversed rate x_a that satisfies the condition for some actions a. Assuming that a solution $\boldsymbol{x}$ defines a valid RCAT product-form, for each process k we can always define a Markov-modulated point process $\mathcal{M}_{a,k}$ associated with the activation of action $a \in \mathcal{A}_k$ in the Markov process with generator matrix $\mathbf{Q}_k$. Let the random variable $X_{a,k}$ denote the inter-arrival time between two consecutive activations of action a in $\mathcal{M}_{a,k}$ and define the rate $\lambda_{a,k}$ to be the reciprocal of the mean inter-arrival time $E[X_{a,k}]$. Then we approximate:

$$\lambda_{a,k} = \frac{1}{E[X_{a,k}]} \approx \boldsymbol{\pi}_k \boldsymbol{A}_a \mathbf{1} = x_a \boldsymbol{\pi}_k \mathbf{1} = x_a \tag{11}$$

The principle of the rate approximation is to assume (inexactly) that (11) is a sufficient condition for a product-form solution. Let $\widetilde{\boldsymbol{x}} = (\widetilde{x}_1, \widetilde{x}_2, \ldots, \widetilde{x}_A)$ be an approximate solution that can be found by the approximate ENS program which is defined by replacing RC3 with (11) in ENS and its relaxations. We note that $\widetilde{\boldsymbol{x}}$ includes the exact solution $\widetilde{\boldsymbol{x}} = \boldsymbol{x}$ when it exists; thus AENS is a relaxation of ENS. Note that using the approach introduced in Section 4.2, one may further tighten the relaxation

using a quadratic constraint

$$\boldsymbol{\pi}_k \boldsymbol{A}_a^2 \mathbf{1} = \widetilde{x}_a^2, \quad \forall a \in \mathcal{A}_k \tag{12}$$

which provides a more accurate approximation of x_a by $\widetilde{x}_a$, but involves relaxation of a convex, and thus efficiently solvable, quadratically-constrained program. Such an extension is left for future work. . The above approximation can be applied to all programs introduced in the preceding sections, e.g, for LPR we define the rate approximation ALPR.

6.2 Structural Approximation

Example cases where RC1 is violated are models with blocking, where a cooperating process is not allowed to synchronize passively owing to capacity constraints, *e.g.*, a queue with a finite size buffer. Similarly, violations of RC2 are examplified by models with priorities, where a low priority action is disabled until higher priority tasks are completed. Structural approximation iteratively updates the rate matrices $\boldsymbol{A}_a$ and $\boldsymbol{P}_b$ in order to account for the blocked or disabled status of certain transitions. Then, the search algorithm presented in Section 5 is run normally, if needed using rate approximation to address any violations of RC3 introduced by the updates. The updating process is detailed in the pseudo-code reported in Appendix 11. First, we correct the blocked (respectively disabled) transitions in $\boldsymbol{P}_b$ (respectively $\boldsymbol{A}_a$) by hidden transitions in the synchronising process that do change the state of the passive (respectively active) process. For $\boldsymbol{A}_a$, such hidden transitions need to be set to the reversed rate of action a in order to satisfy RC3. Next, a local iteration is done to scale the rates of the active (respectively passive) process to account approximately for the probability that the event could not occur in the passive (respectively active) process prior to the updates. Note that the particular way in which $\boldsymbol{A}_a$ and $\boldsymbol{P}_b$ are updated may be customized to reflect how the particular class of models under study handles the specific types of blocking or job priorities. For example, the pseudo-code applies to the case of blocking followed by retrials; variants are discussed in Section 7.5.

Example

Consider two small processes k and m with $N_k = 2$ and $N_m = 3$ states. Suppose there is a single action $a = 1$, $a \in \mathcal{A}_k$, $a \in \mathcal{P}_m$. The rate and local transition matrices are

$$\boldsymbol{L}_k = \begin{bmatrix} 0 & 0 \\ 10 & 0 \end{bmatrix}, \qquad \boldsymbol{A}_a = \begin{bmatrix} 10 & 15 \\ 0 & 0 \end{bmatrix}, \boldsymbol{L}_m = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad \boldsymbol{P}_a = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

Process k has a high transition rate between its two states and the one $1 \rightarrow 2$ requires synchronization with process m. However, when process m is in state 1, no passive action is enabled (all zeros in the first row of $\boldsymbol{P}_a$). Hence, k is prevented from transiting from state 1 to state 2.

The structural approximation sets $\boldsymbol{P}_a(1,1) = 1$ and corrects the rates in process k to account for the blocking effects. In the resulting model, $\boldsymbol{L}_k$ and $\boldsymbol{L}_m$ are unaffected, instead

$$\boldsymbol{A}_a^{(1)} = \begin{bmatrix} 10 + \alpha_{a,1}15 & (1-\alpha_{a,1})15 \\ 0 & 0 \end{bmatrix}, \; \boldsymbol{P}_a^{(1)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

where $\alpha_{a,1} = \boldsymbol{\pi}_m^{(0)}[1]$ and $\boldsymbol{\pi}_m^{(0)}$ is the equilibrium probability distribution for process m in the model for iteration $n = 0$ having rate matrices $\boldsymbol{A}_a^{(0)} = \boldsymbol{A}_a$ and $\boldsymbol{P}_a^{(0)} = \boldsymbol{P}_a^{(1)}$. In this way, we have adjusted the active rates in such a way that, for the fraction of time where process m is in state 1, process k has the rate of action a's transitions to another state proportionally reduced. For this example, it is found that the fraction of the joint probability mass incorrectly placed by the product-form approximation converges after 4 iterations to 5.9%, while it is 45% if we just add $\boldsymbol{P}_a^{(1)}(1,1) = 1$ and do not apply corrections to $\boldsymbol{A}_a^{(1)}$.

