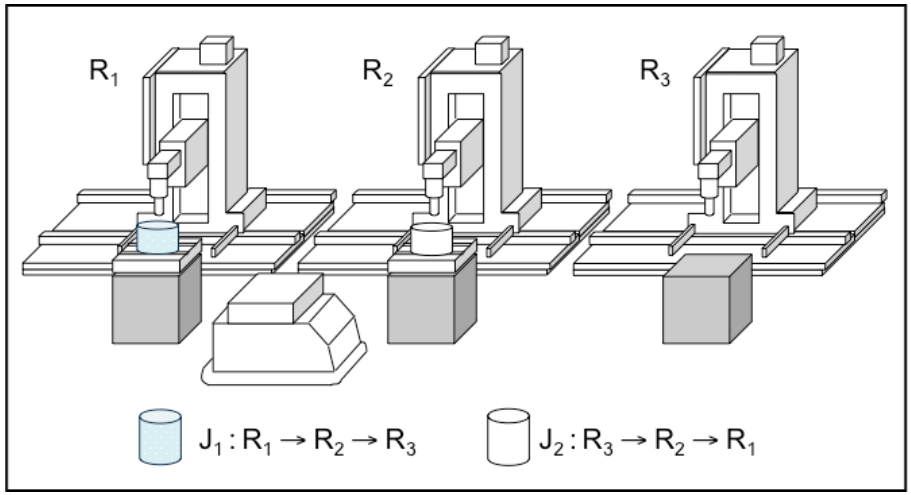
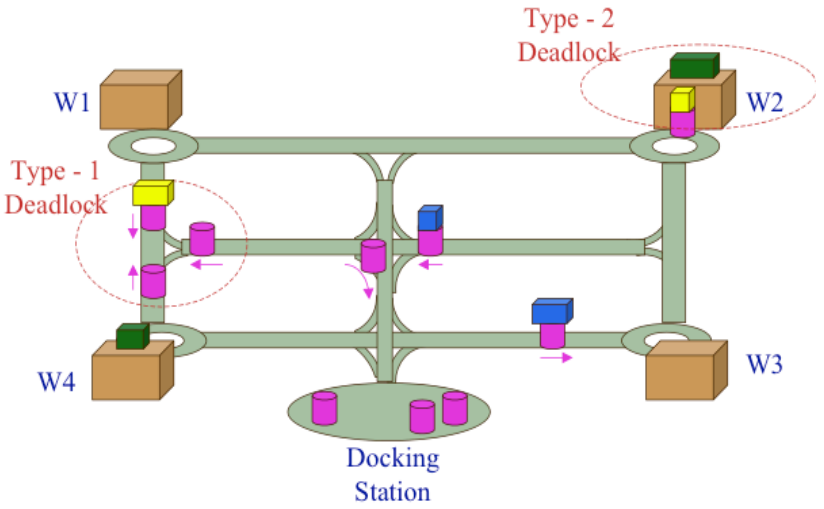
Real-Time Management of Complex Resource Allocation Systems: Necessity, Achievements and Further Challenges

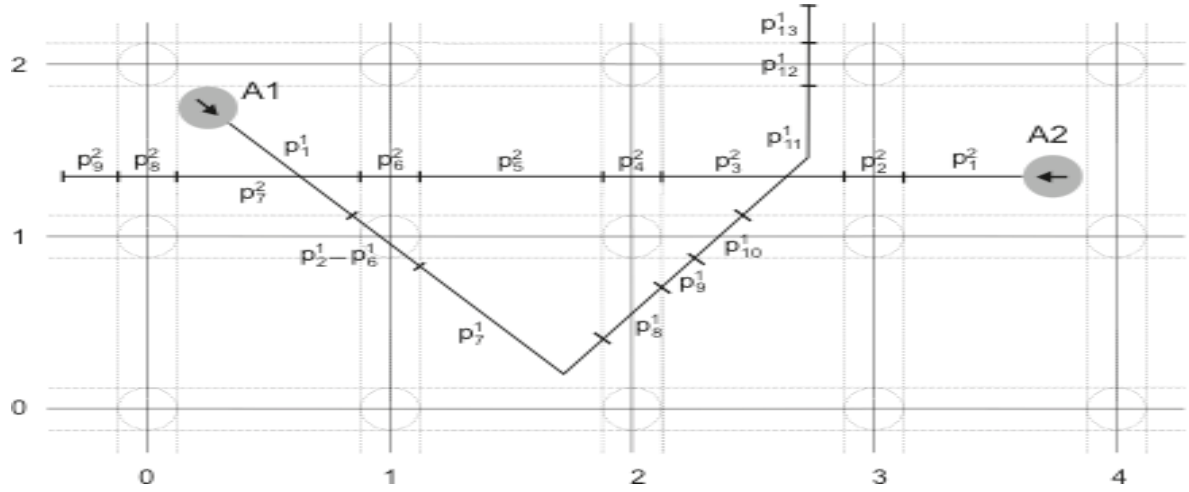
Spyros Reveliotis
School of Industrial & Systems Engineering
Georgia Institute of Technology

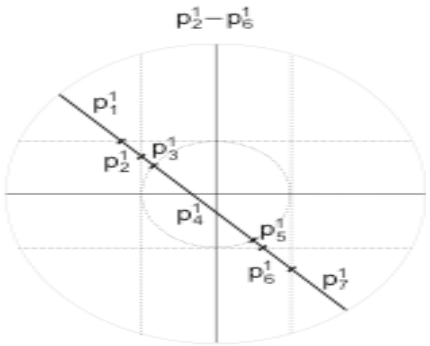
DCDS 2015
Cancun, Mexico

Motivation: Resource allocation in flexible automation and its underlying challenges

