4J-01

自己適応における動的な優先順位変更のための制御器生成コストの削減に関する研究

惣津美穂† 片江将希† 田邉萌香† 鄭顕志‡ 深澤良彰† 本位田真一‡

†早稲田大学 ‡国立情報学研究所

1 はじめに

システムの故障や、ユーザーの予期せぬ行動などの、複雑な環境の変化に対して安定して要求を満たすことが求められるソフトウェアが出てきている。 そのために、システム自身が考えて環境の変化に適応する自己適応システムが現在注目されている。自己適応システムは、システム自身が環境を監視、解析し、どのような処理を行えば適応できるのかを計画、実行するものである。

近年、制御工学の概念を取り入れ、様々な非機能要求を満たすことを保証する既存研究が存在している[1][2]。そこでは、保証できる要求には予め優先順位が定められており、順位が高いものから順に保証されていく。しかし、実際の環境ではその順位は一定ではなく、変化する可能性がある。また、優先順位変更を実現するためにはあらゆる順位に対する制御器を事前に用意する必要があるため、入力が複雑になり、制御器生成コストが嵩む。

そこで、本論文では、二部グラフを用いて、要求間の関係性を明確にし、生成する制御器の数を削減する手法を提案する。これにより、優先順位変更を達成するための制御器生成コストを削減する。

2 自己適応システム

自己適応システムとは、システム自身が環境変化に対して、与えられた要求を満たすように自らが考えて適応するシステムである。例えば、クラウドアプリケーションにおいて、システムが予期せぬ量のユーザーからのアクセスを受けた時のことを考える。このとき、従来であれば、サーバーを増加させる、性能の良いものに変えるなどの行動を人手で行わなければならない。自己適応システムでは、システム自らが決定し、これを実行する。

現在、自己適応システムと制御工学の技術を組み合わせた研究が存在している。制御工学とは、数学モデルを利用して、制御対象を評価し、望み通りの挙動を達成することを目的とした研究分野である。これを自己適応システムと組み合わせることで、対象のシステムを数学的に評価し、要求を達成することを保証することが可能となる。

既存研究としてBrownout[1]と呼ばれるシステムがある。このシステムはクラウドアプリケーションにおいて、ユーザーからの突発的なアクセスに対して、ノブと呼ばれるシステム内部の値を操作して、サーバーへの負荷を減らすというものである。ここでは、推薦機能を付与するユーザーの割合がノブとなっている。ノブの値は、サーバーからの応答時間を入力として、制御器によって決められている。この制御器を生成するためには、システムの入出力を数学モデル化し、評価する必要がある。このときにかかる手間を、本論文では制御器生成コストと呼ぶ。

Brownout では、制御することができるのが応答時間、一つのみであった。これに対して、彼らの次の論文[2]は、多数の品質を制御することを目標としている。それを実現するため、制御したい品質に対して優先順位をあらかじめ決定しておき、順位が高い品質から制御するように制御器を生成している。その際、優先度が高い品質に影響を与えないように、順位が高いものを制御するために既に使われたノブは、優先順位の低い品質を制御する際には使わず、残ったノブだけで制御するという手法を採っている。

品質の優先順位は実行時間中に変化する可能性がある。しかし、先行研究では、品質の優先順位を設計時に決定し変更することはできない。なぜならば、制御器が操作できるノブはそれが制御する品質の優先順位によって異なるためである。従って、優先順位の変更を可能にするためには、品質の優先順位の全可能性に対する制御器、即ち、品質の階乗個分の制御器を生成しなければならない。それには多大な制御器生成コストがかかる。しかし、その制御器の中には全く同じものが存在する可能性がある。優先順位の変更を実現するためには、このような無駄な制御器の生成を防ぐことで、制御器生成コストを減らすことが必要である。

3 例題

本論文では、スマートハウスの例題を用いて説明する。

スマートハウスは、部屋にある家電をシステムで一括に管理するというものである。今回は夏場のスマートハウスを想定しており、例題に取り上げる家電、家具(本手法におけるノブ)は、エアコン、ガス冷房、除湿器、照明、カーテンの５つで

Study on Reduction of Control Generation Cost for Dynamic Priority Change in Self Adaptive System

†Miho Souzu, Masaki Katae, Moeka Tanabe, Yoshiaki Fukazawa, Waseda University

‡Kenji Tei, Shinichi Honiden, National Institute of Informatics

表1　品質とノブの関係

	温度	湿度	セキュリティ	電力	ガス	明るさ
エアコン	昼夜			昼夜		
ガス冷房	昼夜				昼夜	
除湿器		昼夜		昼夜		
照明				昼夜		昼夜
カーテン			昼夜			昼