7 Examples and Case Studies

7.1 Example: LPR *and* TLPR

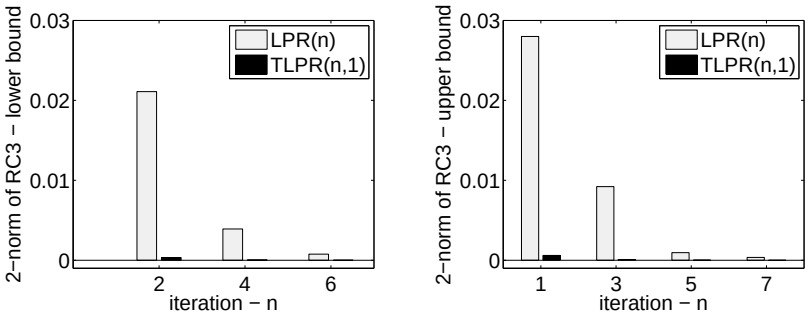
We use a small example to illustrate and compare typical levels of tightness obtained by TLPR and LPR. The results are shown in Figure 3, where the 2-norm for the current optimal solution with respect to the RC3 formula is evaluated for LPR(n) and TLPR$(n,1)$. The algorithm, described in Section 5, increases the lower bound on the reversed rates in each iteration. The model is composed of $M = 2$ agents that interact over $A = 2$ actions a and b with $\mathcal{A}_1 = \{a\}$, $\mathcal{P}_1 = \{b\}$, $\mathcal{A}_2 = \{b\}$, and $\mathcal{P}_2 = \{a\}$. Process 1 has $N_1 = 2$ states, process 2 is defined by $N_1 = 4$ states. Rates of active actions and local transitions are given in Table 1. The passive rate matrices have $\boldsymbol{P}_a(1,4) = \boldsymbol{P}_a(2,1) = \boldsymbol{P}_a(3,2) = \boldsymbol{P}_a(4,3) = 1$ and $\boldsymbol{P}_b(2,1) = 1$.

For this example, the LP solver finds a product-form in both cases, with reversed rates $x_a = 0.659039$ and $x_b = 0.646361$. Linear programs here and in the rest of the paper are generated from MATLAB using YALMIP [32] and solved by IBM ILOG CPLEX's parallel barrier method with 16 software threads [28]. CPU time is 28ms for LPR(n) (20 variables, 82 constraints and bounds) and 51ms for TLPR$(n,1)$ (20 variables, 106 constraints and bounds).

The case studies in Section 7.3 and 7.4 focus on exact product-form solutions and are used to evaluate the proposed methodology against state-of-the-art techniques,

Table 1 two processes cooperating on $A = 2$ action types.

element	value	element	value	element	value
$\boldsymbol{L}_1(1,2)$	1.000000	$\boldsymbol{A}_a(1,1)$	0.312700	$\boldsymbol{A}_b(2,1)$	0.758394
$\boldsymbol{L}_1(2,1)$	0.092800	$\boldsymbol{A}_a(1,2)$	0.012900	$\boldsymbol{A}_b(2,2)$	0.000096
$\boldsymbol{L}_2(1,2)$	0.624292	$\boldsymbol{A}_a(2,1)$	0.384000	$\boldsymbol{A}_b(3,2)$	0.684848
$\boldsymbol{L}_2(2,3)$	0.867884	$\boldsymbol{A}_a(2,2)$	0.644700	$\boldsymbol{A}_b(3,3)$	0.521905
$\boldsymbol{L}_2(3,4)$	0.823686	$\boldsymbol{A}_b(1,1)$	0.180881	$\boldsymbol{A}_b(4,3)$	0.073012
$\boldsymbol{L}_2(4,1)$	0.999997	$\boldsymbol{A}_b(1,4)$	0.574032	$\boldsymbol{A}_b(4,4)$	0.064987



(a) Lower Bound (b) Upper Bound

Fig. 3 Example showing increased tightness of TLPR compared to McCormick's convexification in LPR. The metric is the 2-norm of the error on RC3 at the current iteration of the search algorithm. Note that TLPR finds the product-form at iteration 4, while LPR takes 7 iterations.

namely Buchholz's method [9] and INAP [34]. Conversely, Section 7.5 illustrates the accuracy of rate and structural approximations on two models with resource constraints.

7.2 Example: ZPR

The example in this subsection illustrates certain benefits of the zero-potential relaxation over LPR and TLPR. Consider the toy model studied in [34, Fig. 5] composed of $M = 2$ processes $m = 1$ and $k = 2$ defined over $N_m = 4$ and $N_k = 3$ states. The processes cooperate on actions $a \in \mathcal{A}_k$ and $b \in \mathcal{A}_m$ and are defined by the rate and transitions matrices given in Appendix 10. On this model, all relaxations find a product-form solution associated to the reversed rates $\boldsymbol{x} = (0.70, 1.90)$. LPR requires 14 linear programs to converge to such solution with $\epsilon_{tol} = 10^{-4}$ tolerance. Conversely, ZPR obtains the same solution in just 6 linear programs. Noticeably, at the first iteration ZPR achieves a 2-norm for the residual of RC3 that is achieved by LPR only after 5 linear programs. This provides a qualitative idea of the benefits of ZPR over LPR. Comparatively to TLPR, instead, ZPR offers similar accuracy, in-

cluding in this example where TLPR completes after 5 linear program. However, we have found ZPR numerically more robust than TLPR on several instances.

