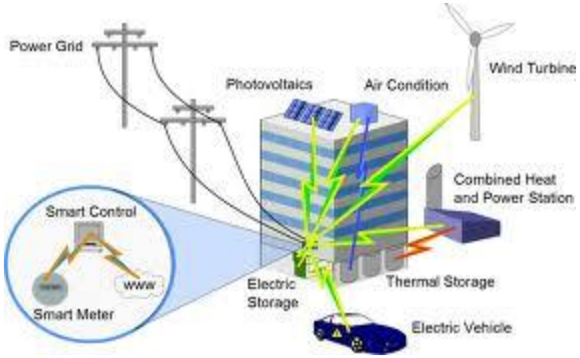
SASS: Self-Adaptation using Stochastic Search

Zack Coker, David Garlan,
Claire Le Goues
School of Computer Science
Carnegie Mellon University

Smart Grids

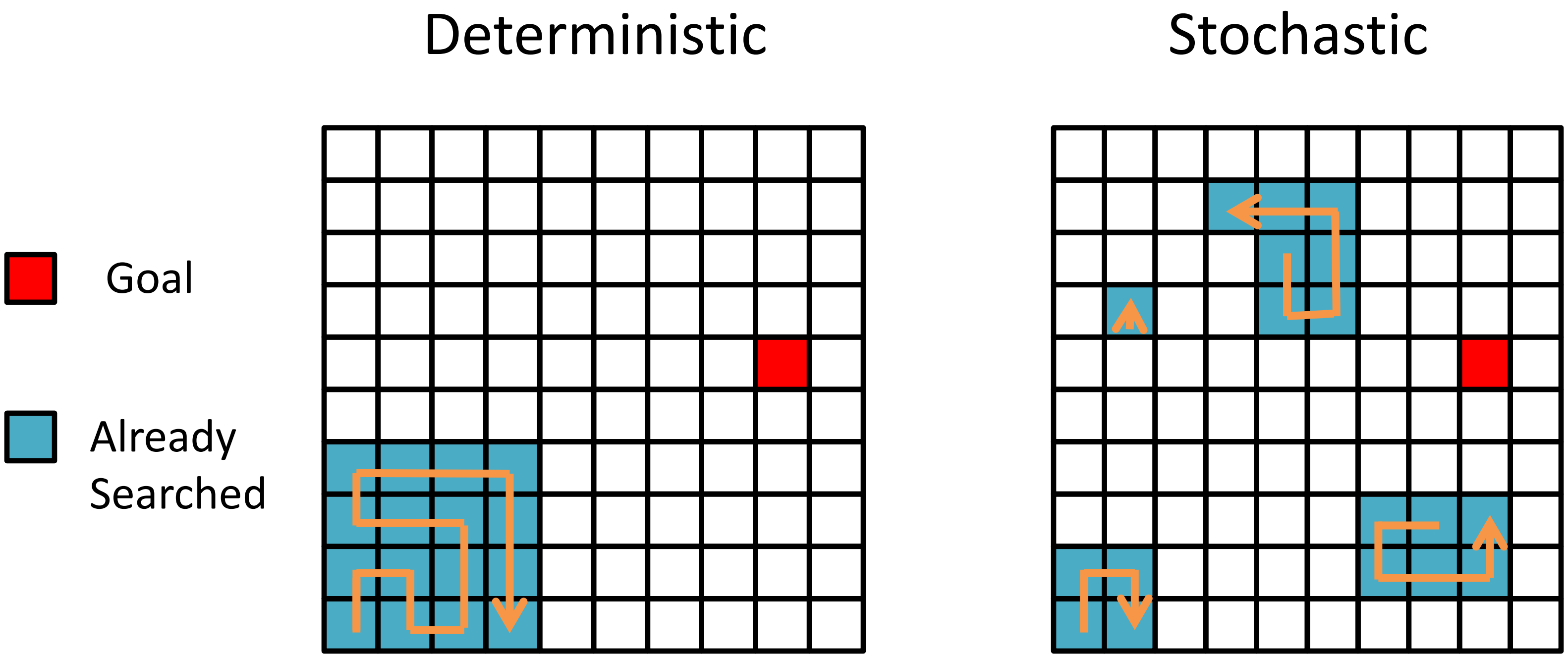
Smart Traffic Systems



Smart Buildings

Smart Farms

Stochastic Search