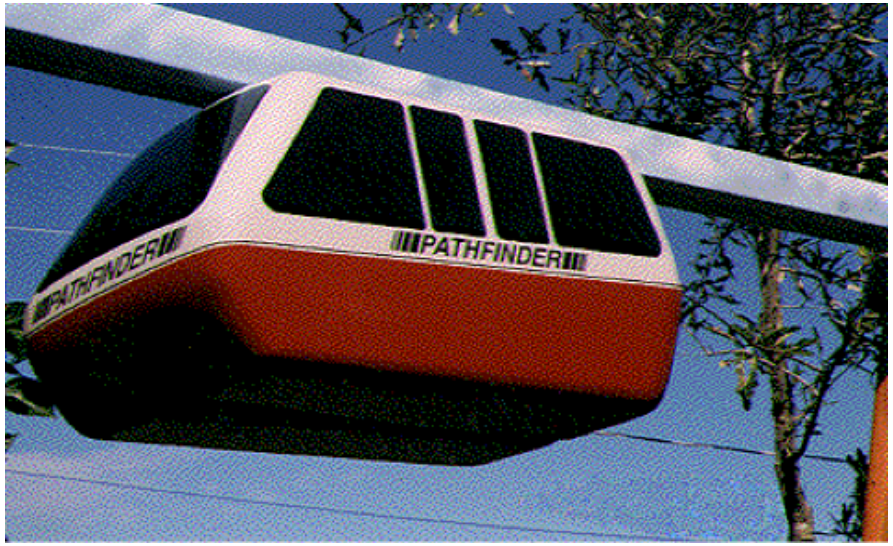


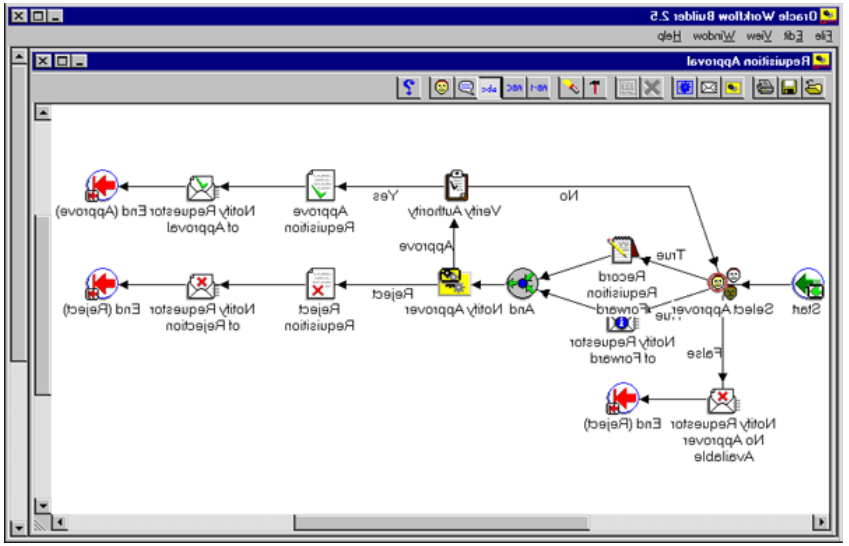


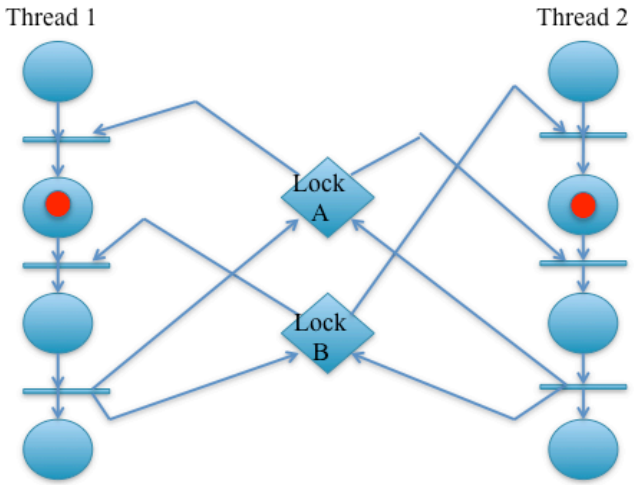


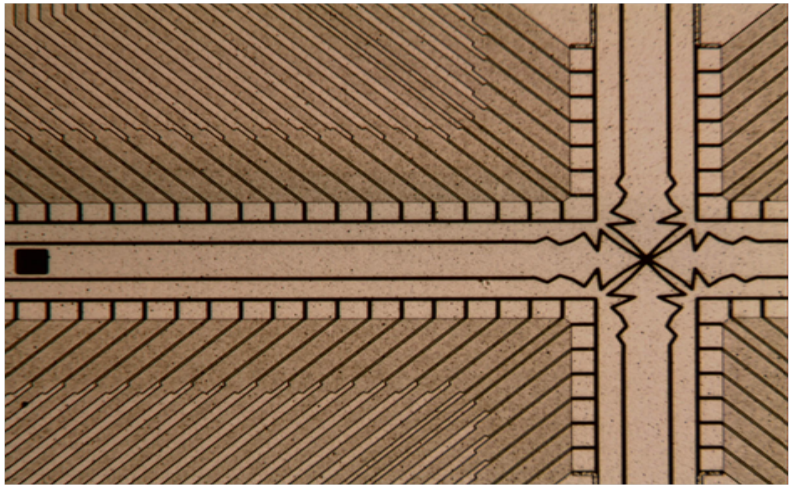


The ubiquitousness of the deadlock problem





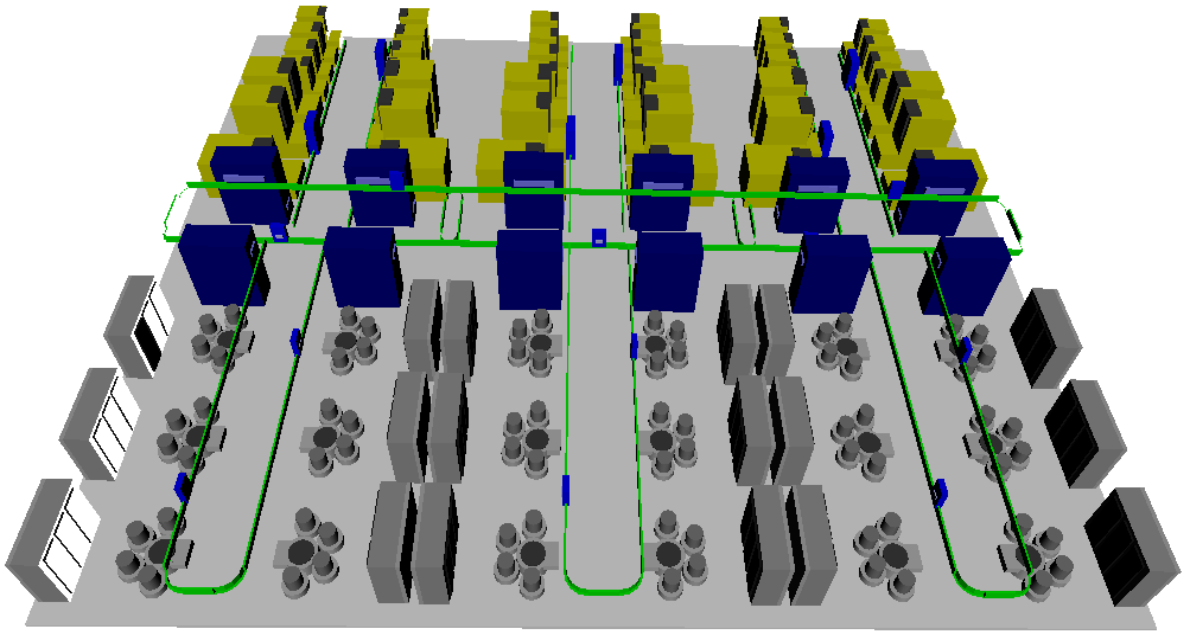




Ion trap, GTRI

Dijkstra, 1965; Havender, 1968; Habermann, 1969; Coffman *et al*, 1971; Holt, 1972

The current state of art



- Current practice deals with the deadlock problem through *ad hoc* and typically very conservative solutions.

- Need for a formal paradigm that enables a rigorous problem investigation and an effective trade-off between computational complexity and operational efficiency.

A modeling abstraction: Sequential Resource Allocation Systems (RAS)

- A set of (re-usable) resource types $R = \{R_i, i = 1,...,m\}$.
- Finite capacity C_i for each resource type R_i.
- A set of process types $J = \{J_j, j = 1,...,n\}$.
- A (partially) ordered set of process stages for each job type, $\{p_{jk}, k = 1,...,\lambda_j\}$.
- A resource requirements vector for each process stage p, $a_p[i]$, $i = 1,...,m$.
- A timing distribution D(p) characterizing the execution time of each process stage p.
- Resource Allocation Protocol: In order to advance from stage p to stage q, a process must be allocated the resource differential $(a_q\text{-}a_p)^+$ and only then it releases the unnecessary resource set expressed by $(a_p\text{-}a_q)^+$.

Sequential RAS deadlock: A RAS state in which there exists a subset of processes s.t. every process in this subset in order to proceed requires some resource(s) currently allocated to some other process in this subset.

A RAS taxonomy

Structure of the process sequential logic

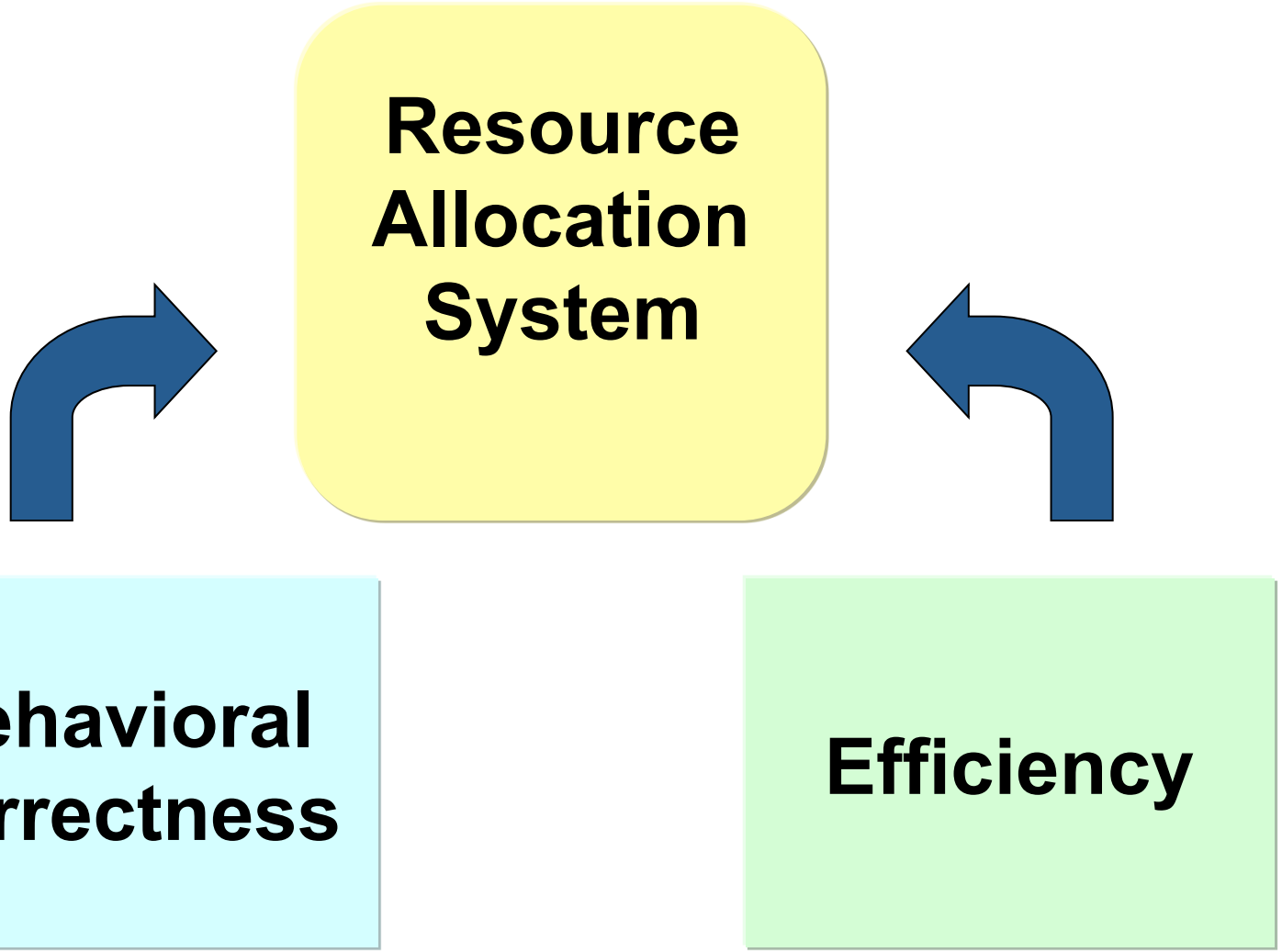
- Linear: each process is defined by a linear sequence of stages
- Disjunctive: A number of alternative process plans encoded by an acyclic digraph
- Merge-Split or Fork-Join: each process is a fork-join network
- Complex: a combination of the above behaviors

Structure of the stage resource requirement vectors

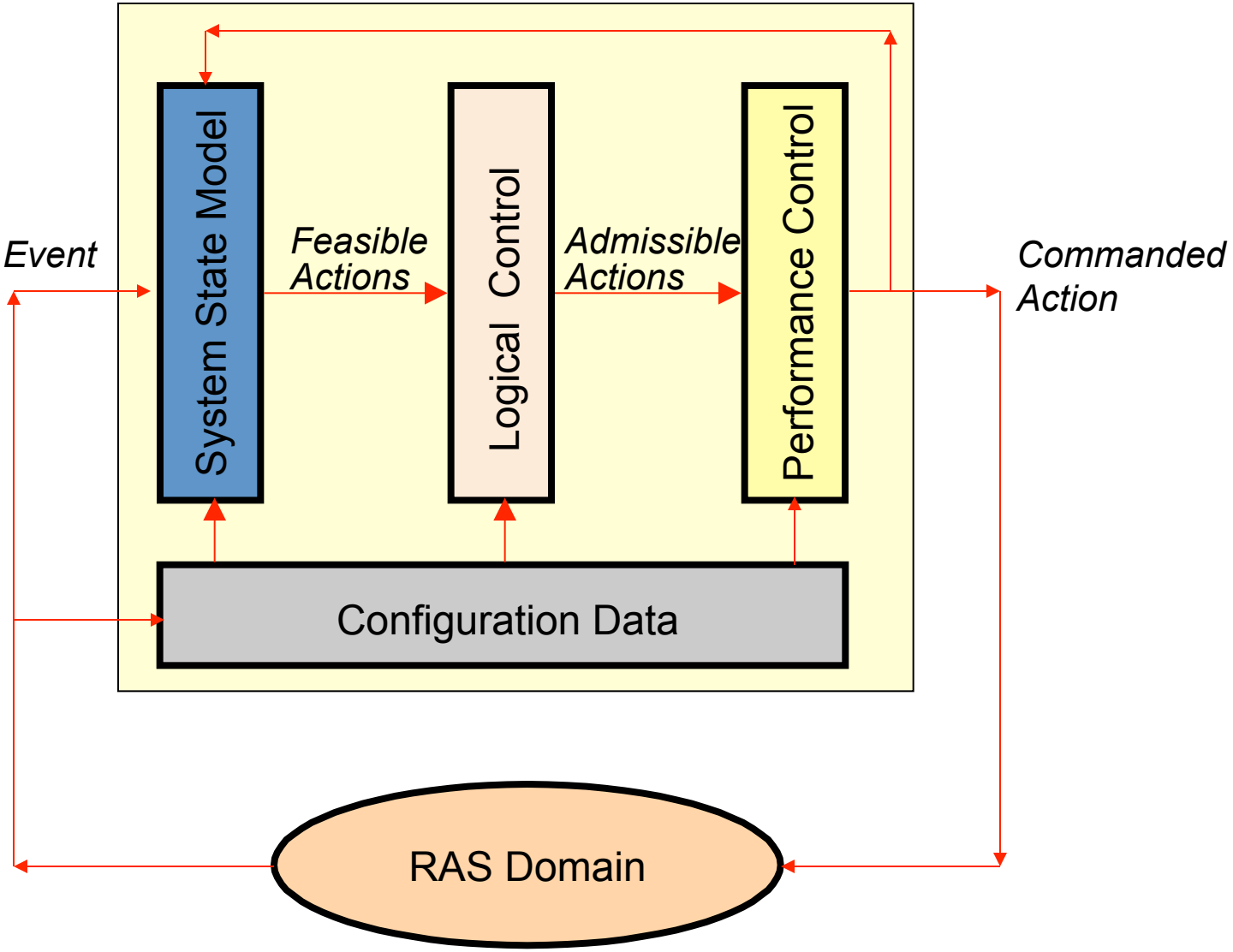
- Single-unit: each stage requires a single unit from a single resource
- Single-type: each stage requires an arbitrary number of units, but all from a single resource
- Conjunctive: Arbitrary number of units from different resources

Some additional interesting features: ***Uncontrollable*** transitions w.r.t. timing and/or routing, ***Unit*** resource capacities, etc.

