Javaによる異種協調制約解消システムの開発

Java Implementation of a Heterogeneous Constraint Solving System

番原 睦則 [1)]　田村 直之 [2)]　井上 克巳 [3)]　川村 尚生 [4)]
Mutsunori BANBARA　Naoyuki TAMURA　Katsumi INOUE　Takao KAWAMURA

1) 神戸大学学術情報基盤センター学術情報処理研究部門
(〒657-8501 神戸市灘区六甲台町 1-1, e-mail: banbara@kobe-u.ac.jp)

2) 神戸大学学術情報基盤センター学術情報処理研究部門
(〒657-8501 神戸市灘区六甲台町 1-1, e-mail: tamura@kobe-u.ac.jp)

3) 神戸大学工学部電気電子工学科
(〒657-8501 神戸市灘区六甲台町 1-1, e-mail: inoue@eedept.kobe-u.ac.jp)

4) 鳥取大学工学部知能情報工学科
(〒680-8552 鳥取市湖山町南 4-101, e-mail: kawamura@ike.tottori-u.ac.jp)

ABSTRACT. Recent development of cooperative constraint solving systems suggests a successful direction to extend constraint programming to be more intelligent and more efficient. However, the entirely almost systems have supported only homogeneous constraint solvers. We have developed the HECS (HEterogeneous Constraint Solving) system that enables real-time collaboration among heterogeneous constraint solvers. Each solver works separately on an individual thread and communicates with the others through shared blackboard. In addition to constraint solvers for finite domains of integers and boolean values, our system supports local-search-based solvers and a Mathematica interface.

1 背景

本ソフトウェアは平成 14 年度 IPA 未踏ソフトウェア創造事業 (紀 PM) で開発したものである.

制約プログラミングは,「ユーザは解決したい問題を制約の形で宣言的に記述するだけで, あとは制約ソルバがその制約を満たす解を求めてくれる」という問題解決手法といえる. 1990 年代に商用の制約ソルバが登場して以来, 制約プログラミング手法に基づく生産スケジューリング, 資源割り当てなどの実用的なシステムが数多く開発・運用されている. またごく最近では企業向けの SCM/ERP システムにも応用されている.

近年, 問題の多様化・複雑化に伴って協調制約解消の研究が進み,「複数の制約ソルバをいかに有効に組み合わせて使うか」が重要な研究課題になっている. しかしながら, これまで開発されたシステムは同種の制約ソルバしか扱えないものが多かった. さらにグリッド計算などの分散コンピューティング環境の進歩に伴い, 今後ますます様々な異種の制約ソルバのための協調制約解消アルゴリズムの重要性が高まると考えられる.

2 目的

本提案の目的は, ネットワーク上に分散している複数の異種の制約ソルバを協調的・競争的に並列動作させることにより, 単一のアルゴリズムをチューニングする以上の効果を得ることである.

3 システムの概要

開発した協調制約解消システム HECS は, 各種ソルバとそれらを管理するスケジューラ部から構成されている.

- 複数の制約ソルバを管理するスケジューラ:
  このスケジューラは, 並列動作する複数の制約ソルバを協調・競争させ, 効率よく解を求めるためのものであり, 以下のような機能をもつ.
  - 問題記述
  - 複数の制約ソルバの並行実行
  - 複数の制約ソルバの分散実行