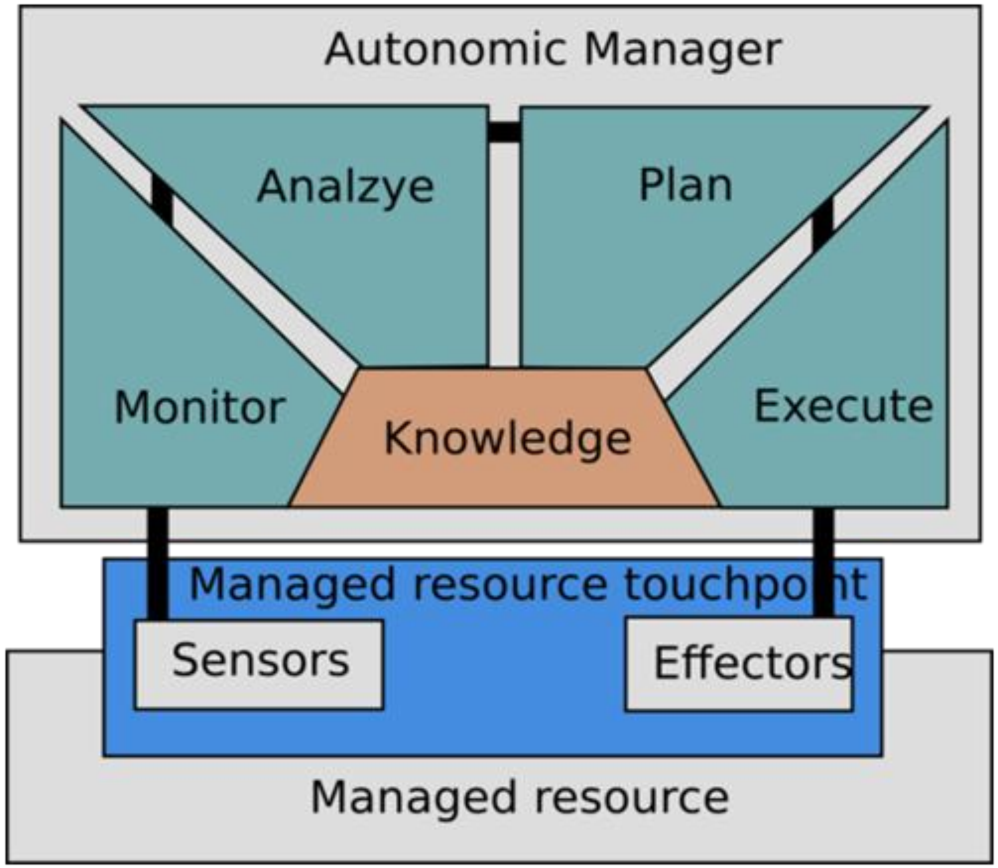


Stochastic Search

Stochastic Search

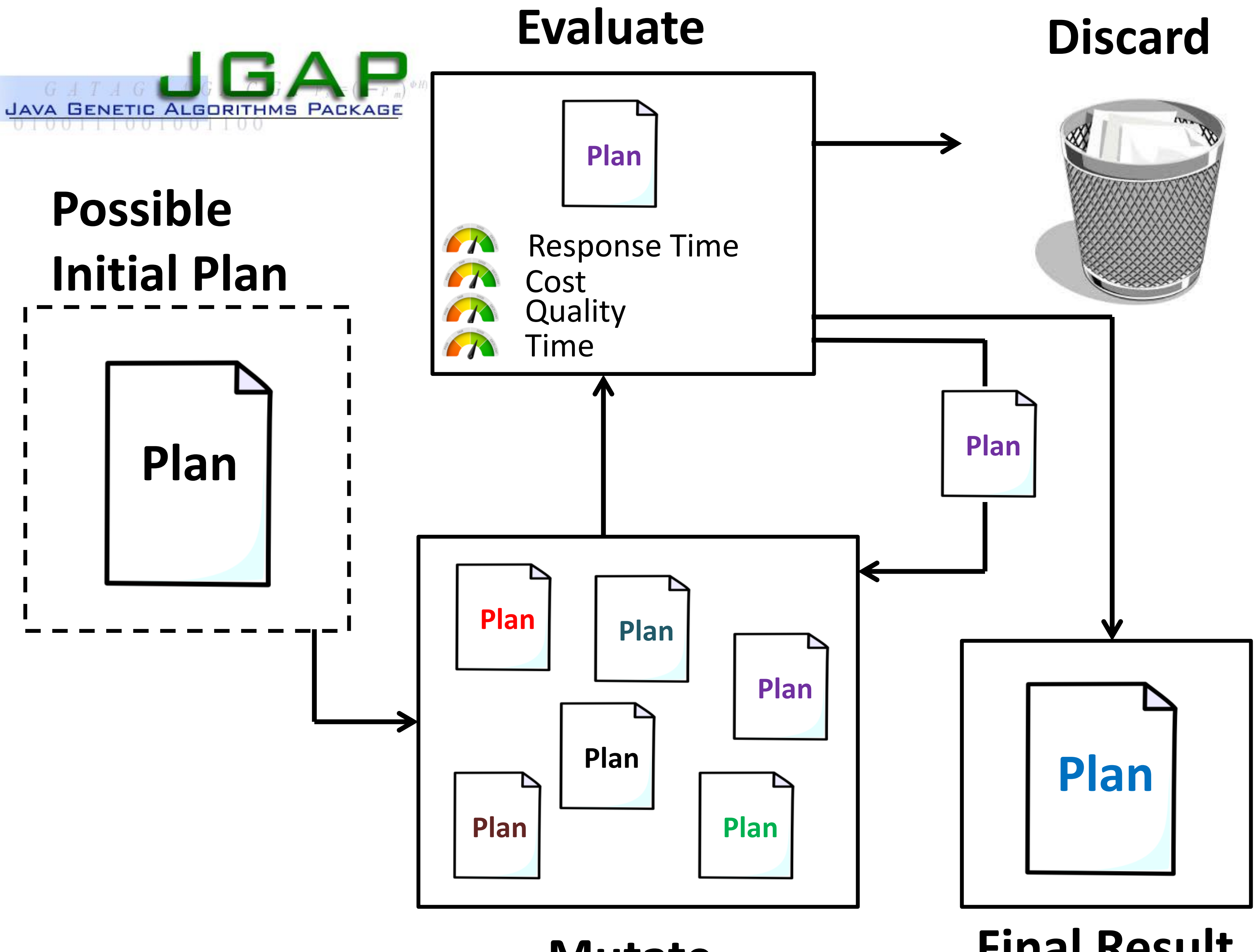
- B. H. C. Cheng, A. J. Ramirez, and P. K. McKinley, “Harnessing evolutionary computation to enable dynamically adaptive systems to manage uncertainty”
- G. G. Pascual, M. Pinto, and L. Fuentes, “Run-time adaptation of mobile applications using genetic algorithms”
- P. Zoghi, M. Shtern, and M. Litoiu, “Designing search based adaptive systems: A quantitative approach”
- M. Harman, Y. Jia, W. B. Langdon, J. Petke, I. H. Moghadam, S. Yoo, and F. Wu, “Genetic improvement for adaptive software engineering (keynote)”

