最適化問題への応用のための並列制約論理型言語の拡張

4L－11　今野 和浩 *† 長塚 雅明 *‡ 小林 直樹 * 松岡 聡 * 米澤 明憲 *

* 東京大学理学部情報科学科　* 東京大学工学部計数工学科

† 現在 日本アイ・ビー・エム (株) 東京基礎研究所　‡ 現在 富士通 (株) HPC本部 第二開発統括部

E-mail: konno@trl.ibm.co.jp

1 はじめに

並列制約論理型言語 PARCS は大規模並列計算機上での実装に向いた宣言的な言語であり、優先度に基づく並列実行制御、整数有限領域上の制約解消と枝刈り、暗黙の OR 並列実行などの特徴を持つ。

PARCS の枝刈り機構はある探索枝から将来分岐する枝の数を劇的に減らすことができる一方で、ある探索枝での結果が他の枝の探索の実行に大きな影響を及ぼす並列の α-β 剪定や分枝限定法などのアルゴリズムにおける枝刈りはこれまで記述できなかった。本研究ではこのような最適化のためのアルゴリズムを記述するために優先度の制御と枝刈りの新しい枠組を提案する。

2 並列制約論理型言語 PARCS

PARCS プログラムの見かけは Prolog のものとよく似ているが、CHIP[3] のような有限領域に関する記述ができる、カットがないなどの点で異なる。また、ゴール内の制約[1]の実行順序はプログラムの文面から静的には決まらず、ランタイムシステムが変数の束縛状態などの実行時の情報によって自動的に優先度を与えることで実行を進める。

PARCS では並列実行のための特別な記述は不要であるが、富士通 AP1000、Thinking Machines CM-5 などの分散メモリ大規模並列計算機上で N- 女王問題、覆面算、準群問題などの組合せ探索問題について良い台数効果を示している。詳しくは [1][2] を参照のこと。

PARCS では静的な実行順序やカットが存在しないため、一度 OR 分岐したゴールはお互いに無関係となってしまい、このため、PARCS は基本的に全解探索の能力しか持っていなかった。PARCS での記述に向いている探索問題のような問題では組合せ爆発を起こしやすく、問題の制約を満たす解全てを求めるのは実行時間・メモリ量などの面から非現実的な場合がある。最適化問題のように制約を満たす多くの解の内の特定のものだけに興味がある場合も多く、このような場合、たとえ全解探索ができても、一度全ての解を求めてから最適解を選ぶのは非効率的である。不必要に探索空間を広げずに最適解に近い枝のみを探索するために探索戦略の記述を行えるようにしたい。また、ある枝の計算結果を用いて、それ以上計算を進めても無駄な (より良い解が得られる見込みがない) 計算途中の OR 枝を刈るという動作を行わせたい。

PARCS の宣言的な記述と整合性を保ちつつ上記の問題に対応するため、優先度指定式と単一解構文を導入する。

3 優先度指定式

PARCS の並列実行制御は各ゴールの優先度付けによって行われるので、この優先度を制御することで深さ優先や幅優先、或いは何らかのヒューリスティクスに基づく探索を行うことが可能となる。ユーザが明示的に探索戦略を制御するために、PARCS の文法を拡張し、節定義や質問の末尾に優先度指定式を付加できるようにする。区切り子として `@` を使う。優先度指定式に解の評価関数を記述することで、望ましい探索枝から探索するようにスケジューリングを制御できる。

ゴール中のある述語が優先度指定式を持つ節とレゾリューションする場合、ゴールの優先度をどのように更新するかは節の優先度指定式で記述できる。例えば、ゴール `:-p(X),q(X)@2.` が節 `p(X):-r(X)@+3.` とマッチすれば新しいゴールは `:-q(X),r(X)@5.` となり (和)、節 `q(X):-s(X)@<3.` とマッチすれば `:-p(X),s(X)@2.` となる (より小さい値を選択)。優先度指定式の値が大きいゴールほど実行時にプロセッサ内で優先的にスケジュールされる。

優先度指定式には定数や通常の変数からなる式が書けるが、特徴的なものとして、PARCS の有限領域を扱う領域変数の上限 / 下限値を与える `upper/lower_bound` 項を式中に書けることが挙げられる。この項には、領域変数が特定の値に束縛されるまでの間、評価関数の近似を行う働きがある。

優先度指定式の使用例は [2] を参照されたい。

4 単一解構文

単一解構文 (single solution construct、以下 SSC) は、カットがなく基本的に全解探索しかできない PARCS に OR ゴール間にまたがる枝刈りの能力を持たせ、かつ、最

Extension to a Parallel Constraint Logic Programming Language for Applications in Optimization Problems.