7.3 Closed Stochastic Model

Next, a challenging model of a closed network comprising three queues, indexed by $k = 1, 2, 3$, that cooperate with a parallel system modeled by the stochastic Petri net shown in Figure 4, indexed by $k = 4$. This Petri net abstracts a generic parallel system where some operations are synchronized between two servers, e.g., mirrored disk drives. The special structure of this model has been shown recently to admit a product-form for certain values of the transition rates [23]. The places P_1 and P_2 receive tokens, representing disk requests, from transitions T_1, T_{12} and T_2. Such transitions are passive, meaning that they are activated by other components. The other transitions are active and fire after exponentially distributed times when their input places all have at least one token. The rates of the underlying exponential distributions are $\sigma_1 = 0.4$ for T_1', $\sigma_2 = 0.1$ for T_2', and $\sigma_{12} = 0.33$ for T_{12}'. Place P_1 receives passively jobs from transition T_1' and actively outputs into T_1 with rate $\mu_1 = 0.5$ (actions 4 and 1, respectively); similarly, place P_2 receives jobs from T_2 and feeds T_2' with rate $\mu_2 = 0.6$ (actions 3 and 6). Similarly, queue $k = 3$ receives from T_{12} and outputs to T_{12}' with rate $\mu_3 = 0.9$ (actions 2 and 5). Thus, $N_k = +\infty$, $k = 1, \ldots, M$, and the model is a cooperation of $M = 4$ infinite processes on $A = 6$ actions. In the RCAT methodology, any cooperating process is considered in isolation with all its (possibly infinite) states, even if part of a closed model. This is consistent with the fact that the specific population in the model affects the computation of the normalizing constant, but not the structure of the product-form solution for a joint state [38].

In addition, note that the model is a pure cooperation, due to the lack of local transitions, having dependency graph

$$\boldsymbol{G} = \begin{pmatrix} 0\,0\,0\,1 \\ 0\,0\,0\,1 \\ 0\,0\,0\,1 \\ 1\,1\,1\,0 \end{pmatrix}$$

Since $\boldsymbol{G}$ has rank $r = 2$ there are $M - r = 2$ degrees of freedom in assigning the reversed rate vector $\boldsymbol{x} = (x_1, x_2, \ldots, x_6)$. Specifically, it is shown in [23] that the following necessary conditions hold for a product-form solution $x_1 = x_4, x_2 = x_5, x_3 = x_6, x_5 = \sigma_{12} x_4 x_6 (\sigma_1 \sigma_2)^{-1}$. We apply our method and the INAP algorithm in [34] to determine a product-form solution of type (1). INAP is a simple fixed-point iteration that starting from a random guess of the vector $\boldsymbol{x}$ progressively refines it until finding a product-form solution. For an action a, the refinement step averages the value of the reversed rates of action a in all states of the active process. Buchholz's method in [9] cannot be used on the present example

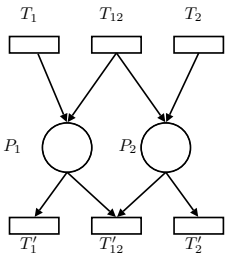


Fig. 4 Petri net process with 6 transitions and 2 places. The process abstracts a system where some operations may be synchronized between servers, e.g., a parallel storage system.

because it does not apply to closed models. For both INAP and our method, we truncate the queue state space to $N_k = 75$ states, the Petri net to $N_4 = 100$ states. Thus, the product-form solution we obtain is valid for closed models with up to $N = 75$ circulating jobs.

Numerical Results. The best performing relaxation on this example is $\mathsf{TLPR}(n, 1)$, which returns after $35.82s$ a solution

$$\boldsymbol{x}^{tlpr} = (0.4023, 0.3323, 0.1004, 0.4014, 0.3315, 0.1003)$$

that matches the RC3 conditions with a tolerance of 10^{-3}. Since the tolerance of the solver is $\epsilon_{tol} = 10^{-4}$, we regard this as an acceptable value considering that $\mathsf{TLPR}(n, 1)$ describes a tight feasible region which may require the LP to apply numerical perturbations. Note that this is a standard feature of modern state-of-the-art LP solvers. $\mathsf{LPR}(n, 1)$ provides a more accurate solution $\boldsymbol{x}^{lpr} = (0.4008, 0.3305, 0.1001, 0.4001, 0.3300, 0.1001)$ but requires 234.879s of CPU time to converge and 124 linear programs. INAP seems instead to suffer a significant loss of accuracy with this parameterization and does not converge. The returned solution after 48.64s and $15,000$ iterations is

$$\boldsymbol{x}^{inap} = (0.3651, 1.0464, 0.6566, 0.3236, 1.0411, 0.4307)$$

which is still quite far from the correct solution, especially concerning the necessary condition $x_3 = x_6$. We have further investigated this problem and observed that, in contrast to our algorithm, INAP ignores ergodicity constraints; hence most of the mass in this example is placed in states near to the truncation border. This appears to be the reason for the failed convergence.

7.4 A G-Network Model

We next consider a generalized, open queueing network, where customers are of positive and negative types, *i.e.*, a G-network [20]. These models enjoy a product-form solution, but this is not generally available in closed-form and requires nu-

Table 2 Reversed rates returned by LPR for the G-network. The indexing is identical to the one in [4].