Control of Sequential RAS: Logical vs Performance Control



An Event-Driven RAS Control Scheme



The "human anchoring" of the presented research

Past and Current Students:

- Jonghun Park
- Jingyoung Choi
- Theologos Bountourelis
- Ahmed Nazeem
- Hongwei Liao
  (co-adv. with S. Lafortune)
- Zhennan Fei
  (co-adv. with K. Akesson)
- Ran Li
- Stephen Daugherty

Research Sponsors:

- Various NSF grants from
  - CMMI / MES
  - ECS & ECCS
- The ISyE Virtual Factory Lab and the Keck Foundation
- The Georgia Tech Research Institute

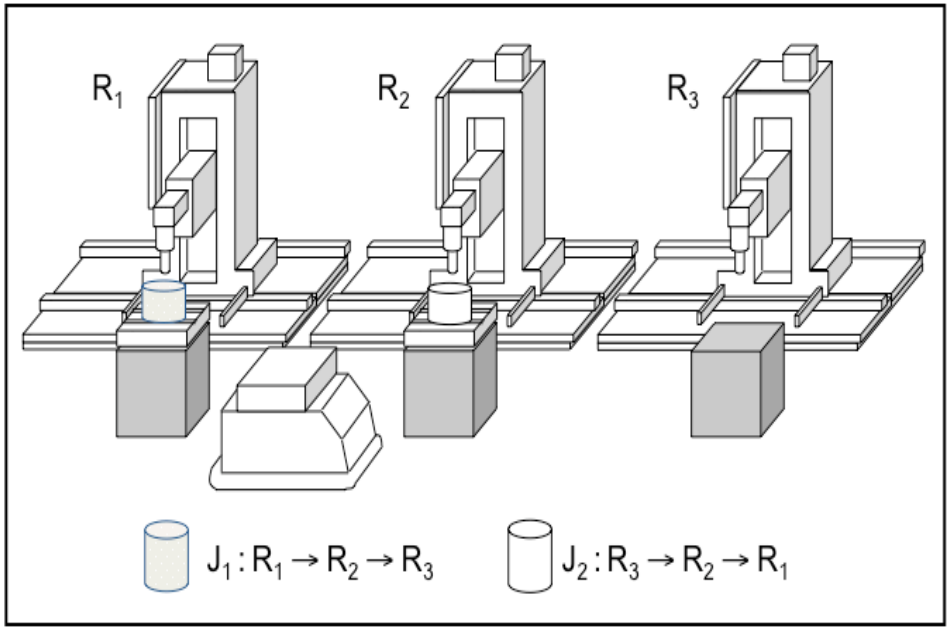
Other research groups and collaborators:

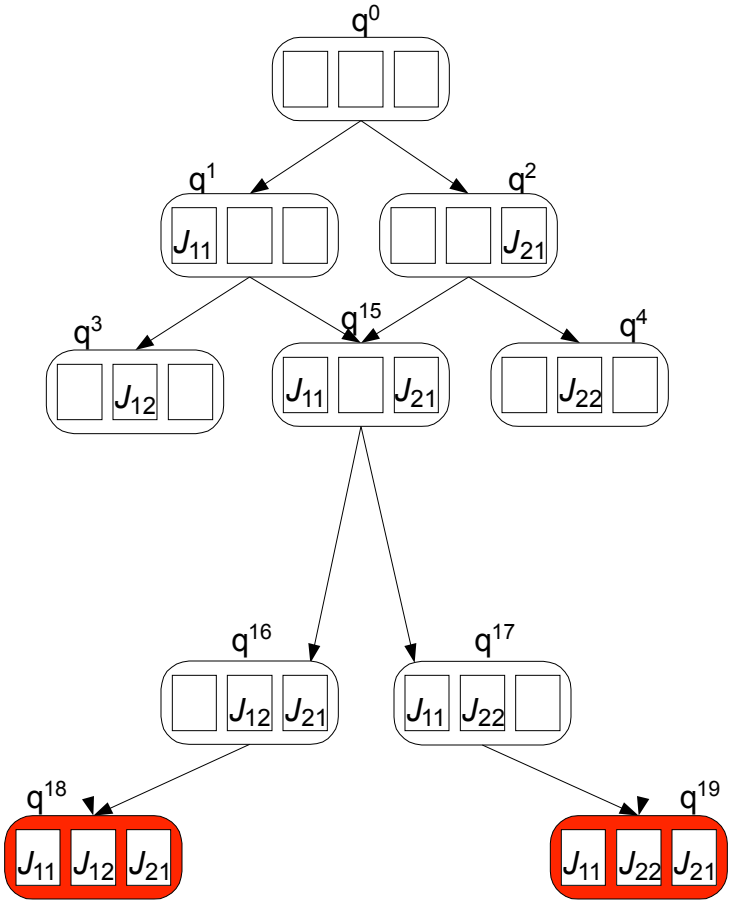
- E. Dijkstra, A. Habermann, J. Havender, R. Holt, E. Coffman, M. Elphick, A. Shoshanni,
- M. Gold, T. Araki, Y. Sugiyama and T. Kasami

- P. Ramadge and W. M. Wonham
- P. Antsaklis, J. Moody and M. Iordache
- A. Giua and M. Silva
- N. Viswanadham,Y. Narahari and T. Johnson
- Z. Banaszak and B. Krogh
- P. Ferreira, M. Lawley and his colleagues
- E. Roszkowska
- J. Ezpeleta, J.-M. Colom and their colleagues
- M. P. Fanti and her colleagues
- M. Zhou, Z. Li and their colleagues
- X. Xie, M. Jeng and their colleagues
- S. Lafortune, Y. Wang and their colleagues
- L. Piroddi and R. Cordone
- K. Barkaoui and his colleagues

- X. Cao and C. Cassandras
- D. Bertsekas, J. Tsitsiklis and their colleagues

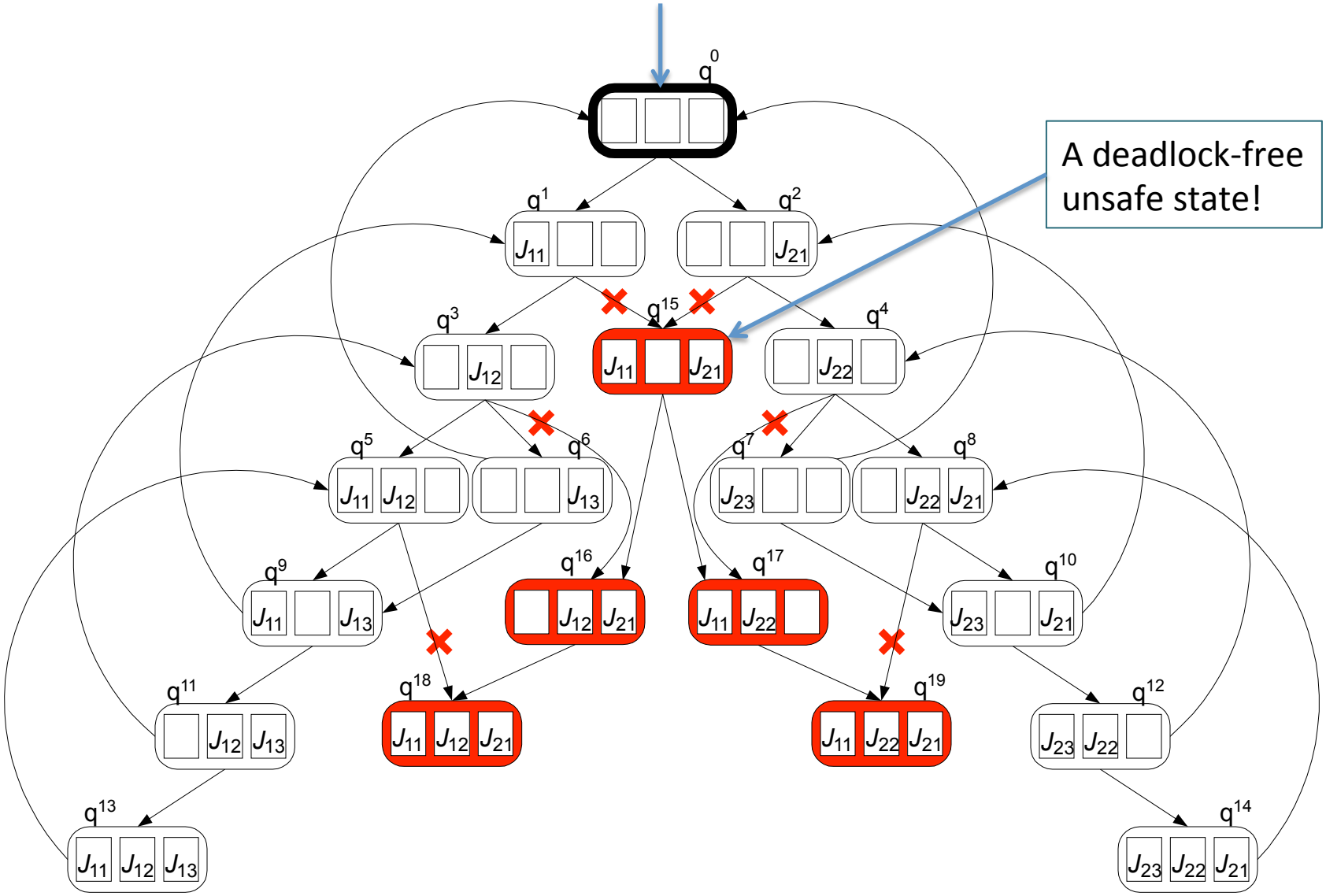
Finite State Automata (FSA)-based modeling of RAS behavior





The RAS state is defined by the number of active process instances at each processing stage.

