Accurate Performance Estimation for Stochastic Marked Graphs by Bottleneck Regrowing

Ricardo J. Rodríguez and Jorge Júlvez

{rjrodriguez, julvez}@unizar.es

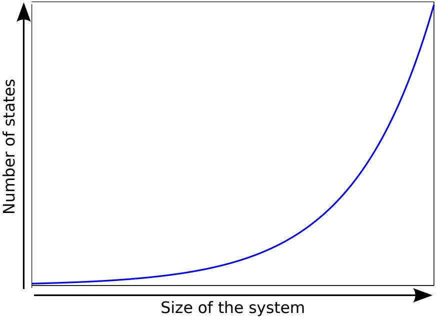
Universidad de Zaragoza
Zaragoza, Spain

September 24^{th}, 2010

EPEW'10: 7^{th} European Performance Engineering Workshop
Bertinoro, Italy

Motivation (I): the need of requirement verification

- New system: problem of verification of requirements
- Performance of an industrial system → real need
- Many systems modelled as Discrete Event Systems (DES)
- Increasing size → exact performance computation unfeasible
  - State explosion problem



Motivation (II): performance evaluation approaches

- Exact analytical measures
  - Need exhaustive state space exploration
- Performance bounds: overcoming state explosion problem
  - Reduced running time, BUT how good (i.e., accurate) is the bound?

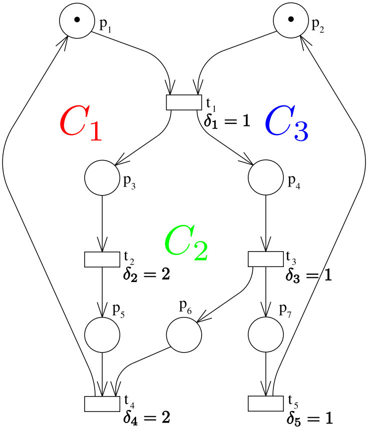
Motivation (II): performance evaluation approaches

- Exact analytical measures
  - Need exhaustive state space exploration
- Performance bounds: overcoming state explosion problem
  - Reduced running time, BUT how good (i.e., accurate) is the bound?

Our approach

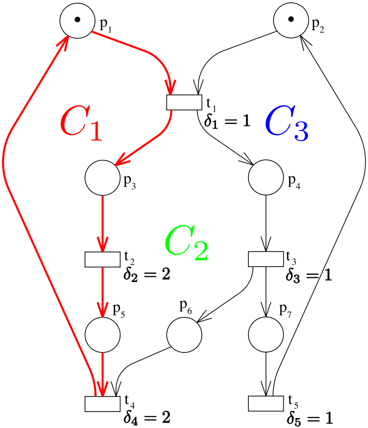
- Iterative algorithm
- Sharper (i.e., closer) bounds
  1. Initial bottleneck cycle (most restrictive)
  2. Add set of places likely to constraint
- Outputs:
  - Improved performance bound
  - New bottleneck

Motivation (III): a small example



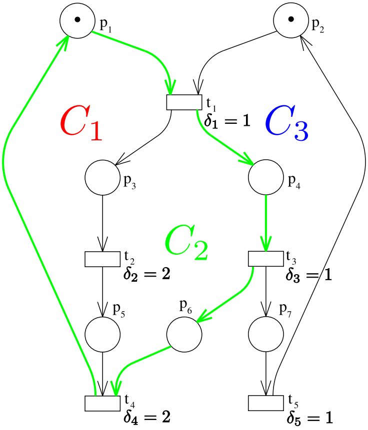
- $r_{C_1} = \frac{1}{5}$, $r_{C_2} = \frac{1}{4}$ and $r_{C_3} = \frac{1}{3}$
- Bottleneck cycle $\rightarrow$ minimum ratio
- Throughput bound: $\frac{1}{5} = 0.2$
- Lowest ratio token/delay $\rightarrow \{p_1, p_4, p_6\}$
- New thr bound: 0.1875 (6.25% lower)
- Seek next constraint cycle non trivial
- *Tight* marking and slack

Motivation (III): a small example



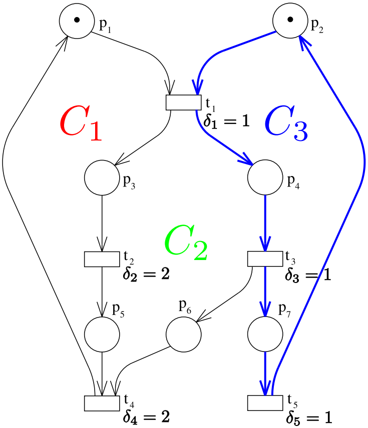
- $r_{C_1} = \frac{1}{5}$, $r_{C_2} = \frac{1}{4}$ and $r_{C_3} = \frac{1}{3}$
- Bottleneck cycle $\rightarrow$ minimum ratio
- Throughput bound: $\frac{1}{5} = 0.2$
- Lowest ratio token/delay $\rightarrow \{p_1, p_4, p_6\}$
- New thr bound: 0.1875 (6.25% lower)
- Seek next constraint cycle non trivial
- *Tight* marking and slack

Motivation (III): a small example



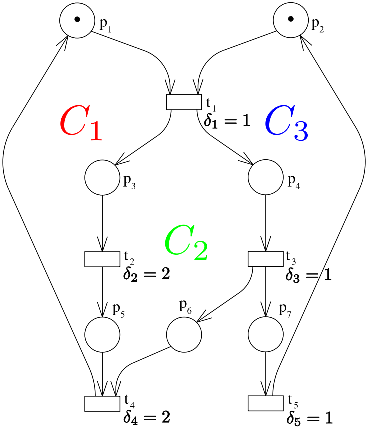
- $r_{C_1} = \frac{1}{5}$, $r_{C_2} = \frac{1}{4}$ and $r_{C_3} = \frac{1}{3}$
- Bottleneck cycle $\rightarrow$ minimum ratio
- Throughput bound: $\frac{1}{5} = 0.2$
- Lowest ratio token/delay $\rightarrow \{p_1, p_4, p_6\}$
- New thr bound: 0.1875 (6.25% lower)
- Seek next constraint cycle non trivial
- *Tight* marking and slack

