GUIを対象とした線形計算による制約階層解消系の高速化

細部 博史

GUIの構築において，制約の優先度を実現する制約階層を容易に利用できるようにするため，我々はこれまでに，線形計算を用いたインクリメンタルな制約解消系HiRiseを提案している．本研究では，HiRiseをさらに高速化するために，多くの実アプリケーションにおいて制約階層が(1)疎な構造を持ち，(2)多数の不連結な成分からなっている，という性質を積極的に利用する手法を提案する．そして，提案した手法の導入によるHiRiseの高速化に関する実験結果を報告する．

1 はじめに

グラフィカルユーザーインターフェース(GUI)の構築において，グラフィカルオブジェクト群のレイアウトなどのために**制約**を用いる研究が，これまでに数多く行われてきた．特に，このようなアプローチでは，オブジェクトの振舞いを適切に制御するために，制約に**優先度**を指定できるようにすることが有効であることが指摘され，**制約階層** [2]と呼ばれる定式化に基づくものを中心に様々な研究がなされてきた[5][6].

制約階層を用いてインタラクティブなGUIを構築するには，差分的に処理を進める**インクリメンタル**な制約解消が重要とされ，過去に多数のインクリメンタルな制約解消系が提案された．それらの多くは，**局所伝播法** [4][9][10][11]または最適化アプローチ[3]と呼ばれる方式に分類できる．

一方，我々はこれまでに，局所伝播法や最適化アプローチとは異なる，線形計算を用いたインクリメンタルな制約解消系HiRiseを提案し，HiRiseが，巨大な制約階層に対して，既存の制約解消系より優れた能力を発揮することを示した[7].

本研究では，多くの実アプリケーションにおいて制約階層が持っている以下の2つの性質を積極的に利用して，HiRiseをさらに高速化する2つの手法を提案する．

疎性: 制約階層が疎な構造を持っている．

不連結性: 1つの制約階層が多数の不連結な成分からなっている．

そして，提案した2つの手法の導入によるHiRiseの高速化に関する実験結果を報告する．

2 HiRise制約解消系の概要

HiRise制約解消系[7]が扱う制約には，1次等式，stay (変数の値を一定にする)，edit (変数の値を繰り返し更新する)の3種類がある．そして，各制約には，**強さ**と呼ばれる優先度が与えられる．強さは有限個(デフォルトで16)の段階からなり，常に満たすべき必須制約のレベル`required`を最上位とし，その下に`strong`，`medium`，`weak`のような選好制約のレベルが続く．

HiRiseは，内部的に，制約階層を**階層線形系** [6]に変換して処理を行う．制約階層では，優先度は有限段階で

Speeding Up HiRise, a Linear Constraint Hierarchy Solver for Graphical User Interfaces.

Hiroshi Hosobe, 文部省学術情報センター研究開発部, Department of Research and Development, National Center for Science Information Systems.

コンピュータソフトウェア, Vol.17, No.2(2000), pp.25–29.

[小論文] 1999年8月9日受付.

あるが，階層線形系では，優先度は個々の制約ごとに異なり，全順序になっている．

HiRise の主要データ構造は，階層線形系の LU 分解である．これは，**階層独立**である 1 次等式制約 (より優先度の高い制約に対して係数ベクトルが線形独立である制約) の係数ベクトルを集めて係数行列を作り，LU 分解したものである．以下では，LU 分解の対象となる制約を**有効**な制約と呼ぶ．

HiRise における LU 分解は，有効な制約の係数ベクトルからなる係数行列を B とすると，以下のように表現できる．

$$BT_1T_2\cdots T_s = L \qquad (1)$$

ただし，L は全ての対角要素が 1 である下 3 角行列であり，T_k $(1 \le k \le s)$ は，以下の P_k, U_k, Q_k のいずれかの形をした変換行列である*．

$$P_k = \begin{pmatrix} E_{i_k-1} & & & & \\ & 0 & & 1 & \\ & & E_{j_k-i_k-1} & & \\ & 1 & & 0 & \\ & & & & E_{n-j_k} \end{pmatrix}$$