$x_1 = 1.1615$	$x_2 = 1.7424$	$x_3 = 2.3230$	$x_4 = 0.5806$
$x_5 = 0.1162$	$x_6 = 0.2324$	$x_7 = 0.4646$	$x_8 = 0.2324$
$x_9 = 0.5228$	$x_{10} = 0.8712$	$x_{11} = 0.3486$	$x_{12} = 0.6970$
$x_{13} = 1.6262$	$x_{14} = 0.7317$	$x_{15} = 0.2559$	$x_{16} = 0.0852$
$x_{17} = 0.4268$	$x_{18} = 0.0852$	$x_{19} = 1.9355$	$x_{20} = 0.1215$
$x_{21} = 0.2430$	$x_{22} = 0.0241$	$x_{23} = 0.4709$	$x_{24} = 0.1178$

merical techniques to determine it. Hence, G-networks provide a useful benchmark to compare different approaches for automated product-form analysis. The queue parametrization used in this case study is the one given in [4] for a large model with $M = 10$ queues, $A = 24$ actions. Model parameters are given in Appendix 12. The infinite state space is truncated such that each queue has $N_k = 75$ states. The size of the joint state space for the truncated model is $5.63 \cdot 10^{18}$ states, which is infeasible to solve numerically in the joint process.

Numerical Results. For this model, ENS and QCP fail almost immediately, reporting that the magnitude of the gradient is too small. Conversely, LPR returns the solution in Table 2. Quite interestingly, ZPR returns a different set of reversed rates, but these are found to generate the same equilibrium distributions $\boldsymbol{\pi}_k$ for all processes within the numerical tolerance $\epsilon_{tol} = 10^{-4}$. Thus, this case study again confirms that our approach provides valid answers also in models with multiple solutions. A comparison of the convergence speed of LPR and ZPR is given in Figure 5; TLPR fails in this case due to numerical issues since the feasible region is very tight. We have investigated the problem further and found that the barrier method is responsible for such instabilities, and that switching to the simplex algorithm solves the problem and provides the same product-form solution as LPR.

We now compare our technique against Buchholz's method, applied in finding product-forms of type (1). Buchholz's method involves a quadratic optimization technique that minimizes the residual norm with respect to a product-form solution for the model. This is done without explicitly computing the joint state distribution; hence it is efficient computationally. Comparison with the method proposed here is interesting since Buchholz's method seeks local optima instead of the global optima searched for by AUTOCAT. We have verified that, on small to medium scale models, the method is efficient in finding product-forms. However, the local optimization approach for large models does not guarantee to find a product-form solution when one is known to exist. In particular, we used random initial points, and found that, even though the residual errors are similar to those of the optimum solution of LPR, the specific local optimum returned by Buchholz's method can differ substantially in terms of the global product-form probability distribution. In particular, for some local optima, the marginal probability distribution at a queue is not geometric and the error on performance indices can be very large. This confirms the importance of using global optimization methods, such as the one proposed in this paper, for product-form analysis, especially in large-scale models. Furthermore, we believe

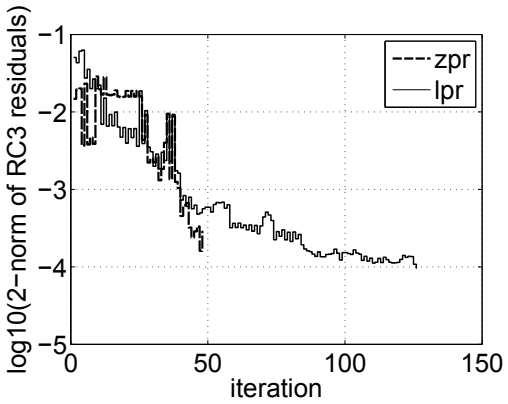


Fig. 5 Convergence speed of LPR and ZPR$(n, 1)$. An iteration corresponds to the solution of a linear program.

that including RC3 in Buchholz's method would help to ensure the geometric structure of the marginal distribution.

7.5 Models with Resource Constraints

Finally, we consider an automated approximation of performance models with resource constraints. We have considered an open queueing network composed of $M = 5$ exponential, first-come first-served queues with finite buffer sizes described by the vector $(B_1, B_2, \ldots, B_M) = (7, 2, +\infty, 3, 10)$. Routing probabilities and model parameters are given in Appendix 13. In particular, arrival rates are chosen so that the equilibrium of the network differs dramatically from that of the corresponding infinite capacity model, where the first queues would be fully saturated. In order to explore the accuracy of rate and structural approximations, we have considered two opposite blocking types: *blocking before service* (BBS) [5], where a job is blocked before entering the server if its target queue is full, and the classical *loss policy* where a job reaching a full station leaves the network. Such policies apply homogeneously to all queues. In both models, we study as the target performance metric the mean queue-length vector $\boldsymbol{n} = (n_1, \ldots, n_M)$ because such values are typically harder to approximate than utilizations as they depend more strongly on the entire marginal probability distributions of the queues.

The BBS model requires structural approximation to improve the accuracy of the initial ALPR rate approximation. In order to adapt the $\boldsymbol{A}_a$ corrections to this specific blocking policy, it is sufficient to delete the term $\alpha_{c,n}\Delta(\boldsymbol{A}_c^{(0)}\mathbf{1})$ from the updating in the structural approximation pseudo-code, thus implying that jobs are not executed while the target station is busy. For this case study, the absolute values of the queue-lengths obtained by simulation are $\boldsymbol{n} = (5.946, 1.262, 0.327, 1.1631, 1.653)$. The

estimates returned by structural approximation converge after the 5th iteration to $\boldsymbol{n}^{sa(5)} = (5.9580, 1.3117, 0.2871, 1.0631, 1.3559)$ with an error on the bottleneck queue of just 0.20%.

