1N－3

インクリメンタルな制約解消機構を用いたロボットタスクスケジューラの設計と実装*

白石陽　納谷太　安西祐一郎†
慶應義塾大学‡

1 はじめに

自律移動ロボットによる荷物配送システムを想定した時、タスクは、ユーザからロボットへ逐次依頼され、時間制約を持つものと考えられる。このようなタスクをスケジューリングするためには、インクリメンタルな制約解消機構を備えたスケジューラが必要となる[1]。ロボットを効率良く移動させるためには、1つのスケジュールだけを求めるのではなく、複数の実行可能なスケジュールの中からできるだけ良いものを選ぶ必要がある。しかし、最終的に選ばれるスケジュールとスケジューリングに要する時間は、制約解消機構に与える制約をどのように生成していくか、すなわち制約生成アルゴリズムに大きく依存する。実際の環境では、スケジューリングに十分な時間を割り当てることができない場合も予想されるので、スケジューラの設計においては、スケジューリングに要する時間を考慮しなければならない。本研究では、制約生成アルゴリズムを3つ提案し、評価する。

2 インクリメンタルなスケジューラ

2.1 スケジューラの構成

本研究で扱うスケジューラの構成を図1に示す。スケジューラはインクリメンタルな制約解消機構を持つが、これはTCLM(時間制約論理モデル)[2]に基づく。タスクは$task(P1, P2, ES, LF)$で与えられ、$P1$(タスク始点)から$P2$(タスク終点)までの荷物の配送を表す。ES(最早開始時刻)、LF(最遅終了時刻)はそれぞれ時間制約を表すパラメータであり、特に指定しなくても良い。ユーザからタスクがインクリメンタルに与えられる度に、まず制約生成部がそのタスクそのものに関する制約リストを生成し、矛盾検出部に渡す。矛盾検出部は、制約が解消されれば真を返し、そうでなければ偽を返す。制約リストのすべての制約が解消されない場合、ここでそのタスクは実行不可能とみなされる。次に制約生成部が生成するのは、他のタスクとの依存関係から決まる制約リストである。この制約リストは、タスク列における前後のタスクとの制約から構成される。ここでいうタスク列とは、実行する順にタスクを並べたものである。制約リストを矛盾検出部に渡した結果が真となれば、そのタスク列からスケジュールが

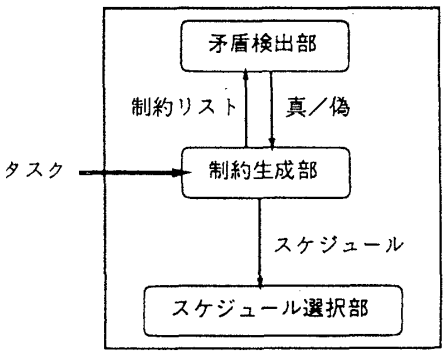

図1: スケジューラの構成

求まる。複数のスケジュールの中からより良いもの選択するためには、複数のタスク列を求め、その制約リストを生成し、矛盾を検出するということを繰り返さなければならない。スケジュールを選択する時の基準については、以下のように決めた。

- スケジュールに含まれるタスク数を最大にする。
- 最後に実行するタスクの終了時刻が早い
- タスクの始点、終点に存在できる時間が長い
- 全移動距離が短い
- 先に依頼されたタスクが先に実行される

2.2 スケジューリング時間に関する問題

スケジューリングに割り当てることのできる時間は、ロボットの状態によって異なる。タスクが依頼された時のロボットの状態は、停止しているか、移動しているかのいずれかである。ロボットが停止している場合、スケジューリングを全て行なってから移動するのか、それとも決められた時間までに得られたスケジュールにしたがって移動するのかが問題となる。ロボットが移動している場合には、タスクの始点や終点に到達するまでの時間がスケジューリング時間として割り当てられる。いずれの場合も、スケジューリングが途中で打ち切られても何らかのスケジュールを持っている必要がある。

スケジューリング時間は、制約生成アルゴリズムに大きく依存するので、スケジューリング時間を短くするような制約生成アルゴリズムを考える必要がある。

*Design and implementation of robot task scheduler using incremental constraint satisfaction
†You Shiraishi Futoshi Naya Yuichiro Anazi
‡Keio University

