プログラミング 6－15
（1996. 3. 27）

制約プログラミングによる木の描画

安達由洋　土田賢省　夜久竹夫

{adachi, kensei}@krc.eng.toyo.ac.jp
yaku@loadstar.chs.nihon-u.ac.jp

東洋大学工学部　〒350 埼玉県川越市鯨井2100
日本大学文理学部　〒156 東京都世田谷区桜上水3-25-40

木の描画問題には，NP完全のものが数多く存在する．NP完全な問題に対する手続き的描画プログラムの開発は描画条件ごとにプログラムを作る必要があり困難な作業である．

本研究では，制約論理プログラミングにより宣言的にかつ柔軟に木の描画問題を扱えるように定式化し，有限領域上で効率良く解く方法を考察した．そしてメタプログラミングの技法を用いて配置条件に対応する制約プログラムを自動的に生成し，そのプログラムを制約パッケージを用いて求めた解を X Window 上に表示するシステムを実現した．このシステムは，NP完全に対応する条件のもとでノード数100の木の描画問題を約１秒で解くことができる．

キーワード：　制約プログラミング，グラフ描画，情報の可視化，
計算の複雑さ，NP完全，論理プログラミング

Tree Drawing by Using Constraint Programming

Yoshihiro ADACHI　Kensei TSUCHIDA　Takeo YAKU

Toyo University, 2100 Kujirai Kawagoe Saitama 350 Japan
Nihon University, 3-25-40 Sakurajyousui Setagaya Tokyo 156 Japan

A tree drawing problem under some layout conditions becomes NP-complete.
We studied methods to program tree drawing problems by using constraint logic programming, and to solve them efficiently in finite domains. We have implemented a system which generates a constraint program for a tree drawing problem automatically by using meta-programming in Prolog. The system can solve a tree drawing problem of 100 nodes under a NP-complete condition in about a second.

Keywords: constraint programming, graph drawing, visualization,
computational complexity, NP-complete, logic programming

1 はじめに

木の描画問題には，ノードのレイアウト条件に依存して線形時間で解ける問題，NP完全のクラスに属する問題，計算量のクラスがまだ分かっていない問題などがある[1]．そして，NP完全問題に対する手続き的プログラムを開発するには描画条件ごとにプログラムを作る必要があり，また開発されたプログラムの保守や拡張は困難な作業となる．

一方，制約プログラミングは効率的なアルゴリズムが無い問題や存在してもプログラミングが非常に難しい問題などに対する解決パラダイムとして人工知能分野で注目され研究されてきた．近年，化学工場や石油精製工場での生産スケジューリング，ビル内のコンピュータと電話のネットワーク設計あるいは旅客機乗務員の乗務計画など実際的な問題解決にも適用され始めて，その有効性が確認されている[2]．しかしながら，制約プログラミングの形式的モデルと利用できるツールとの間のギャップにより，制約プログラミングを実用的問題へ適用する際の難しさも指摘されており[3]，また制約プログラミングを用いて実問題を効果的に解決するには，その問題に対する知識を利用する必要があることも報告されている[4]．

本研究では，制約論理プログラミングにより宣言的にかつ柔軟に木の描画問題を扱えるよう定式化し，有限領域上で効率良く解く方法を考察した．そしてメタプログラミングの技法を用いて配置条件に対応する制約プログラムを自動的に生成し，そのプログラムを制約パッケージを用いて求めた解を X Window 上に表示するシステムを Prolog を用いて実現した．このシステムは，NP完全に対応する条件のもとでノード数100の木の描画問題を約１秒で解くことができる．

2 木の描画問題の定式化

木の描画問題の形式的な定義と，レイアウト条件の制約式による定式化を行う．なお，この節の内容は文献[5]に基づいている．

2．1 木構造図の定義

定義１．木構造Ｔは５項組Ｔ＝(Ｖ, Ｅ, ｒ, width, depth)である．ここで，Ｖはセルの集合，Ｅはエッジの集合，(Ｖ, Ｅ)は順序木であり，ｒ∈Ｖはルートセルである．写像width：Ｖ→Ｚはセルの幅関数，写像depth：Ｖ→Ｚはセルの深さ関数である． □