Motivation (III): a small example



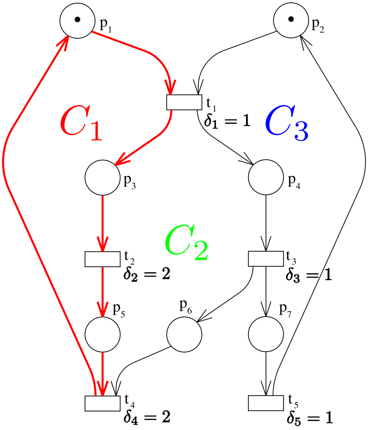
- $r_{C_1} = \frac{1}{5}$, $r_{C_2} = \frac{1}{4}$ and $r_{C_3} = \frac{1}{3}$
- Bottleneck cycle $\rightarrow$ minimum ratio
- Throughput bound: $\frac{1}{5} = 0.2$
- Lowest ratio token/delay $\rightarrow \{p_1, p_4, p_6\}$
- New thr bound: 0.1875 (6.25% lower)
- Seek next constraint cycle non trivial
- *Tight* marking and slack

Motivation (III): a small example



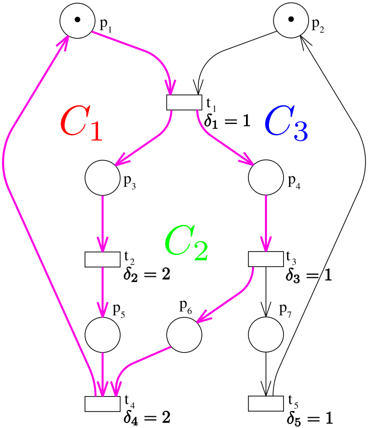
- $r_{C_1} = \frac{1}{5}$, $r_{C_2} = \frac{1}{4}$ and $r_{C_3} = \frac{1}{3}$
- Bottleneck cycle $\rightarrow$ minimum ratio
- Throughput bound: $\frac{1}{5} = 0.2$
- Lowest ratio token/delay $\rightarrow \{p_1, p_4, p_6\}$
- New thr bound: 0.1875 (6.25% lower)
- Seek next constraint cycle non trivial
- *Tight* marking and slack

Motivation (III): a small example



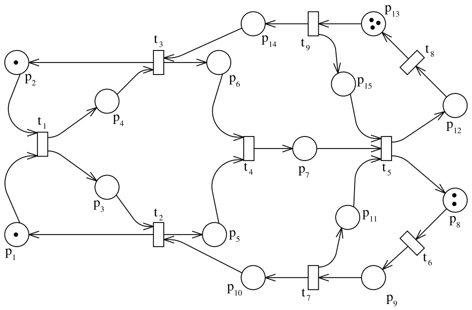
- $r_{C_1} = \frac{1}{5}$, $r_{C_2} = \frac{1}{4}$ and $r_{C_3} = \frac{1}{3}$
- Bottleneck cycle → minimum ratio
- Throughput bound: $\frac{1}{5} = 0.2$
- Lowest ratio token/delay → $\{p_1, p_4, p_6\}$
- New thr bound: 0.1875 (6.25% lower)
- Seek next constraint cycle non trivial
- *Tight* marking and slack

Motivation (III): a small example



- $r_{C_1} = \frac{1}{5}$, $r_{C_2} = \frac{1}{4}$ and $r_{C_3} = \frac{1}{3}$
- Bottleneck cycle $\rightarrow$ minimum ratio
- Throughput bound: $\frac{1}{5} = 0.2$
- Lowest ratio token/delay $\rightarrow \{p_1, p_4, p_6\}$
- New thr bound: 0.1875 (6.25% lower)
- Seek next constraint cycle non trivial
- *Tight* marking and slack

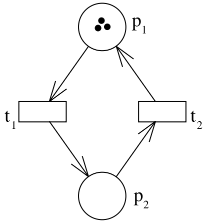
Motivation (IV): running example



- Performance bound 12.9% lower than the initial one
  - More iterations: a bound just 0.3% greater than the real performance
- Benefits of the proposed method:
  - Efficient (uses linear programming)
  - Accurate (converges in few iterations)

Some basic concepts (I): Stochastic Marked Graph

- Petri Net system: $\mathcal{S} = \langle P, T, \mathbf{Pre}, \mathbf{Post}, \mathbf{m_0} \rangle$
- Marked graph (MG): ordinary PN such that each place has exactly one input and exactly one output arc
- *Stochastic Marked Graph* (SMG): MG and a vector δ, where $\delta(t)$ is the mean of the exponential firing time distribution associated to each transition $t \in T$
- SMG's transitions work under *infinite* server semantics (assumed)
- Steady state throughput χ: average number of firing counts per u.t.



