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

Andrea Marin, Samuel Rota Bulò

Università Ca' Foscari di Venezia
Dipartimento di Informatica
Italy

2009

Presentation outline

1. Introduction
   - Product-form models
   - Sketch of the results
   - RCAT by example
2. Algorithm definition
   - Two agents: the simplest case
   - Non-optimized algorithm
3. Examples
   - Jackson queueing networks
   - G-networks
   - Other cases
4. Conclusion

Markov process: steady state analysis

- Steady-state analysis: analysis of the system (if possible) when $t \to \infty$
- Γ: state space
- q_{ij}: transition rate from states i to j, $i \neq j$, $i, j \in \Gamma$. Let $\mathbf{Q} = [q_{ij}]$ with $q_{ii} = -\sum_{j \neq i} q_{ij}$
- $\pi(i)$: probability of observing state i when $t \to \infty$ (limiting distribution), $\boldsymbol{\pi} = [\pi(i)]$

Theorem (Stationary distribution)

If the CTMC is ergodic the limiting distribution is unique and independent of the initial state. The stationary distribution is given by:

$$\boldsymbol{\pi}\mathbf{Q} = \mathbf{0} \ \wedge \ \boldsymbol{\pi}\mathbf{1} = 1$$

Compositionality and steady state analysis: product-form

- Consider model S consisting of sub-models $S_1, \ldots, S_N$
- Let $m = (m_1, \ldots, m_N)$ be a state of model S and $\pi(m)$ its steady state probability
- S is in product-form with respect to $S_1, \ldots, S_N$ if:

$$\pi(m) \propto g_1(m_1) \cdot g_2(m_2) \cdots g_N(m_N)$$

  where $g_i(m_i)$ is the steady state probability distribution of S_i appropriately parametrised
- The cardinality of the state space of S is proportional to the product of the state space cardinalities of its sub-models $\Rightarrow$ product-form models can be studied more efficiently!

Some problems. . .

- How to decide if a model yields a product-form solution?
  - list of instances: BCMP theorem, Coleman/Henderson Stochastic Petri Nets, G-networks. . .
  - general criteria: Markov implies Markov property, RCAT (and extensions),. . .
- How to find the correct parametrisation for the sub-models?
  - solving the linear system of traffic equations for BCMP/Jackson queueing networks
  - solving the non-linear traffic equations for G-networks
  - solving the rate-equations for RCAT

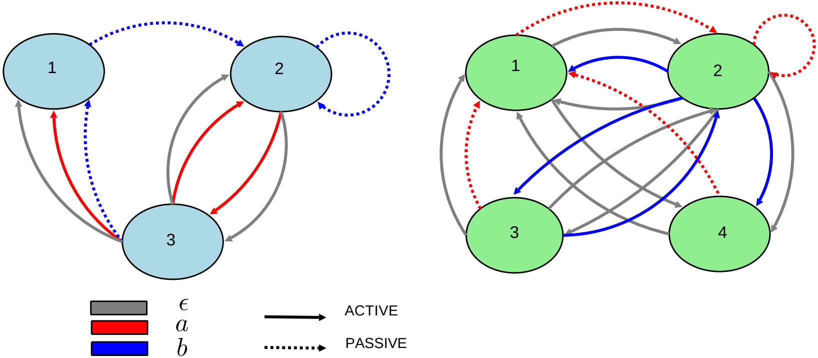
Sketch of the results

We propose an algorithm that...

- decides if a model S has a product-form solution with respect to a set of sub-models $S_1, \ldots, S_N$
- derive the correct parametrisation for the sub-models
- compute the unnormalised steady-state solution

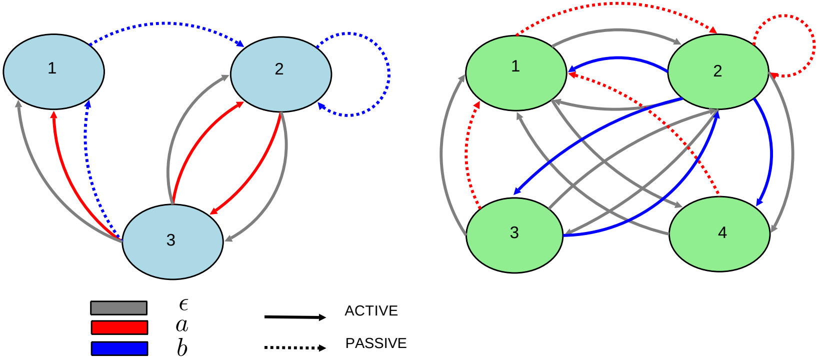
The algorithm is based on the Reversed Compound Agent Theorem (RCAT) [Harrison, 2003] and its extensions.

