7Q−7

要員スケジュールシステムにおける探索制御手法の導入

岡本一弘　中重亮　小野山隆

日立ソフトウェアエンジニアリング（株）

1．はじめに

計画問題は幅広い応用分野に関わる問題であり、なかでもスケジューリング問題はニーズが高く特に重要なテーマの一つである。近年、このスケジューリング問題をＣＳＰ（Constraint Satisfaction Problem、制約充足問題）として定式化して解こうというアプローチが盛んであるが、ＣＳＰ自体はＮＰ完全であることが知られており、実世界の応用にその枠組を適用するためには解の導出方法に工夫が必要となる。

著者らはＣＳＰによって定式化されたスケジューリング問題を解く、汎用的なスケジューリングシステムの試作を行っている。このシステムの核となる探索ドライバでは、問題の特徴に合わせて探索の制御についてのパラメータの設定やヒューリスティックスなどを記述し、これを利用して効率の良い探索を行うことを考えている。しかしながら、現在のところ「どのような問題に対してどのような制御を行うことが効果的か」といった探索制御手法の利用技術は確立していない。

本報告ではＣＳＰとして定式化されたスケジューリング問題を解く際の探索制御手法の適用可能性を要員配置問題について検討する。また例題に対して探索制御手法を適用した場合の効果の測定結果について述べる。

2．要員配置問題の特徴と探索の制御

ＣＳＰは変数の集合、各変数のドメインの定義、変数間の制約の集合で定義される。要員配置問題の場合、変数として要員または要員を配置する位置（日時、場所、任務）のどちらかをとり、各変数のドメインの値としてもう一方を割り当てるのが一般的である。つまり、「要員を位置に割り当てる」か「位置を要員に割り当てる」かのどちらかである。そして、その割り当て方を規定するのが変数間の制約ということになる。

要員配置問題の特徴としては次のようなものがある。

(1) 制約がきつい。

同じスケジューリング問題でも生産工程のスケジューリングなどと比べて要員配置の方が制約がきついものが多い。一人の要員について着目した場合の勤務日数や勤務パターンに関する制約と、ある一日に着目した場合の要員配置の組合せなどに関する制約を共にうまく満たす解を求めることは難しく、解が存在しない場合もある。そのような場合には許容できる範囲内で制約を緩和するなどして近似解を求めることが必要となる。

(2) 変数間の制約に局所性がある。

これはスケジューリング問題をＣＳＰとしてとらえた場合の共通の特徴と言える。つまり、ある一定期間のスケジュールを作成する場合に、期間中のある時期のスケジュールを決めた影響はその時期から離れたところよりも隣接したところのスケジュールの決定に影響が大きいということである。したがって、要員配置問題ではある日のある勤務の割当はその直前直後の日の勤務の割当から強く制約を受けるが、離れた日の割当から直接制約を受けることは少ない。ＣＳＰの例題としてよく用いられるｎ－クイーン問題で、隣接する行または列からも、離れた行または列からも殆ど同じように制約を受けるのとは対照的である。

(3) 各要員のスケジュールにパターン性がある。

(2) にも関連するが、ある日の（またはある日までの）ある要員の割当が決まるとその日以降の割当が連続的に決まることが多い。したがって、各要員のスケジュールには特定の割当パターンがしばしば現れる。また、制約によって特定の割当パターンしか許さない要員配置問題もある。

一方、探索を制御する手法はさまざまなものが考えられているが[1]、これまでに述べた特徴を利用して効率的に解を導出するための手法としてここでは次のようなものを考えている。

(a) 変数の順序付け

変数間の制約に局所性がある問題では値を割り当てられた変数によって制約を強く受ける変数から順に割り当てることが有効な場合が多い。したがって、それぞれの時点で隣接する変数で割当済みのものが多いものから割り当てていく。

(b) 変数値の固定

変数間の制約に局所性がある問題では、順調に割当が進んでいる時にはある段階でそれ以前の割当を固定し、そこから先の探索では固定済みの変数との間の制約を無視または簡単化してバックトラックの発生を抑制する。

(c) バックトラックの戻り幅の制限

大幅なバックトラックを起こした場合には探索木の中でその時に探索している枝の周辺には解が存在しなかったり、あるいは制約を緩めなければ解を得ることが難しかったりすることが多い。そこで、あらかじめ決めておいた戻り幅を越えるバックトラックが発生した場合にはそこで探索を中断し、制約の緩和、探索方法の修正、オペレータの介入などによって探索を続行できるようにする。

(d) パターンの利用

パターン性のはっきりした問題では、パターンとしてひと固まりとなっている変数のグループに対してそのグループに含まれる変数をまとめて割り当てる。

Search Control Methods for Work Scheduling System
Kazuhiro Okamoto, Ryo Nakasige, Takashi Onoyama
Hitachi Software Engineering Co.,Ltd.