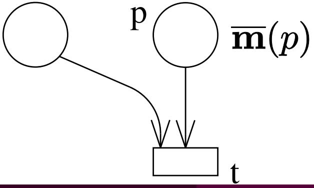
Some basic concepts (II): Critical Cycle (1)

Little's law

- Average number of customers L in a queue: $L = \lambda \cdot W$
- In a SMG: each pair $\{p, t\}$, where $p^\bullet = \{t\}$, can be seen as a simple queueing system

$$\overline{\mathbf{m}}(p) = \chi(p^\bullet) \cdot \overline{\mathbf{s}}(p) \qquad (1)$$

- $\overline{\mathbf{s}}(p)$ =average waiting time + average service time ($\delta(p^\bullet)$ in our case) $\rightarrow \delta(p^\bullet) \leq \overline{\mathbf{s}}(p)$



$$\overline{\mathbf{m}}(p) \geq \chi(p^\bullet) \cdot \delta(p^\bullet) \qquad (2)$$

Some basic concepts (II): Critical Cycle (2)

- Note that MGs have a single minimal t-semiflow equal to **1** $\rightarrow$ same steady state throughput for every transition

$$\textit{Maximize } \Theta :$$

$$\hat{\mathbf{m}}(p) \geq \delta(p^\bullet) \cdot \Theta \quad \forall p \in P \tag{3a}$$

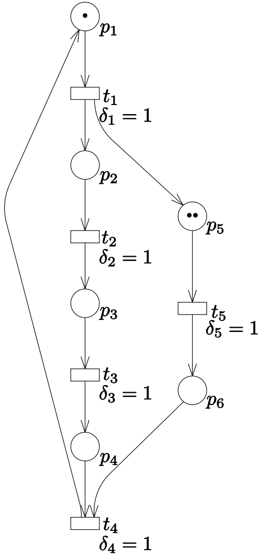
$$\hat{\mathbf{m}} = \mathbf{m_0} + \mathbf{C} \cdot \sigma \tag{3b}$$

$$\sigma \geq 0 \tag{3c}$$

- Θ is an upper throughput bound

Campos, J. Performance Bounds. *Performance Models for Discrete Event Systems with Synchronizations: Formalisms and Analysis Techniques*, Ed. KRONOS, 1998

Some basic concepts (II): Critical Cycle (3)

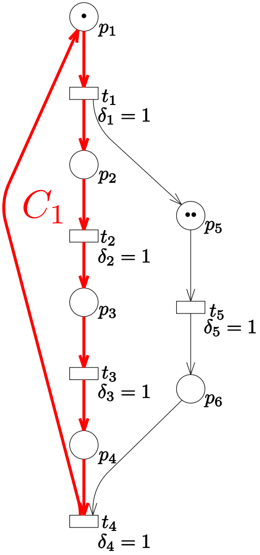


Concept of *slack*: μ

$$\hat{\mathbf{m}}(p) \geq \delta(p^\bullet) \cdot \Theta \longrightarrow \mathbf{m}(p) = \delta(p^\bullet) \cdot \Theta + \mu(p)$$

- $\mu(p) = 0$ if p belongs to critical cycle
- Value of vector μ will depend on the algorithm used by the LP solver
- The lower the slack, the higher the probability that place will constraint

Some basic concepts (II): Critical Cycle (3)

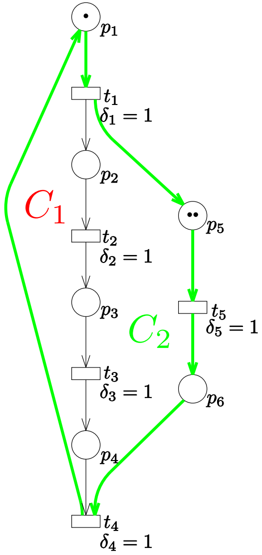


Concept of *slack*: μ

$$\hat{\mathbf{m}}(p) \geq \delta(p^\bullet)\cdot\Theta \longrightarrow \mathbf{m}(p) = \delta(p^\bullet)\cdot\Theta + \mu(p)$$

- $\mu(p) = 0$ if p belongs to critical cycle
- Value of vector μ will depend on the algorithm used by the LP solver
- The lower the slack, the higher the probability that place will constraint

Some basic concepts (II): Critical Cycle (3)

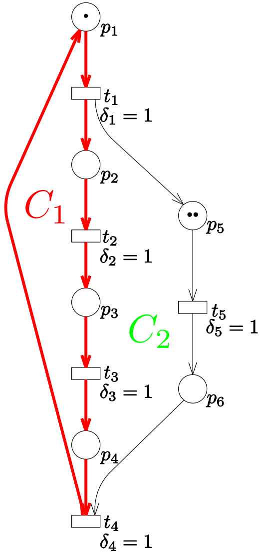


Concept of *slack*: μ

$$\hat{\mathbf{m}}(p) \geq \delta(p^\bullet)\cdot\Theta \longrightarrow \mathbf{m}(p) = \delta(p^\bullet)\cdot\Theta + \mu(p)$$

- $\mu(p) = 0$ if p belongs to critical cycle
- Value of vector μ will depend on the algorithm used by the LP solver
- The lower the slack, the higher the probability that place will constraint

Some basic concepts (II): Critical Cycle (3)

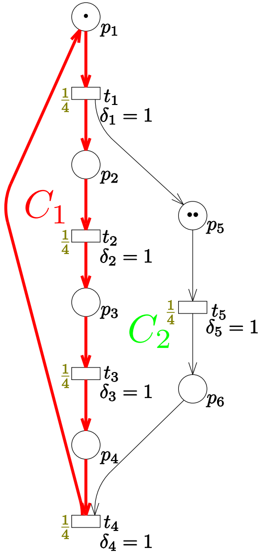


Concept of *slack*: μ

$$\hat{\mathbf{m}}(p) \geq \delta(p^\bullet) \cdot \Theta \longrightarrow \mathbf{m}(p) = \delta(p^\bullet) \cdot \Theta + \mu(p)$$