$$U_k = \begin{pmatrix} E_{i_k-1} & & \\ & \frac{1}{b'_{i_k i_k}} & -\frac{b'_{i_k,i_k+1}}{b'_{i_k i_k}} \cdots -\frac{b'_{i_k n}}{b'_{i_k i_k}} \\ & & E_{n-i_k} \end{pmatrix}$$

$$Q_k = \begin{pmatrix} E_{i_k-1} & & \\ & & 1 \\ & E_{n-i_k} & \end{pmatrix}$$

直観的に，(1) は，$T_1, T_2, \ldots, T_s$ が，係数行列 B を下 3 角行列 L に変換すると理解できる．

3 提案する高速化手法

本節では，HiRise 制約解消系のための 2 つの高速化手法を提案する．

3.1 疎性を利用する手法

実際の GUI アプリケーションにおいては，階層線形系全体が巨大でも，個々の制約は数個の変数だけを参照することが多い．このため，有効な制約の係数ベクトルからなる係数行列は，多くの要素が 0 である，疎な行列になる．

しかし，係数行列を下 3 角行列へ変換する過程で，係数行列で要素が 0 だった位置に，フィルイン [8]と呼ばれる，非零要素を生じさせてしまうことがある．このようなフィルインの発生が多いと，疎の度合を低下させてしまい，疎行列の有効性を活かせなくなってしまう可能性がある．

そこで，本研究では，疎行列のオーダリング法 [8]を採り入れることで，HiRise の LU 分解の過程で，フィルインの発生を抑えるようにした．

3.1.1 基本アイデア

単純な階層線形系の定式化 [6]では，1 つの階層線形系の中で，制約の優先度は 1 つずつ異なり，全順序になっている．しかし，この定式化を修正して，制約を，必ず満たされるべき必須制約と，より低い優先度を持つ選好制約に分けることも可能である．この場合，必須制約の集合は，通常の線形系 (連立 1 次方程式) として見なすことができる．

このような階層線形系では，必須制約の間に優先度の違いがないので，自由にそれらを解く順序を選択できる．従って，係数行列の必須制約に対応する部分では，行の順序を自由に変えることができる．このため，この部分に対しては，疎行列のオーダリング法を適用できる．

3.1.2 アルゴリズム

上記のアイデアに基づき，疎行列のための核の生成と Tewarson の方法 [8]を，HiRise の LU 分解を最初から求める処理に以下のように導入する．

1. 以下の 2 段階の処理で，核の生成を行う．
   (a) 1 個の変数だけを参照している必須制約を探す．そのような制約があれば，その非零要素をピボット位置に移動した後，さらにこの操作を繰り返す (ただし，これ以降は，ここで処理した変数を無視する)．
   (b) 他の必須制約に参照されていない変数を持つ必須制約を探す．そのような制約があれば，その非零要素をピボット位置に移動して，未処理の変数に対応する要素を消去した後，さらにこの操作を繰り返す (ただし，これ以降は，この制約を無視する)．

* E_m は $m \times m$ 単位行列を表す．

2. 残った必須制約の集合について，順次，LU 分解を行う．ピボットにする変数と制約の選択は，Tewarson の方法に従う．すなわち，その時点までの LU 分解を適用して得られる行列を調べて，(その制約に対応する行における未処理の非零要素の個数 -1) $\times$ (その変数に対応する列における未処理の非零要素の個数 -1) が最小となることを基準とする．

3.2 不連結性を利用する手法

局所伝播法では，制約階層をデータフローグラフとして管理する．このため，制約階層の中で関連のない部分は，グラフ中の異なる不連結な成分に分かれる．局所伝播法は，基本的に，グラフをたどることによって処理を進めるので，制約階層全体としては巨大でも，その時点で対象としている成分が小さければ，高速に制約解消を行うことができる．

一方，従来の HiRise 制約解消系では，制約階層 (内部的には階層線形系) 全体を 1 つの LU 分解で管理していた．この方式でも，そのアルゴリズムの高速性のために，ある程度まで対処できるが，それでも，大量の細かい不連結な成分からなる制約階層に対しては，局所伝播法に比べて不利になる．