Safe vs. Unsafe Region and the Optimal Deadlock Avoidance Policy



Complexity Considerations

- State Safety is an NP-complete problem in sequential RAS
  (by reduction of the SAT or 3SAT problem)

- State Transition Diagram (STD) size for SU-RAS:

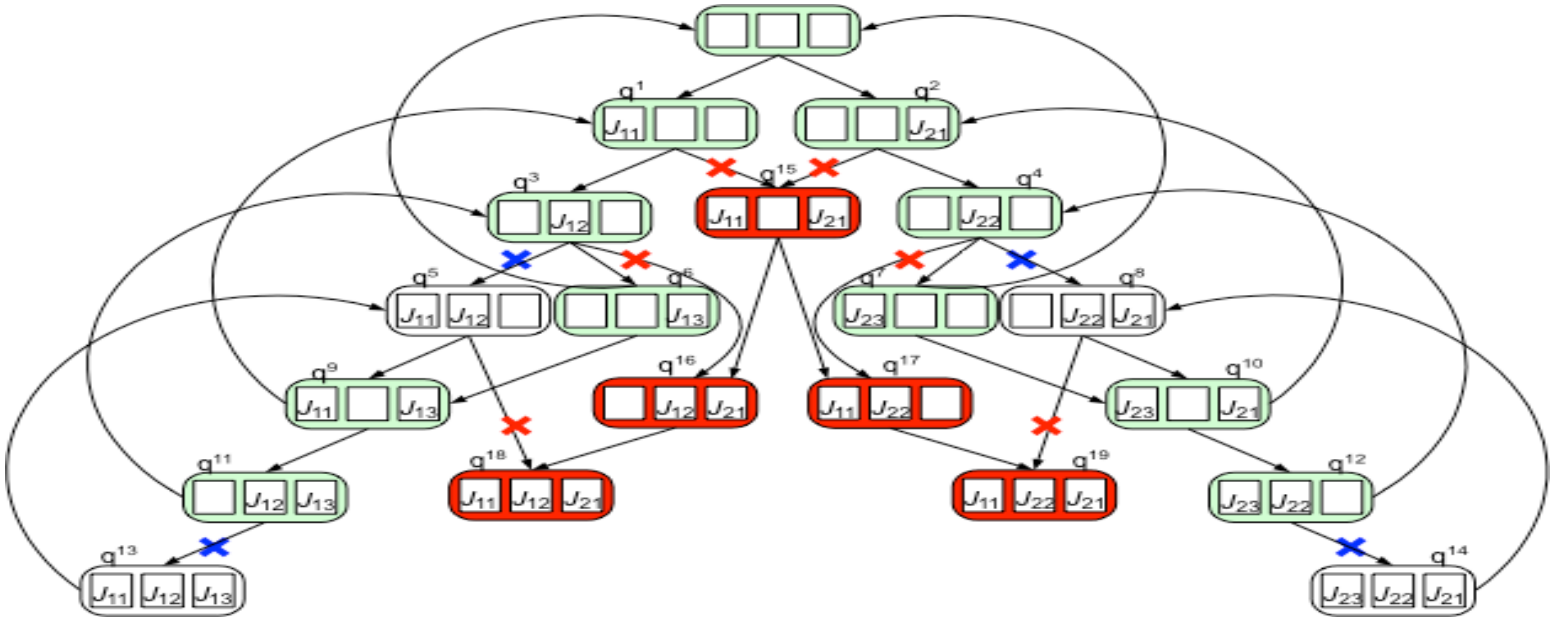
$$O\left(\binom{C+Q}{Q}^{m}\right)$$

where:

- C = max resource capacity
- Q = max number of stages supported by a resource
- m = number of resource types

Efficient suboptimal solutions: Correct Polynomial-Kernel DAPs

- ***Sub-optimal*** one-step-lookahead policies based on state properties that are polynomially verifiable and identify a strongly connected component of the RAS state space containing the empty state



  - **"Ordered"** states and **Banker's algorithm**: Focus on "termination" sequences that are polynomially identifiable
  - ***"Algebraic"*** PK-DAPs: Defined by polynomially-sized sets of linear inequalities on the RAS state
    - RUN (Resource Upstream Neighborhood -- available for Disjunctive / Conjunctive RAS)
    - RO (Resource Ordering – appropriate for single-type linear RAS

Special Structure: RAS admitting optimal LES of polynomial complexity

1. **"Nested"** allocation: Resources are released in the reverse order from which they were acquired. (every safe state has an polynomially identifiable termination sequence).

2. **No deadlock-free unsafe states**: Unsafety ≡ deadlock (substitute state safety test with deadlock test, which has polynomial complexity)

Theorem: In a D-SU-RAS where every resource has at least two units of capacity, the optimal logical control policy is *polynomially* implementable w.r.t. the underlying RAS "size" through one-step lookahead for deadlock.

"**Corollary":** For most practical configurations, deadlock-free ***buffer*** allocation in flexibly automated production systems is an easy problem (!)

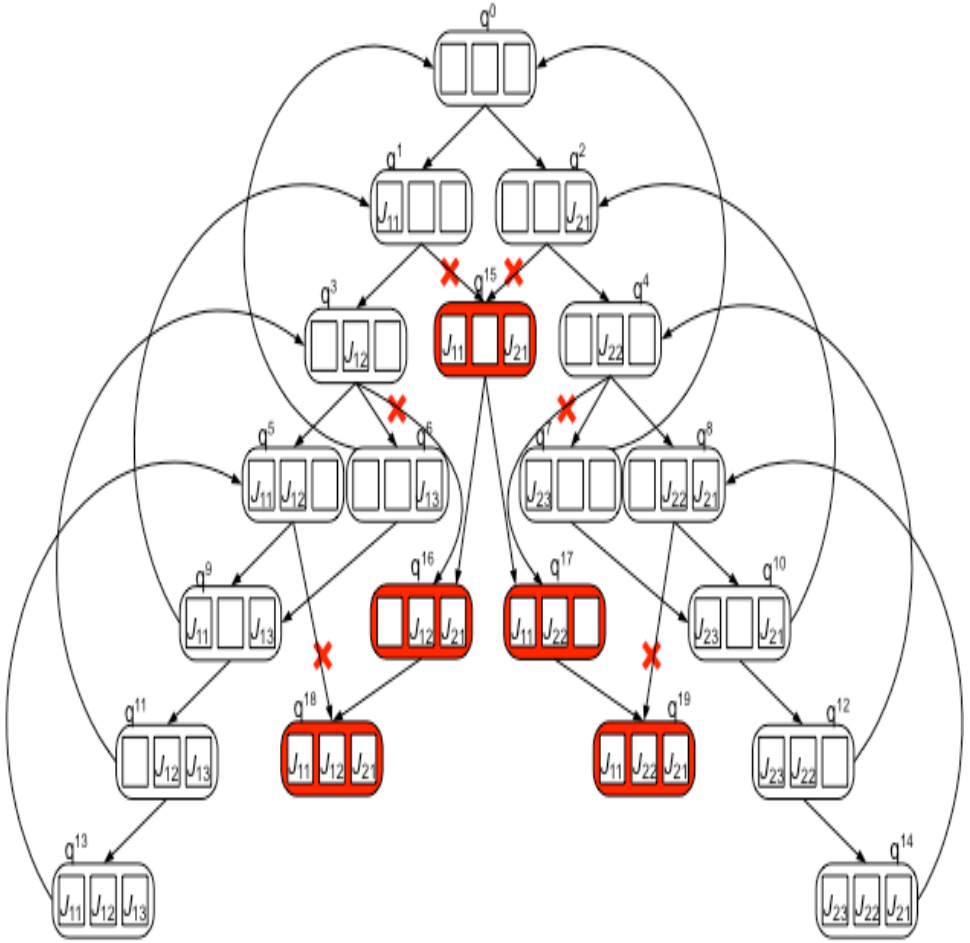
Designing the maximally permissive LES (for D/C-RAS) through classification theory

Compute the reachable state space of the considered RAS

Use standard supervisory control theory to classify the various states into safe and unsafe.

Design a compact classifier that effects the aforestated dichotomy of the reachable state space.

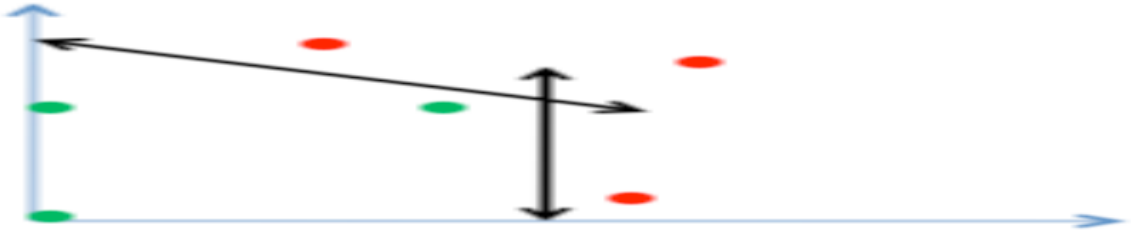
Use this classifier on-line in order to accept or reject tentative transitions.



Some enabling ideas and results

1. The "monotonicity" property for D/C-RAS state safety:

$$s \text{ is safe } \wedge\ s' \leq s \Rightarrow s' \text{ is safe.}$$



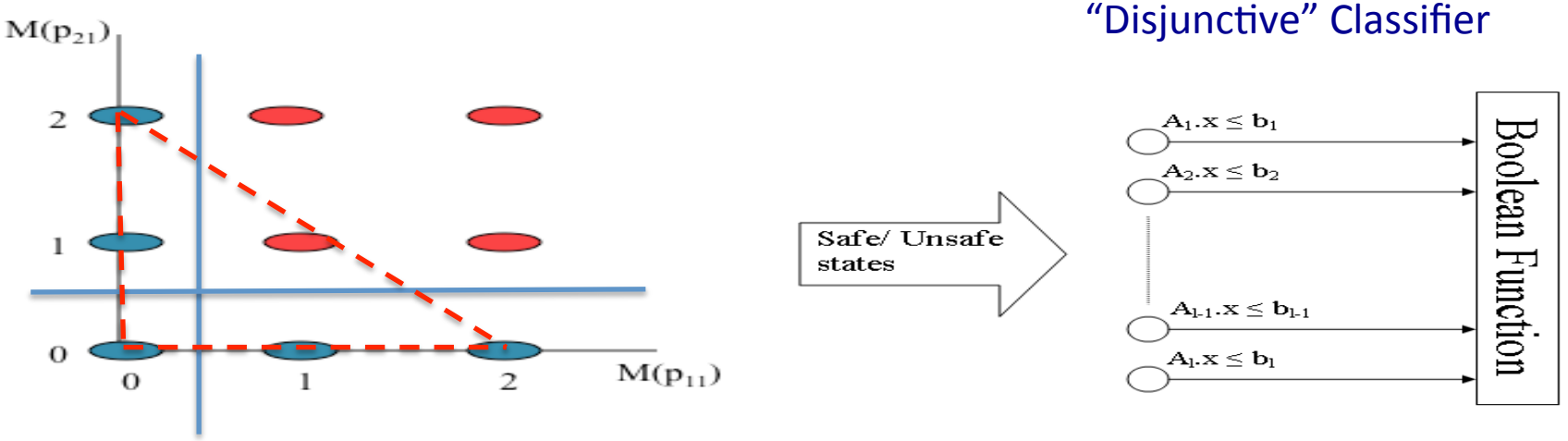
Focus the classifier construction only on the correct classification of maximal safe and minimal unsafe states.

2. Consider only "boundary" unsafe states, i.e. states that can be reached in one transition from some safe state.

3. Ignore state components corresponding to stages that cannot be entangled in deadlock (this is attained automatically by the considered methods).

Classifier "architectures"

- Another implication of the monotonicity of state safety:
  The safe subspace of D/C-RAS can be represented as *the union of a set of polytopes*,
  with each polytope defined by a *maximal* safe state.