For the loss model, we found that queue-lengths are estimated accurately by the ALPR rate approximation alone, after adding hidden transitions to the $\boldsymbol{P}_b$ matrices to correct RC1. In particular, $\boldsymbol{n}^{ra} = (5.0792, 0.9599, 0.2688, 0.5050, 0.5273)$ where the result of the simulation is $\boldsymbol{n} = (5.4877, 1.0642, 0.2766, 0.5248, 0.5536)$ which has an average relative gap of 5.72%.This confirms the quality of the rate approximation in the loss case. Note that both in this case and in the BBS model computational costs are less than 5 minutes.

We remark that we have also tried to apply Buchholz's method to these examples, but similarly to the model of Section 7.4, the technique converges to a local optimum that differs from the simulated equilibrium behavior. Conversely, INAP does not apply to approximate analysis.

7.6 Closed PH Queueing Network

We now describe an example of approximate analysis of closed queueing networks. For illustration purposes, we focus on a machine repairman model comprising a single server first-come first-served queue in tandem with an infinite server station. The same methodology can be used for larger models. The infinite server station has exponentially distributed service times with rate $\mu_2 = 20$ job per second. The queue has phase-type (PH) service times (we point to [8] for an introduction to PH models). The distribution chosen has two states and representation $(\boldsymbol{\alpha}, \boldsymbol{T})$ with initial vector $\alpha = (1, 0)$ and PH subgenerator

$$\boldsymbol{T} = \begin{bmatrix} -1.2705 & 0.0118 \\ 0.0457 & -0.0457 \end{bmatrix}.$$

This PH model generates hyper-exponential service times with mean 0.9996, squared coefficient of variation 9.9846, and skewness 19.6193. With this parameterization, the model is solved for a population $N = 15$ jobs by direct evaluation of the underlying Markov chain obtaining a throughput $X_{ex} = 0.6303$ jobs per second. This is lower than the throughput $X_{pf} = 0.6701$ job per second provided by a corresponding product-form model where the PH service time distribution is replaced by an exponential distribution.

We then approximate the solution of this model by AUTOCAT and study its relative accuracy. In order to cope with the lack of explicit constraints to find feasible reversed rates different from the degenerate ones $\boldsymbol{x} = (x_1, x_2) = \boldsymbol{0}$, we use the following iterative method. Initially, we set $x_1 = X^{pf}$. Based on this educated guess, we run our approximation method based on the LPR formulation to find an approximate value for x_2. This allows the model to be solved after computing numerically the normalizing constant of the equilibrium probabilities, and readily provides

an estimate $X^{(1)}$ for the network throughput. In the following iteration we assign $x_1 = X^{(1)}$ and re-optimize to find a new value of x_2 and corresponding throughput $X^{(2)}$. This iterative scheme is reapplied until convergence is achieved[1].

For the model under study, this approximation provides a sequence of solutions $X_{lpr}^{(1)} = 1.0004$, $X_{lpr}^{(2)} = 0.6291$, $X_{lpr}^{(3)} = 0.6089$, $X_{lpr}^{(4)} = 0.6107$, and $X_{lpr}^{(5)} = 0.6105$ job per second, for a total of 40 solver iterations. The last solution provides a relative error on the exact one of -3.14% compared to the 6.31% error of the product-form approximation, thus reducing the approximation error by about 50%.

We have also compared accuracy with a recent iterative approximation technique for closed networks, inspired by RCAT and proposed in [14, 15]. This technique involves replacing each $-/PH/1$ queueing station by a load-dependent station such that the state probability distribution for a model with M queues is

$$\Pr(n_1, n_2, \ldots, n_M) = G^{-1} \prod_{i=1}^{M} F_i(n_i)$$

where n_i is the number of jobs in queue i, G is a normalization constant, and

$$F_i(n_i) = \begin{cases} 1 - \rho_i & n_i = 0 \\ \rho_i(1-\eta_i)\eta_i^{n_i} & n_i > 0 \end{cases}$$

where η_i is the largest eigenvalue of the rate matrix for the quasi-birth death process obtained by studying the ith station as an open $PH/PH/1$ queue with appropriate input process and utilization ρ_i. We point to [15] for further details on this construction and here we just stress that this particular approximation differs from the AUTOCAT one by using only the slowest decay rate of the queue-length marginal probabilities for such a $PH/PH/1$ queue, whereas in this paper we developed more general approximations that do not resort to asymptotic arguments to simplify the model and that may be applied also to stochastic systems other than queueing networks.

The comparison with the method proposed in [14, 15] reveals that the throughput returned by the approximation is $X = 0.5843$ jobs per second with a relative error of -7.30%. While classes of models exist where it can be shown that this method is far more accurate than the product-form one [15], this example convincingly illustrates a case where the AUTOCAT approximation is the most accurate available.

8 Conclusion

We have introduced an optimization-based approach to product-form analysis of performance models that can be described as a cooperation of Markov processes, e.g. queueing networks and stochastic Petri nets. Our methodology consists of solv-

[1] Note that all test cases did converge, but we no rigorous convergence proof is available.

ing a sequence of linear programming relaxations for a non-convex optimization program that captures a set of sufficient conditions for a product-form. The main limitation of our methodology is that we cannot represent cooperations involving actions that synchronize over more than two processes. However, multiple cooperations are useful only in specialized models, e.g., queueing networks with catastrophes [19]. We believe that such extension is possible, although it may require a sequence of independent product-form search problems to be solved. Hence, computational costs of such solutions should be evaluated for models of practical interest.