Self-Adaptive System

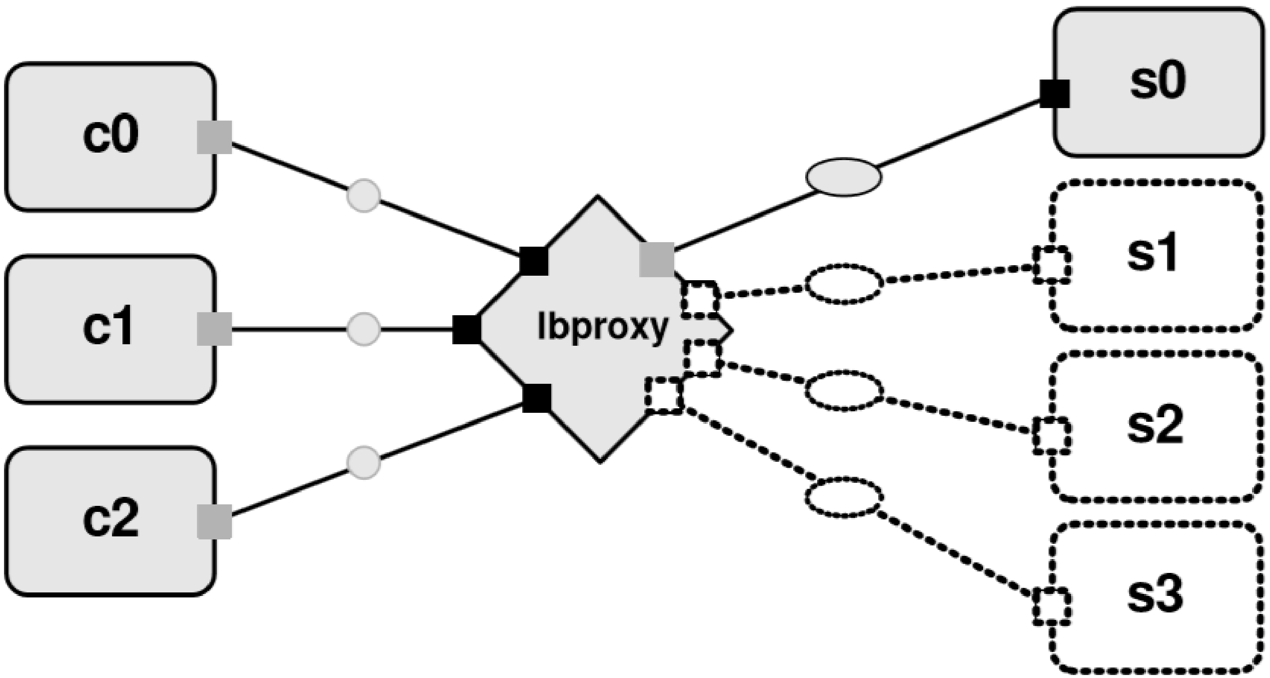


Requirements for Stochastic Techniques

- Possible Solution Creation
  - Randomly create a tree of tactics
- Objective Function
  - Example: -20 $responseTime_{norm}$ – $cost_{norm}$ + $quality_{norm}$ – $time_{norm}$
  - Use PRISM to evaluate the score of each final state in the plan and the probability of reaching each final state


JGAP
JAVA GENETIC ALGORITHMS PACKAGE
Evaluate
Discard
Possible
Initial Plan
Plan
Plan
Response Time
Cost
Quality
Time
Plan
Plan
Plan
Plan
Plan
Plan
Plan
Plan
Mutate
Final Result