  実装方法としては, Prolog から Java へのトランスレータ処理系 Prolog Cafe[1] をマルチスレッド化し, この Prolog Cafe 上でスケジューラの実装を行った. また, 分散環境における (制約ソルバの) スケジューリングを実現するために, Prolog Cafe を拡張したモバイルエージェントシステム記述言語処理系 Maglog の開発も行った.
- 整数制約ソルバ, および確率的制約ソルバ:
  Java 上の制約プログラミング用クラスライブラリ Cream[2] を開発した. また, Cream 上で分岐限定法, シミュレーティッド・アニーリング法, タブー探索法に基づく 3 種類の確率的制約ソルバを開発し, これらを使った近似最適解探索が可能となった.
- ブール制約ソルバ:
  Java による並行動作・協調 SAT ソルバ multisat[3] を開発した. このシステムでは確率的 SAT ソルバを含む各種 SAT ソルバ (bf, satz, walksat, zchaff) が利用できる. プランニング問題を SAT 問題に変換して解く SAT プランニングにも応用可能である.
- Mathematica:
  Mathematica の Java インタフェイス J/Link を用いて, Prolog Cafe の Mathematica インタフェイスを実装した. 計画問題などのソルバとして利用できる.

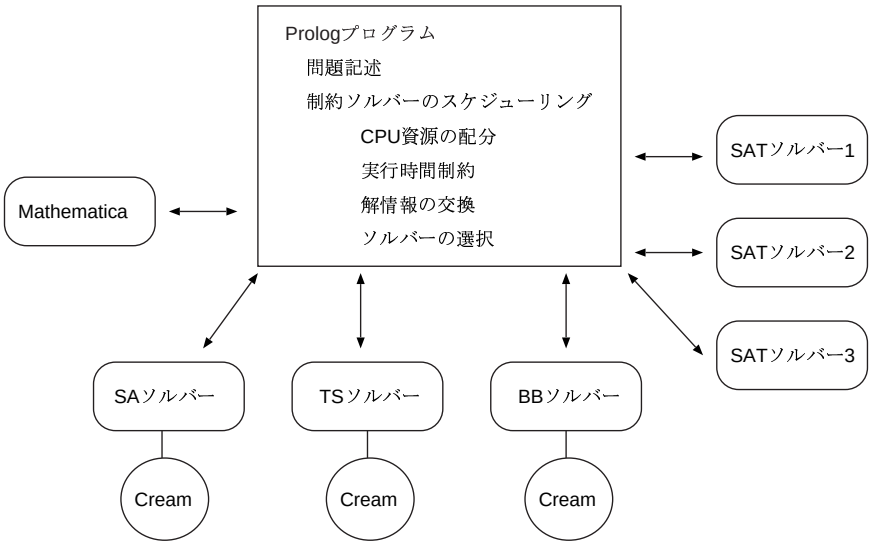

図 1: システムの概要

図 1 に HECS システムの概略図を示す．このシステムはすべて Java で実装されており，Java から各制約ソルバを統一的に取り扱うことができるというメリットもある．また，各構成要素はそれ自身単体のソフトウェアとしても利用可能である．

4 全体構成と動作環境

協調制約解消システム HECS は，以下のサブシステムから構成されている．

- Prolog Cafe: Prolog から Java へのトランスレータ処理系 (担当：番原，付録 A 節参照)
- Cream: Java 上の制約プログラミング用クラスライブラリ (担当：田村，付録 B 節参照)
- multisat: 並行動作・協調 SAT ソルバ (担当：井上，付録 C 節参照)
- Maglog: Prolog Cafe を拡張したモバイルエージェントシステム記述言語処理系 (担当：川村，付録 D 節参照)
- Prolog Cafe の Mathematica インターフェイス (担当：田村)

HECS システムは構成要素を含めすべて Java で実装されているため，Java2 の動作する環境であれば利用可能である．ただし，HECS, Prolog Cafe, Maglog のコマンドツールが Perl および Ruby で記述されているため，それら及び Windows 上では Cygwin がインストールされている環境が望ましい．また, Mathematica インターフェイスを利用するためには，ウルフラムリサーチ社の数学統合環境ソフトウェア Mathematica および Mathematica の Java インターフェイスである J/Link が必要である (Mathematica インターフェイスを利用しない場合は，必要ない)．なお，Mathematica の最新バージョンは 4.2 であり，Windows 版, Mac 版, Unix 版 (Linux を含む) が販売されている．