3 制約生成アルゴリズム

ここでは、3つの制約生成アルゴリズムを提案する。

- アルゴリズム1
  前のスケジューリングで得られたタスクの実行順序は変えずに、新しいタスクを前から順にタスクの間に入れて実行可能なスケジュールが生成されるか調べていく。
- アルゴリズム2
  移動距離を計算しながら探索を行ない、定められた下限値より移動距離が大きくなったらバックトラックを起こす。下限値は、スケジュールが求まる度に移動距離の小さい方の値が設定される。
- アルゴリズム3
  タスクを時間制約のあるものと、全く時間制約のないものの2つに分けて考える。時間制約のあるタスクの中には、事前に順序関係が決められるものがある。したがって、このようなタスクを探索の前にチェックしておき、探索時にそれを調べていくことで、探索空間を制限できる。時間制約のないタスクについては、それらのタスクが連続して並んでいたら、その中で移動コストが最小になるようにタスクを並び替える。

アルゴリズム2,3については、タスク列を求めるために探索を行なっているが、それぞれ探索空間の制限の仕方が異なっている。

4 シミュレーション結果

先に述べた3つのアルゴリズムについて、スケジューリングに要する時間を比較し、全解探索で得られる最適なスケジュールが得られるかどうか調べた。シミュレーションにおいて、タスクはランダムな7個のタスクを順番に与え、これを1つのタスクセットとして、このタスクセットを100個について調べた。スケジューリングに要する時間の比較を、図2に示す。図中のalgo0は、アルゴリズムとして全解探索を用いた場合である。図2より、アルゴリズム1が最も速く、他のアルゴリズムとの差は、タスク数が多くなるにつれて大きくなっていくことがわかる。アルゴリズム1のスケジューリング時間が短いのは、生成された制約の数が少ないからである。

一方、全解探索で得られる最適なスケジュールが得られるかどうかについては、アルゴリズム3を用いた場合には、タスク数が多くなってもほぼ100%の割合で得られた。一方、アルゴリズム1、アルゴリズム2では、タスク数が多くなると最適なスケジュールが得にくくなり、タスク数が6,7の時、アルゴリズム1では60%、アルゴリズム2では30%ぐらいの割合に落ちてしまった。

5 考察

シミュレーションの結果から有効と思われるアルゴリズムは、単純にスケジューリング時間を短くするという点からはアルゴリズム1、最適なスケジュールを求めるという点からはアルゴリズム3である。したがって、どちらを選ぶかが問題になるが、これはロボットの状態に依存する。

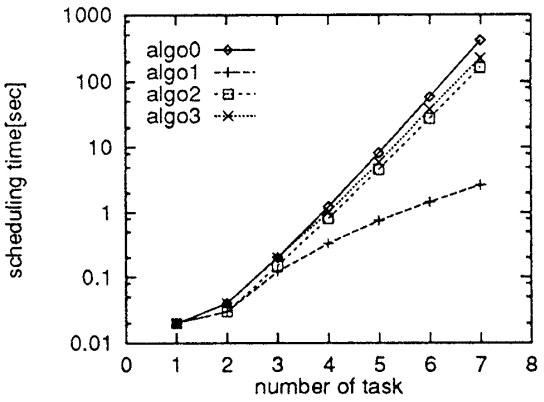

図2: スケジューリング時間

ロボットが移動している時にタスクが依頼される場合、ロボットが向かっている地点に到達するまでの時間が長いほど、アルゴリズム3を使った方が良い。時間が短い場合には、アルゴリズム1を用いて得られるスケジュールとアルゴリズム3を用いたスケジューリングの途中で得られるスケジュールのどちらが良いかが問題となる。

ロボットが停止している場合には、スケジューリングがすべて終ってから移動するのではなく、まず行き先を決めて移動し始めてから、最適なスケジュールを求めていくのが良いと考えられる。このような場合、行き先を決めるのには、時間はかからない方がよい。したがって、まずアルゴリズム1を用いてスケジュールを出し、移動し始めたら新たにアルゴリズム3を用いて最適なスケジュールを求めるというアプローチも考えられる。

6 まとめ

本研究では、インクリメンタルな制約解消機構を持つタスクスケジューラにおいて、3つの制約生成アルゴリズムを提案し比較を行なった。その結果、スケジューラの設計と実装においてスケジューリング時間を考慮することが重要であることを確認した。今後の課題としては、制約生成アルゴリズムを一意に決めてしまうのではなく、状況によって適当なアルゴリズムを選ぶことが挙げられる。

参考文献

[1] Futoshi Naya, Kazuo Hiraki, Yuichiro Anzai "Incremenatal Constraint Satisfaction for Robot Task Scheduling" International Conference on Automation Robotics and Computer Vision(ICARCV'92), RO-9.3.1-9.3.5,1992

[2] 丸一智巳, 安西祐一郎, "時間制約論理モデルとその応用" 人工知能学会誌, Vol.6,No.3,pp.97-105,1991