USING MARKOV CHAIN AND GRAPH THEORY CONCEPTS TO ANALYZE BEHAVIOR IN COMPLEX DISTRIBUTED SYSTEMS

Christopher Dabrowski Fern Hunt

Information Technology Laboratory
U.S. National Institute of Standards and Technology
Gaithersburg MD

Presented at

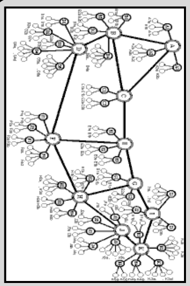
The 23rd European Modeling and Simulation Symposium

Rome, Italy
Tuesday, September 13, 2011

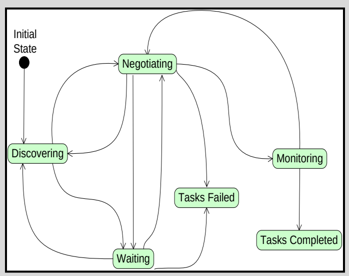
Summary

Goal: To develop scalable modeling tools for monitoring complex distributed systems and predicting catastrophic performance degradations.

- **Use *Discrete Time Markov Chain* (*DTMC*):**
  - Develop *time-inhomogeneous* model of system behavior.
  - Perturb DTMC *transition probability matrices* (TPMs) to simulate alternative system evolutions → ***Identify failure scenarios***



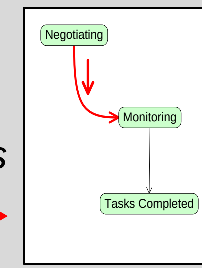
System to state model



State model to perturbable TPM

	Initial	Wait	Disc	Ngt	Mon	Comp	Failed
Initial	0.9697	0	0.0303	0	0	0	0
Wait	0	0.8363	0.0673	0.0918	0	0	0.0046
Disc	0	0.0355	0.6714	0.2931	0	0	0
Ngt	0	0.4974	0.0182	0.2882	0.1961	0	0.0001
Mon	0	0	0	0.0003	0.9917	0.0080	0
Comp	0	0	0	0	0	1.0	0
Failed	0	0	0	0	0	0	1.0

Identify failure scenarios



Problem: To design an efficient approach for analyzing DTMCs

Solution approach: Use minimal s-t cut set analysis to identify ***critical state transitions*** in a directed graph of a DTMC & relate to ***failure scenarios***

– Introduce algorithms that ***reduce search space*** to find minimal s-t cut sets

→ ***Use of combination of analysis techniques not reported before***

Outline

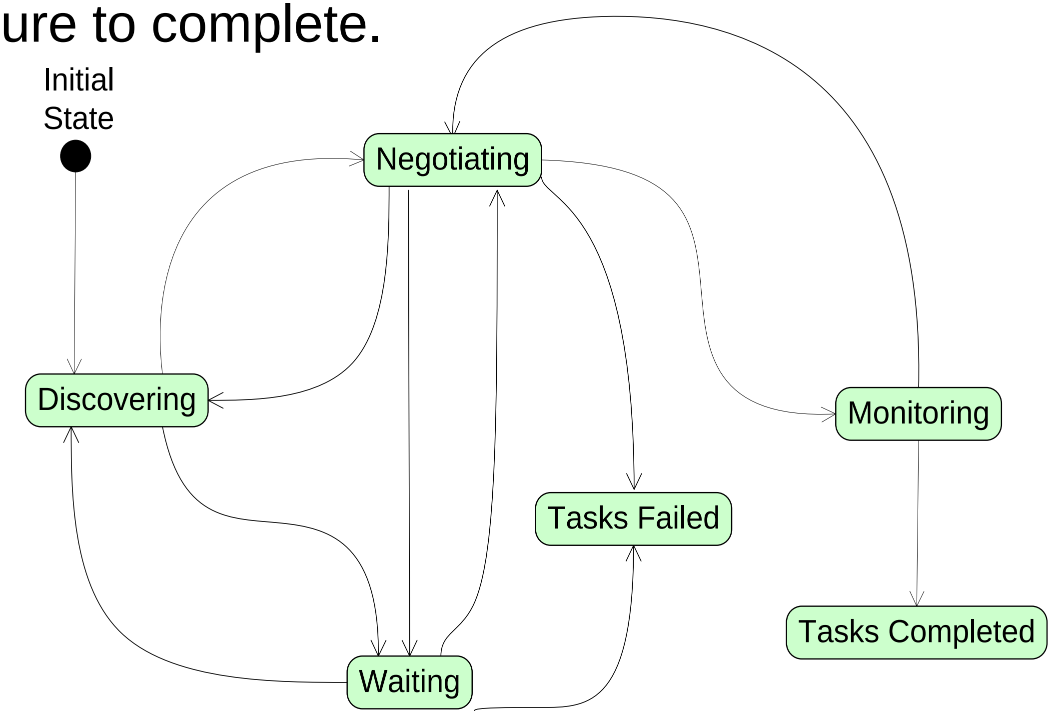
1. ***DTMC concepts and model development***
2. Perturbing a DTMC to identify a failure scenario
3. Using minimal s-t cut set analysis and node contraction to reduce search for failure scenarios
4. Conclusions