5 開発の目標と達成した外部仕様

HECS システム開発にあたって，実現を目標とした項目は以下の通りである．

1) 複数の制約ソルバの並行実行機能
2) 複数の制約ソルバの分散実行機能
3) 複数の制約ソルバ間での解情報の交換機能
4) 実行時間制約下での解探索機能（与えられた時間内での最善解探索機能）
5) 複数の制約ソルバに対して CPU 資源を動的に割り当てるスケジューリング機能
6) 複数のソルバから最適なソルバを選択する機能
7) 整数有限領域上で，最適解の探索および準最適解の確率的探索を行うソルバ (上記の並行実行機能等に対応したもの)
8) ブール制約問題に対する，SAT ソルバおよび確率的 SAT ソルバ (上記の並行実行機能等に対応したもの)
9) 実数領域上での線形な制約条件に対して，最適解の探索を行うソルバ (上記の並行実行機能等に対応したもの)

本年度のプロジェクト以前に開発したソフトウェアとしては，Prolog から Java へのトランスレータ処理系である Prolog Cafe および Java 上の制約プログラミング用クラスライブラリである Cream があるが，上記の機能は備えていなかった．

上記の目標のそれぞれに対して，達成した外部仕様との対比は以下の通りである．

1) 複数の制約ソルバの並行実行機能
   Prolog Cafe のマルチスレッド化，および Cream のマルチスレッド化により達成できた．
2) 複数の制約ソルバの分散実行機能
   Prolog Cafe を拡張したモバイルエージェントシステム記述言語処理系である Maglog を開発することにより，枠組は完成した．しかし，具体的な問題について分散実行を実現することはできなかった．
3) 複数の制約ソルバ間での解情報の交換機能
   整数有限領域上のソルバ間での解情報の交換は達成できた．
4) 実行時間制約下での解探索機能
   整数有限領域上のソルバに対しては達成できた．
5) 複数の制約ソルバに対して CPU 資源を動的に割り当てるスケジューリング機能
   検討だけに留まり，達成できなかった．
6) 複数のソルバから最適なソルバを選択する機能
   検討だけに留まり，達成できなかった．
7) 整数有限領域上で，最適解の探索および準最適解の確率的探索を行うソルバ
   Cream にこれらの機能を追加することにより達成で

きた.

8) ブール制約問題に対する，SAT ソルバおよび確率的 SAT ソルバ
multisat の開発により達成できた.
9) 実数領域上での線形な制約条件に対して，最適解の探索を行うソルバ
Prolog Cafe の Mathematica インターフェイスを開発することにより達成できた.

6 内部仕様

HECS の実装技術のうち重要なものは，以下の通りである.

- Prolog Cafe サブシステム: Prolog プログラムを Java プログラムに変換し実行する．複数のプログラムを並行実行するマルチスレッド機能，Cream や multisat 等の Java プログラムを利用する機能，Mathematica システムを利用する機能を持つ.
- Cream サブシステム: 分岐限定法による最適解の探索機能，準最適解の確率的探索機能，複数の探索を組み合わせた並行探索機能，および解情報の交換による並行探索における協調機能を持つ．また，ベンチマークであるスケジューリング問題にも対応している.
- multisat サブシステム: ブール制約問題に対する解の探索機能，確率的な解の探索機能，複数の探索を組み合わせた並行探索機能を持つ.
- Maglog サブシステム: 複数のマシンに分散した複数のエージェント，およびそれらの協調の場である複数のフィールドからなるマルチエージェント・システムを記述できる．Prolog Cafe の拡張として開発されており，プログラムはすべて Java に変換され実行される.

7 検査方法

HECS システムをインストールし，`examples` ディレクトリ中のプログラムを下記のように実行することで，対応する外部仕様が実現されていることを確認できる.