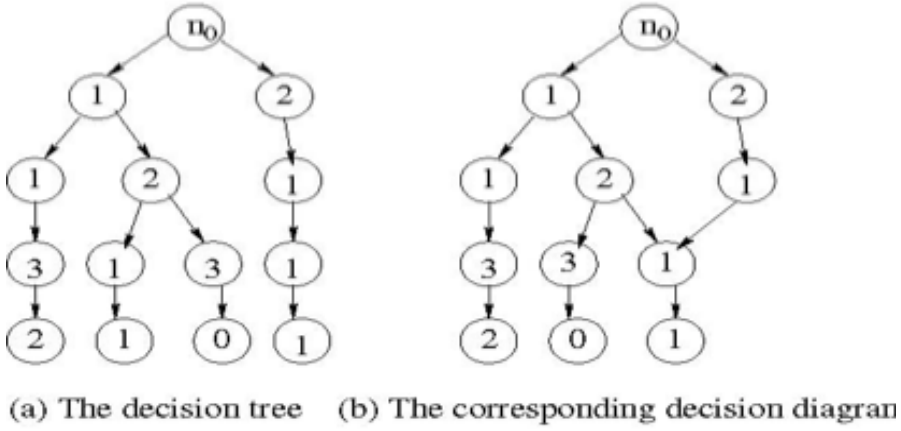
"Disjunctive" Classifier

- If the convex hull of the safe states does not contain any unsafe states, then the sought classifier can be expressed by a set of linear inequalities; such a classifier is called "linear".

- Structurally minimal classifiers representing the target DAP can be obtained through MIP formulations and other more streamlined methods adapted from combinatorial optimization theory.

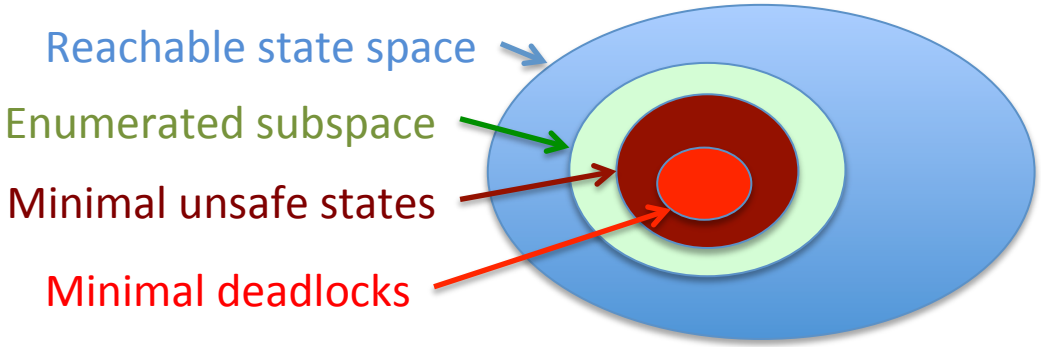
"Non-parametric" classifiers

Identify and store only the (minimal) boundary unsafe states using an efficient ("symbolic") representation.



(a) The decision tree (b) The corresponding decision diagram

Efficient searching for the (minimal) boundary unsafe states:

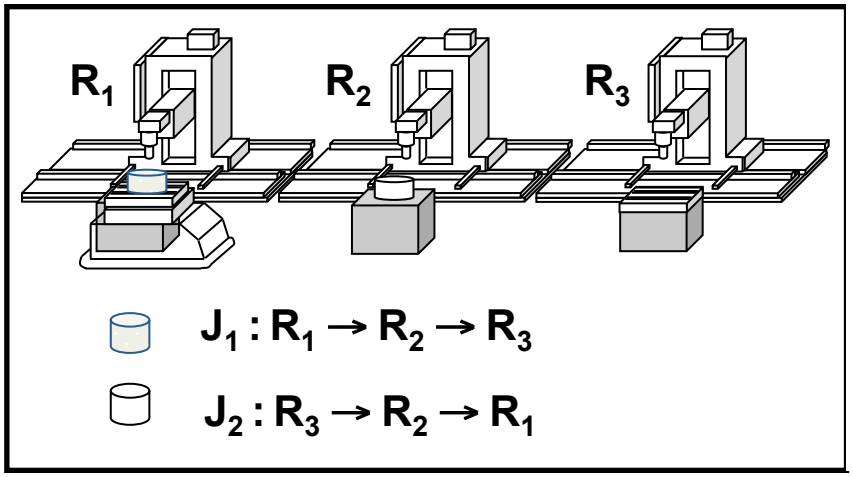


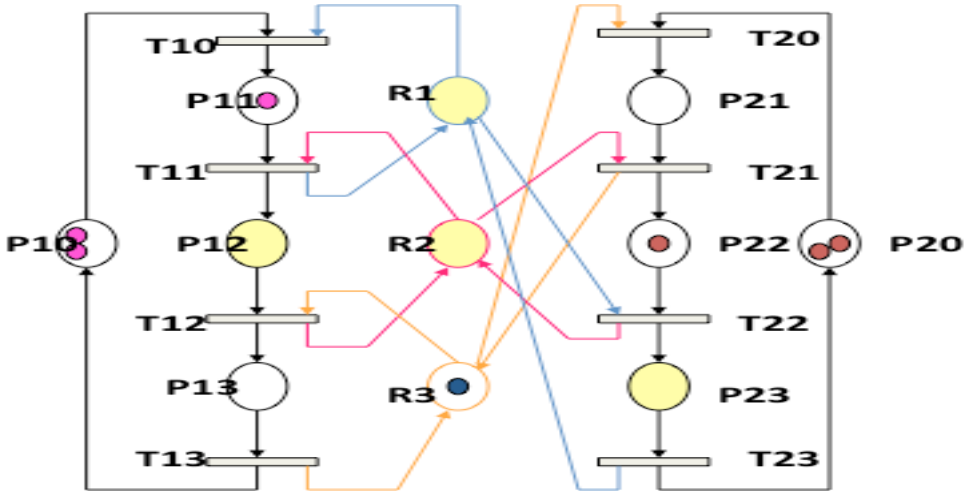
Some experimental results

# of Resources	# of Stages	Reachable states	Boundary states	Min Boundary	Comp. Time (sec)	Max # of Nodes
8	42	4.0×10^5	2.0×10^5	5.3×10^3	8	2.2×10^5
16	27	2.4×10^6	7.4×10^5	1.6×10^4	10	2.2×10^5
10	42	1.9×10^7	7.6×10^5	1.0×10^4	33	9.2×10^5
14	42	2.9×10^7	7.7×10^6	2.7×10^4	205	2.1×10^6
40	27	1.0×10^8	1.2×10^7	4.3×10^3	313	2.3×10^6
64	36	3.9×10^9	1.2×10^8	1.9×10^4	59	2.9×10^6
45	45	1.2×10^9	7.2×10^7	5.1×10^2	460	7.9×10^6
48	36	3.5×10^9	9.3×10^7	8.1×10^3	74	3.6×10^6
60	39	3.7×10^9	2.6×10^8	5.1×10^3	99	2.4×10^6

The employed symbolic algorithms are implemented in *Supremica* and run on a standard desktop (2.66 GHz Intel Core Quad CPU, 10GB RAM) running Windows 7.

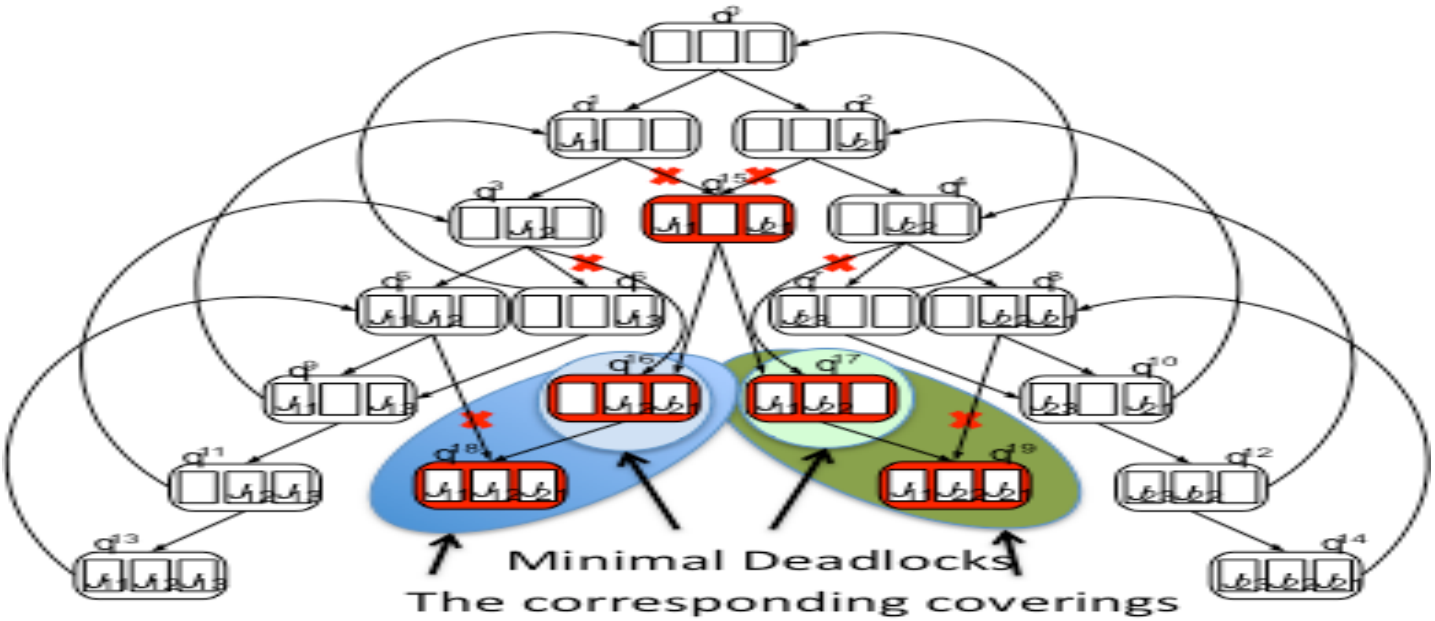
Petri net – based modeling and analysis of the considered RAS