State model of a grid computing system

- Basis → Large-scale discrete-event simulation of grid computing system (Mills and Dabrowski 2008)
  - Grids "rent" compute resources – CPUs, memory, disk

- **FOCUS: *Lifecycle of a grid system task -*** stages in processing of grid task
  - Each state represents phase in processing of grid task
  - **Tasks Completed –** successful completion of a task
  - **Failed State** – failure to complete.

State Diagram of Task Lifecycle



Building a Discrete Time Markov Chain (DTMC) model

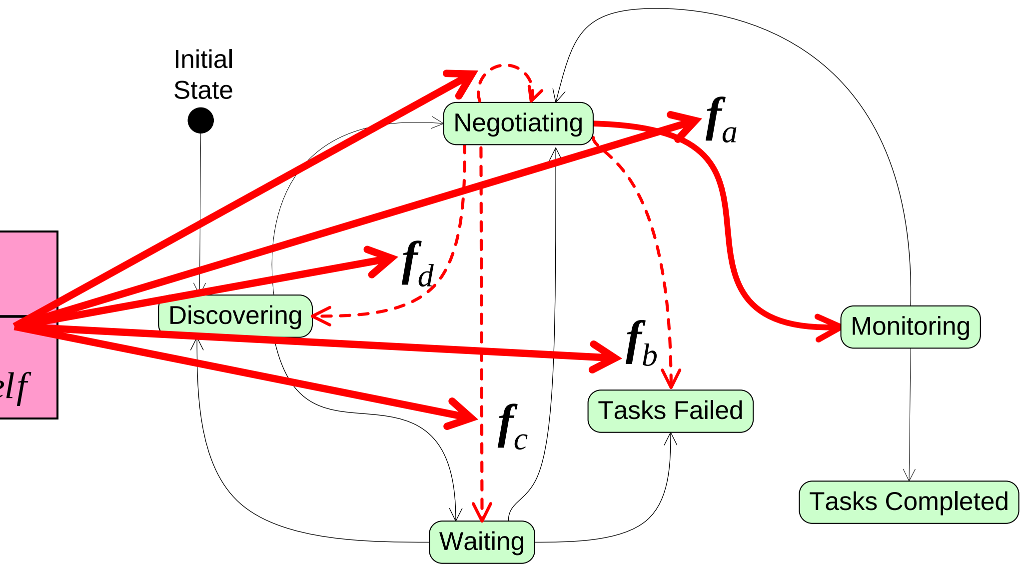
- *Markov chains* are state models where probability of transition from one state to another does not depend on past history: $\Pr(X_{n+1} = x | X_n = x_n, \ldots, X_1 = x_1) = \Pr(X_{n+1} = x | X_n = x_n)$ for sequence of states X_n, X_{n+1}, X_{n+2}…….
- In a *discrete time* Markov chain, system evolves in discrete time steps.
- Probability state i transitions to state j, $\boldsymbol{p_{ij}}$, is the proportion of total number of transitions from state i to other states, where $\boldsymbol{f_{ij}}$ are frequencies. Note: if $i = j$, a ***self transition*** occurs.

