Decentralization of Control Loop for Self-Adaptive Software through Reinforcement Learning

Kishan Kumar Ganguly, Kazi Sakib
Institute of Information Technology
University of Dhaka, Dhaka, Bangladesh

Presented By
Kishan Kumar Ganguly
Institute of Information Technology
University of Dhaka
Presentation Outline

• Introduction
• Background
• Related Work
• Problem and Motivation
• Running Example
• Proposed Approach
• Experimental Evaluation
• Results
• Discussion
• Conclusion and Future Work
Introduction

• Self-adaptive systems change their behavior at runtime to conform to their goals
• Goals are generally non-functional requirements
  – Response time
  – Throughput etc.
• Decentralization has become a widespread concept [1]
• Designing self-adaptive software with decentralized control loop is still a research challenge [1]–
  – A control loop helps to respond to the goal violation at runtime without interrupting its service
In self-adaptive system with decentralized control loops, multiple control loops coordinate to satisfy some goals.

These goals can be divided as:

- Local goal: component-level goal [2]
- Global goal: system-level goal [2]

A goal violation leads to:

- Coordination of multiple control loop (information sharing) to take adaptation decisions
- These adaptation decisions / actions are selection of variants (software variants with different configurations)
- Adaptation decisions must satisfy local and global goals
Background: Centralized Self-Adaptive Systems

Centralized Control Loop

Analyze → Plan → Monitor → Execute

Actions

Goal: Response time ≤ 3ms

Goal: Response time ≤ 2ms

Goal: Response time ≤ 6ms

Goal: Response time ≤ 8ms

Goal: Response time ≤ 14ms

Goal: Response time ≤ 1ms
Background: Decentralized Self-Adaptive Systems

Goal: Response time \( \leq 3\text{ms} \)

Goal: Response time \( \leq 2\text{ms} \)

Goal: Response time \( \leq 8\text{ms} \)

Goal: Response time \( \leq 1\text{ms} \)

Goal: Response time \( \leq 6\text{ms} \)

Goal: Response time \( \leq 14\text{ms} \)

12/10/2017
Related Work

Weyns et al. [4]
- Proposed a generic model for decentralized self-adaptive system
- It was highly abstract and did not explicitly address coordination of multiple control loops

Sykes et al. [5]
- Proposed a distributed self-assembly approach
- Used aggregated gossip for coordination
- Lacked dynamism, used static strategies

Schmerl et al. [13]
- Proposed five patterns for decentralized self-adaptation
- Facilitated further research into this domain

Grassi et al. [2]
- For self-adaptive service assembly
- Used gossip protocol for service specification dissemination
- Selected best service based on predefined utility
- Problem: static utility value

Wang et al. [3]
- Dynamic service composition based on reinforcement learning
- Opponent model-based coordination
- Can be further improved using opponent model for both learning and action selection and updating weight of violated goal for more importance

Caporuscio et al. [6]
- Performed TD learning on global quality functions of services to estimate their long term quality
- Service matching was performed by choosing the service that has maximum quality
- Local goals and coordination was not explicitly considered
Problem and Motivation

**Problem**
- Writing static strategies or action selection rules, similar to some centralized control loops is not practical due to large state space
- In a specific state, action selection rules of a local control loop depend on the strategies followed by other control loops
- Reward functions (that calculate goal conformance [3]) need to be defined in such a way that these successfully capture both local and global goal violations

**Motivation**
- Reinforcement learning provides a great opportunity to introduce dynamism into the self-adaptive decisions
- Each local control loop needs to estimate the strategies used by other ones which can be done through an opponent model along with multiagent reinforcement learning
- The reward functions can be aggregated with a dynamic weight to provide more importance to the violated goals
The Tele Assistance System (TAS) is a service-based system that provides medical service to patients [9].

Three services are used –

- MedicalAnalysisService – Checks the vital parameters of the patient and takes actions
- DrugService – Change drug or dosage of the drug
- AlarmService – Provides alarm in case of emergency

MedicalAnalysisService variants with different configurations
- MedicalAnalysisService1, MedicalAnalysisService2 and MedicalAnalysisService3

AlarmService similarly has three variants
## Running Example

<table>
<thead>
<tr>
<th>Goal</th>
<th>Attribute</th>
<th>Goal Type</th>
<th>Goal Threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>MedicalAnalysisService must have response time less than or equals 5.6 ms</td>
<td>Response Time</td>
<td>Local &amp; Threshold</td>
<td>5.6</td>
</tr>
<tr>
<td>AlarmService must have response time less than or equals 5.2 ms</td>
<td>Response Time</td>
<td>Local &amp; Threshold</td>
<td>5.2</td>
</tr>
<tr>
<td>MedicalAnalysisService must have failure rate less than or equals 0.12</td>
<td>Failure Rate</td>
<td>Local &amp; Threshold</td>
<td>0.12</td>
</tr>
<tr>
<td>AlarmService must have failure rate less than or equals 0.</td>
<td>Failure Rate</td>
<td>Local &amp; Threshold</td>
<td>0.1</td>
</tr>
<tr>
<td>At least one service must have failure rate less than 0.08</td>
<td>Failure Rate</td>
<td>Global &amp; Threshold</td>
<td>0.08</td>
</tr>
<tr>
<td>Average cost per service must be minimized</td>
<td>Average Cost Per Service</td>
<td>Global &amp; Minimization</td>
<td>-</td>
</tr>
</tbody>
</table>
## Running Example