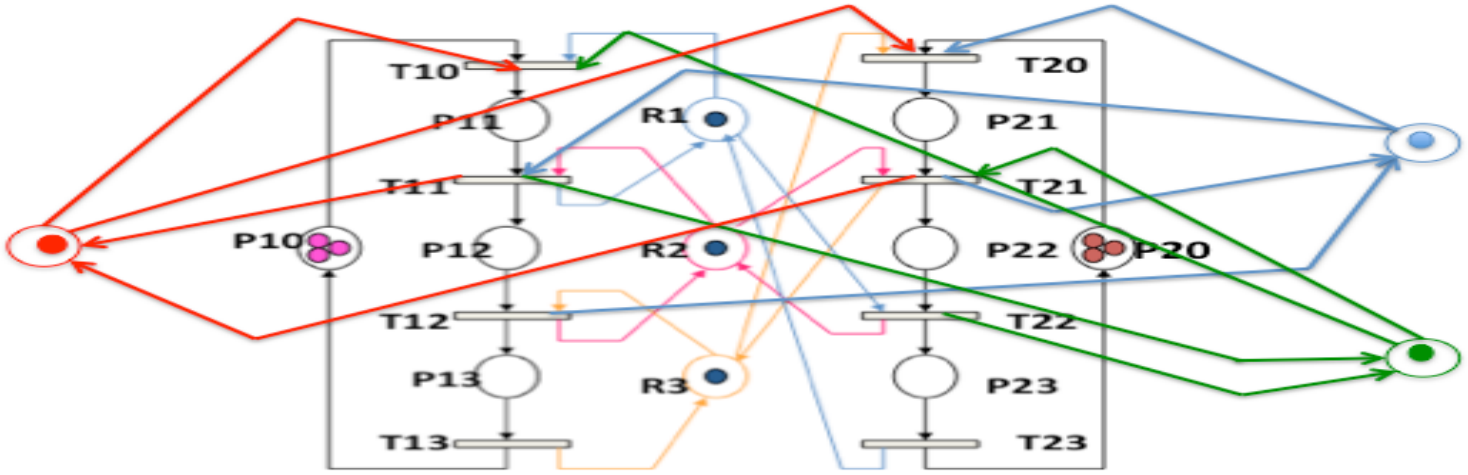




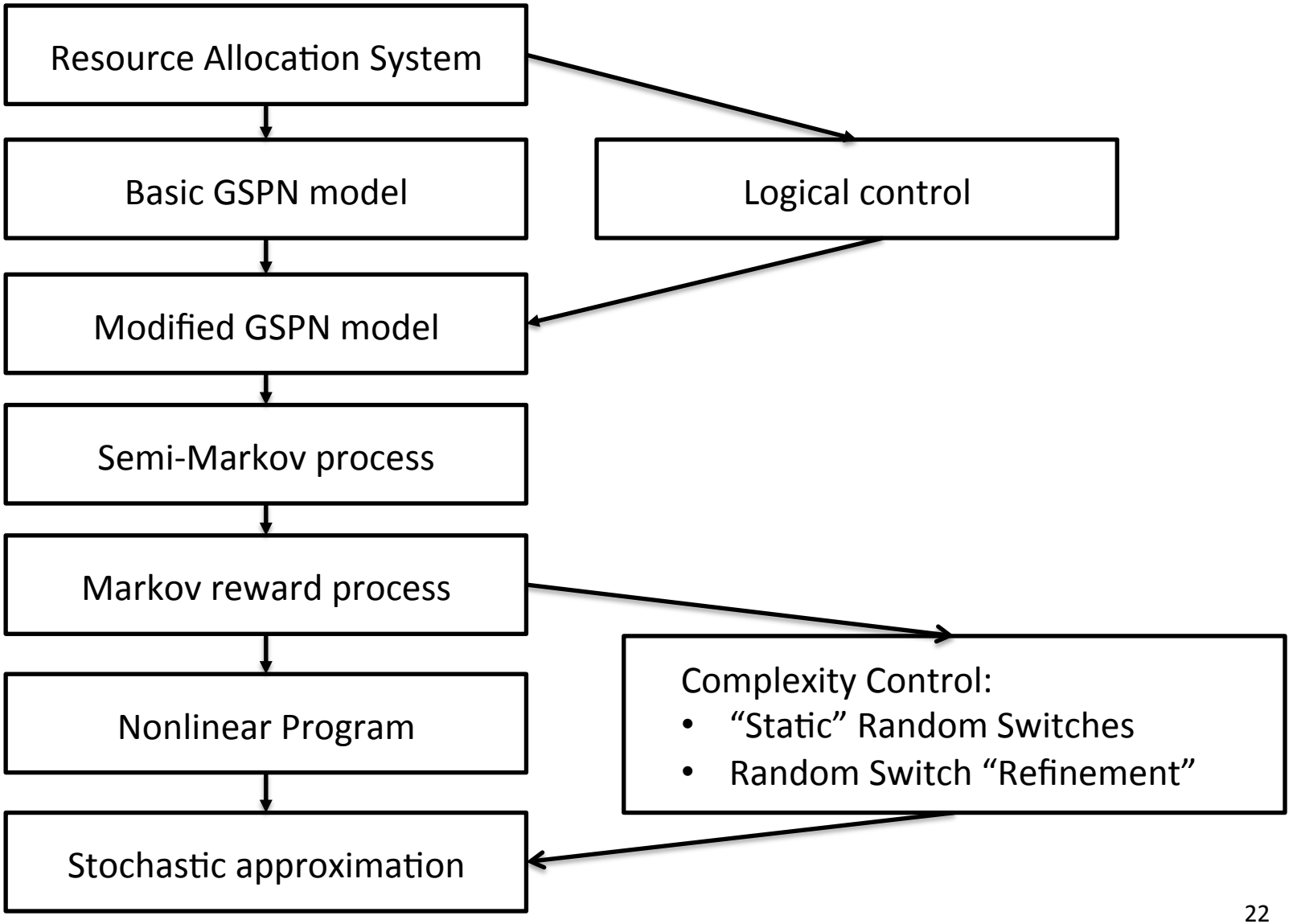
1. Allow for explicit modeling of the RAS structure, and thus, they can support effectively structural analysis.
2. RAS deadlock is interpreted through the structural concept of "empty (or, more generally, deadly marked) siphon".
3. For any given marking, the maximal empty (or deadly marked) siphon is polynomially detectable (another manifestation of the polynomial detectability of RAS deadlock).
4. The logic of the siphon detection algorithm(s) can be translated to a Mixed Integer Programming formulation that can be extended to an analytical tool for assessing the liveness and reversibility of the RAS-modeling PNs.
5. Can enable an incremental synthesis of the maximally permissive DAP through "monitors".

Example: incremental synthesis of the maximally permissive DAP

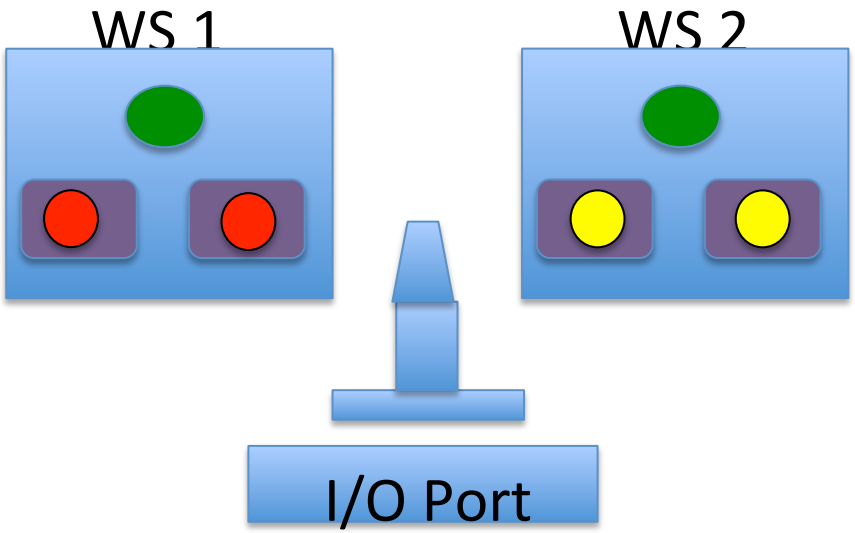




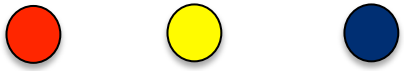
Towards the RAS performance control: a basic methodology



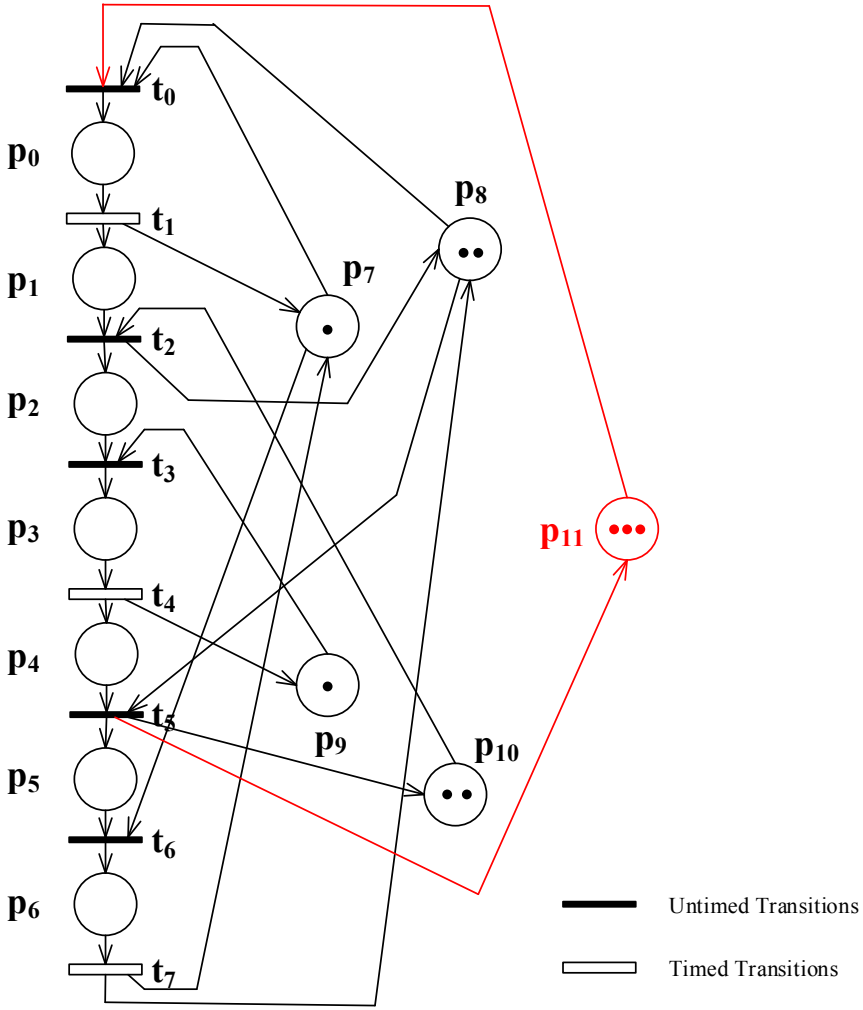
The proposed method through an example: Scheduling of a capacitated re-entrant line