$$p_{ij} = \frac{f_{ij}}{\sum_{k=1}^{n} f_{ik}}$$

Observe system (large scale simulation) and obtain frequencies for all transitions

Example:

$$p\,(\text{Negotiating} \rightarrow \text{Monitoring}) = \boxed{\frac{f_a}{f_a + f_b + f_c + f_d + f_{self}}}$$



→ ***Produce Transition Probability Matrix (TPM)***

Result is set of TPMs for *m* time periods

- ***Key Concept:*** Observation of system over time yields series of TPMs for ***m*** successive time periods → ***a piece-wise homogenous DTMC*** (Rosenberg, Solan, and Vielle, 2004) → captures change over time.

  Transition frequencies recorded over 7200s time periods in large-scale simulation
- Summary TPMs -- weighted average of m periods

8 hours (5 time periods*)

To

From

	Initial	Wait	Disc	Ngt	Mon	Comp	Failed
Initial	0.9697	0	0.0303	0	0	0	0
Wait	0	0.8363	0.0673	0.0918	0	0	0.0046
Disc	0	0.0355	0.6714	0.2931	0	0	0
Ngt	0	0.4974	0.0182	0.2882	0.1961	0	0.0001
Mon	0	0	0	0.0003	0.9917	0.0080	0
Comp	0	0	0	0	0	1.0	0
Failed	0	0	0	0	0	0	1.0

Similar analysis 640hours (321 periods). See paper

- A DTMC that has ***absorbing states*** (tasks enter and never exit), e.g., Tasks Completed & Tasks Failed → ***absorbing chain***. States that can be re-entered are ***transient states*** (Waiting, Discovering, Negotiating, and Monitoring)

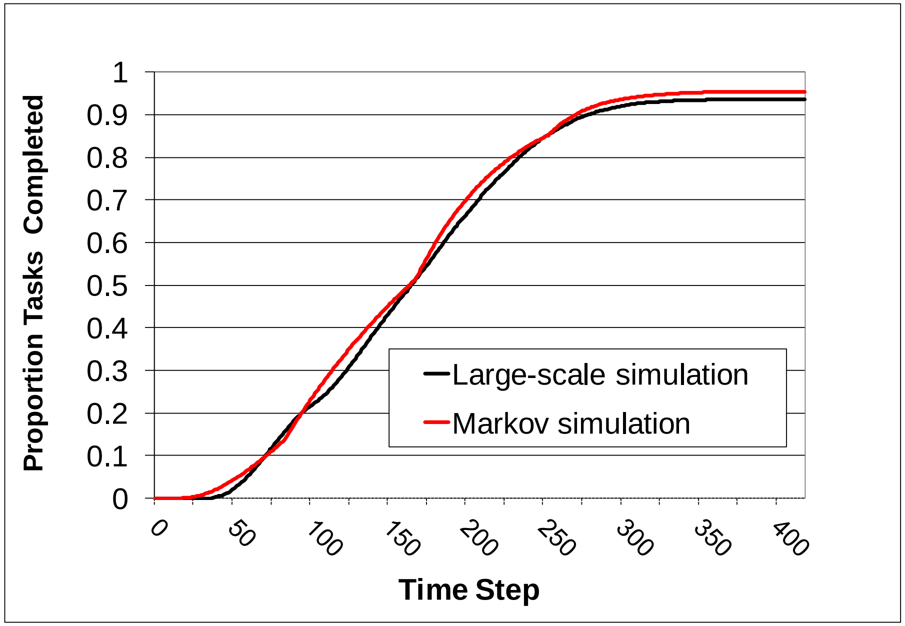
*Extra period for clean-up operations

DTMC simulates system evolution

- Set of TPMs for successive time periods (7200 s)
- System evolves in discrete time steps (85 s per step)
- Vector v_n shows system state at any step n:
  - consists of 7 elements → one for each state
- Matrix multiplication: $Q^T \cdot v_n = v_{n+1}$ with Q_i for related time period.