ある。ノブが影響を与える品質は、温度、湿度、セキュリティ、電力、ガス代、明るさの 6 つである。表 1 に品質とノブの関係性を示す。この表では昼に品質とノブの間に関係があるものには昼と、夜に関係があるものは夜と表に書いてある。例えば、エアコンは昼、夜ともに気温に影響を与え、カーテンは昼のみ明るさに影響を与えるということを示す。また、環境は昼、夜と変化することを想定している。

優先順位の変更による制御器の増加数は従来手法であれば、制御すべき品質 1 つに対し、6!通りの制御器を生成することになる。また、昼から夜へと環境が変化した場合、カーテンと明るさの関係性がなくなる。そのため、すべての制御器を再生成しなければならず、結果として 6!×2 個の制御器が必要となる。

4　提案手法の概要

優先順位を実行時間中に変化させる場合、膨大な制御器を事前に生成しなければならず、ユーザーへの負担が大きくなるという問題点があり、また、その中には無駄に生成している制御器も存在する。これに対して、我々は本手法で、二部グラフを用いることを提案する。二部グラフとは、点を線で結ぶグラフの一つであり、点の集合を二つの素な集合に分割できるものを指す。二部グラフを用いることにより、生成すべき制御器の中から重複しているものを取り除き、ユーザーへの負担を減らすことが出来る。

本手法の利点は以下の通りである。

- 品質の優先度を実行時間中に切り替えることができる。
- 制御器を切り替えるためのユーザーへの負担を軽くすることができる。

図 1 のように品質とノブを分けて列記し、関係があるものを結ぶ。実線は昼と夜の関係性、点線は昼のみの関係性を表している。制御器を生成したい品質と辺の数が 2 本以内でつながる品質だけを取出し、その中で優先順位を考える。昼の場合の温度の例を挙げると、2 本以内でつながる品質は、電力とガスの 2 つである。また、考慮すべき優先順位は以下の 4 通りである。

A)　1 位　温度、2 位　湿度、3 位　ガス
B)　1 位　湿度、2 位　温度、3 位　ガス
C)　1 位　ガス、2 位　温度、3 位　湿度
D)　1 位　湿度、2 位　ガス、3 位　温度

これにより、その品質にとって関係のない品質

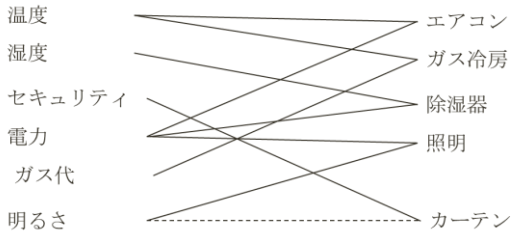

図 1　品質とノブの関係性の二部グラフ

との優先順位を考慮しなくてよいため生成すべき制御器を減らすことができる。

次に、環境が昼から夜に変化する場合を考える。このとき変化したのは、カーテンと明るさの関係がなくなったという点である。ここでなくなった辺に接していた 2 点から長さ 1 以内でつながる点の品質の制御器のみ変更すればよい。今回、当てはまる品質は明るさとセキュリティのみである。

5　評価

ここでは、スマートハウスの例題を用いて評価を行う。今回は温度の制御器の生成を想定する。従来では必要な制御器は全ての優先順位を考慮するため、6!=720 個用意する必要がある。本手法を用いると、前章の例より、4 個だけでよくなり、従来の手法より大幅に生成すべき制御器を少なくすることができた。

また、環境が昼から夜に変化した場合、従来は 720×2 個の制御器をそれぞれ生成する必要があったが、本手法では明るさとセキュリティ以外の品質の制御器には影響がないため変更しなくてよい。さらに、その二つの品質もそれぞれ 2 個、1 個の追加であるため、大幅に生成すべき制御器を少なくすることができ、制御器生成コストを減らすことができた。

6　おわりに

本手法では多数の品質を制御する自己適応システムに対し二部グラフを用いて、生成すべき制御器を減らし、実行時間中に優先順位を切り替えることを可能にした。

また、環境が変化した場合でも用意すべき制御器の数を減らすことができた。

今後は、ノブと品質の関係が変わったときに、実行中に生成できるようにすることを目標としたい。

参考文献

[1] C. Klein, M. Maggio, K. E. Årzén and F. Hernández-Rodriguez, ″Brownout: Building More Robust Cloud Applications,″ in ICSE, 2014
[2] Antonio Filieri, Henry Hoffmann, Martina Maggio, ”Automated Multi-objective Control for Self-Adaptive Software Design ”,in FSE,2015