Finally, we plan to study the effects of integrating new constraints into the linear programs, such as costs or bounds on the variables that may help in determining a particular reversed rate vector among a set of multiple feasible solutions. For instance, for models which enjoy bounds on their steady-state that may be expressed as linear programs, e.g., stochastic Petri nets [33], this could enable the generation of exact or approximate product-forms that are guaranteed to be within the known theoretical bounds.

References

1. F. A. Al-Khayyal and J. E. Falk. Jointly Constrained Biconvex Programming. *Math. of Oper. Res.*, 8(2):273–286, 1983.
2. A. Argent-Katwala. Automated product-forms with Meercat. *Proc. of SMCTOOLS*, Oct 2006.
3. G. Balbo, S. C. Bruell, and M. Sereno. Product form solution for generalized stochastic Petri nets. *IEEE TSE*, 28(10):915–932, 2002.
4. S. Balsamo, G. Dei Rossi, and A. Marin. A numerical algorithm for the solution of product-form models with infinite state spaces. In *Proc. of EPEW 2010*.
5. S. Balsamo, R. O. Onvural, and V. De Nitto Personé. *Analysis of Queueing Networks with Blocking*. Kluwer Academic, MA, USA, 2001.
6. F. Baskett, K. M. Chandy, R. R. Muntz, and F. G. Palacios. Open, closed, and mixed networks of queues with different classes of customers. *Journal of the ACM*, 22(2):248–260, 1975.
7. D. Bertsimas and J. Tsitsiklis. *Introduction to Linear Optimization*. Athena, 1997.
8. G. Bolch, S. Greiner, H. de Meer, and K. S. Trivedi. *Queueing Networks and Markov Chains*. John Wiley and Sons, 1998.
9. P. Buchholz. Product Form Approximations for Communicating Markov Processes. In *Proc. of QEST*, 135–144. IEEE, 2008.
10. P. Buchholz. Product form approximations for communicating Markov processes. *Perform. Eval.*, 67(9): 797-815, Elsevier, 2010.
11. S. Burer and A. N. Letchford. On nonconvex quadratic programming with box constraints. *SIAM J. on Optimiz.*, 20(2):1073–1089, 2009.
12. X. R. Cao. The relations among potentials, perturbation analysis,and Markov decision processes. *Discr. Event Dyn. Sys.*, 8(1):71–87, 1998.
13. X. R. Cao and D. J. Ma. Performance sensitivity formulae, algorithms and estimates for closed queueing networks with exponential servers. *Perform. Eval.*, 26:181–199, 1996.
14. G.Casale, P.G.Harrison, M.G.Vigliotti. Product-Form Approximation of Queueing Networks with Phase-Type Service. *ACM Perf. Eval. Rev.* 39(4), 2012.
15. G.Casale, P.G.Harrison. A Class of Tractable Models for Run-Time Performance Evaluation In *Proc. of ACM/SPEC ICPE*, 2012.
16. E. de Souza e Silva and P. M. Ochoa. State space exploration in Markov models. In *Proc. of ACM SIGMETRICS*, 152–166, 1992.

17. N. Dijk. *Queueing Networks and Product Forms: A Systems Approach*. John Wiley and Sons, Chichester, 1993.
18. J. M. Fourneau, B. Plateau, and W. Stewart. Product form for stochastic automata networks. In *Proc. of ValueTools*, 1–10, 2007.
19. J. M. Fourneau and F. Quessette. Computing the steady-state distribution of G-networks with synchronized partial flushing. In *Proc. of ISCIS*, 887–896. Springer, 2006.
20. E. Gelenbe. Product-form queueing networks with negative and positive customers. *J. App. Prob.*, 28, 3:656–663, 1991.
21. GNU GLPK 4.8. `http://www.gnu.org/software/glpk/`.
22. P.G. Harrison and T. Lee. Separable equilibrium state probabilities via time reversal in Markovian process algebra. *Theor. Comput. Sci*, 346:161–182, 2005.
23. P.G. Harrison and C. Llado. A PMIF with Petri net building blocks. In *Proc. of ICPE*, 2011.
24. P. G. Harrison. Turning back time in Markovian process algebra. *Theor. Comput. Sci*, 290(3):1947–1986, 2003.
25. P. G. Harrison. Reversed processes, product forms and a non-product form. *Linear Algebra and Its Applications*, 386:359–381, July 2004.
26. P. G. Harrison, J. Hillston. Exploiting quasi-reversible structures in Markovian process algebra models. *Computer J.*, 38(7):510–520, 1995.
27. J. Hillston. A compositional approach to performance modelling, University of Edinburgh, Ph.D. Thesis, 1994.
28. IBM ILOG CPLEX 12.0 User's Manual, 2010.
29. J. R. J. Jackson. Jobshop-like queueing systems. *Management Science*, 10(1):131–142, 1963.
30. F. P. Kelly. Networks of queues with customers of different types. *J. of App. Prob.*, 12(3):542–554, 1975.
31. F. P. Kelly. *Reversibility and Stochastic Networks*. John Wiley & Sons, New York, 1979.
32. J. Löfberg. YALMIP : A toolbox for modeling and optimization in MATLAB. In *Proc. of CACSD*, 2004.
33. Z. Liu. Performance Analysis of Stochastic Timed Petri Nets Using Linear Programming Approach. *IEEE TSE*, 11(24):1014–1030, 1998.
34. A. Marin and S. R. Bulò. A general algorithm to compute the steady-state solution of product-form cooperating Markov chains. In *Proc. of MASCOTS*, 1–10, 2009.
35. A. Marin and M. G. Vigliotti. A general result for deriving product-form solutions in Markovian models. In *Proc. of ICPE*, 165–176, 2010.
36. G. P. McCormick. Computability of global solutions to factorable nonconvex programs. *Math. Progr.*, 10:146–175, 1976.
37. R. R. Muntz. Poisson departure processes and queueing networks. Tech. Rep. RC 4145, IBM T. J. Watson Research Center, N.Y., 1972.
38. R. D. Nelson. The mathematics of product form queuing networks. *ACM Comp. Surveys*, 25(3):339–369, 1993.
39. J. Nocedal and S. J. Wright. Numerical Optimization. Springer-Verlag, 1999.
40. Y. Saad. Iterative Methods for Sparse Linear Systems. SIAM, 2000.
41. B. Plateau. On the stochastic structure of parallelism and synchronization models for distributed algorithms. *SIGMETRICS*, 147–154, 1985.