8-hour period

(421 time steps and 5 TPMs)



End system state vector $\mathbf{v}_{421}$ approximates result of discrete event large-scale simulation, i.e., *Tasks Completed*

Outline

1. DTMC concepts and model development
2. ***Perturbing a DTMC to identify a failure scenario***
3. Using minimal s-t cut set analysis and node contraction to reduce search for failure scenarios
4. Conclusions

TPM perturbation

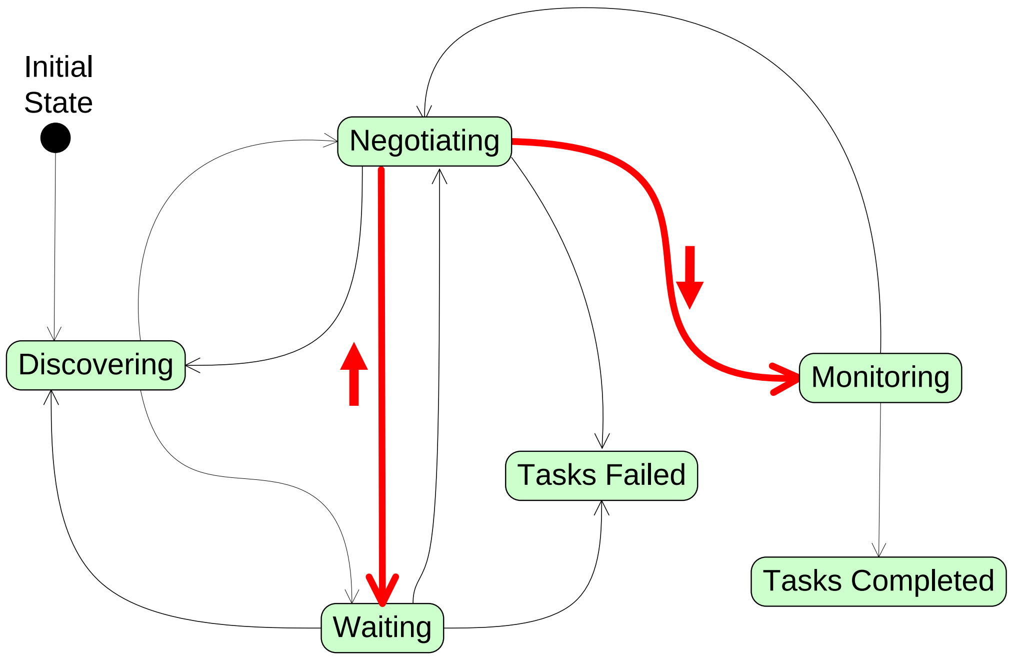
- *Modifying state transition probabilities changes behavior and outcome of Markov simulation*

EXAMPLE:

Decrease *p* (*Negotiating* → *Monitoring*)

Increase *p* (*Negotiating* → *Waiting*)

	Wait	Ngt	Mon
Wait	0.8363	0.0918	0
Ngt	0.4974	0.2882	0.1961
Mon	0	0.0003	0.9917



→ *changes proportion of requests that enter* ***Tasks_Completed*** *absorbing state*

Using critical transitions to predict failure scenarios

- Markov simulation of perturbed critical transitions over multiple time periods (time inhomogeneous evolution) drives down performance
- *Can be related to failure scenarios:*