ZNN.com

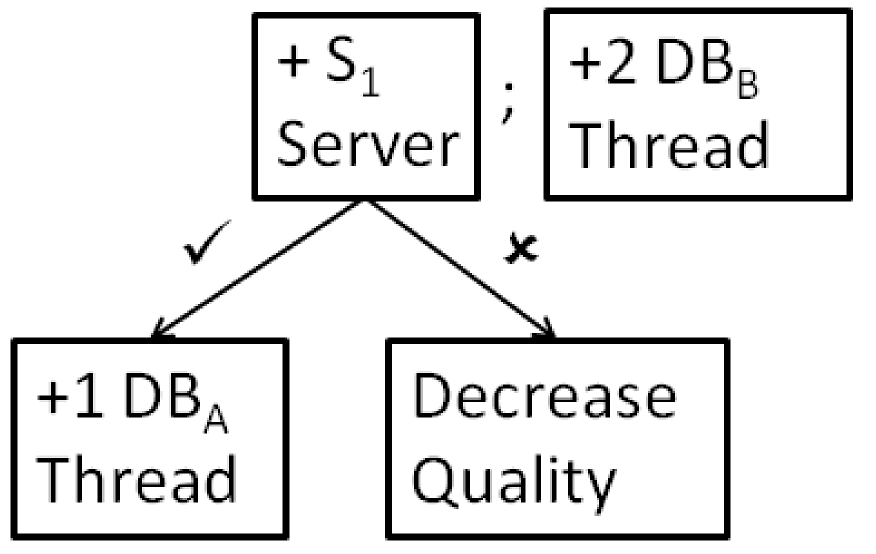


Proof of Concept

Basic Action	Precondition	Cost ($)	Resp. Time	Time	Failure Rate
Add S_1 Server	$S_1 < S_1^{max}$	15	-5	600	0.1
Add S_2 Server	$S_2 < S_2^{max}$	20	-5	600	0.1
Remove S_1 Server	$S_1 > 1$	-15	5	600	0.1
Remove S_2 Server	$S_2 > 1$	-20	5	600	0.1
Add DB_A Thread	$T_A < T_A^{max}$	0	-2	180	0.2
Add 2 DB_B Threads	$T_B < T_B^{max}$-1	0	-1	180	0.2
Increase Quality	Quality set low	0	$2S$	1	0.3
Decrease Quality	Quality set high	0	-$2S$	60	0.3

Plans

[(if (add-server(1))
(add-database-thread(A,1))
(decrease-quality))),
add-database-thread(B,2)]