木構造でセルとは，長方形の箱である．セルｐ∈Ｖの垂直方向の長さは width(ｐ)で表され，ｐの幅と呼ばれる．また，ｐの水平方向の長さは height(ｐ)で表され，ｐの深さと呼ばれる．

定義２．木構造Ｔ＝(Ｖ, Ｅ, ｒ, width, depth)の配置とは，関数π：Ｖ→Ｚ×Ｚのことである．π_x(ｐ)とπ_y(ｐ)はそれぞれπ(ｐ)のｘ座標とｙ座標を表す． □

定義３．Ｔ＝(Ｖ, Ｅ, ｒ, width, height)を木構造，πをＴの配置とするとき，Ｄ＝(Ｔ, π)を木構造図と呼ぶ． □

木構造図とは，セルをその座標上におき，エッジを線分で結んだ図である．本論文では，ｘ座標を水平方向に，ｙ座標を垂直方向にとり，セルの左上の点の座標をセルの座標とする．

定義４．Ｔ＝(Ｖ, Ｅ, ｒ, width, height)を木構造，πをＴの配置，Ｄ＝(Ｔ, π)を木構造図とする．Ｄの幅およびＤの深さは次の関数でそれぞれ定義される：

$$\mathrm{width}(D) \equiv \max\{|\pi_y(p)+\mathrm{width}(p)-\pi_y(q)| : \forall p, q \in V\},$$

$$\mathrm{depth}(D) \equiv \max\{|\pi_x(p)+\mathrm{depth}(p)-\pi_x(q)| : \forall p, q \in V\}.$$ □

定義５．Ｔ＝(Ｖ, Ｅ, ｒ, width, height)を木構造，πをＴの配置，Ｄ＝(Ｔ, π)を木構造図とする．Ｄの面積は次のように定義される

$$\mathrm{area}(D) \equiv \mathrm{width}(D) \times \mathrm{height}(D) \in Z.$$ □

定義６．木構造図の描画問題とは以下のように定義される：木構造Ｔ＝(Ｖ, Ｅ, ｒ, width, height)に対して，ある美的制約を満たしながらarea(Ｄ)を最小にする配置πを見つけることである．ただし，Ｄ＝(Ｔ, π)である． □

さらに，以下の関数を導入する．

$$\mathrm{Index}(p) \equiv \begin{cases} 0 & : p \text{がルートセルのとき} \\ i & : p \text{が} p \text{の親セルの} i \text{番目の子セルのとき} \end{cases}$$

Child(ｐ)≡ｐの子セルの集合
Parent(ｐ)≡ｐの親セル
Level(ｐ)≡セルｐとルートセルとの間のエッジの数

2．2　木構造図描画の制約条件

木構造図の全てのセルの座標が分かれば，容易に木構造図を描画することができる．したがって，木構造図の描画問題はセルの２次元座標への割当問題と見なせる．

木の描画でよく採り入れられている制約をもとにして，以下のように木構造Ｔの配置に関する条件を導入する．

条件B0　エッジが直線で描かれるとき，どのエッジも他のエッジやセルと交差しない．　□

条件B1^a　（セルのｘ座標は祖先セルの深さの総和で決まる）．ｐを木構造図Ｄのセル，$p_0=r, p_1, \cdots, p_m=p$をルートセルｒからｐに至るパス上のセルとする．このとき，

$$\pi_x(p)=\sum_{(0\leq i\leq m-1)}(\mathrm{depth}(p_i)+1).$$　□

条件B1^b　（同じレベルのセルはすべて同じｘ座標を持つ）．任意のｐ，ｑ∈Ｖに対して，Level(ｐ)＝Level(ｑ)ならば$\pi_x(p)=\pi_x(q)$．　□

条件B2　（親セルは子セルの中央に位置する）．木構造図Ｄに対して，セルｐがｋ個の子セル$q_1, \cdots, q_k$（Index(ｑi)＝ｉ，$1\leq i\leq k$）を持つとき，$\pi_y(p)$は次式を満たす：$\pi_y(p)+\lfloor \mathrm{width}(p)/2 \rfloor = \pi_y(q_1)+\lfloor (\pi_y(q_k)+\mathrm{width}(q_k)-\pi_y(q_1))/2 \rfloor$．　□