Pairwise interacting agents



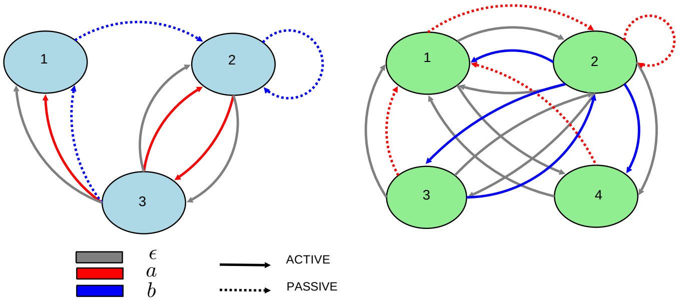
- PEPA-like cooperation with an active and a passive agent
- Active transitions have a rate
- Passive transitions have a unspecified rate
- Active/Passive transitions occur only simultaneously with the rate of the active ones

RCAT conditions



- Passive transitions enabled in every state
- Active transitions incoming in every state
- Same reversed rate for all active transitions

RCAT parametrisation and solution



- Parametrisation: replace all the passive transitions with the reversed rate of the corresponding active transitions
- Solution: let g_1 and g_2 be the solution of the parametrised agents, then the solution π of the cooperating agent is in product-form:

$$\pi \propto g_1 \cdot g_2$$

Intuition of the iterative scheme

The algorithm steps:

1. guess g_1 and g_2
2. use g_1 to parametrise g_2
3. use g_2 to parametrise g_1
4. compute the new g_1 and g_2
5. test RCAT conditions
   - Satisfied? $\Rightarrow$ END
   - Not satisfied? $\Rightarrow$ STEP 2

- How do we compute the reversed rate to parametrise the other agent?
- What if the model is *not* in product-form?
- Does the iterative scheme always converge?

The re-parametrisation phase

What do we have?

- An hypothetical steady-state distribution g_i, $i = 1, 2$

What do we need to compute?

- The reversed rates of all the active transitions
  - Let j, k be states of agent i
  - Assume an active transition from j to k with rate r
  - Its reversed rate is $g(j)/g(k) \cdot r$

What happens if the reversed rates are different?

- There could still be product-form (remember g_i is hypothetical!)
- We need to compute *one* rate to re-parametrise the other agent
- We use the mean of the computed reversed rates as the parameter

Computing new g_i and convergence

Computing g_i and product-forms

- g_i are computed using the global balance equation system
- If g_i at step n are identical to g_i at step $n-1$
  - Constant reversed rates for active transitions $\Rightarrow$ product-form solution found
  - Otherwise $\Rightarrow$ no product-form found

Convergence

- Convergence has been proved for specific cases
- A maximum number of iterations is used to avoid infinite loops

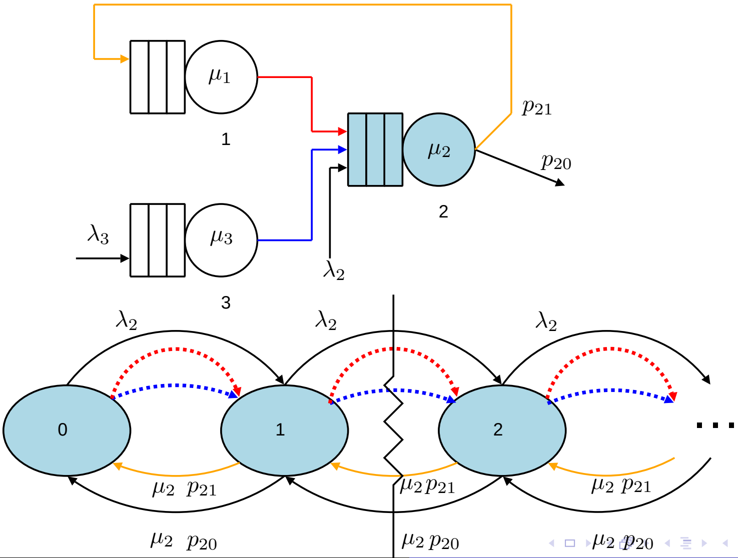
Algorithm definition: notation

- N: number of agents
- $\mathcal{S}_k$, $1 \leq k \leq N$: state space of agent k
- α, β, γ: states of an agent
- $\lambda_k(a, \alpha, \beta)$: in agent k, the rate of the active transition from state α to state β labelled by a
- g_k, $1 \leq k \leq N$: hypothetical stationary distribution of agent k
- n: number of iterations performed
- g_k^{prev}: stationary distribution computed at step $n-1$
- M: maximum number of iterations
- ϵ: tolerance