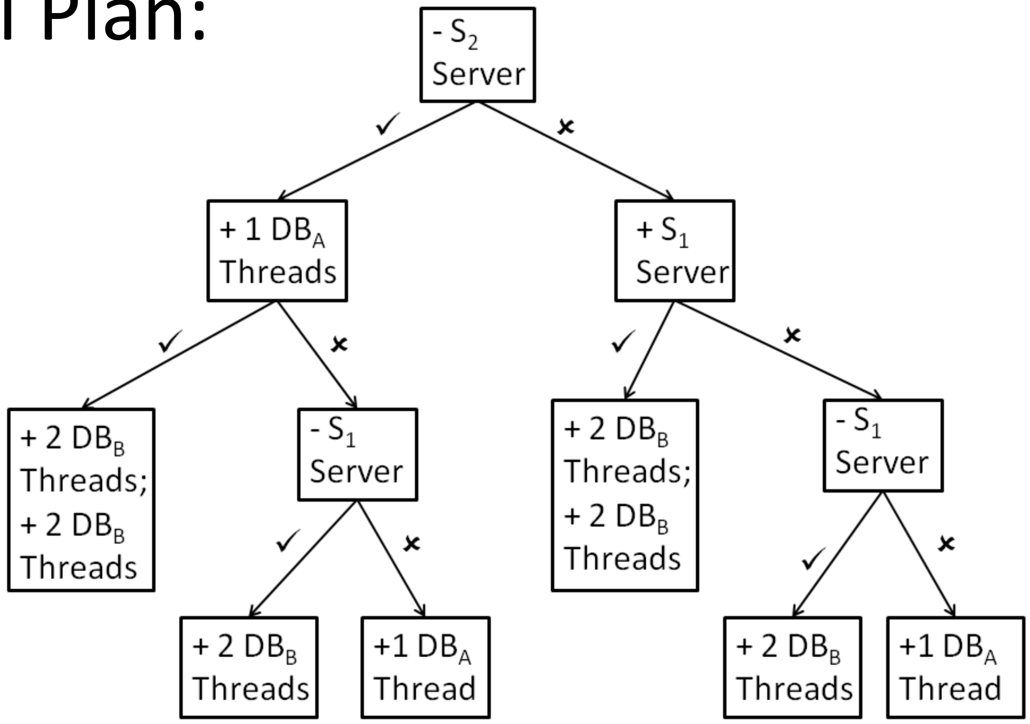


Experiments

- Improving a bad plan
  - Show that subpar inputs do not cause subpar results
- Comparing different utility functions
  - Understand how changes in the utility function affects plans
- Planning with similar utility functions
  - Can plans be used to create new plans for a similar goal?