- $\mu(p) = 0$ if p belongs to critical cycle
- Value of vector μ will depend on the algorithm used by the LP solver
- The lower the slack, the higher the probability that place will constraint

Some basic concepts (II): Critical Cycle (3)

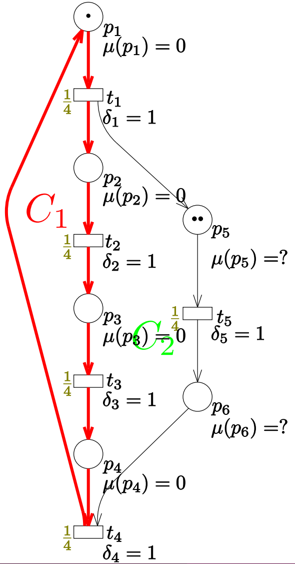


Concept of *slack*: μ

$$\hat{\mathbf{m}}(p) \geq \delta(p^\bullet) \cdot \Theta \longrightarrow \mathbf{m}(p) = \delta(p^\bullet) \cdot \Theta + \mu(p)$$

- $\mu(p) = 0$ if p belongs to critical cycle
- Value of vector μ will depend on the algorithm used by the LP solver
- The lower the slack, the higher the probability that place will constraint

Some basic concepts (II): Critical Cycle (3)

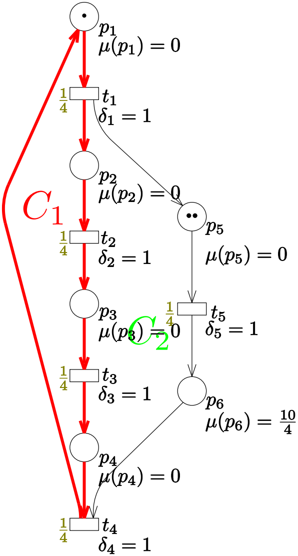


Concept of *slack*: μ

$$\hat{\mathbf{m}}(p) \geq \delta(p^\bullet)\cdot\Theta \longrightarrow \mathbf{m}(p) = \delta(p^\bullet)\cdot\Theta + \mu(p)$$

- $\mu(p) = 0$ if p belongs to critical cycle
- Value of vector μ will depend on the algorithm used by the LP solver
- The lower the slack, the higher the probability that place will constraint

Some basic concepts (II): Critical Cycle (3)



Concept of *slack*: μ

$$\hat{\mathbf{m}}(p) \geq \delta(p^\bullet)\cdot\Theta \longrightarrow \mathbf{m}(p) = \delta(p^\bullet)\cdot\Theta + \mu(p)$$

- $\mu(p) = 0$ if p belongs to critical cycle
- Value of vector μ will depend on the algorithm used by the LP solver
- The lower the slack, the higher the probability that place will constraint

Some basic concepts (III): Tight Marking (1)

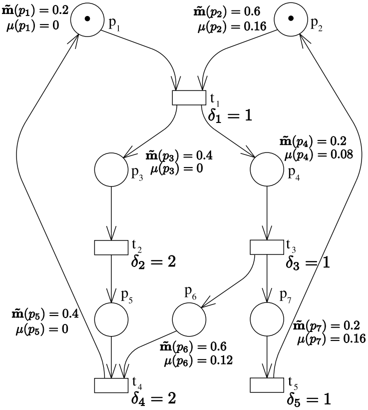
Tight marking vector ($\tilde{\mathbf{m}}$)

$$\begin{aligned} \tilde{\mathbf{m}} &= \mathbf{m_0} + \mathbf{C} \cdot \sigma && \text{(4a)} \\ \forall\ p: \quad \tilde{\mathbf{m}}(p) &\geq \delta(p^\bullet) \cdot \Theta && \text{(4b)} \\ \forall\ t\ \exists\ p \in {}^\bullet t: \quad \tilde{\mathbf{m}}(p) &= \delta(p^\bullet) \cdot \Theta && \text{(4c)} \end{aligned}$$

- Computed by solving the following LPP:

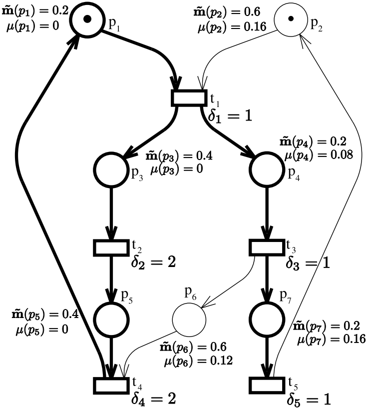
$$\begin{aligned} &\mathit{Maximize}\ \Sigma\sigma: \\ &\qquad \delta(p^\bullet) \cdot \Theta \leq \tilde{\mathbf{m}}(p) \quad \text{for every } p \in P \\ &\qquad \tilde{\mathbf{m}} = \mathbf{m_0} + \mathbf{C} \cdot \sigma \\ &\qquad \sigma(t_p) = k \end{aligned} \qquad (5)$$

Some basic concepts (III): Tight Marking (2)



- Tight place p: $\tilde{\mathbf{m}}(p) = \delta(p^\bullet) \cdot \Theta$
- Considering tight places (and their input and output transitions) $\rightarrow$ kind of tree
  - Critical cycle is the root
  - All transitions are reached

Some basic concepts (III): Tight Marking (2)



- Tight place p: $\tilde{\mathbf{m}}(p) = \delta(p^\bullet) \cdot \Theta$
- Considering tight places (and their input and output transitions) $\rightarrow$ kind of tree
  - Critical cycle is the root
  - All transitions are reached