APPENDIX

9 Infinite Processes

Numerical optimization techniques generally require matrices of finite size. In both ENS and its relaxations, we therefore used exact or approximate aggregations to truncate the state spaces of any infinite processes. Let $C+1$ be the maximum acceptable matrix order. Then we decompose the generator matrix and its equilibrium probability vector of an infinite process k as

$$\boldsymbol{Q}_k = \begin{bmatrix} \boldsymbol{Q}_k^{C,C} & \boldsymbol{Q}_k^{C,\infty} \\ \boldsymbol{Q}_k^{\infty,C} & \boldsymbol{Q}_k^{\infty,\infty} \end{bmatrix}, \quad \boldsymbol{\pi}_k = \left[\boldsymbol{\pi}_k^{C}, \boldsymbol{\pi}_k^{\infty}\right],$$

where $\boldsymbol{Q}_k^{C_1,C_2}$ is a $C_1 \times C_2$ matrix. Similar partitionings are also applied to the transition matrix $\boldsymbol{L}_k$ and to the rate matrices $\boldsymbol{A}_a$ and $\boldsymbol{P}_b$, $a \in \mathcal{A}_k, b \in \mathcal{P}_k$. We define the truncation such that the total probability mass in the first C states is 1 relative to the numerical tolerance of the optimizer, *i.e.* $\boldsymbol{\pi}_k^{\infty}\mathbf{1} < \epsilon_{tol}$. Notice that the latter condition can also be used to determine ergodicity of the infinite process. Furthermore, from condition RC1 (respectively RC2) we need to account for the cases where passive (respectively active) actions associated with the first C states are only enabled in $\boldsymbol{P}_k^{C,\infty}$ (respectively only incoming from $\boldsymbol{A}_k^{\infty,C}$). Such problems are easily handled by adding one fictitious state to the truncated set $\{1, 2, \ldots, C\}$. For example, for $\boldsymbol{A}_a$ and $\boldsymbol{P}_b$, we consider the truncated matrices

$$\boldsymbol{A}_a = \begin{bmatrix} \boldsymbol{A}_a^{C,C} & \boldsymbol{A}_a^{C,\infty}\mathbf{1} \\ \mathbf{1}^T\boldsymbol{A}_a^{\infty,C} & 0 \end{bmatrix}, \; \boldsymbol{P}_b = \begin{bmatrix} \boldsymbol{P}_b^{C,C} & \boldsymbol{P}_b^{C,\infty}\mathbf{1} \\ \mathbf{1}^T\boldsymbol{P}_b^{\infty,C} & 0 \end{bmatrix},$$

where $\mathbf{1}$ is now an infinite column of ones. Note that the fictitious state is excluded from the validation of the conditions RC1 and RC2, thus the value of the diagonal rate on the last row is irrelevant with respect to finding a product-form.

Finally, we comment on the choice of the parameter C for a given process k. Since this determines the number of states N_k for the truncated process, an optimal choice of this value can provide substantial computational savings. Let us first note that starting from a small C, it is easy to integrate additional constraints or potential vectors in the linear formulations for a value $C' > C$. Recall that we propose in the rest of the paper a sequence of linear programs in order to obtain a feasible solution $\boldsymbol{x}$. Then, if a linear program is infeasible, this can be either due to a lack of a product-form or due to a truncation where C is too small. The latter case can be readily diagnosed by adding slack variables similarly to the ones in QCP to the ergodicity condition and verifying if this is sufficient to restore feasibility. In such a case, the C value is updated to the smallest value such that feasibility is restored in the main linear program.

10 ZPR Example Model

$$\boldsymbol{A}_a = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0.2170 \\ 2.9105 & 2.2575 & 0 \end{bmatrix} \qquad \boldsymbol{P}_a = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{A}_b = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 5.65 & 0 & 0.52 & 2.13 \\ 0 & 7.00 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad \boldsymbol{P}_b = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{L}_m = \begin{bmatrix} 0 & 8 & 0 & 3 \\ 6.15 & 0 & 8.28 & 7.67 \\ 15 & 9.70 & 0 & 0 \\ 16 & 0 & 0 & 0 \end{bmatrix} \qquad \boldsymbol{L}_k = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 3.78 \\ 3.09 & 2.74 & 0 \end{bmatrix}$$