- (1) 複数の制約ソルバの並行実行機能，(3) 複数の制約ソルバ間での解情報の交換機能，(4) 実行時間制約下での解探索機能，(7) 整数有限領域上で最適解の探索および準最適解の確率的探索を行うソルバ，については以下の通り (図 2 のジョブショップ問題に対する最適解探索の実行例を参照).

```
$ bin/hecs examples/jssp/ft10.hecs
```

- (1) 複数の制約ソルバの並行実行機能，(8) ブール制約問題に対する SAT ソルバおよび確率的 SAT ソルバ，については以下の通り.

```
$ bin/hecs examples/sat/p2.hecs
```

- (9) 実数領域上での線形な制約条件に対して最適解の探索を行うソルバについては以下の通り.

```
$ bin/hecs examples/mma/mma1.hecs
```

8 性能評価と今後の課題

当初の目標のうち HECS 全体の枠組に関するものは実現できたという点で，当初の目標の大半は達成できたと自己評価している.

今後の課題は，「解の交換，CPU 資源の割り当て，ソルバの選択/切替え」などの機能をもつ，複数ソルバのための高度なスケジューリング・システムの構築である．CPU 数が 1 つの集中管理型スケジューリングを想定し問題を整理すると，

- 実行時間 $\mathcal{T} = [0, T]$.
- ソルバの数 N.
- 時間 $t \in \mathcal{T}$ におけるソルバ i の暫定解 $x_i(t)$ および暫定値 $f_i(t)$.
- ソルバーの選択/切替え：
  - 単位時刻ごとに割り付けるソルバを選択する.
  - 任意の時刻で（例えば解が交換されるたびに）ソルバを切替える.
- 解の交換：
  - どの時点（例えばソルバが切替えられた時点）で
  - どのソルバ（現在 CPU が割り当てられているソルバ）に
  - どうのような解（例えば現時点で最良の暫定解 $f^*(t)$）を渡す.

などが考えられる．この問題は形式的には，シングルマシン・スケジューリング問題に属するものと捉えられる．しかし，リアルタイム・スケジューリングの枠組で，これまで提案されているアルゴリズム (スケジューリング/ディスパッチング手順) はジョブの処理時間や納期 (デッドライン)，周期をベースにしたものがほとんどであり，ここでのスケジューラにそのまま使うことはできない.

ここでのスケジューラでは，事前にソルバの処理時間 (1 試行に要する時間）すら与えられない状況下でのスケジューリング問題が対象となる．すなわち，ソルバを選択する基準をスケジューリング・アルゴリズムのなかで見積もることが必要となる．例えば，このようなスケジューリング基準として，各ソルバ i に対して，

- これまでに割り当てられた CPU 時間の割合 $a_i(t)/t$,
- 現時点での暫定解の良さ $f_i(t)/f^*(t)$,
- 暫定解の更新履歴 $h_i(t)$(さらなる加工が必要)
- 1 回の試行あたりの解 (評価値) のゲイン $\Delta f_i(t)/f_i(t)$

などが考えられる．これらのスケジューリング基準から 1 つあるいは複数選び，これをソルバの選択/切替え手順に用いれば，リアルタイム・スケジューリングを実現できるのではないかと考えている．具体的には，遺伝的機械学習 (Pitt アプローチ)，強化学習の利用を検討している.

9 入手方法

HECS の Web ページ

`http://kaminari.istc.kobe-u.ac.jp/hecs/`

からソフトウェアおよびマニュアルのダウンロードが可能である．ただし，各サブシステムのソースコードは含まれていないので，上記 Web ページからリンクしている各々のサブシステムの Web ページからダウンロードする必要がある.

HECS はフリーソフトウェアである．multisat サブシステム中の walksat と zchaff プログラム以外は，Free Software Foundation によって公開されている GNU General Public License の第二版以降に従う限り，誰でも自由に複製，改変，再配布することができる.

なお，利用している walksat プログラムは Daniel Jackson によって開発されたものを改変したものであり，オリジナルは `http://sdg.lcs.mit.edu/~dnj/walksat/` で公開されている.