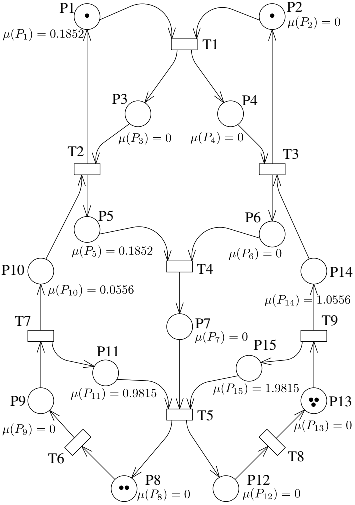
Graph regrowing strategy (I): algorithm

- Input data: SMG, accuracy
- Output data: sharper performance bound, bottleneck

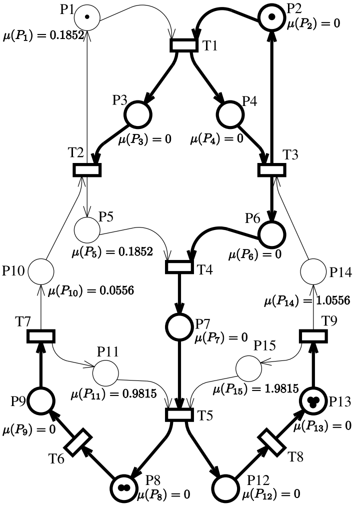
Algorithm steps

1. Calculate initial upper throughput bound and initial bottleneck cycle
2. Calculate tight marking and slacks
3. Iterate until no significant improvement is achieved
    1. Look for place with minimum slack and add it
    2. Calculate new throughput bound

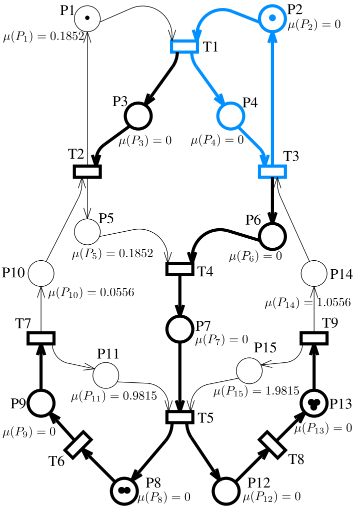
Graph regrowing strategy (II): running example



Graph regrowing strategy (II): running example

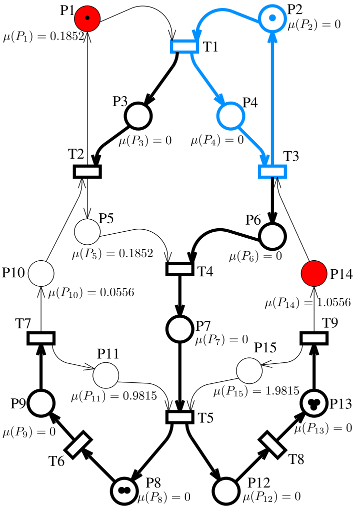


Graph regrowing strategy (II): running example



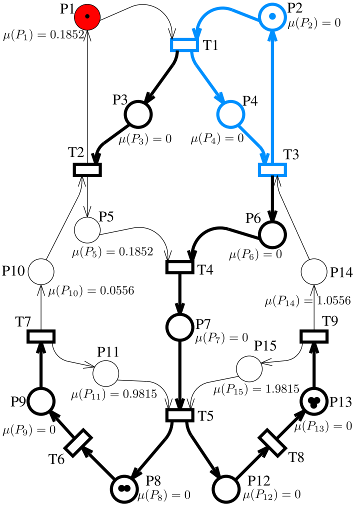
Iteration step	Candidates places	Added	Θ	$\%_{last}$	$\%_{initial}$
0	-	-	0.3704	-	-

Graph regrowing strategy (II): running example



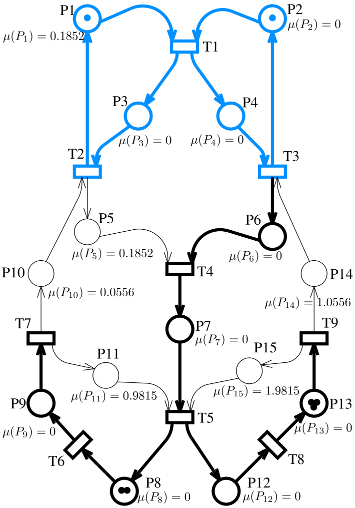
Iteration step	Candidates places	Added	Θ	$\%_{last}$	$\%_{initial}$
0	p_1, p_{14}	-	0.3704	-	-

Graph regrowing strategy (II): running example



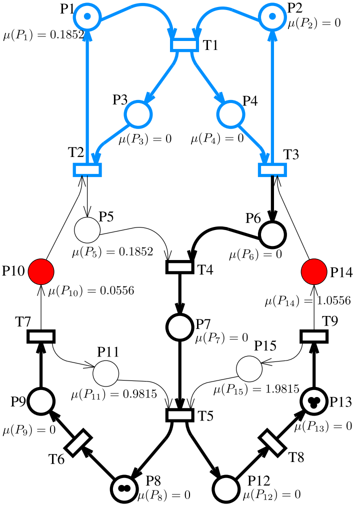
Iteration step	Candidates places	Added	Θ	$\%_{last}$	$\%_{initial}$
0	p_1 , p_{14}	p_1	0.3704	-	-

Graph regrowing strategy (II): running example



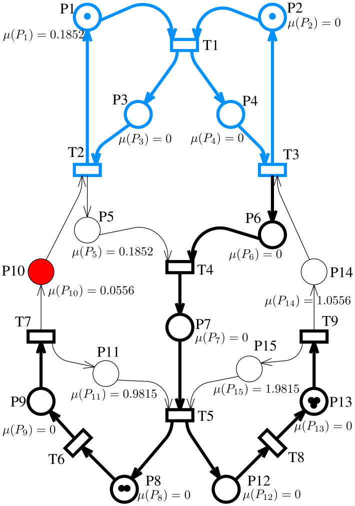
Iteration step	Candidates places	Added	Θ	$\%_{last}$	$\%_{initial}$
0	p_1 , p_{14}	p_1	0.3704	-	-
1	-	-	0.322581	12.9%	12.9%