条件B3　（兄弟セルは同じｘ座標上に上から下に順番に配置される）．木構造図Ｄに対して，セル$p_0, p_1, \cdots, p_k$を兄弟セルとし，Index(p_i)＝ｉ$(1\leq i\leq k)$とする．このとき，$\pi_y(p_i)<\pi_y(p_{i+1})$かつ$\pi_x(p_i)=\pi_x(p_{i+1})$，$(1\leq i\leq k)$である．　□

条件B4　（どのエッジも他のエッジと交差しない）．任意のｐ，ｑ∈Ｖに対して，$\pi_x(p)+\mathrm{depth}(p)=\pi_x(q)+\mathrm{depth}(q)$かつ$\pi_y(p)<\pi_y(q)$のとき，

$$\max\{\pi_y(p')+\mathrm{depth}(p') \mid \forall p' \in \mathrm{Child}(p)\} \leq \min\{\pi_y(q')+\mathrm{depth}(q')+1 \mid \forall q' \in \mathrm{Child}(q)\}.$$

□

条件B5　（同型な部分木構造図は平行移動に関して合同に配置される）．任意の部分木構造図$D_1=(T_1, \pi)$と$D_2=(T_2, \pi)$に対してT_1とT_2が位相同形な部分木構造のとき，任意の$q_1\in V_1$とそれに対応する$q_2\in V_2$に対して

$$\pi_x(q_1)-\pi_x(q_2)=\pi_x(T_1\text{のルートセル})-\pi_x(T_2\text{のルートセル}),$$

$$\pi_y(q_1)-\pi_y(q_2)=\pi_y(T_1\text{のルートセル})-\pi_y(T_2\text{のルートセル}).$$

□

条件B6　（どのセルも他のセルと交差しない）．木構造図Ｄ＝(Ｔ，π)とセルｐ∈Ｖに対して，集合pseudo-area(ｐ，π)を次のように定義する：

$$\text{pseudo-area}(p, \pi)\equiv\{(x,y) \mid \pi_x(p)<x<\pi_x(p)+\mathrm{depth}(p)+1,\ \pi_y(p)<y<\pi_y(p)+\mathrm{width}(p)+1\}.$$

このとき，すべてのｐとｑ($p\neq q$)に対して

$$\text{pseudo-area}(p, \pi)\cap\text{pseudo-area}(q, \pi)=\phi$$

□

条件B4とB6より，条件B0が導かれる．条件B6はすべてのセルがｘ座標とｙ座標ともに互いに少なくとも１離れていることを意味する．

これらの描画条件はソフトウェア開発教育の中で，木構造図をプログラム流れ図に適用した経験により得られた．

2．3　美的制約と描画アルゴリズムの複雑さ

前述した条件を結合して，整数格子上の木構造図に関する美的制約C1とC2を構成する．

C1＝B1^a∧B2∧B3∧B4∧B5∧B6

C2＝B1^a∧B2∧B3∧B4∧B6

描画問題の難しさは美的制約に依存する．

定理１[1]．与えられた木構造Ｔ，正の整数Ｗ，および美的制約C1に対して，width(Ｄ)≦Ｗを満たす配置πが存在するか否かの判定問題はNP完全である．　□

Ｎ≠ＮＰを仮定すれば，美的制約C1のもとで最小配置を与える多項式時間アルゴリズムは存在しない．さらに，制約C1から条件B5を除いた制約C2のもとでは多項式時間アルゴリズムが存在するか否かも分かっていない．

3　木構造図描画システム

制約論理プログラミングを用いた木構造図描画システムについて説明する．このシステムは，前節で定義した美的制約C1とC2を主に扱っている．

3．1　システムの構造

木構造図描画システムは，図１に示すように制約生成系，制約解消系，表示系から構成される．

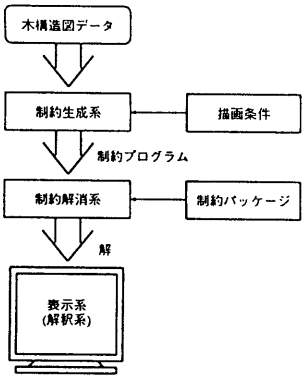

図１　木構造図描画システムの構成

本システムの入力は木構造データを次のようにPrologの項表現したものである．

セル・データ　：cell(ID, Width, Depth)