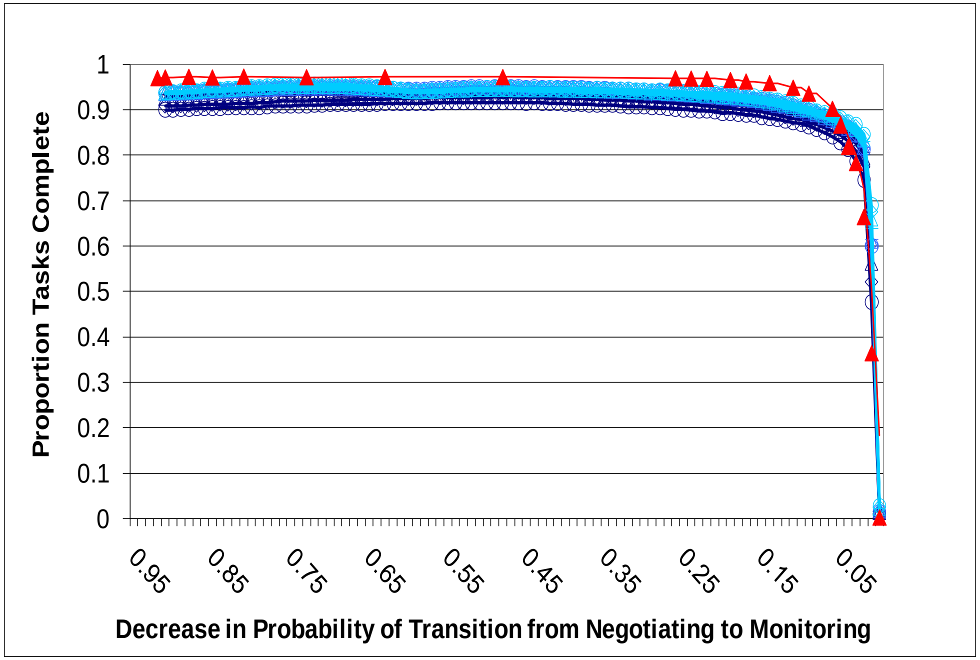
    *ex. System-wide f*ailure of Negotiation components due to spread of Trojan virus

EXAMPLE:

Decrease *p* (*Negotiating* → *Monitoring*)

Increase *p* (*Negotiating* → *Waiting*)

	Wait	Ngt	Mon
Wait	0.8363	0.0918	0
Ngt	↑0.4974	0.2882	↓0.1961
Mon	0	0.0003	0.9917



→ Predict the performance of the system being modeled.

Computability of finding critical state transitions

Unfortunately, there may be many perturbation combinations to examine in a large problem

- Developed *exhaustive perturbation algorithm* (Dabrowski and Hunt 2009) which iterates over rows of TPM representing transient states. For all columns in each row,
  - raises the transition probability of one column
  - lowers transition probabilities of one or more other non-zero columns in the same row.
- See (Dabrowski and Hunt, 2011) for analysis of larger DTMC in which multiple rows must be perturbed together to find combinations of state transitions which together are critical increases.

→ ***Exhaustive search over all perturbation combinations infeasible for larger problems***

Outline

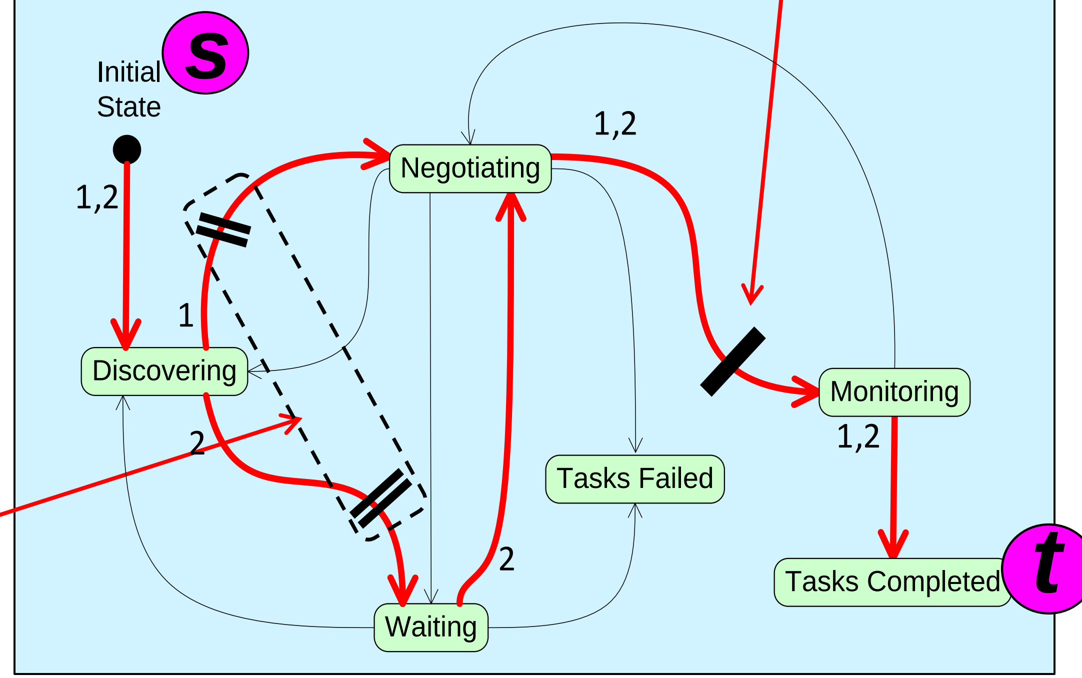
1. DTMC concepts and model development
2. Perturbing a DTMC to identify a failure scenario
3. ***Using minimal s-t cut set analysis and node contraction to reduce search for failure scenarios***
4. Conclusions