10 参加企業及び機関

日本エンジェルズ・インベストメント株式会社
(プロジェクト実施管理組織)

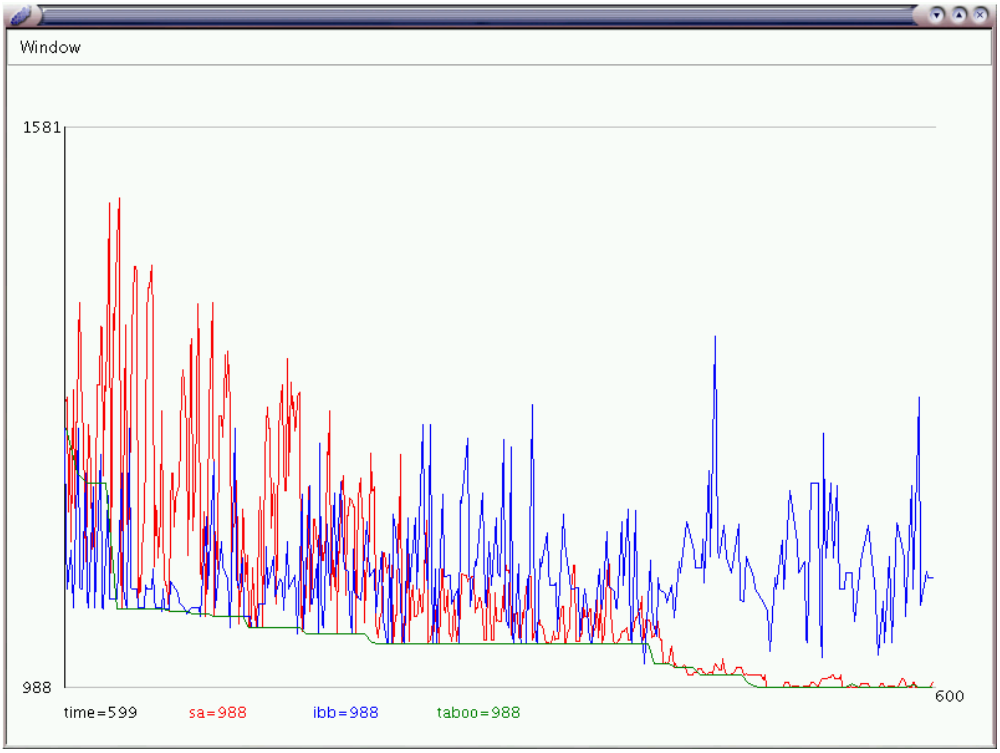

図 2: ジョブショップ問題に対する最適解探索の実行例

11 謝辞

本開発のプロジェクトマネージャーである紀 信邦氏に心から感謝致します．本開発にオブサーバとして参加していただいた玉置 久氏に感謝致します．また，プログラム作成に協力していただいた学生アルバイトの上田 盛慈氏，鵜飼 訓史氏，大西 秀志氏，谷澤 勉氏，榊原 一紀氏に感謝致します．

参考文献

[1] Mutsunori Banbara and Naoyuki Tamura. Translating a linear logic programming language into Java. Electronic Notes in Theoretical Computer Science, Vol. 30, No. 3, 2000.

[2] Shuji Ohnishi, Hiroaki Tasaka, and Naoyuki Tamura. Efficient representation and operations of descrete sets for constraint programming. In Ninth International Conference on Principles and Practice of Constraint Programming (CP'03), Octorber 2003. (投稿中).

[3] 上田盛慈, 鵜飼訓史, 井上克巳, 番原睦則, 田村直之, 川村尚生. Sat ソルバの並列実行に関する一考察. 電子情報通信学会 人工知能と知識処理研究会, 5 月 2003. (発表予定).