Graph regrowing strategy (II): running example



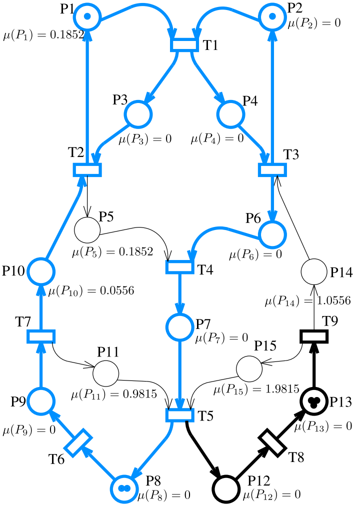
Iteration step	Candidates places	Added	Θ	$\%_{last}$	$\%_{initial}$
0	p_1 , p_{14}	p_1	0.3704	-	-
1	p_{10} , p_{14}	-	0.322581	12.9%	12.9%

Graph regrowing strategy (II): running example



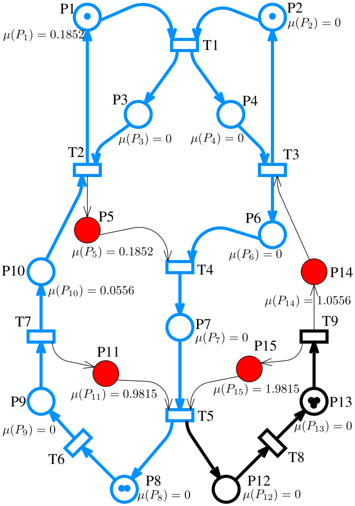
Iteration step	Candidates places	Added	Θ	$\%_{last}$	$\%_{initial}$
0	p_1, p_{14}	p_1	0.3704	-	-
1	p_{10}, p_{14}	p_{10}	0.322581	12.9%	12.9%

Graph regrowing strategy (II): running example



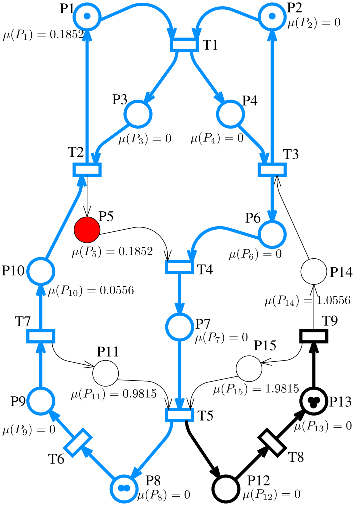
Iteration step	Candidates places	Added	Θ	$\%_{last}$	$\%_{initial}$
0	p_1 , p_{14}	p_1	0.3704	-	-
1	p_{10} , p_{14}	p_{10}	0.322581	12.9%	12.9%
2	-	-	0.297914	7.647%	19.563%

Graph regrowing strategy (II): running example



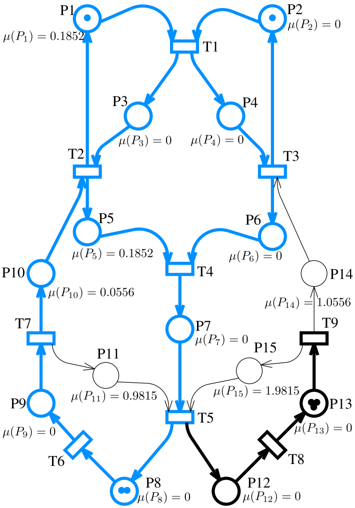
Iteration step	Candidates places	Added	Θ	$\%_{last}$	$\%_{initial}$
0	p_1, p_{14}	p_1	0.3704	-	-
1	p_{10}, p_{14}	p_{10}	0.322581	12.9%	12.9%
2	p_5, p_{11}, p_{14}, p_{15}	-	0.297914	7.647%	19.563%

Graph regrowing strategy (II): running example



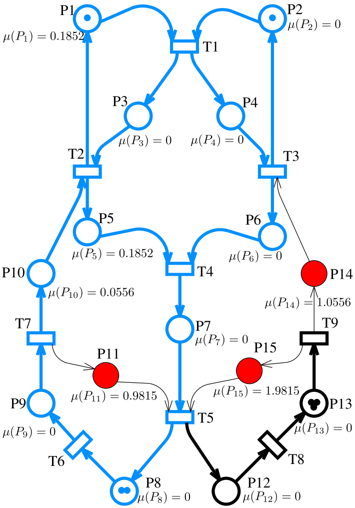
Iteration step	Candidates places	Added	Θ	$\%_{last}$	$\%_{initial}$
0	p_1, p_{14}	p_1	0.3704	-	-
1	p_{10}, p_{14}	p_{10}	0.322581	12.9%	12.9%
2	p_5, p_{11}, p_{14}, p_{15}	p_5	0.297914	7.647%	19.563%