エッジ・データ：edge(ID1, ID2)

ここでID, ID1, ID2はセルの識別子(名前)であり，WidthとDepthはそれぞれセルの幅と深さの値である．この入力に対して，制約生成系は指定した描画条件に対応する制約Prologプログラムをメタプログラミングにより生成する．制約解消系は生成されたプログラムを実行して有限領域で制約式を解きその結果得られたセルの配置データをファイルに出力する．なお，制約解消にはIF/Prolog制約パッケージを用いている．表示系は，セルの配置データとエッジデータをもとに X window上に木構造図を描画する．

３．２　制約プログラムの生成

美的制約C1とC2において，各セルのｘ座標は条件B1*よりアルゴリズム的に容易に決定できる．そこで，各セルのｙ座標を決定するための制約プログラムのみを生成している．図２に示す例をもとに制約表現の生成法を説明する．ただし，セルの中に書かれている数字をセルの名前とし，簡単のため各セルの幅と深さはともに１とする．

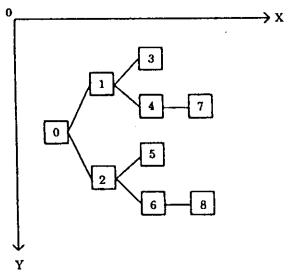

図２　木構造図の例

条件B3∧条件B6の制約式

有限領域での比較演算子 ?>= を用いて以下のように記述される：

$\pi_y(2)$?>= $\pi_y(1)+1+1$

$\pi_y(4)$?>= $\pi_y(3)+1+1$

$\pi_y(5)$?>= $\pi_y(4)+1+1$

$\pi_y(6)$?>= $\pi_y(5)+1+1$

$\pi_y(8)$?>= $\pi_y(7)+1+1$

たとえば，最後の式ではセル８の座標は，セル７の座標に幅１とセル間の最小ギャップ１を加えた値より大きいことを表している．

条件B2の制約式

有限領域の制約では除算は使えない．条件B2で与えられた式の両辺を２倍し，各セルの幅が１を考慮すると例えば次の等式による制約式が得られる．

$2*\pi_y(0)$?= $\pi_y(1)+\pi_y(2)+1-1$ または

$2*\pi_y(0)$?= $\pi_y(1)+\pi_y(2)+1$

これら２式は，セル０がセル１とセル２の中央に位置するための制約式である．この式をもとにシステムをインプリメントしたところ制約を解くのに時間がかかった．そこで次のように不等式制約に置き換えバックトラックを無くしたところ数倍以上の速度向上があった．

$2*\pi_y(0)$?>= $\pi_y(1)+\pi_y(2)+1-1$ かつ

$2*\pi_y(0)$?=< $\pi_y(1)+\pi_y(2)+1$

ここでは幅１の場合について説明したが，幅が１でない場合も偶数か奇数かをチェックする事により類似の式が導ける．

条件B5の制約式

同型性のチェックは，各セルに対してそのセルをルートとする部分木構造をリストで表現したラベル付けを行い，そのラベルのユニフィケーション(パターンマッチ)で実行している．各セルには次の形式のラベルが付けられる．

[Width, Height|[第１子のラベル, …]]

図２の木構造図例におけるセルとそのラベルを幾つか示す：

セル名	ラベル
1	[1, 1, [1, 1], [1, 1, [1, 1]]]
2	[1, 1, [1, 1], [1, 1, [1, 1]]]
3	[1, 1]
4	[1, 1, [1, 1]]
5	[1, 1]
6	[1, 1, [1, 1]]

