**グリッドコンピューティングにおける制約条件付きスケジューリング問題へのACO法の適用**

13H023 恩田　征

**1．はじめに**

マシンを複数繋ぐことと、それらを統合的に利用することの間には大きなギャップが存在する。複数のマシンを繋ぐことはグリッドの要件ではあるが、それだけではリソースを有効に活用することは難しい。多数のタスクから構成される大規模なジョブ群に対して、多数のリソースの統合的活用を検討するのが、グリッドコンピュータにおけるスケジューリング問題である。本研究では、スケジューリング問題に焦点を当て、先行制約を持つ問題に対し、最適化手法であるACO法(Ant Colony Optimization)の適用を試みる。特に、制約を考慮するノード空間では解候補の選択対象から除外されるノードが存在するため，選択される可能性のあるノードに対してのみ蒸発する改良を試みる。

**2．スケジューリング問題**

本研究では次の想定で、スケジューリング問題を考える。

1. ネットワークには1つのスケジューラとM個のマシンが接続されている。
2. ネットワークには、J個ジョブが同時に投入される。
3. ジョブjはN(j)個から構成されている。
4. 各マシンのメモリサイズにより、処理できるタスクに関して制限がある。(容量制約)
5. 各ジョブのタスク間には処理順に関する先行制約が存在する場合がある。

と想定する。スケジューリング問題とは、容量制約と先行制約を満たしながら「ジョブの処理完了時刻が最も早くなるように、全てのタスクについて処理リソースとそのリソースでの処理開始時刻を決めること」となる。

**3．最適化手法**

**3.1 ACOの適用**

ACO 法はアリがフェロモンに基づいて餌を探索する行動を模した多点探索法である．解候補の空間をノード空間とし、複数のアリエージェントが、ノード上に蓄積されたフェロモン量をたよりに最適解の探索を行う． まず、解候補を表現するノード空間として、ジョブ内タスクの先行制約を満たす割り当て順序を決めるノード空間(処理順ノード空間)、タスクをマシンへ割り当てる順序を決めるノード空間(配置順ノード空間)、割り当てマシンを決めるノード空間(割り当てノード空間)の3つのノード空間を定義する。処理順ノード空間ではタスクの先行制約を考慮し、配置順ノード空間では全てのタスクの割り当て順序を考慮し、割り当てノード空間ではタスクの容量制約を考慮する。そして、ガントチャート上に時間的先行制約を考慮してタスクを前詰め配置し、各タスクの処理開始時刻を得る。

**3.2 ノード空間**

ジョブ内先行制約を踏まえたジョブ内タスクの処理順を決める処理順ノード空間の構成について記す。各ジョブの処理順ノード空間は 図1に示すようなノード空間でジョブjにおけるノード(α,β)は、β番目に割当てられるタスクがαであることを意味する。したがって、処理順ノード空間はN(j)×N(j)の長方形の空間となる。なお、一度選択されたタスク番号はそれ以降の処理において、選択候補から除外する。また、アリが選択できるのは先行制約を満たすタスク番号のみである。

・・・

1　　　2　　　・・・　　β　　・・・　N(j)

タスク番号

処理順

　N(j)

1

2

α

１

２

・・・

・・・

α

N(j)

・・・

１

２

・・・

α

α

α

・・・

・・・

N(j)

N(j)

N(j)

２

２

１

1

・・・

・・・

・・・

・・・

・・・

・・・

・・・

・・・

・・・

・・・

・・・

図1 ジョブjの処理順ノード空間

各ジョブの先行制約を満たしつつ作業のガントチャートの配置順を決めるノード空間は、図2に示すような空間とする。ノード(α,β)は、β番目に配置されるタスクがジョブαに属していることを意味する。配置順ノード空間はJ×Uの横非常に長い空間となる。

Uはスケジューリングの対象となるジョブに含まれるタスクの総数である。また、ジョブごとにタスクリストを準備し、処理順ノード空間で決定された処理順にタスクを登録する。

・・・

1　　　2　　・・・　　β　　・・・　U

ジョブ番号

　Ｊ

1

2

α

１

２

・・・

・・・

α

Ｊ

・・・

１

２

・・・

α

α

α

・・・

・・・

Ｊ

Ｊ

Ｊ

２

２

１

１

・・・

・・・

・・・

・・・

・・・

・・・

・・・

・・・

・・・

・・・

・・・

配置順

図2 配置順ノード空間

タスクの割り当てマシンを決めるノード空間は、図3に示すようなノード空間で、ジョブjにおけるノード(α,β)はタスクβをマシンαに割当てることを意味する。したがって、割当てノード空間はN(j)×Mの長方形の空間となる。なお、アリが選択できるのは、容量制約を満たすマシン番号のみである。もし、タスクサイズよりマシンαのメモリが小さいなら、タスクβはマシンαに割当てることができないので、ノード(α,β)の初期フェロモン量を0、それ以外の割当て可能なノードの初期フェロモン量は適当な正の値に設定する。　これにより、ノード(α,β)の初期選択確率は0となり、以降のサイクルにおいても選択されることはない。

・・・

1　　　2　　・・・　　β　・・・　N(j)

マシン番号

タスク番号

　Ｍ

1

2

α

１

２

・・・

・・・

α

M

・・・

１

２

・・・

α

α

α

・・・

・

・・・

M

M

M

２

２

１

１

・・・

・・・

・・・

・・・

・・・

・・・

・・・

・・・

・・・

・・・

・・・

図3 割当てノード空間

通常のACOでは，一世代終了するごとに，一定の蒸発率を用いて，処理順ノード空間，配置順ノード空間および割当てノード空間のノード上のフェロモン蓄積量を蒸発・減少させることになる．ただし，本研究では，処理順ノード空間では先行制約のため選択の対象外となるノードが存在する．また，配置順ノード空間でも各ジョブの選択回数制約から選択の対象外となるノードが存在する．そこで，制約上除外されるノードに対しては蓄積フェロモンの蒸発を行わず，選択される可能性のあるノードに対しての蒸発を行うという蒸発率の変更を考えた．

**4．数値実験**

以下のように問題を設定し、数値実験を行った。

* + マシン台数10台 , 1ジョブの含むタスク数30~80個
  + ジョブ数 5個 , タスク総数280個
  + 容量制約率　40% , 0%
  + アリ数　50匹 世代数　10000

「なし」は通常のACO法であり、「あり」は蒸発率変更を含んだ結果である。

　図4において、Job数が増えるほど、処理時間の増加割合が増加することは判明したが、蒸発率変更の有無は最良解の収束性に影響を与えていないことが判明した。しかし、図5の階層の複雑化を比較した場合、蒸発率変更によって最良解が悪化するという結果が出た。

**５．おわりに**

蒸発率の変更がどのようにして悪化しているのかということを今後の課題とし、蒸発率の変更式を見直し、プログラムを改善していく必要があると思われる。