モデル予測制御と離散制御器合成による外部環境の動的特性を考慮した適応制御手法

安曽徳康[1]　小川雅俊[1]　松塚貴英[1]　鄭顕志[2]

概要：自己適応システムの分野において，周辺環境のモデルと要求や制約の論理式から要求を満足する実行処理手順を自動合成する離散制御器合成という方法がある．離散制御器合成による自己適応システムは機能要求を満たす適応は可能であるが，実行処理手順に複数の解が存在する場合は分岐先処理を選定する必要があり，「実行処理手順」と「その分岐先処理の性能指標の数値」の双方の動的な最適化による適応が行えない．本研究では「実行処理手順の最適化」と「その分岐先処理の性能指標の数値の最適化」の双方の動的な適応管理を実現することを目的とした適応制御手法を構築した．具体的には，離散制御器合成を用いて機能要求を満足する実行処理手順を導出すると同時に，環境に設置されたセンサを用いて逐次取得した観測データから逐次的に同定した予測モデルを用いて性能指標値を予測し，複数の性能指標値により総合評価する評価関数を満足するように選択肢を最適化する．一例として，旅行管理システムに適用した結果，各性能指標の優先度に基づく選択肢の変更や天候の変化により各移動手段に生じる遅延時間を考慮した選択肢の変更などが適切に選択されることを確認し，その有効性を確認した．

Adaptive Control Method Considering Dynamic Characteristics of External Environment by using Model Predictive Control and Discrete Controller Synthesis

Noriyasu Aso[1]　Masatoshi Ogawa[1]　Takahide Matsutsuka[1]　Kenji Tei[2]

1. 緒言

ソフトウェアシステム開発では，利用可能な資源および外部周辺環境の経時的な変化に伴うソフトウェアの修正や更新などの業務工数の増大が大きな課題となっている．ソフトウェアシステムと外部周辺環境の双方を逐次監視して，その変化に合わせて継続的にシステムを適応させるフィードバックループを構成する自己適応システム[1-3]への関心が高まっている．

自己適応システムの実現を支援する技術の１つに，要求を満足する動作仕様を自動生成する離散制御器合成[4]と呼ばれる技術がある．離散制御器合成は，外部周辺環境を Labelled Transition System (LTS)でモデル化した状態遷移モデルと論理式で記述された要求を用いて，要求を満足する動作仕様を自動生成する技術である．ここでの動作仕様とは要求を満たす実施可能な実行処理手順であり，その解が複数存在する場合は実行処理手順において分岐が存在する．最適な実行手順を選択するには，分岐先で実行され得る処理の性能指標を評価する必要がある．しかしながら，離散制御器合成は外部環境を LTS で表現するため，性能指標が取りうる値を定量的にモデル化できず，値の範囲を離散状態として表現する必要があるため組み合わせが膨大になり扱いづらいという課題がある．また，複数の性能指標値にトレードオフが生じる場合も同様に困難である．つまり，離散制御器合成を用いた自己適応システムは機能要求を満たす適応は可能だが，「実行処理手順」と「その分岐先処理の性能指標の数値」の双方の動的な最適化が行えず，性能を最適化する適応が行えない．

本研究では，「実行処理手順」と「その分岐先処理の性能指標の数値」の双方の動的な適応管理を実現することを目的とする．離散制御器合成を活用し，想定環境における実行処理手順を生成した上で，その分岐先処理の選定においては，Model Predictive Control（MPC）を用いて複数の性能指標の動的特性を考慮した適応的分岐処理選択を行う適応制御手法を提案する．MPC を用いることで，管理すべき性能指標に関する要因の将来のダイナミクスを数理モデルにより予測し，その将来の予測結果が望ましい結果となるように所定の制約条件下での分岐先処理の選択を行うことが可能となる．

2. 提案する適応制御技術

本提案手法の構成図を Fig.1 に示す．周辺環境を LTS によりモデル化した状態遷移モデルと要求や制約の論理式を入力とし，離散制御器合成を用いて機能要求を満足する実行処理手順を導出する．同時に，環境に設置されたセンサを用いて観測データを逐次取得し，予測モデルをシステム同定し,同定した予測モデルを用いて性能指標値を予測し，複数の性能指標値を総合評価する評価関数を満足するように，分岐先処理を適応制御で選択することで性能面を考慮した最適化を行う．本構成により環境変化に対応した最適なサービスを動的に選択することができる．

1 (株) 富士通研究所
FUJITSU LABORATORIES LTD. Kawasaki, Kanagawa 211-8588, Japan
2 早稲田大学
WASEDA University, Shinjuku, Tokyo, 169-8555, Japan

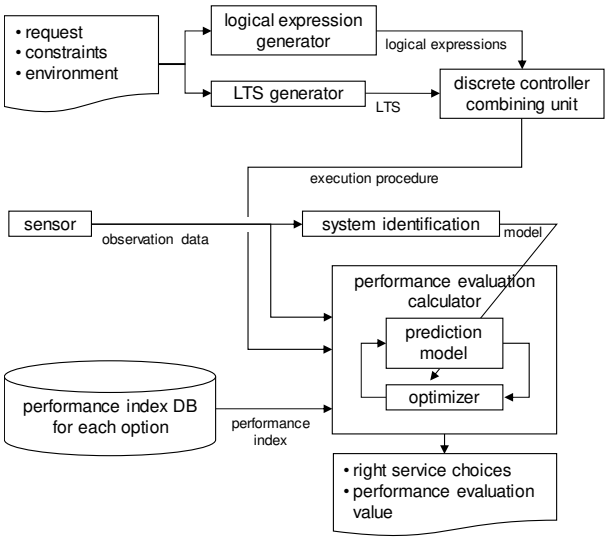

Fig.1 Proposed method.