Minimal s-t cut set analysis

- A DTMC is a directed graph
- Minimal s-t cut set: edges (transitions) that disconnect all paths from vertex ***s*** (*Initial* state) to vertex ***t*** -- desired absorbing states *Tasks_Complete*

Single-transition cut

*Cut sets contain **critical Transitions** where perturbation reduces performance*



Multiple-transition cut

For two paths from s to t, there are 3 single-transition s-t cut sets and 2 two-transition s-t cut sets. For related discussion of trap states, see paper.

Result of applying minimal s-t cut set analysis

- ***Reduces perturbation combinations to examine to focus on most critical***
- ***2x magnitude less computation time*** over exhaustive perturbation algorithm
- Finds all critical transitions found by exhaustive perturbation, including those involving > 1 state, verified by large-scale simulation (bolded and shaded entries)
  - → All related to failure scenarios. See (Dabrowski and Hunt 2009)

Results of exhaustive perturbation of TPMs and minimal s-t cut set analysis for the 8- and 640-hour cases

(a) *row* = *Discovering*

	Element reduced→0	Element raised	Proportion of *Tasks Complete* 8-hour	Proportion of *Tasks Complete* 640-hour	s-t cut exists
1	*Waiting*	*Discovering*	0.957	0.935	No
2	*Waiting*	*Negotiating*	0.959	0.935	No
3	*Discovering*	*Waiting*	0.939	0.935	No
4	*Discovering*	*Negotiating*	0.963	0.935	No
5	*Negotiating*	*Waiting*	0.894	0.933	No
6	*Negotiating*	*Discovering*	0.651	0.932	No

(b) *row* = *Waiting*

	Element reduced→0	Element raised	Proportion of *Tasks Complete* 8-hour	Proportion of *Tasks Complete* 640-hour	s-t cut exists
1	*Waiting*	*Discovering*	0.974	0.937	No
2	*Waiting*	*Negotiating*	0.981	0.939	No
3	*Discovering*	*Waiting*	0.937	0.934	No
4	*Discovering*	*Negotiating*	0.963	0.936	No
5	*Negotiating*	*Waiting*	0.818	0.843	No
6	*Negotiating*	*Discovering*	0.939	0.932	No

(c) *row* = *Negotiating*

	Element reduced→0	Element raised	Proportion of *Tasks Complete* 8-hour	Proportion of *Tasks Complete* 640-hour	s-t cut exists
1	*Waiting*	*Discovering*	0.974	0.937	No
2	*Waiting*	*Negotiating*	0.985	0.938	No
3	*Waiting*	*Monitoring*	1.000	0.939	No
4	*Discovering*	*Waiting*	0.954	0.935	No
5	*Discovering*	*Negotiating*	0.957	0.935	No
6	*Discovering*	*Monitoring*	0.967	0.936	No
7	*Negotiating*	*Waiting*	0.923	0.931	No
8	*Negotiating*	*Discovering*	0.941	0.933	No
9	*Negotiating*	*Monitoring*	0.988	0.938	No
10	*Monitoring*	*Waiting*	**0.000**	**0.000**	Yes
11	*Monitoring*	*Discovering*	**0.000**	**0.000**	Yes
12	*Monitoring*	*Negotiating*	**0.000**	**0.000**	Yes