本報告ではこの中から変数の順序付けを適用した場合の効果について報告する。

3．システムの概要

現在試作中の汎用的なスケジューリングシステムはワークステーション上でＣｏｍｍｏｎ Ｌｉｓｐを使用して作成している。図１はシステム構成の概略である。本システムの核となるのが探索ドライバであり、与えられた問題に関する記述と探索制御に関する記述を元にスケジュールの作成を行なう。スケジューリングインタフェースはスケジューリング操作の容易化と分かりやすい形でのスケジュール表示を行う。またアプリケーション記述インタフェースは問題記述および探索制御記述の容易化を行う。例えば、パラメータを与えると最終的な記述（Ｃｏｍｍｏｎ Ｌｉｓｐのプログラムの形式）を生成する自動生成機能などがこの中に含まれている。

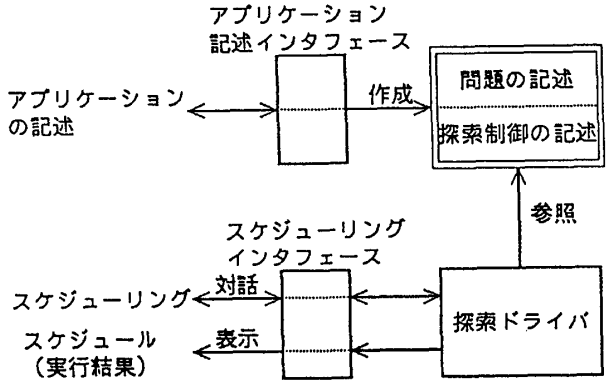

図１　システム構成の概略

4．例題による実験と評価（地図の塗り分け問題）

ここでは、要員配置問題に近い性質を持つ例題として４つの色（例えば、青、赤、白、黄）を使って与えられた地図を隣接する区域が同じ色にならないように塗り分ける問題を例にとる。各領域ｉにその色を示す変数 Ｖｉ を対応付けると、この問題は次のように定義できる。

・変数の集合：｛V_1,V_2,…,Vn｝
・各変数 Ｖｉ のドメイン Ｄｉ ＝ ｛青,赤,白,黄｝
　（ｉ ＝ 1,…,n ）
・制約の集合：
　｛Ｖｉ ≠ Ｖｊ ｜ 領域 ｉ と領域 ｊ は隣接している｝

この例題に探索制御手法（変数の順序付け）を適用した場合の効果を測定する。

地図としては東京２３区とそれを囲む領域に関するものをを用いる。解の探索方法については探索空間上の解の存在位置のバラツキを考慮して、見つける解の個数を１個、１０個、全ての解の３種類について実験する。なお、全解探索については処理時間を短縮するため、実験方法を以下のように二通りに変更して実験を行なった。

(1) 領域数を減らす。（２３＋１→８＋１）
(2) 一部の領域の色をあらかじめ指定する。（４ヶ所）

また、変数の割当方法については順序付けの有無の比較を行う。つまり、

(a) ランダムに与えられた順序に従って割り当てる。
(b) 割当済みの隣接領域数が最大の領域から割り当てる。

以上、２種類の方法を実験する。この場合、ランダムに選んだ変数の順序によって割当のしやすさがかなり変化するので１０種類の順序について実験し、処理コストの最大、最小、平均を見ることにした。処理コストは探索のために掛かった USER CPU時間と探索のステップ数について調べた。

結果は表１の通りである。

表１　変数の順序付けの効果（地図の塗り分け問題）

探索方法	順序付け	結果（上：USER CPU時間(s)）（下：ステップ数）		
		平均	最小	最大
１解探索	なし	59.452	1.38	246.24
		7991.8	83	35263
	あり	4.128	3.84	4.38
		33	25	65
全解探索領域減	なし	15.692	8.58	21.30
		2522.6	969	4721
	あり	9.724	8.06	10.76
		1065	777	1161
全解探索先指定	なし	74.964	26.34	118.38
		9768.8	3373	15167
	あり	7.698	6.90	8.32
		357.4	297	405
10解探索	なし	74.402	3.26	266.04
		10122.4	287	35509
	あり	5.784	5.02	7.08
		134.4	63	263

各方法で、平均USER CPU時間で約１．５～１５倍、平均ステップ数で約２．４～２４０倍、順序付けを行った場合の方が処理が速く、またステップ数が削減されている。ステップ数の減少に比べてUSER CPU時間の短縮が顕著でないのは、順序付けのためのオーバヘッドがあるためと考えられる。また実験方法ごとの結果の比較では、領域数を減らした時には順序付けの効果が少なくなっている。

5．おわりに

現在、本システムを実際の要員配置問題（勤務スケジュール作成）に適用する実験を行っている。実際的な問題ではここで述べた変数の順序付けのような一つの手法の適用だけでなく、複数の手法を組み合わせて適用しなければならない場合も多い。複数の探索制御手法の効果的な組合せ方法、あるいはそれを効率良く得る方法を与えることが今後の課題である。

[参考文献]
[1] Mark S.Fox and Monte Zweben: Knowledge Based Scheduling, AAAI-91 Tutorial(1991)