3. 試作した旅程管理システム

提案手法を出張旅程管理システムに適用した．交通機関(バス，電車，飛行機)や宿泊地(A，B)などの実行環境を反映した環境モデルを構築し，Fig.2 に示す出張者の出張時のイベントの順番や業務遂行の場所，時間などの要求と制約を満足した実行処理手順を自動生成する．環境予測モデルは，出張中の各地の降水量の推移データから各時刻における交通機関の選択肢(Bus-A1，A2 など)の到着時刻に生じる遅延時間を予測するモデルを構築し，予測ステップごとに逐次更新する．構築したモデルにより各交通機関の到着時刻に基づく選択肢候補や性能指標の一つである移動時間を動的に更新する．旅行期間中の各予測ステップの宿泊先，移動手段の選択肢の空き状況，交通手段の到着時刻などを考慮し，複数の性能指標（快適性，金額，移動時間など）を総合的に評価する評価関数 F を用い，F を最小化するように選択肢を選定する最適化計算を行う．Fig2 のシナリオを満足する選択肢の組み合わせは 51.3 万通りあるため，計算コストの低減を目的に，例えば“move to A”といったイベントの選択肢を集約し，実行処理手順を宿泊と移動に関する合計 6 イベントに簡略化し，全体の大域的な最適化により解を導出する予測最適化手法を構築した．

4. 評価

各選択肢の組み合わせを網羅的に計算し最適解を選択する手法と本最適化手法の計算時間の比較を行った結果，網羅的に計算する手法が約 770 秒であったのに対し，本最

Day1: home >> move to A (arrival time limit: 12:30) >>
work at the office in A (13:00-18:00) >> stay at the hotel in A
Day2: work at the office in A (09:00-16:00) >>
move to B (arrival time limit: 21:45) >> stay at the hotel in B
Day3: work at the office in B(09:00-18:00) >> stay at the hotel in B
Day4: move to C (arrival time limit: 12:30) >>
work at the office in C (13:00-18:00) >> stay home

Fig.2 Business travel scenario.

Table1 Comparison result between the proposed method and the conventional method

	selected result	normal arrival time	arrival time in rainy weather
case1	Bus-B1	21:37	21:49 (limit over)
case2	Train-B1	20:15	20:16

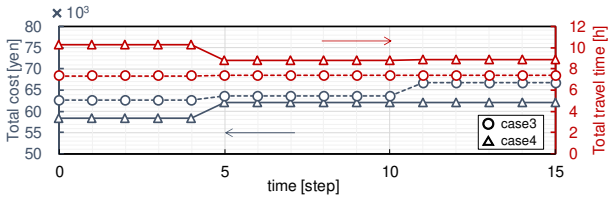

Fig.3 Optimization results for case 3 and case4.

適化手法では約 17 秒であり，効率よく最適解を得ることができることを確認した．次に予測モデルを用いた提案する適応制御手法の有効性を検証した．天候変化により遅延が生じる 5 ステップ目の“move to B”の最適化結果を Table1 に示す．動的予測モデルを用いない手法(case1)は，到着制限時刻 21:45 を超過する一方で，動的予測モデルを用いた本提案手法(case2)は，制約を満足しており，環境変化に合わせて適切に選択肢が導出されていることを確認した．最後に，性能指標の最適化において，移動時間を優先したもの(case3)と合計金額を優先したもの (case4)を比較した結果を Fig.3 に示す. 双方ともに上限金額 80,000 円を満足し，優先する性能指標を考慮して最適解が得られていることを確認した．

5. 結言

提案した環境予測モデルにより外部環境の動的特性を考慮した適応制御手法を旅程管理システムに適応して評価した結果，各性能指標の優先度に基づく選択肢の変更や天候の変化により各移動手段に生じる遅延時間を考慮した選択肢の変更などが適切に選択されることを確認し，その有効性を確認した．

参考文献

[1] Luciano Baresi and Carlo Ghezzi: The disappearing boundary between development-time and run-time, In the FSE/SDP workshop on Future of software engineering research, pp.17-22, 2010.
[2] Frank D. Macias-Escriva, Rodolfo Haber, Raul del Toro and Vicente Hernandez: Self-adaptive systems: A survey of current approaches, research challenges and applications, Journal of Expert Systems with Applications, Volume 40, Issue 18, pp. 7267-7279, 2013.
[3] D. Weyns, Software Engineering of Self-Adaptive Systems: An Organised Tour and Future Challenges, Chapter for Handbook of Software Engineering, p.43, Springer 2017.
[4] N. D'Ippolito, D. Fischbein, M. Chechik, and S. Uchitel. Mtsa: The modal transition system analyser. In 2008 23rd IEEE/ACM International Conference on Automated Software Engineering, pp. 475-476, Sept 2008.