これより，セル１をルートとする部分木構造とセル２をルートとする部分木構造が同型であること

をラベルのユニフィケーションにより検出できる．この手法により，図２の木構造図に対して生成した条件B5に対応する制約式は次のものである．

$\pi_y(1)-\pi_y(2)$?= $\pi_y(3)-\pi_y(5)$
$\pi_y(1)-\pi_y(2)$?= $\pi_y(4)-\pi_y(6)$
$\pi_y(4)-\pi_y(6)$?= $\pi_y(7)-\pi_y(8)$
$\pi_y(4)-\pi_y(6)$?= $\pi_y(7)-\pi_y(8)$

本システムでは上記のように同型性の条件として冗長な制約式を生成している．これは，冗長性を無くすためのPrologプログラムが実行時間がかかり，冗長な制約式を解くほうがはるかに速いという実験結果に基づいている．

変数の領域の決定

木構造図Ｄ＝(Ｔ，π)の幅は，すべてのセルを１ずつ離してｙ方向に一列に並べたときの幅以下である．したがって，各変数の領域を０からDomainまでとする．ただし，ｐ∈Ｖに対して

Domain＝$\sum_p$(width(p)＋１)－１．

以上の各条件に対応する制約式をもとに，本システムが自動的に生成したPrologプログラムを以下に示す．

```
solve_constraints([(8,6,A),(7,6,B),(6,4,C),
 (5,4,D),(4,4,E),(3,4,F),(2,2,G),(1,2,H),(0,0,I)]) :-
      [A,B,C,D,E,F,G,H,I] in 0 .. 17,
      A ?>= B + 1 + 1,
      C ?>= D + 1 + 1,
      C ?= A,
      D ?>= E + 1 + 1,
      E ?>= F + 1 + 1,
      E ?= B,
      G ?>= H + 1 + 1,
      2 * G ?>= D + C + 1 - 1,
      2 * G ?=< D + C + 1,
      2 * H ?>= F + E + 1 - 1,
      2 * H ?=< F + E + 1,
      2 * I ?>= H + G + 1 - 1,
      2 * I ?=< H + G + 1,
      E - C ?= B - A,
      E - C ?= B - A,
      H - G ?= E - C,
      H - G ?= F - D,
      minimize_maximum(label([A,B,C,D,E,F,G,H,I]),
         [A,B,C,D,E,F,G,H,I]).
```

ここで，minimize_maximum(?Goal,?List)は制約パッケージの組み込み述語で，Goalを満たす条件のもとでList要素の最大値を最小化する．したがって，このゴールsolve_constraints/1が成功したとき，最小幅の木構造配置が得られる．

４　システムの実行と性能評価

木構造図描画システムの実行例とその性能について説明する．なお，すべての実験は SUN SPARC station20 上で実行している．

図３は，16個のセルを持つ木構造図を描画制約C1のもとで描画した画面である．制約Prologプログラム生成に0.03秒，ゴールを実行する(制約を解く)時間は百分の１秒以下で測定できない．

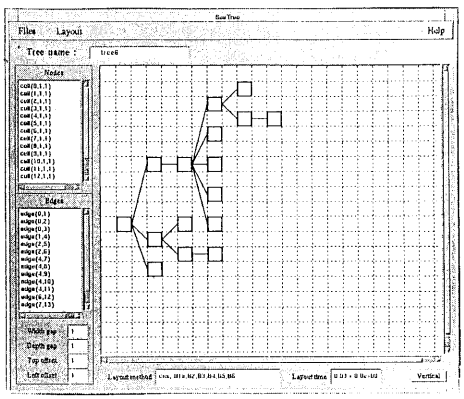

図３　描画制約C1での描画画面

図４では同じ問題を描画制約C2のもとで解いている．処理時間にはほとんど差がない(速すぎて差が測定できない)．図３と図４を比べてみると，同型性の条件B5が入っている図３のほうが木構造図の幅が１大きいことが分かる．