11 Structural approximation pseudo-code

Input: RLX$\in$ {ALPR, ATLPR, AZPR}, $\boldsymbol{L}_k$, $\boldsymbol{A}_a$, $\boldsymbol{P}_b$, $\mathcal{A}_k$, $\mathcal{P}_k$, $\forall k$, a; **Output**: $\boldsymbol{x}$, $\boldsymbol{\pi}_k$, $\boldsymbol{Q}_k$, $\forall k$
ignore RC1 and RC2, get approximate product-form solution $\boldsymbol{x}^{(0)}$ by RLX
for $k = 1, \ldots, M$ `/* correct RC1 and RC2 */`
  for all $a \in \mathcal{A}_k$ **do** $\boldsymbol{A}_a(j,j) = x_a^{(0)}, \forall j \in \mathcal{J}_b,\ \mathcal{J}_b = \{j \mid \sum_i \boldsymbol{A}_a[i,j] = 0\}$ **end for all**
  for all $b \in \mathcal{P}_k$ **do** $\boldsymbol{P}_b(i,i) = 1,\ \forall i \in \mathcal{I}_b,\ \mathcal{I}_b = \{i \mid \sum_j \boldsymbol{P}_b[i,j] = 0\}$ **end for all**
$\alpha_{c,0} = 1$; $\boldsymbol{A}_c^{(0)} = \alpha_{c,0}\boldsymbol{A}_c, \quad c = 1, 2, \ldots, A$;
$\beta_{c,0} = 1$; $\boldsymbol{P}_c^{(0)} = \beta_{c,0}\boldsymbol{P}_c, \quad c = 1, 2, \ldots, A$;
while current iteration number $n \geq 1$ is less than the maximum number of iterations
  get by RLX an approximate product-form solution $\boldsymbol{x}^{(n)}$ for $\boldsymbol{L}_k$, $\boldsymbol{A}_a^{(n)}$, $\boldsymbol{P}_b^{(n)}$
  for $c = 1, \ldots, A$, where $c \in \mathcal{A}_k$ and $c \in \mathcal{P}_m$ `/* update blocking probabilities */`
    $\alpha_{c,n} = \sum_{i \in \mathcal{I}_c} \boldsymbol{\pi}_m(\boldsymbol{x}^{(n)})[i]$; $\boldsymbol{A}_c^{(n)} = (1 - \alpha_{c,n})\boldsymbol{A}_c^{(0)} + \alpha_{c,n}\Delta(\boldsymbol{A}_c^{(0)}\mathbf{1})$
    $\beta_{c,n} = \sum_{j \in \mathcal{J}_c} \boldsymbol{\pi}_k(\boldsymbol{x}^{(n)})[j]$; $\boldsymbol{P}_c^{(n)} = (1 - \beta_{c,n})\boldsymbol{P}_c^{(0)} + \beta_{c,n}\Delta(\boldsymbol{P}_c^{(0)}\mathbf{1})$
  end for
  if $\max_c(||\alpha_{c,n} - \alpha_{c,n-1}||_2, ||\beta_{c,n} - \beta_{c,n-1}||_2) \leq \epsilon_{tol}$ **return** $\boldsymbol{x}^{(n)}$, $\boldsymbol{\pi}_k(\boldsymbol{x}^{(n)})$, $\boldsymbol{Q}_k(\boldsymbol{x}^{(n)})$
end while
return $\boldsymbol{x}^{(n)}$, $\boldsymbol{\pi}_k(\boldsymbol{x}^{(n)})$, $\boldsymbol{Q}_k(\boldsymbol{x}^{(n)})$

12 G-Network Case Study

We report the parameters for the G-network given in [4]. The network consists of $M = 10$ queues with exponentially distributed service times having rates $\mu_1 = 4.5$ and $\mu_i = 4.0 + (0.1)i$ for $i \in [2, 10]$. The external arrival rate defines a Poisson process with rate $\lambda = 5.0$. The routing matrix for (positive) customers has in row i and column j the probability $r^+_{i,j}$ of a (positive) customer being routed to queue j, as a positive customer, upon leaving queue i. In this case study, this routing matrix is given by

$$\boldsymbol{R}^+ = [r^+_{i,j}] = \begin{bmatrix} 0 & .2 & .3 & .4 & 0 & 0 & 0 & 0 & 0 & 0 \\ .1 & 0 & 0 & 0 & .2 & 0 & 0 & .2 & 0 & 0 \\ 0 & 0 & 0 & 0 & .3 & .5 & .2 & 0 & 0 & 0 \\ .3 & 0 & 0 & 0 & 0 & 0 & .7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & .3 & 0 & .5 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & .2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Conversely, the probability $r^-_{i,j}$ of a customer leaving queue i and becoming a negative signal upon arrival at queue j is

$$\boldsymbol{R}^- = [r^-_{i,j}] = \begin{bmatrix} 0 & 0 & 0 & 0 & .1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & .4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & .1 & 0 & .1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & .1 & 0 & 0 & 0 & 0 & 0 \\ .1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & .2 & 0 & .05 & 0 \end{bmatrix}$$

13 Loss and BBS Models

The model is composed of $M = 5$ queues that cooperate on a set of $A = 12$ actions, one for each possible job movement from and inside the network. The routing probabilities $R[k, j]$ from queue k to queue j are as follows:

$$R = \begin{bmatrix} 0.16 & 0 & 0.04 & 0.50 & 0.30 \\ 0.08 & 0.29 & 0.02 & 0.08 & 0.52 \\ 0 & 0 & 0.78 & 0 & 0 \\ 0.29 & 0.24 & 0 & 0.25 & 0.22 \\ 0 & 0.49 & 0 & 0.20 & 0 \end{bmatrix}$$

Service times are exponential at all queues with rates $mu_k = k$, $k =, 1 \ldots, 5$. For a queue k, the probability of departing from the network is $r_{k,0} = 1 - \sum_{j=1}^{5} R[k, j]$. The Poisson arrival rates from the outside worlds are given by the vector:

$$\boldsymbol{\lambda} = (0.6600, 0.1500, 0.0750, 0.1650, 0.4500).$$