Non-optimized algorithm

Randomly initialize g_k for all $k = 1, \ldots, N$
$n = 0$
repeat
  for $j, k = 1, \ldots, N$ **do**
    foreach $a \in (\mathcal{P}_j \cap \mathcal{A}_k)$ **do**
      $\Lambda \leftarrow \left\{ \lambda_k(a, \alpha, \beta) \frac{\pi_k(\alpha)}{\pi_k(\beta)} \; : \; \alpha, \beta \in \mathcal{S}_k \right\}$
      foreach $\alpha, \beta \in \mathcal{S}_j \; : \; \lambda_j(a, \alpha, \beta) > 0$ **do**
        $\lambda_j(a, \alpha, \beta) \leftarrow \text{mean}(\Lambda)$
  Update g_k for all $k = 1, \ldots, N$
  $n \leftarrow n + 1$
until $n > M$ **or** $\forall k = 1, \ldots, N. \; \|g_k - g_k^{prev}\| < \epsilon$;
if *the reversed rates are not constant* **then**
  fail: MARCAT product-form not identified
return $\{g_k\}_{k=1,\ldots,N}$

Modelling Jackson QNs



Application of the algorithm

- We use just two states 0 and 1
- Without loss of generality assume $g_i(0) = 1$
- Each iteration computes:

$$g_k(1)^{(n+1)} = \frac{\sum_{\ell=1}^{N} x_{\ell k}^{(n)} + \gamma_k}{\mu_k}$$

  where: $x_{\ell k}^{(n)} = g_\ell(1)^{(n)} \mu_\ell p_{\ell k}$
- Traffic equation system of the Jackson Theorem:

$$e_i = \gamma_i + \sum_{\ell=1}^{N} e_\ell p_{\ell i}$$

- The iteration scheme is the Jacobi algorithm applied to the traffic equations
  - the matrix of coefficient is irreducibly diagonally dominant $\Rightarrow$ the scheme always converges

G-networks: Modelling and algorithm application: results

- The modelling technique is analogous to that of Jackson QN
- The iteration scheme becomes:

$$g_k(1)^{(n+1)} = \frac{\gamma^+ + \sum_{\ell=1}^{N} x_{\ell k}^{+(n)}}{\mu_k + \gamma_k^- + \sum_{\ell=1}^{N} x_{\ell k}^{-(n)}},$$

where:

  - $x_{\ell k}^{+(n)} = \pi_\ell(1)^{(n)} \mu_\ell p_{\ell k}^+$
  - $x_{\ell k}^{-(n)} = \pi_\ell(1)^{(n)} \mu_\ell p_{\ell k}^-$
  - γ_k^+ and γ_k^-: positive and negative arrival rates to G-queue k
  - $p_{\ell k}^+$ ($p_{\ell k}^-$): pr. of joining G-queue k as positive (negative) customer after being served by G-queue ℓ

- The scheme is identical to the well-know iterative scheme for the computation of the steady-state distribution of G-networks

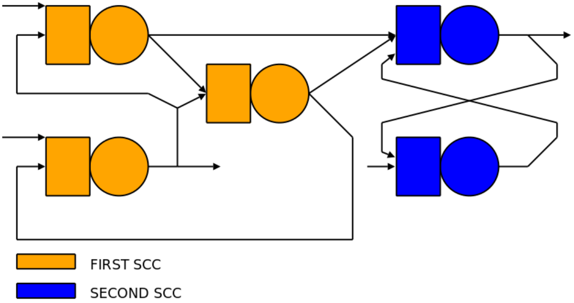
Random instances

- We developed a generator of random instances of agents with finite state spaces whose cooperation is in product-form
- Tested over 100 agents with 100 states each
- The algorithm has always converged to the correct solution
- The convergence speed is fast (always less that 20 iterations)
- New positive tests have been done with product-form QN with blocking

Optimizations

Provided optimizations:

- Active self-loop
  - The reversed rate of this transition is equal to its forward rate
- Parallel computation of g_k
- Compute the strong connected components $\Rightarrow$ Use of Tarjan algorithm to define the order of solution



Conclusions

Properties of the algorithm:

- Iterative algorithm to decide and compute the product-form solution of interacting agents
- Based on the Reversed Compound Agent Theorem
- It is not necessary to derive the traffic equations or to perform symbolic computations
- Complexity $O(INn^3)$ with:
  - I: number of iterations
  - N: number of agents
  - n: number of states of an agent
- Easy implementation
- Convergence proved only for special cases

Future works

- Extending the algorithm to deal with more complex models
  - e.g. G-networks with partial flushing (Fourneau [year])
- Proving the convergence for a larger model class
- Implementing the algorithm within a user-friendly tool
  - Work in progress...

Thanks for the attention