<table>
<thead>
<tr>
<th>Goal Attribute</th>
<th>Value Range</th>
<th>Corresponding Category</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Failure Rate</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[0, 0.002)</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td>[0.002, 0.08)</td>
<td>Medium</td>
</tr>
<tr>
<td></td>
<td>[0.08, 0.1)</td>
<td>High</td>
</tr>
<tr>
<td></td>
<td>[0.1, (\infty))</td>
<td>Extreme</td>
</tr>
<tr>
<td><strong>Average Cost Per Service</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[0, 2)</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td>[2, 5)</td>
<td>Medium</td>
</tr>
<tr>
<td></td>
<td>[5, (\infty))</td>
<td>High</td>
</tr>
<tr>
<td><strong>Individual Service Response Time</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[0, 2.5)</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td>[2.5, 5.2)</td>
<td>Medium</td>
</tr>
<tr>
<td></td>
<td>[5.2, (\infty))</td>
<td>High</td>
</tr>
</tbody>
</table>
Proposed Approach

- A state is represented by a specific combination of the different goal attribute values in different agents
  - $\{\{\text{low,extreme}\}, \{\text{high,high}\}, \{\text{low}\}\}$ is a state that expresses that MedicalAnalysisService has low response time and extreme failure rate, AlarmService has high response time and failure rate and globally average cost per service is low

- An action is considered as choosing a specific variant

- The action set of a specific agent consists of all of its variants

- The joint action set is the action selection of all the agents
  - $\{\{\text{MedicalAnalysisService}_1, \text{MedicalAnalysisService}_2, \text{MedicalAnalysisService}_3\}, \{\text{AlarmService}_1, \text{AlarmService}_2, \text{AlarmService}_3\}\}$
Proposed Approach

- Reward functions measure goal conformance
- For minimization and maximization goal, if $G$ is the set of the values of a goal attribute, a reward function is defined as $R : G \rightarrow [0, 1]$
- For threshold goals, it is defined as $R : G \times T \rightarrow [0, 1]$
- The reward function value is restricted between 0 and 1
Proposed Approach

- Reward functions for TAS –

\[
\begin{align*}
\text{Global Failure Rate } & \quad r_{g lf} = \\
& = \begin{cases} \\
\frac{(2 \times th_{g lf} - \min(f^1_r, f^2_r, \ldots, f^n_r))}{2 \times th_{g lf}} & \text{if } (th_{g lf} - \min(f^1_r, f^2_r, \ldots, f^n_r)) > 0 \\
\frac{1}{2} - \frac{\min(f^1_r, f^2_r - 2 \times th_{g lf})}{2 \times (1 - th_{g lf})} & \text{if } (th_{g lf} - \min(f^1_r, f^2_r, \ldots, f^n_r)) \leq 0
\end{cases}
\end{align*}
\]

\[
\begin{align*}
\text{Local Response Time Reward } & \quad r_t = \\
& = \begin{cases} \\
\frac{2 \times th_t - t^i}{2 \times (th_{g lf} - t^i_{max})} & \text{if } th_t - t^i > 0 \\
\frac{1}{2} - \frac{th_t - t^i + 1}{2 \times (th_t - t^i_{max} + 1)} & \text{if } th_t - t^i < 0 \\
\frac{1}{2} & \text{if } th_t - t^i = 0
\end{cases}
\end{align*}
\]
Proposed Approach

- Reward functions for TAS –

\[
rf = \begin{cases} 
\frac{2 \times th_f - f^i}{2 \times (th_f - f^i_{max})} & \text{if } th_f - f^i > 0 \\
1 & \text{if } th_f - f^i < 0 \\
\frac{1}{2} & \text{if } th_f - f^i = 0 
\end{cases}
\]

Average Cost Per Service Reward

\[
rc = \frac{1}{\frac{1}{n} \sum_{i=1}^{n} \bar{c}^i + 1}
\]

Total Reward

\[
tot_r = 0.25 \times r_{glf} + 0.25 \times r_t + 0.25 \times r_f + 0.25 \times rc
\]

Threshold-based reward functions provide [0, 0.5) values for goal violation and [0.5, 1] values for goal conformance
Proposed Approach