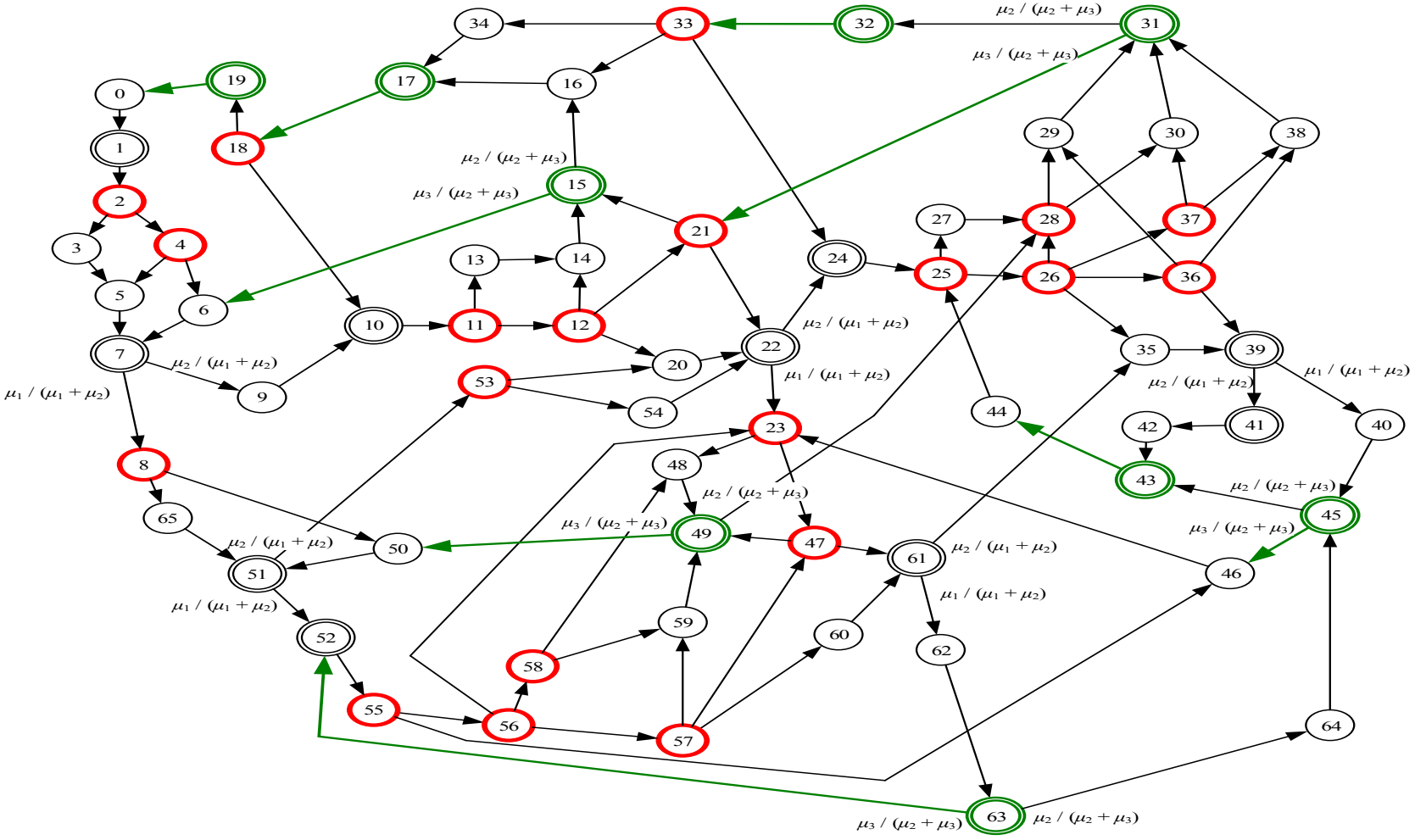
Process route:
WS1 -> WS2 -> WS1



Maximally Permissive DAP
J1 + J2 ≤ 3



The underlying semi-Markov process and the corresponding scheduling problem



Tangible Markings

Vanishing Markings

Tangible Markings with Rewards

Decision Points

Basic Problem Formulation

- Let
  - $\Xi_i = <\xi_{ij}: j=1,\ldots,k(i)>$ the random switch for vanishing marking v_i
  - ε = a minimal degree of randomization in each Ξ_i
  - Δ = the stationary policy defined by the pricing of Ξ_i for all v_i
  - $Q(\Delta)$ = the infinitesimal generator for the CTMC induced by Δ
  - $\pi(\Delta)$ = the stationary distribution of the above CTMC
  - r = the vector collecting the reward rates at the tangible markings of the semi-Markov process
- Problem formulation

$$\max_{\Delta} \eta(\Delta) = \pi(\Delta)^T \bullet r$$

s.t.

$$\pi(\Delta)^T Q(\Delta) = \mathbf{0}^T$$

$$\pi(\Delta)^T \bullet \mathbf{1} = 1.0$$

$$\Xi_i^T \bullet \mathbf{1} = 1.0 \quad \text{for all } v_i$$

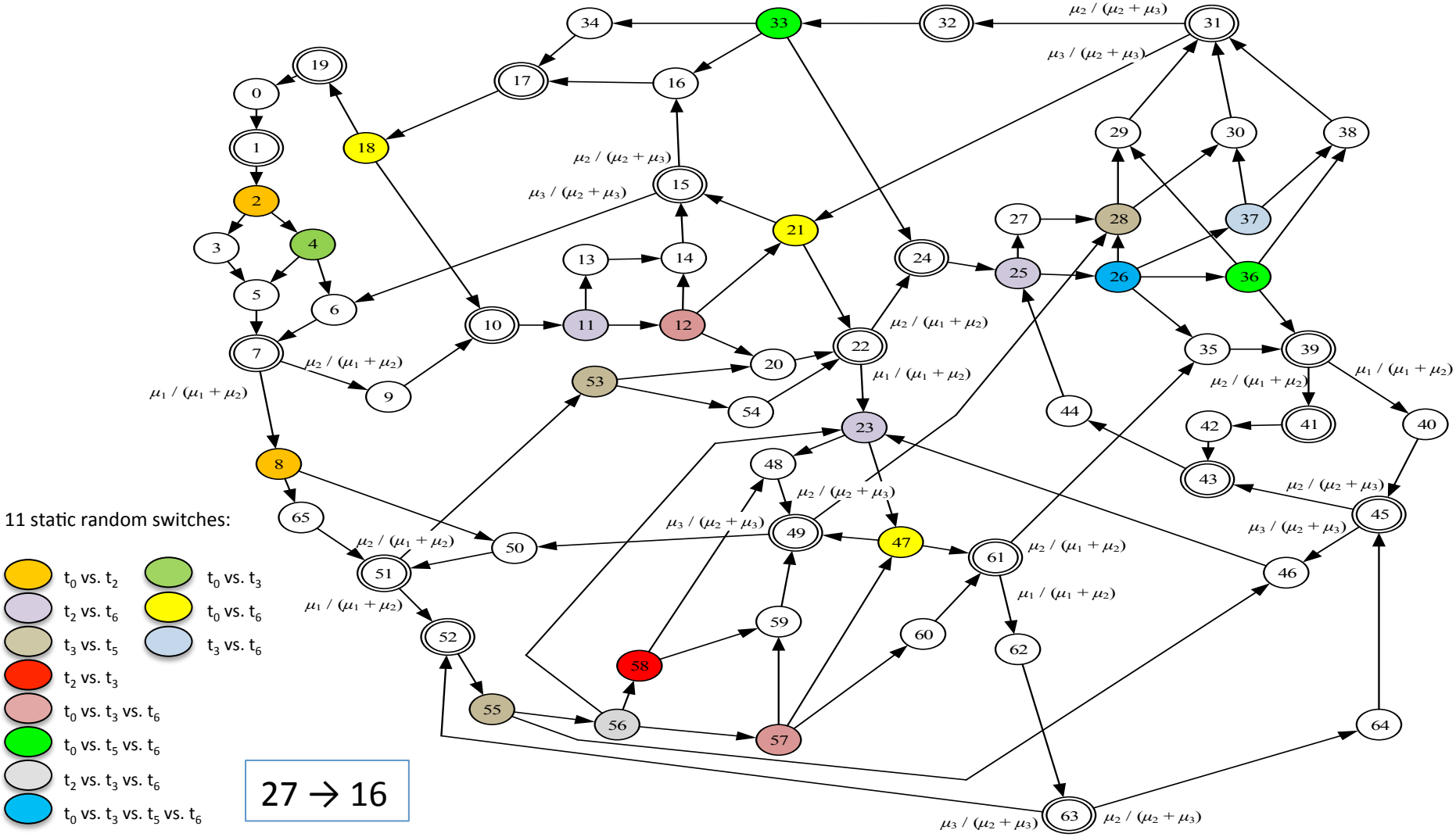
$$\varepsilon \leq \xi_{ij} \quad \text{for all } v_i \text{ and all } j \text{ in } \{1,\ldots,k(i)\}$$

Complexity Considerations

- **Computational Challenge**
  - An explosion of v_i => An explosion of decision variables ξ_{ij}
- **Proposed Solution Methodology**
  - Restrict the problem formulation to policies Δ admitting a more parsimonious representation
    - *Static* random switches: Defined only from the set of the enabled untimed transitions and not by the marking itself, i.e.,

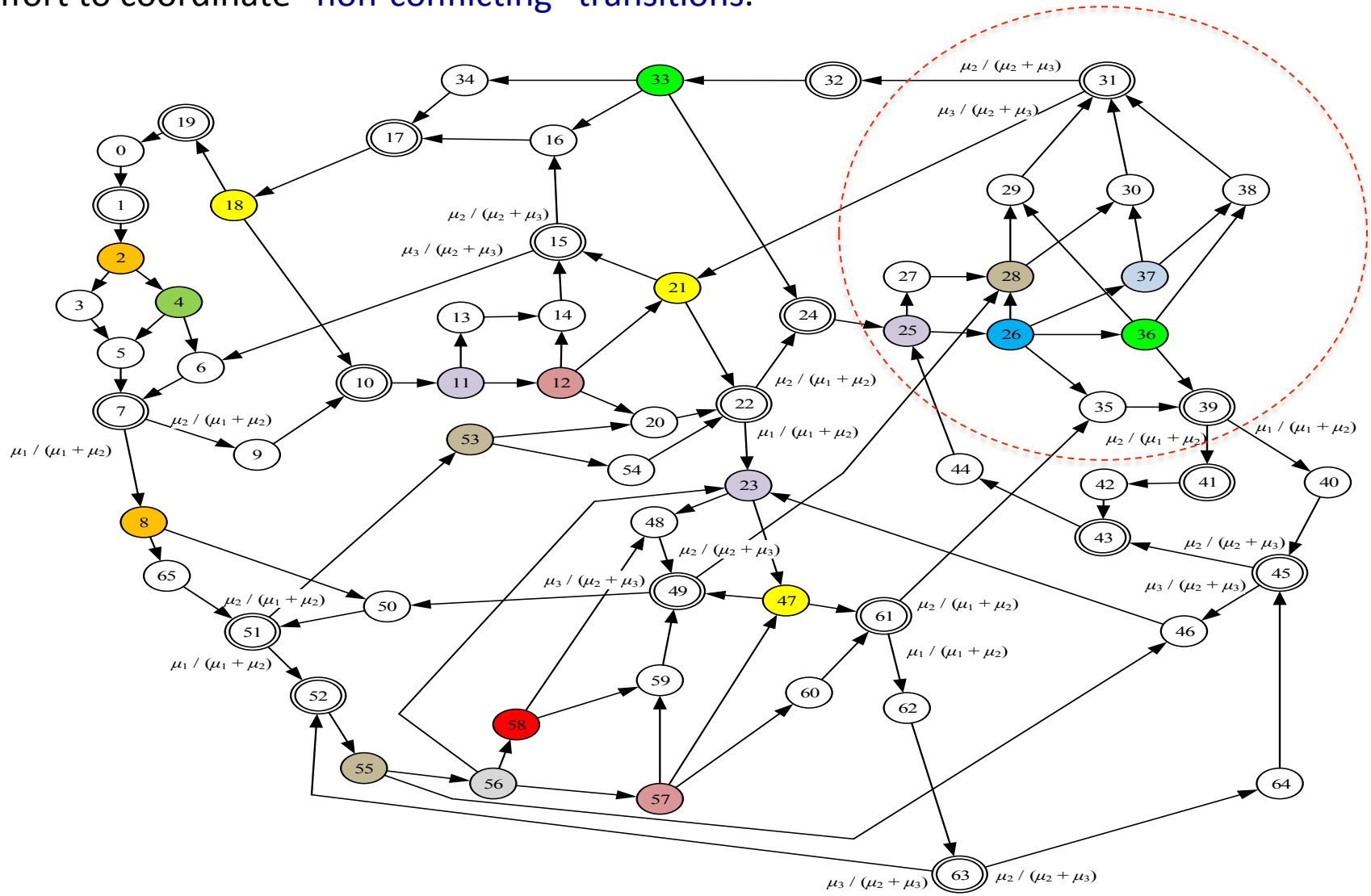
      $\Xi_i = \Xi_j$ if the vanishing markings v_i and v_j activate the same set of untimed transitions
    - The corresponding policy space Δ contains all the "static-priority" policies
    - Hence, in the manufacturing setting, this scheme can optimize over all the possible combinations of the currently used heuristics at the different stations
    - Mathematically, the proposed restriction corresponds to a state space aggregation
    - Hence, we can refine the obtained solution through (partial) disaggregation