(d) *row* = *Monitoring*

	Element reduced→0	Element raised	Proportion of *Tasks Complete* 8-hour	Proportion of *Tasks Complete* 640-hour	s-t cut exists
1	*Negotiating*	*Monitoring*	0.982	0.937	No
2	*Negotiating*	*Tasks Comp*	0.982	0.938	No
3	*Monitoring*	*Negotiating*	**0.028**	**0.186**	Yes
4	*Monitoring*	*Tasks Comp*	0.980	0.949	No
5	*Tasks Comp*	*Negotiating*	**0.001**	**0.006**	Yes
6	*Tasks Comp*	*Monitoring*	**0.002**	**0.016**	Yes

(e) *row* = *Initial*

	Element reduced→0	Element raised	Proportion of *Tasks Complete* 8-hour	Proportion of *Tasks Complete* 640-hour	s-t cut exists
1	*Discovering*	*Initial*	***0***	***0***	Yes
2	*Initial*	*Discovering*	0.970	0.988	No

Monitoring → Tasks Completed

Negotiating → Monitoring

Initial → Discovering

Tractability of minimal s-t cut set analysis

However, number of potential s-t cut sets in large directed graphs poses barriers

- Implemented minimal s-t cut set enumeration algorithm described in (Provan and Shier 1996)
  - Complexity is *O* |*E*| for each s-t cut set that exists, where |*E*| is the number of edges in the graph. (Other algorithms surveyed were similar)

- Example: One Markov chain directed graph of order 40 contained > 4×10^8 minimal s-t cut sets even though related TPM was sparse (required > 193 hours to compute).

→ ***Minimal s-t cut set enumeration algorithms may not be computationally efficient for large Markov chain problems***

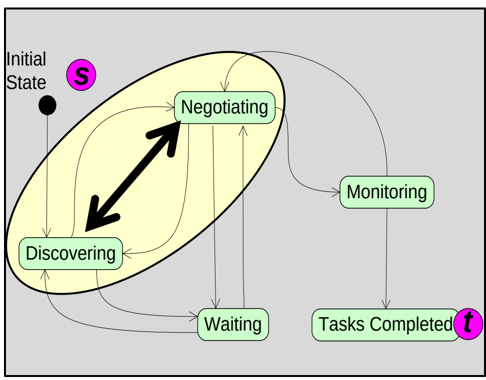
Using the node Contraction algorithm to find minimal s-t cut sets

Finds minimal s-t cut sets probabilistically → though not guaranteed to find all
(Also, <u>*finds multiple transition s-t cut sets*</u>)

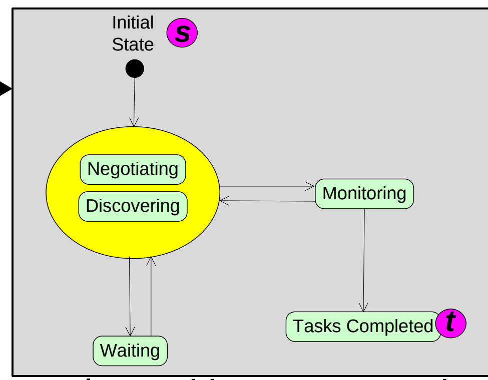
<u>Overview of steps in single repetition</u>.

Psuedo code in (Dabrowski, Hunt, and Morrison, 2010)

Minimal s-t cut set

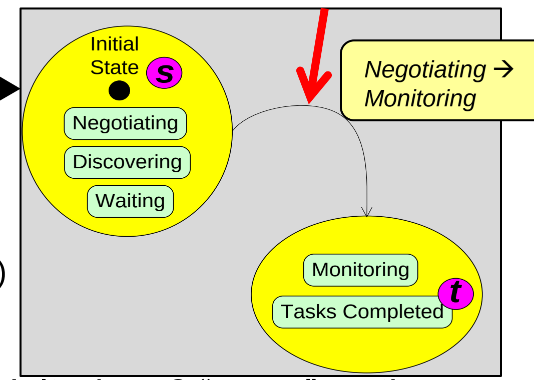


Randomly choose 2 connected vertices to contract.



Replace with new vertex that assumes edges of both contracted vertices

Repeat steps. (Ensure ***s*** and ***t*** are not in same contracted vertex)



Halt when 2 “mega” vertices remain. Assumed edges between “mega” vertices are cut set.