以下では，この問題を解決するために，HiRise において不連結な成分を扱う手法について述べる．

3.2.1 基本アイデア

制約階層の不連結な成分ごとに，別々の階層線形系を作って管理する．この場合，制約の追加によって複数の成分が連結されるときに，別々に求められていた LU 分解を効率的に統合する必要がある．

制約階層の 2 つの不連結な成分に対応する階層線形系を考える．元の成分が不連結なので，それらの変数の集合に重なりはない．このため，階層線形系を 1 つにまとめても，各制約の階層独立性は変化しない．従って，まとめる前の 2 つの階層線形系について，有効な制約を取り出したものをそれぞれ $B\boldsymbol{x}=\boldsymbol{c}$, $B'\boldsymbol{x}'=\boldsymbol{c}'$ とすると，それらをまとめた階層線形系から有効な制約を取り出したものは以下のように書ける．

$$\begin{pmatrix} B & 0 \\ 0 & B' \end{pmatrix}\begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{x}' \end{pmatrix}=\begin{pmatrix} \boldsymbol{c} \\ \boldsymbol{c}' \end{pmatrix} \qquad (2)$$

ここで B, B' を LU 分解したものをそれぞれ $BT_1T_2\cdots T_s=L$, $B'T_1'T_2'\cdots T_{s'}'=L'$ とすると，以下の式が成り立つ．

$$\begin{pmatrix} B & 0 \\ 0 & B' \end{pmatrix}\begin{pmatrix} T_1 & 0 \\ 0 & E \end{pmatrix}\cdots\begin{pmatrix} T_s & 0 \\ 0 & E \end{pmatrix} \times\begin{pmatrix} E & 0 \\ 0 & T_1' \end{pmatrix}\cdots\begin{pmatrix} E & 0 \\ 0 & T_{s'}' \end{pmatrix}=\begin{pmatrix} L & 0 \\ 0 & L' \end{pmatrix}$$

これは，(2) の係数行列を (1) の形で LU 分解したものと見なせる．従って，このようにすれば，別々に解かれた複数の階層線形系を容易に 1 つにまとめることが可能である．

3.2.2 アルゴリズム

上記のアイデアを利用すれば，制約階層の不連結な成分の扱いは，以下のように制約の追加の処理を行うことで対処できる．

1. 新たに追加される制約が参照する変数について，それらが属する階層線形系を調べる．そのような階層線形系が複数ある場合，それらの階層線形系を 1 つにまとめ，先に述べた方法で，それらの LU 分解を 1 つに統合する．
2. 新しい制約が属する階層線形系に対して，従来と同様の方法で，制約追加の処理を行う．

4 実　　験

本研究で提案した高速化手法を評価するために，巨大で複雑な制約階層を生成するアプリケーションを用いて実験を行った．実装には Java を用い，コンパイルと実行には，Microsoft 製のコンパイラーと仮想機械†を用いた．実験の環境には，Windows NT 4.0 Workstation で動作する，Pentium II 450 MHz を搭載した PC/AT 互換機を使用した．

実験には，図 1の木の編集のアプリケーションを用いた．このアプリケーションでは，同じ親ノードを持つ部分木が隣接し，さらに葉ノードの間隔が一定になるように，レイアウトのための制約を定義している．また，木の構造を決定する制約には全て必須制約を用い，さらにいくつかの選好制約を利用して，木の振舞いを制御できるようにした．

本研究で提案した 2 つの高速化手法の効果を調べるた

† JVC Ver. 6.00.8424 と WJView Ver. 5.00.3176.