Example (cont.): static random switches and the induced state aggregation



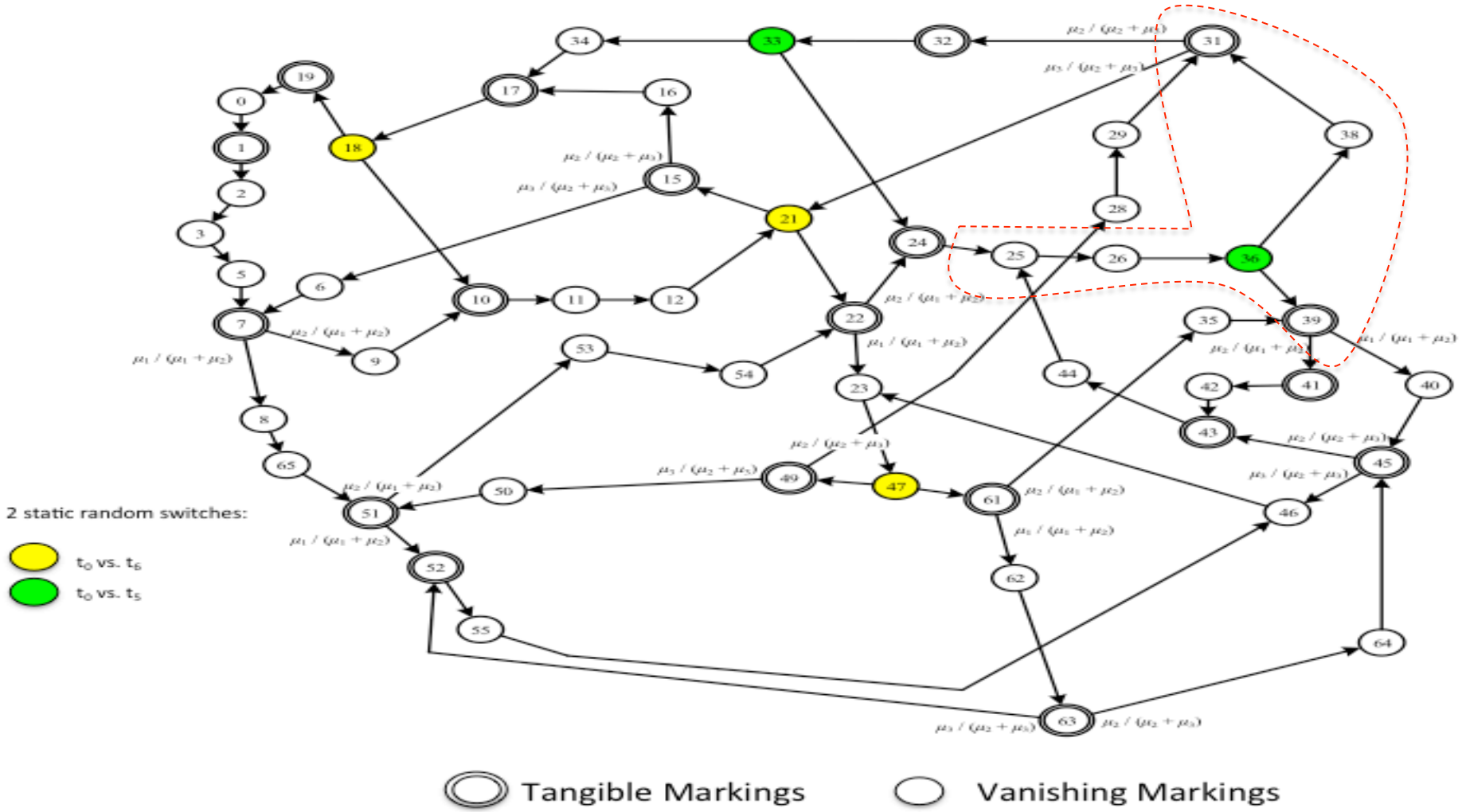
Random switch refinement

Eliminate "redundancies" in the connectivity of the tangible markings that might result from an effort to coordinate "non-conflicting" transitions.



Random switch refinement (cont.)

Eliminate "redundancies" in the connectivity of the tangible markings that might result from an effort to coordinate "non-conflicting" transitions.



A "simulation optimization" strategy (for the uniformized process)

- The modified scheduling problem is solved through **stochastic approximation**.
- Let
  - P(Ξ) = the one-step transition probability matrix for the uniformized MC that corresponds to the policy Δ defined by the pricing of Ξ
  - g(Ξ) = the corresponding relative value function

- A sensitivity formula from Markov reward process theory:

$$\frac{\partial \eta(\Xi)}{\partial \xi} = \pi(\Xi)^T \cdot \frac{\partial}{\partial \xi} P(\Xi) \cdot g(\Xi)$$

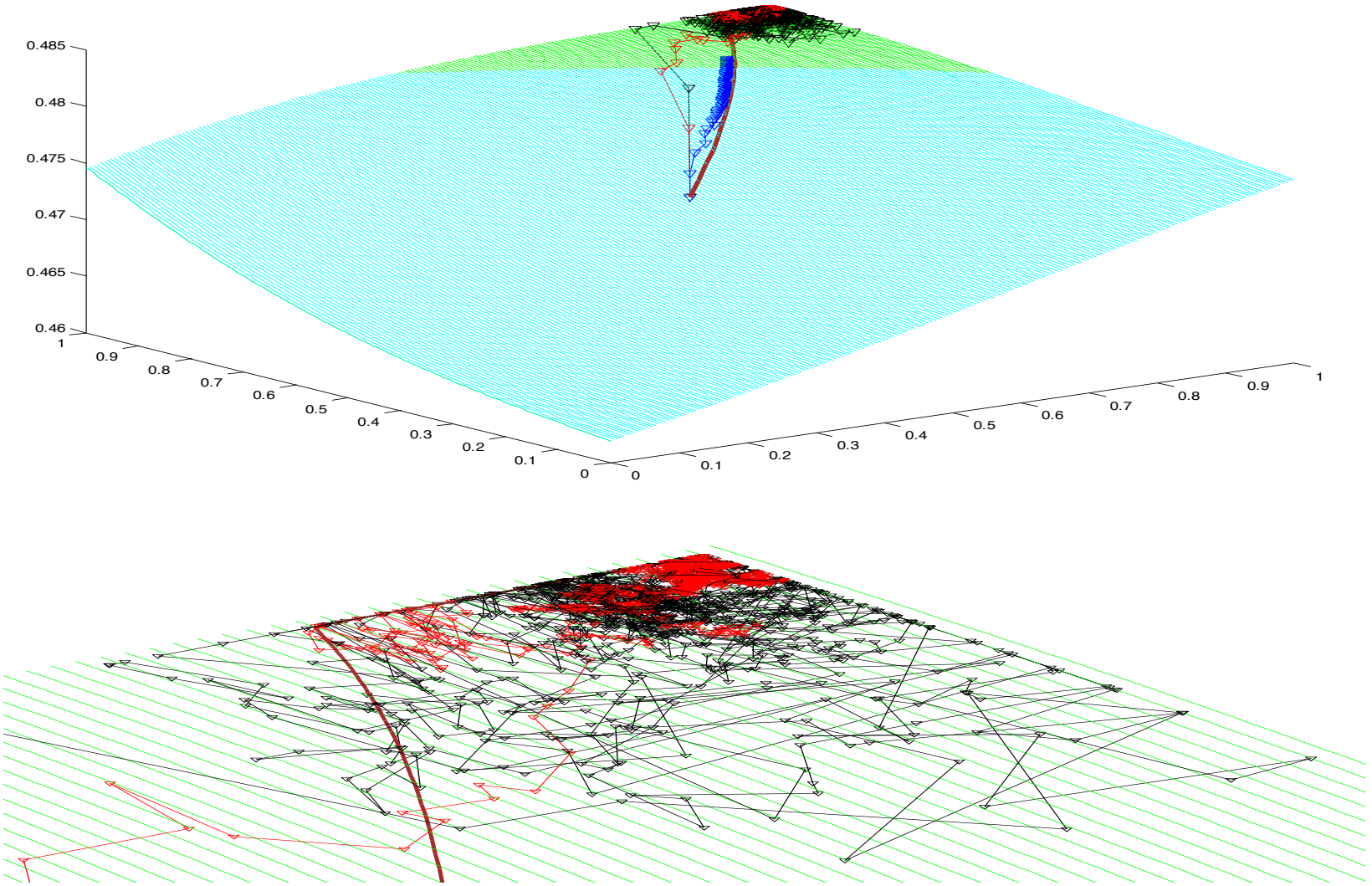
- A gradient estimator:

$$\frac{\partial \eta(\Xi)}{\partial \xi} = \frac{E[\sum_{k=u_\nu}^{u_{\nu+1}-1} (r(m_k) - \eta(\Xi)) \cdot \Lambda_k(\Xi)]}{E[u_{\nu+1} - u_\nu]}$$

- where:

$$\Lambda_k(\Xi) = \sum_{j=u_{\nu(k)}}^{k} \frac{\frac{\partial}{\partial \xi} p(m_{j-1}, m_j; \Xi)}{p(m_{j-1}, m_j; \Xi)}$$

Example (cont.): the converging behavior of the SA algorithm



Concluding Remarks

- The RAS modeling framework is a powerful abstraction capturing the complex resource allocation dynamics that take place in a very broad range of technological applications.
- These dynamics must be controlled for, both, logical correctness and performance; this suggests a natural decomposition to two major sub-problems.
- DES theory provides pertinent modeling frameworks for the representation of the dynamics addressed by each sub-problem.
- The analytical power of the standard DES models has been substantially augmented through their specialization in the RAS domain.
- In particular, the RAS logical control problem is very extensively researched, and the corresponding results can provide optimal solutions for most practical realizations of the considered RAS.

Remaining Challenges and Further Opportunities

- Strengthen the existing methodology for the solution of the performance control problem:
  - More “robust” SA algorithms w.r.t. the detection of improving directions and the applied termination criteria.
  - Development of efficient single-sample-path versions of the current SA algorithms for real-time computation / “learning” of the optimal policy.
  - Identification of new pertinent policy spaces.
- Consideration of additional (e.g., “fairness”) constraints on the RAS behavior.
- Exploration of distributed and/or decentralized approaches.
- Consideration of “resource capacity losses” due to failure or other disruptions.
- “Packaging” of the existing theory in a set of computational tools that will enable its effective implementation in various practical applications.
- Development / training of control engineers with the necessary background to effectively use and promote the presented developments.

Thank you!

The theoretical foundations of the presented framework