- Because selection of vertices to contract is random, multiple repetitions produce different results, yielding collection of cut sets
- For single repetition, best algorithms find a minimal s-t cut set in $O\ |V|^2$ *in undirected graphs* where $|V|$ is the number of vertices (Karger and Stein, 1996).
- Computational cost can be bounded by limiting number of repetitions

Experimental results of applying node contraction

- Chose four large problems, for which minimal s-t cut sets could be computed by cut set enumeration algorithm of (Provan and Shier 1996) in reasonable time.
- Compared node contraction and cut set enumeration algorithm to see if node contraction could find the ***most critical cut sets***.
- Criticality of ***minimal s-t cut sets*** determined by three ranking criteria (sorts A, B, C), based on idea that cut sets with fewest transitions were most critical → *related to most likely failure scenarios*.

		Minimal s-t cut set enumeration		Proportion (in %) of 100 top-ranked minimal s-t cut sets ranked by criteria A, B that were found by the node contraction algorithm							
Number	Order	Number of cut sets	Time (in hours)	After 10,000 repetitions				After 100,000 repetitions			
				Time	Sort A	Sort B	Sort C	Time	Sort A	Sort B	Sort C
1	50	530,432	332 s	640 s	80	100	96	---	---	---	---
2	50	28,230,288	21.6	171 s	93	98	65	1710 s	99	100	99
3	50	27,242,634	36.0	218 s	67	100	100	2288 s	88	100	100
4	40	422,060,801	193.6	106 s	30	80	62	1051 s	37	100	100

Result: node contraction could find most critical cut sets, with exceptions, with further 2x reduction in computational cost

Outline

1. DTMC concepts and model development
2. Perturbing a DTMC to identify a failure scenario
3. Using minimal s-t cut set analysis and node contraction to reduce search for failure scenarios
4. ***Conclusions***

Conclusions

- Approach to finding critical transitions & failure scenarios in DTMCs uses combination of techniques *not previously reported*
  - Time inhomogeneous representation to capture change over time
  - Markov simulation and quantitative performance analysis (thresholds)
  - Minimal s-t cut set analysis

- Results show potential of minimal s-t cut set analysis to identify critical transitions and related failure scenarios at reduced computation cost
  - Generally 2x less than exhaustive perturbation of all combinations in TPM

→ ***Indicates potential for predictive use***

- For larger problems, node contraction algorithm shows potential to find critical transitions through reduced search, though needs further investigation

- Areas of further work
  - Investigate other approaches to finding cut sets in large problems (ex. other node contraction algorithms (Karger and Stein, 1996), min-max flow algorithms, & eigensystem analysis (Hunt, Morrison, and Dabrowski, 2011)
  - Investigate applicability to other domains.

References

Dabrowski, C. and Hunt, F., 2009. Using Markov Chain Analysis to Study Dynamic Behavior in Large-Scale Grid Systems. *Proceedings of the Seventh Australasian Symposium on Grid Computing and e-Research–Volume 99*, pp. 29–40. January 21, Wellington (New Zealand).

Dabrowski, C., Hunt, F. and Morrison, K., 2010. *Improving the Efficiency of Markov Chain Analysis of Complex Distributed Systems.* National Institute of Standards and Technology, Interagency Report 7744.

Hunt, F., Morrison, K. and Dabrowski, C., 2011. Spectral Based Methods That Streamline the Search for Failure Scenarios in Large-Scale Distributed Systems. *Nineteenth IASTAD International Conference on Modeling and Simulation*. June 22–24, Crete (Greece).

Karger, D. and Stein, C., 1996. A New Approach to the Minimum Cut Problem. *Journal of the ACM*, 43, 601–640.

Mills, K. and Dabrowski, C., 2008. Can Economics-based Resource Allocation Prove Effective in a Computation Marketplace? *Journal of Grid Computing*, 6 (3), 291–311.

Rosenberg, D., Solan, E. and Vielle, N., 2004. Approximating a Sequence of Observations by a Simple Process. *The Annals of Statistics*, 32 (6), 2742–2775.