Graph regrowing strategy (II): running example



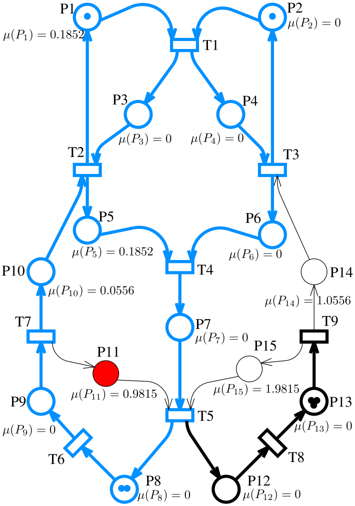
Iteration step	Candidates places	Added	Θ	$\%_{last}$	$\%_{initial}$
0	p_1 , p_{14}	p_1	0.3704	-	-
1	p_{10} , p_{14}	p_{10}	0.322581	12.91%	12.91%
2	p_5 , p_{11} p_{14} , p_{15}	p_5	0.297914	7.647%	19.563%
3	-	-	0.288401	3.193%	22.137%

Graph regrowing strategy (II): running example



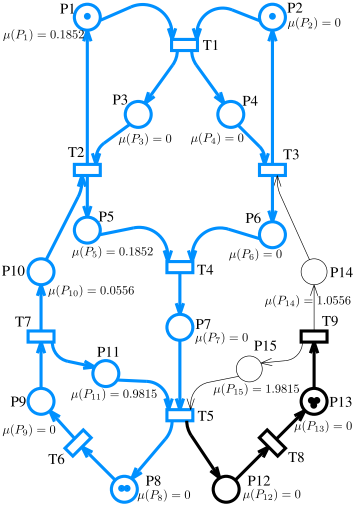
Iteration step	Candidates places	Added	Θ	$\%_{last}$	$\%_{initial}$
0	p_1, p_{14}	p_1	0.3704	-	-
1	p_{10}, p_{14}	p_{10}	0.322581	12.9%	12.9%
2	p_5, p_{11}, p_{14}, p_{15}	p_5	0.297914	7.647%	19.563%
3	p_{11}, p_{14}, p_{15}	-	0.288401	3.193%	22.137%

Graph regrowing strategy (II): running example



Iteration step	Candidates places	Added	Θ	$\%_{last}$	$\%_{initial}$
0	p_1 , p_{14}	p_1	0.3704	-	-
1	p_{10} , p_{14}	p_{10}	0.322581	12.9%	12.9%
2	p_5 , p_{11} , p_{14} , p_{15}	p_5	0.297914	7.647%	19.563%
3	p_{11} , p_{14} , p_{15}	p_{11}	0.288401	3.193%	22.137%

Graph regrowing strategy (II): running example



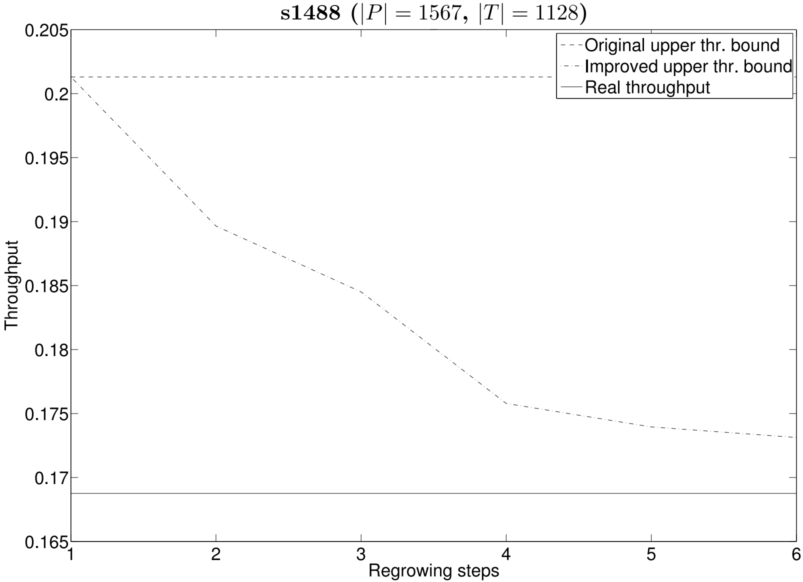
Iteration step	Candidates places	Added	Θ	$\%_{last}$	$\%_{initial}$
0	p_1, p_{14}	p_1	0.3704	-	-
1	p_{10}, p_{14}	p_{10}	0.322581	12.9%	12.9%
2	p_5, p_{11}, p_{14}, p_{15}	p_5	0.297914	7.647%	19.563%
3	p_{11}, p_{14}, p_{15}	p_{11}	0.288401	3.193%	22.137%
4	-	-	0.288401	0%	22.137%

Experiments (I): description of the experiments

Benchmarking and used tools

- ISCAS benchmarking
  - Strongly connected components of the ISCAS graphs
  - Initial marking randomly selected in $[1 \ldots 10]$
  - Delay of transitions randomly selected in $[0.1 \ldots 1]$
- Strategy implemented in MATLAB (linprog)
- Simulation tool: GreatSPN
  - Confidence level 99%; accuracy 1%
- Host: Pentium IV 3.6GHz, 2GB DDR2 533MHz RAM

Experiments (II): Gets close to the real thr. after few steps



Experiments (III): results of improvement

Graph	Size		% Size		Regrowing steps	Initial thr. bound	Θ
	$\|P\|$	$\|T\|$	$\|P'\|$ (%)	$\|T'\|$ (%)			
s1423	1107	792	79 (7.13%)	76 (9.59%)	3	0.236010	0.235213 (0.34%)
s1488	1567	1128	91 (5.8%)	86 (7.62%)	6	0.201300	0.173127 (13.99%)
s208	27	24	27 (100%)	24 (100%)	3	0.409390	0.377683 (7.75%)
s27	54	44	19 (35.18%)	18 (40.9%)	1	0.305960	0.304987 (0.31%)
s349	187	146	26 (13.9%)	24 (16.44%)	2	0.340320	0.327867 (3.66%)
s444	92	68	14 (15.21%)	12 (17.64%)	2	0.181670	0.181260 (0.22%)
s510	1038	734	45 (4.33%)	40 (5.45%)	5	0.133030	0.117819 (11.43%)
s526	113	92	18 (15.93%)	16 (17.39%)	2	0.313490	0.305860 (2.43%)
s713	271	208	11 (4.06%)	10 (4.8%)	1	0.428720	0.427840 (0.2%)
s820	1162	848	40 (3.44%)	38 (4.48%)	2	0.161060	0.147483 (8.43%)
s832	1293	948	84 (6.5%)	78 (12.04%)	5	0.239429	0.208798 (12.79%)
s953	415	312	88 (11.36%)	82 (26.28%)	6	0.369214	0.337811 (8.50%)