• Weights need to be updated at runtime for providing more importance on reward function values indicating goal violation

• For example, consider four reward functions $r_1, r_2, r_3, r_4$
  – For reward values 0.7, 0.8, 0.3, 0.6, the total reward value is 0.6
  – For reward values 0.6, 0.8, 0.5, 0.5, the total reward value is also 0.6
  – The first reward value should be less than the second

• Solution –
  – The weight for 0.3 can be updated to 0.3542 from 0.25
  – The remaining weight 0.6458 can be equally distributed among the other three agents
  – The total reward value becomes 0.558
Proposed Approach
Action Selection through Q-Learning

Algorithm 1 Algorithm for Multiagent Q-learning

1: $Q_i(s, a) \leftarrow 0, \forall s, a, i$
2: $\text{decay} \leftarrow d_f$
3: while $\text{TerminationCondition} \neq \text{true}$ do
4:   for $i = 1$ to $n$ do
5:       $a_i \leftarrow \text{selectAction}()$
6:       $a^{-i} \leftarrow \text{receiveOtherAgentActions}()$
7:       $a \leftarrow a_i \cup a^{-i}$
8:       observe transitioned state $s'$ and reward $r_i$
9:   for $i = 1$ to $n$ do
10:      $Q_i(s, a) \leftarrow Q_i(s, a) + \alpha \times [r_i(s, a) + \gamma \max_{a_i} Q_i(s', a_i) - Q_i(s, a)]$
11:      $\alpha \leftarrow \alpha - \text{decay} \times \alpha$
12:   end for
13: end for
14: end while
An opponent model chooses the maximum next state strategy based on other agents’ strategies [8]

\[ Q_i(s, a_i) = \sum_{a^{-i} \in A_{-i}} P(s, a^{-i}) \times Q_i(s, a_i, a^{-i}) \]

\[ P(s, a^{-i}) = \frac{F(s, a^{-i})}{\sum_{a^{-i'} \in A_{-i}} F(s, a^{-i'})} \]

Opponent Model
Proposed Approach
Action Selection through Q-Learning

- The $\epsilon$-greedy strategy is used for action selection
- This provides a balance between exploration and exploitation in Q-learning [11]
- $\epsilon$-greedy strategy –

An $\epsilon$ value is prespecified where $0 \leq \epsilon < 1$

A value $p$ between 0 and 1 is randomly selected

If $p < \epsilon$, the agent chooses a random action

Otherwise, it selects the action with maximum Q-value in the current state
Experimental Evaluation

• The Tele Assistance System is used to evaluate the proposed approach
• It is extended to support decentralized adaptation by adding a control loop to each of the services
• The approach is compared to two techniques –
  – The first one chooses actions randomly
  – The second one learns and chooses actions based on maximum Q-values without considering opponent models
• Parameters –
  – $\alpha$, $\gamma$ and $\varepsilon$ were chosen to be 0.1, 0.9 and 0.8 respectively
  – A small number 0.001 was chosen as the decay factor
  – 1.25 was chosen as the value of $fr$

Chosen through empirical experimentation
Results: Comparison of Reward
Results: Comparison of Reward
Results: Effectiveness of the Dynamic Weight Update Technique
• The reward values are over 0.5 in most cases which indicates adaptation
• Reward values are more stable and total reward values are higher when opponent model is considered
• The weight update mechanism supports adaptation of multiple goals as the reward values remain stable over time
• Q-learning over joint actions become computationally challenging in large scale systems
• A promising direction towards solving this problem can be the use of sparse cooperation Q-learning (future work) [12]
Conclusion and Future Work

- A decentralized control loop for self-adaptive software has been proposed –
  - Considering other agent strategies
  - A better total reward calculation mechanism

- It was evaluated on a Tele Assistance System where it was observed that –
  - Reward stays over threshold
  - Reward values are stable over time
  - Both indicates successful self-adaptation

- Future Work –
  - Applying to large scale systems
  - Self-tuning the required parameters to achieve the highest reward
  - Tool support
References


References


Thank You
Background: Action Selection

Service

Goal: Maintaining response time ≤ 3ms

Components

A → B → Variants

Analyze → Plan → Execute

Monitor

Control Loop

Replace C2 with C1

Variants

C1

C2

C3
Proposed Approach

Lemma 1. Let $n_r$, $n_a$ be the total number of reward functions and the number of reward functions requiring weight update respectively. Let $r_i$, $th$ and $fr$ be the $i$–th reward function value requiring weight update, reward function threshold and the fraction of the initial reward used to update the weight respectively. The weight of the $i$–th reward function is updated with $\Delta w_i = k \times (th - r_i)$, where $k = \frac{1}{fr \times n_r \times (n_a \times th - \sum_{i=1}^{n_a} r_i)}$.

- For the previous example –
  - $r_i = 0.3$
  - $th = 0.5$
  - $fr = 2.4$