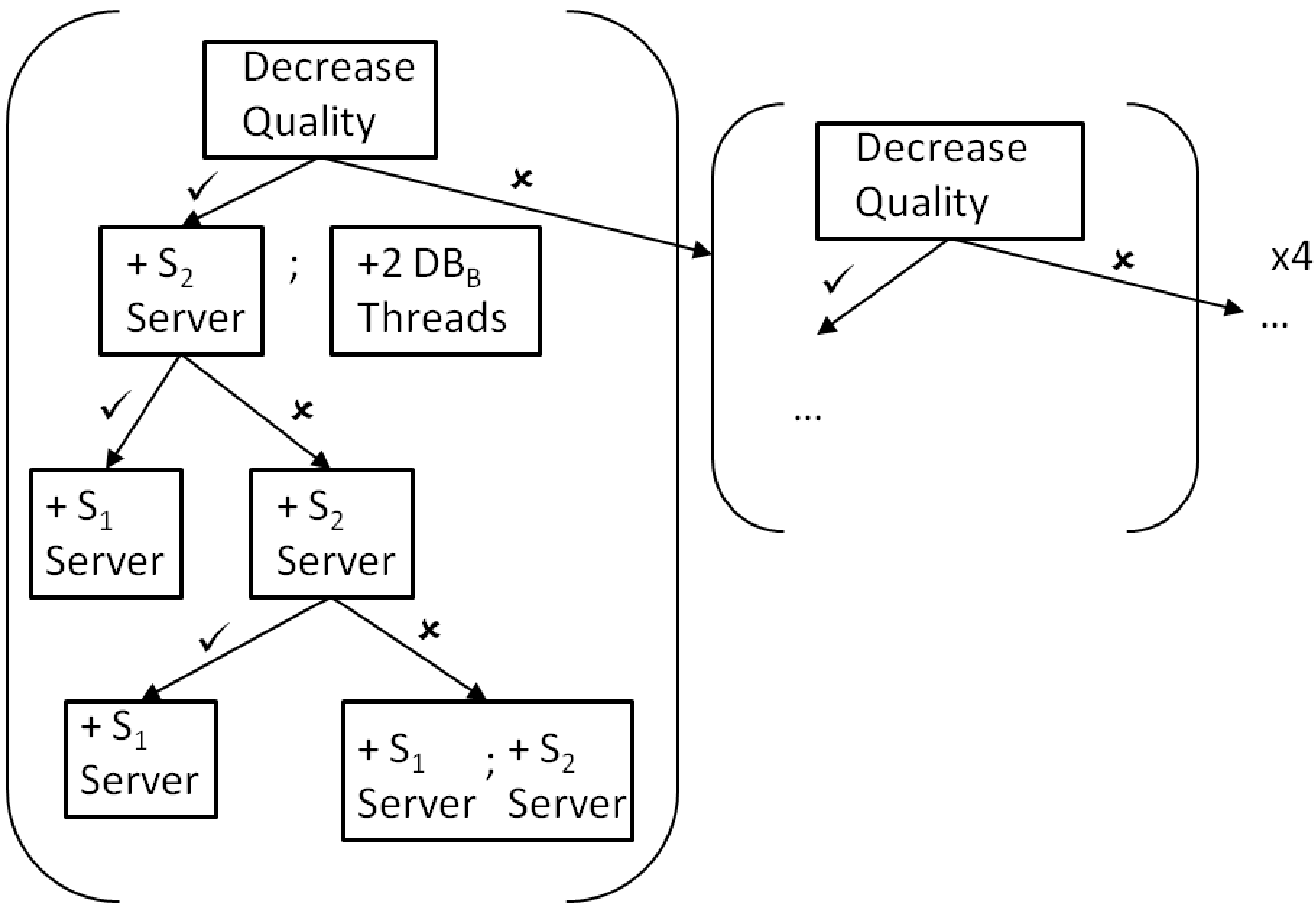
Improving a Bad Plan

- Utility Function: -20 $responseTime_{norm} - cost_{norm} + quality_{norm} - time_{norm}$
- Initial Plan:



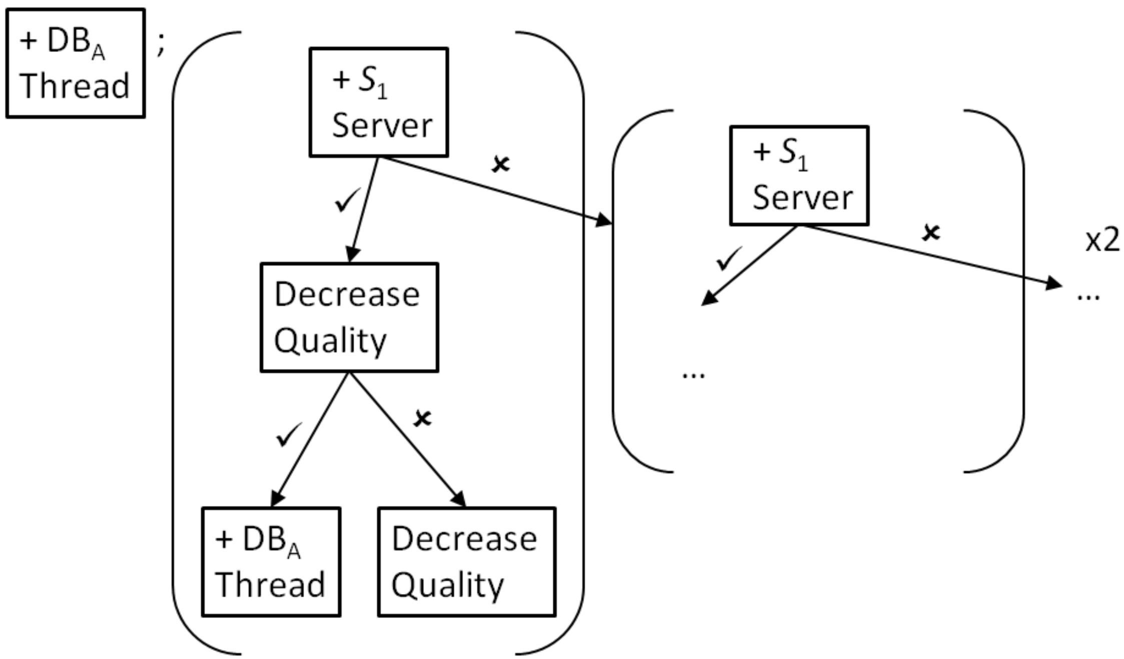
Improving a Bad Plan

- Results:



Comparing Utility Changes

- Utility Function : -10 $responseTime_{norm} - cost_{norm} + quality_{norm} - time_{norm}$
- Results:



