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

Deterministic

Stochastic

- Goal
- Already Searched
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”
• 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: $-20 \text{responseTime}_{\text{norm}} - \text{cost}_{\text{norm}} + \text{quality}_{\text{norm}} - \text{time}_{\text{norm}}$
  – Use PRISM to evaluate the score of each final state in the plan and the probability of reaching each final state
Possible Initial Plan

Evaluate
- Response Time
- Cost
- Quality
- Time

Mutate

Discard

Final Result

Plan

Plan

Plan

Plan

Plan

Plan

Plan

Plan

Plan
## Proof of Concept

<table>
<thead>
<tr>
<th>Basic Action</th>
<th>Precondition</th>
<th>Cost ($)</th>
<th>Resp. Time</th>
<th>Time</th>
<th>Failure Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add $S_1$ Server</td>
<td>$S_1 &lt; S_1^{max}$</td>
<td>15</td>
<td>-5</td>
<td>600</td>
<td>0.1</td>
</tr>
<tr>
<td>Add $S_2$ Server</td>
<td>$S_2 &lt; S_2^{max}$</td>
<td>20</td>
<td>-5</td>
<td>600</td>
<td>0.1</td>
</tr>
<tr>
<td>Remove $S_1$ Server</td>
<td>$S_1 &gt; 1$</td>
<td>-15</td>
<td>5</td>
<td>600</td>
<td>0.1</td>
</tr>
<tr>
<td>Remove $S_2$ Server</td>
<td>$S_2 &gt; 1$</td>
<td>-20</td>
<td>5</td>
<td>600</td>
<td>0.1</td>
</tr>
<tr>
<td>Add DB(_A) Thread</td>
<td>$T_A &lt; T_A^{max}$</td>
<td>0</td>
<td>-2</td>
<td>180</td>
<td>0.2</td>
</tr>
<tr>
<td>Add 2 DB(_B) Threads</td>
<td>$T_B &lt; T_B^{max-1}$</td>
<td>0</td>
<td>-1</td>
<td>180</td>
<td>0.2</td>
</tr>
<tr>
<td>Increase Quality</td>
<td>Quality set low</td>
<td>0</td>
<td>2$S$</td>
<td>1</td>
<td>0.3</td>
</tr>
<tr>
<td>Decrease Quality</td>
<td>Quality set high</td>
<td>0</td>
<td>-2$S$</td>
<td>60</td>
<td>0.3</td>
</tr>
</tbody>
</table>
Plans

\[
\begin{align*}
&\text{if (add-server(1))} \\
&\quad (\text{add-database-thread(A,1)}) \\
&\quad (\text{decrease-quality})), \\
&\text{add-database-thread(B,2)}
\end{align*}
\]
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: \(-20\) \(\text{responseTime} \_{\text{norm}}\)
  \(-\text{cost} \_{\text{norm}} + \text{quality} \_{\text{norm}} - \text{time} \_{\text{norm}}\)

• Initial Plan:
Improving a Bad Plan

- Results:

  - Decrease Quality
    - $+ S_2$ Server
    - $+2 \text{DB}_B$ Threads
  - Decrease Quality
    - $x4$
    - ...
Comparing Utility Changes

- Utility Function: \(-10 \text{ responseTime}_{\text{norm}} - \text{cost}_{\text{norm}} + \text{quality}_{\text{norm}} - \text{time}_{\text{norm}}\)

- Results:
Comparing Utility Changes

• Utility Function: $-2 \text{responseTime}_{\text{norm}} - 2 \text{cost}_{\text{norm}} + \text{quality}_{\text{norm}} - \text{time}_{\text{norm}}$

• Results:
Planning with Similar Utility Functions

- Utility Function: $-10 \text{responseTime}_{\text{norm}} - \text{cost}_{\text{norm}} + \text{quality}_{\text{norm}} - \text{time}_{\text{norm}}$

- 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