- Sharper upper bound in few regrowing steps
  - Improvement varies from 0.2% to 14%
- Uses a very low percentage of the size of the original graph
  - Lower than 10% (in most of cases)

Experiments (III): results of improvement

Graph	Size		% Size		Regrowing steps	Initial thr. bound	Θ
	$\|P\|$	$\|T\|$	$\|P'\|$ (%)	$\|T'\|$ (%)			
s1423	1107	792	79 (7.13%)	76 (9.59%)	3	0.236010	0.235213 (0.34%)
s1488	1567	1128	91 (5.8%)	86 (7.62%)	6	0.201300	0.173127 (13.99%)
s208	27	24	27 (100%)	24 (100%)	3	0.409390	0.377683 (7.75%)
s27	54	44	19 (35.18%)	18 (40.9%)	1	0.305960	0.304987 (0.31%)
s349	187	146	26 (13.9%)	24 (16.44%)	2	0.340320	0.327867 (3.66%)
s444	92	68	14 (15.21%)	12 (17.64%)	2	0.181670	0.181260 (0.22%)
s510	1038	734	45 (4.33%)	40 (5.45%)	5	0.133030	0.117819 (11.43%)
s526	113	92	18 (15.93%)	16 (17.39%)	2	0.313490	0.305860 (2.43%)
s713	271	208	11 (4.06%)	10 (4.8%)	1	0.428720	0.427840 (0.2%)
s820	1162	848	40 (3.44%)	38 (4.48%)	2	0.161060	0.147483 (8.43%)
s832	1293	948	84 (6.5%)	78 (12.04%)	5	0.239429	0.208798 (12.79%)
s953	415	312	88 (11.36%)	82 (26.28%)	6	0.369214	0.337811 (8.50%)

- Sharper upper bound in few regrowing steps
  - Improvement varies from 0.2% to 14%
- Uses a very low percentage of the size of the original graph
  - Lower than 10% (in most of cases)

Experiments (IV): graph throughput and time comparative

Graph	Original graph thr. CPU time (s)	Θ CPU time (s)	Original graph thr.	Θ	% thr.
s1423	59948.980	8.283	0.222720	0.235270	5.63%
s1488	36717.156	7.165	0.168760	0.172154	2.01%
s208	0.492	0.492	0.376892	0.376892	0%
s27	2166.002	0.954	0.305082	0.306166	0.35%
s349	141.210	0.441	0.328340	0.327398	−0.28%
s444	2278.231	0.205	0.181069	0.181260	0.11%
s510	13669.814	1.358	0.117500	0.118040	0.46%
s526	129.181	0.344	0.270010	0.305860	13.27%
s713	628.503	0.405	0.411630	0.427840	3.94%
s820	20775.811	0.788	0.144770	0.147699	2.02%
s832	16165.863	1.914	0.196920	0.208873	6.07%
s953	453.850	19.155	0.327910	0.338644	3.27%

- Θ CPU time insignificant respect to original thr CPU time
- Improvement varies from very close value to 13% over the real thr
    - Slow cycles far away from critical cycle?
- Negative relative error caused by simulation parameters

Experiments (IV): graph throughput and time comparative

Graph	Original graph thr. CPU time (s)	Θ CPU time (s)	Original graph thr.	Θ	% thr.
s1423	59948.980	8.283	0.222720	0.235270	5.63%
s1488	36717.156	7.165	0.168760	0.172154	2.01%
s208	0.492	0.492	0.376892	0.376892	0%
s27	2166.002	0.954	0.305082	0.306166	0.35%
s349	141.210	0.441	0.328340	0.327398	−0.28%
s444	2278.231	0.205	0.181069	0.181260	0.11%
s510	13669.814	1.358	0.117500	0.118040	0.46%
s526	129.181	0.344	0.270010	0.305860	13.27%
s713	628.503	0.405	0.411630	0.427840	3.94%
s820	20775.811	0.788	0.144770	0.147699	2.02%
s832	16165.863	1.914	0.196920	0.208873	6.07%
s953	453.850	19.155	0.327910	0.338644	3.27%

- Θ CPU time insignificant respect to original thr CPU time
- Improvement varies from very close value to 13% over the real thr
  - Slow cycles far away from critical cycle?
- Negative relative error caused by simulation parameters

Summary

- Proposed approach based on an iterative algorithm
  - Takes initial thr bound and refines it in each iteration
- Accurate upper bound in few iterations
- Efficient and good *accuracy-computational complexity load* trade-off
- Outputs:
  - Accurate estimate for the steady state thr
  - Subnet representing bottleneck of the system

Accurate Performance Estimation for Stochastic Marked Graphs by Bottleneck Regrowing

Ricardo J. Rodríguez and Jorge Júlvez
{rjrodriguez, julvez}@unizar.es

Universidad de Zaragoza
Zaragoza, Spain

September 24^{th}, 2010

EPEW'10: 7^{th} European Performance Engineering Workshop
Bertinoro, Italy