KONNO Kazuhiro (IBM Japan Ltd.), NAGATSUKA Masaaki (Fujitsu Ltd.), KOBAYASHI Naoki (Dept. of Information Science, Univ. of Tokyo), MATSUOKA Satoshi (Dept. of Mathematical Engineering, Univ. of Tokyo), YONEZAWA Akinori (Dept. of Information Science, Univ. of Tokyo)

[1] PARCS では `:-a(X),X+Y=1,Y!=5.` のような質問全体を「ゴール」と呼び、`a(X)` のような述語呼び出しや `X+Y=1` のような単純な制約をまとめて「制約」と呼ぶ。

適化問題において見込みのないOR枝を刈ってどのように探索空間を狭めるかを記述するためのものである。前節で導入した優先度指定式と併せて使用することで、並列実行の柔軟な制御を宣言的に行うことができる。

節定義の本体や質問にSSC ({}で囲われた制約の列)を高々1つ許すようにする。その動作は、SSC内の制約から解が1つも得られなかった場合、そのSSCを含むゴール自体が失敗し、逆に、複数の解が得られた場合にはそのうちの1つの解のみが得られたものとしてSSCの実行を終了するというものである。論理型言語Gödel[4]の単一解演算子と異なるのは、SSCでは枝を刈るための条件を付加することができることである。

SSC内には通常の制約の他に、ゴール間共有変数の初期化を記述できる。共有変数はあるSSCから分岐したゴールの間で値を共有する変数であり、プログラムの文面上は先頭の文字が#であることで他の変数と区別される。また、枝刈り条件を記述するための新しい文を導入する。枝刈り条件文には以下の3種類の条件を記述できる。

- 唯一解の選択条件
- SSCから分岐したゴールが失敗する条件
- 共有変数値の更新条件

各ゴールは、上記の失敗条件とそれぞれの変数束縛、ゴール間共有変数の値を使ってこれ以上計算を進めても無駄であることが分かったら失敗する (自殺による枝刈り)。SSCが入れ子になって実行される場合には、外側のSSC (親) の共有変数を内側のSSC (子) に引数として渡すことができる。共有変数の値は概念的にはゴール間で共有されているが、分散メモリ環境を前提としているので、必ずしも厳密に同期して値の更新が行われるわけではないことを念頭に条件を記述する必要がある。

通常の (SSCの外の) 制約は必ずSSC内の制約に先んじて実行される。この制限がないと、以下のような場合にどの解が得られるのが正しい実行なのかが決定できないからである[2]。

```
p(2,6). p(3,4). q(3,3).
:- { p(X,Y) }_maximize(Y), q(X,3).
```

実装に当たっては、SSCの実行ごとにSSC receiverと呼ぶエージェントをそのSSC内の制約の実行が始まるプロセッサ内に生成する形にする。SSC receiverは(1)SSCが返す唯一解の管理、(2)SSCゴール間の共有変数の管理、(3)SSCゴールの終了判定を行う (各SSCゴールは負荷分散のためにプロセッサ間を移動するので)。

SSCを使ってα-βを記述した例 (部分) を以下に示す。2者によるゲームのプログラムで、述語playを再帰呼出ししてk手先の局面の評価を行っている。評価値は変数Valに入る。自分の手番ではこのValの値を最大化するのが目標である。述語playの記述は、SSCの枝刈り条件と共有変数の初期化の部分以外は通常のOR並列を使った探索と全く変わらない。maxphase(...) := 以下が新たに導入した枝刈り条件の記述文である。#Betaがβ値、#PAlphaが親SSCのα値を表す ($\alpha \leq \beta$となったら探索を打ち切る)。

```
play(N, Board, Val, #PAlpha) :-
  N<k, even(N),                 /* 自分の手番 */
  my_move(Board, NewBoard),     /* OR分岐 */
  {                             /* SSC */
    #Beta<-MININT,
    play(N+1, NewBoard, Val, #Beta)
  }_maxphase(Val, #PAlpha, #Beta).

maxphase(Val, #PAlpha, #Beta) :=
  select max(Val),              /* 解選択条件 */
  fail when #Beta>=#PAlpha,     /* 失敗条件 */
                                /* β値が大き過ぎる */
  #Beta<-Val when Val>#Beta.    /* 共有変数更新条件 */
```

注目すべき点は、探索自体の記述とα-βによる枝刈りの記述が完全に分離されていることである。また、より良い解が得られそうな探索枝を述語my_move中に優先度指定式で記述することで、さらにα-β剪定の効果を高めることが可能である。

5 まとめ

PARCSに優先度指定式と単一解構文 (SSC) を導入し、組合せ探索問題において不必要に探索空間を広げずに最適な解に近い枝を優先的に実行するための記述を宣言的に行えるようにした。また、例としてSSCを用いてα-β剪定を記述し、探索自体の記述と枝刈りの記述を分離できることを示した。

優先度指定式の実装・評価は終わっているが、SSCについては効率の良い大規模並列環境での実装方式をさらに検討している段階である。実装を終え、その効果やオーバーヘッドを評価することが今後の課題である。

参考文献

[1] N. Kobayashi, S. Matsuoka, and A. Yonezawa. Control in Parallel Constraint Logic Programming. In *Proceedings of the Logic Programming Conference '91*, 1991.

[2] K. Konno, M. Nagatsuka, N. Kobayashi, S. Matsuoka, and A. Yonezawa. PARCS: An MPP-Oriented CLP Language. In *Proceedings of the First International Symposium on Parallel Symbolic Computation (PASCO '94)*, pp. 254–263, 1994.

[3] M. Dincbas, P. Van Hentenryck, *et al.* The Constraint Logic Programming Language CHIP. In *Proceedings of the International Conference on FGCS '88*, pp. 693–702, 1988.

[4] P. M. Hill and J. W. Lloyd. *The Gödel Programming Language*, MIT Press, 1994.

[2] 述語qが先に実行されると解X=3, Y=4が得られるが、pが先に実行された場合には解が存在しない。