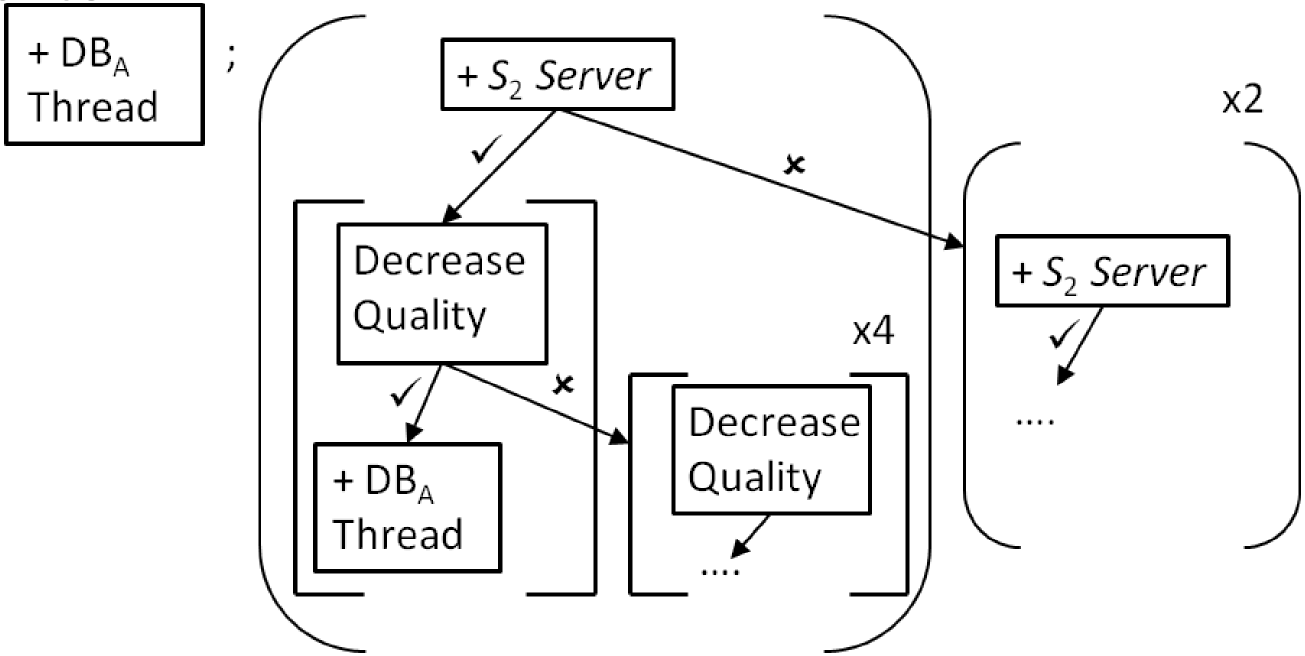
Comparing Utility Changes

- Utility Function : $-2\ responseTime_{norm} - 2\ cost_{norm} + quality_{norm} - time_{norm}$
- Results:

[+ DB_A Thread] ; [Decrease Quality] ; [+ DB_A Thread] ; [+ DB_A Thread] ;
[+ DB_A Thread] ; [- S_2 Server] ; [- S_1 Server]

Planning with Similar Utility Functions

- Utility Function: $-10\ responseTime_{norm} - cost_{norm} + quality_{norm} - time_{norm}$
- Results:



Experiment Conclusions

- The planner can handle erroneous user provided plans
- The planner can handle multiple objectives
- The planner can provide unexpected knowledge about the search space
- The planner can use information from previously generated plans to make new plans

Future Ideas

- Test the planner on a system with more tactics
  - Compare to deterministic planners
- Incorporate feedback from the system monitor
  - If adaption fails, is it likely to fail again?
  - Partially effective adaptations – timing issues
- Adapting similar plans to a new situation
- Catalogue when stochastic techniques are effective
- Improve human trust in plans/ stochastically generated plans

Summary

- Stochastic search shows promise for handling the future complexity of self-adaptive systems
- Demonstrate the benefits of stochastic search with a proof of concept genetic programming planner
- 3 experiments demonstrate the potential of our planner
- There are many research problems for applying stochastic search to self adaptive systems