付録 A Prolog Cafe：Prolog から Java へのトランスレータ処理系

Prolog Cafe は Prolog から Java へのトランスレータ処理系である．Prolog Cafe は Prolog プログラムを Java プログラムにトランスレートし，それを Java コンパイラ (`javac` など) でコンパイルし，実行時システムと共に実行する，という方式をとっている．トランスレータは SICStus Prolog で記述されているがブートストラップされており，Java が動く環境であれば何処でも利用可能である．実行速度は SWI-Prolog より 6〜7 倍遅い程度に抑えられている．

以下に Prolog Cafe の特徴を挙げる．

- オープン・ソース
- 100% Pure Java で実装されており，移植性が高い
- 拡張性が高い (Java のクラスライブラリなどを簡単に取り込める)
- Java からの Prolog の呼び出しが可能
- Prolog からの Java の呼び出しが可能
- マルチスレッド化により，複数ゴールの並行実行が可能

Prolog Cafe は Java™2 SDK, Standard Edition が動作する環境であれば，基本的にはどこでも利用可能である．ただし，開発用のコマンドツールが Perl で記述されているため，Perl バージョン 5.6.1 以上がインストールされている環境が望ましい．Prolog Cafe の最新バージョン (ソースコードを含む) は，

`http://kaminari.istc.kobe-u.ac.jp/PrologCafe/`

から入手可能である．

付録 B Cream: Java 上の制約解消プログラミング用クラスライブラリ

Cream (Constraint Resolution Enhancement And Modules) は，Java 上での制約プログラミングのためのクラス・ライブラリであり，以下の点を特徴とする

- 100% Pure Java
- オープン・ソース
- Java 構文での自然な制約記述が可能
- 制約記述や最適化アルゴリズムの拡張・改良が容易
- 分岐限定法による最適解探索が可能
- スケジューリング問題への対応

- シミュレーティッド・アニーリング法，タブー探索法などの準最適解探索アルゴリズムを利用可能
- 複数探索プログラムの並行実行が可能
- 複数探索プログラム間で，準最適解の交換による協調が可能
- タイムアウトの設定による探索の打ち切り機能

Cream は Java™2 SDK, Standard Edition が動作可能な環境であれば利用できる．Cream の Web ページ

`http://bach.scitec.kobe-u.ac.jp/cream/`

からソフトウェアおよびマニュアルのダウンロードが可能である．

付録 C　multisat: 並行動作・協調 SAT ソルバ

multisat は SAT ソルバの並列実行システムであり，以下の点を特徴とする．

- 100% Pure Java
- オープン・ソース
- 確率的 SAT ソルバを含む 4 種類の SAT ソルバ (bf, Satz, WalkSAT, zChaff) が利用可能
- 問題に対して異種の SAT ソルバを並行的，協調的 (解情報の交換) に解かせ，複数のモデルをより高速に得ることが可能
- プランニング問題を SAT 問題に変換して解く，SAT プランニングにも応用可能.

multisat は Java™2 SDK, Standard Edition が動作する環境であれば，基本的にはどこでも利用可能である．multisat の Web ページ

`http://cslab.eedept.kobe-u.ac.jp/~ueda/multisat/`

からソフトウェアおよびマニュアルのダウンロードが可能である．

付録 D　Maglog: モバイルエージェントシステム記述言語処理系

Maglog は分散コンピューティング環境に対応した論理型言語処理系であり，以下の点を特徴とする．

- 100% Pure Java
- オープン・ソース
- 複数のゴールの並行/分散実行が可能
- 強マイグレーション機能
- モバイルエージェント・システムの開発に適している．
- フィールド (複数のゴールのための共有空間) を利用したメッセージパッシングが可能.

Maglog は Java™2 SDK, Standard Edition が動作する環境であれば，基本的にはどこでも利用可能である．ただし，開発用コマンドツールが Ruby で記述されているため，Ruby がインストールされている環境が望ましい．Maglog の Web ページ

`http://nu.ike.tottori-u.ac.jp/kawamura/maglog/`

から (ソースコードを含む) ソフトウェアがダウンロードが可能である．