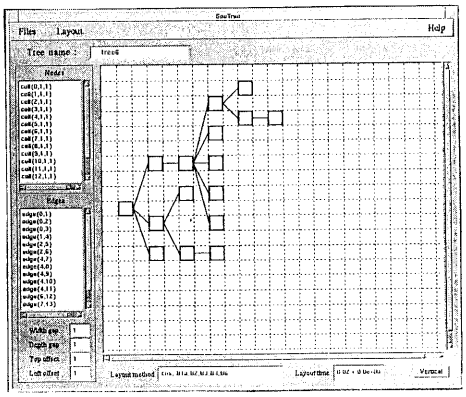

図４　描画制約C2での描画画面

図５は，300個のセルを持つ木構造図を描画制約C1，すなわちNP完全の条件のもとで描画した画面である．制約Prologプログラムの生成に11.22秒，ゴールの実行に0.7秒かかっている．このように本システムでは制約Prologプログラムの生成に時間がかかっており，ゴールの実行は相当大きな問題でも１秒以下である．

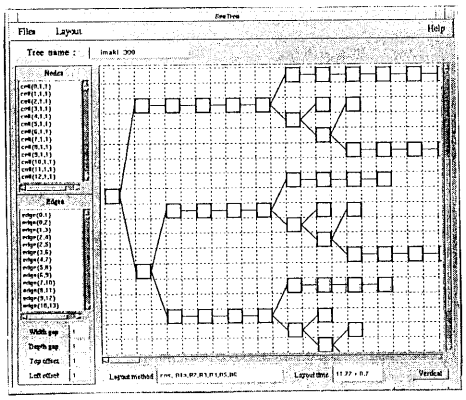

図５　300セルを持つ木構造図の描画

図６には，木構造図のセルの個数と本システムの処理時間(制約プログラム生成時間＋実行時間)の関係をグラフで示した．このグラフでも分かるように，本システムはNP完全に対応する美的制約C1のもとで，セル数100の木構造図の描画問題を約１秒で解くことができる．

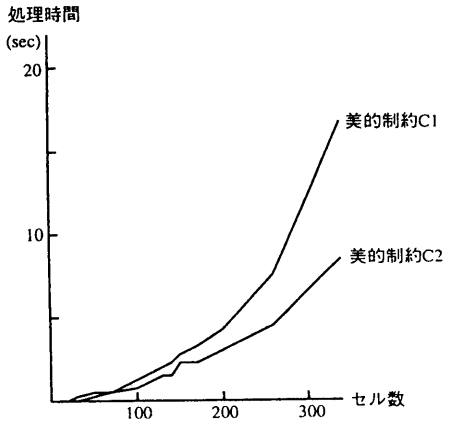

図６　セルの個数と処理時間

5　おわりに

木構造図の描画問題を制約論理プログラミングを用いて解決するための定式化を行い，その処理システムをPrologを用いて実現した．このシステムは，描画条件に対応した制約論理プログラムをメタプログラミング技法により自動的に生成するという特徴を持つ．

また，同じ描画条件でも制約プログラムの表現に依存してその処理速度が大きく異なることを実験的に確認した．すなわち，制約プログラムを効果的に用いるには対象とする問題に関する知識を積極的に利用することが必要となる．我々のシステムでは，同型性のチェック，変数の領域の決定やバックトラックを無くすために木構造図描画に関する知識とPrologプログラミング上の工夫を用いており，これによりNP完全に対する条件のもとでセル数100の木構造図の描画問題を約１秒で解くことができた．

なお，本システムはIF/Prologとその制約パッケージを用いて約2,500Prolog行で実現されている．

参考文献

[1]K. Tsuchida: The Complexity of Drawing Tree-Structured Diagrams, IEICE Trans. Inf. & Syst., Vol. E78-D, No. 7, pp. 901-908 (1995)

[2]H. Simonis: The CHIP System and Its Applications, LNCS Vol. 976, pp. 643-646, Springer-Verlag (1995)

[3]N. Guerinik and V. Caneghem: Solving Crew Scheduling Problems by Constraint Programming, LNCS Vol. 978, pp. 481-498, Springer-Verlag (1995)

[4]J. F. Puget: Applications of Constraint Programming, LNCS Vol. 978, pp. 647-650, Springer-Verlag (1995)

[5]K. Tsuchida et al.: Constraints and Algorithms for Drawing Tree-Structured Diagrams, Proc. of Int. Workshop on Constraints for Graphics and Visualization in CP95, pp. 87-101 (1995)

[6]今木，他：制約プログラミングによる木の美的描画，情報処理学会第52回全国大会，5N-4 (1996)