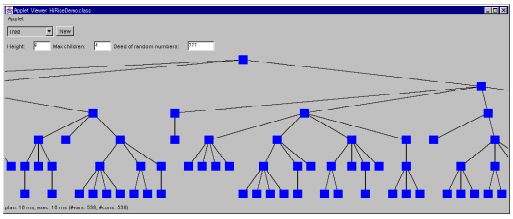

図 1 木を編集するアプリケーション

めに，2 種類の実験を行った．実験 1 は，疎性を利用する手法の効果を調べるもので，501 個のノードからなる木 (変数 2006 個，制約 2010 個) について，その木を編集する際に要する制約解消の時間を測定した．その実験結果 (表 1) から，疎性を利用する手法が，制約解消の高速化に大きく貢献することが確認された．

実験 2 は，不連結性を利用する手法の効果を調べるもので，実験 1 に用いた木を 2 つ生成し，その一方について実験 1 と同じ操作を行い，さらに新たな操作として，一方の木を他方の木の葉ノードに子として接続することを行った．実験の結果 (表 2)，不連結性を利用する手法が，制約解消の高速化に有効であることが確かめられた．

5 関連研究と議論

局所伝播法は，制約階層をデータフローグラフとして管理し，それをたどって，1 つずつ制約を解くことで制約解消を行う．そのため，フィルインの発生はなく，疎性の利用は極めて有効に行われる．しかし，このような方式のために，制約の連立が必要な場合に，解を誤る危険性がある [1]．一方，HiRise は制約の連立を適切に処理でき，本研究の手法を導入しても，そのような問題を生じない．

3.2節で述べたように，制約階層の不連結性への対処についても，局所伝播法では，特別な処理を導入することなく自然に行われる．特に，制約の削除によって 1 つの連結成分が複数の不連結な成分に分割される場合，その後は，それらの成分を別々に処理できる．一方，このようなことは，現在の HiRise ではできない．

局所伝播法では，制約階層を表すデータフローグラフの構造を分析することで，インクリメンタルな制約解消の効率を向上した例がある [10]．その手法は，本研究で述べた疎性や不連結性の利用とは全く異なったアプローチであり，興味深い．ただし，これと類似のアイデアが，HiRise のような数値的アルゴリズムに対しても有効であるかどうかは，現時点で不明である．

最適化アプローチの制約解消系である Cassowary と QOCA では，疎性と不連結性を積極的に利用した高速化は行っていない [3]．ただし，一般の最適化アルゴリズムで利用されている手法を導入して，そのような高速化を実現できる可能性はある．しかし，従来の HiRise に比べても Cassowary は極めて遅く [7]，簡単にその差が縮まるとは考え難い．

6 おわりに

本研究では，HiRise 制約解消系の高速化のために制約階層の疎性と不連結性を利用する 2 つの手法を提案し，それらの有効性を実験により確認した．

今後の課題は，それらの性質の利用をさらに拡大することである．疎性については，その利用を選好制約にまで拡げることを，一方，不連結性については，制約の削除によって連結成分が複数の成分に分割される場合を扱うことを検討している．

参考文献

[1] Borning, A.: Problem with SkyBlue and Cycles, 1995. http://www.cs.washington.edu/research/constraints/solvers/skyblue-cycles.html.

[2] Borning, A., Duisberg, R., Freeman-Benson, B., Kramer, A. and Woolf, M.: Constraint Hierarchies, *Proc. ACM OOPSLA*, 1987, pp. 48–60.

[3] Borning, A., Marriott, K., Stuckey, P. and Xiao, Y.: Solving Linear Arithmetic Constraints for User Interface Applications, *Proc. ACM UIST*, 1997, pp. 87–96.

[4] Freeman-Benson, B. N., Maloney, J. and Borning, A.: An Incremental Constraint Solver, *Comm. ACM*, Vol. 33, No. 1 (1990), pp. 54–63.

[5] 服部隆志: 編集操作におけるマクロと制約の統合, インタラクティブシステムとソフトウェア *IV—WISS'96* (田中二郎, ed.), 近代科学社, 1996, pp. 41–49.

[6] 細部博史, 松岡 聡, 米澤明憲: 階層線形系を用いた効率的な制約階層解消法, インタラクティブシステムとソフトウェア *V—WISS'97* (尾内理紀夫, ed.), 近代科学社, 1997, pp. 129–134.

[7] 細部博史, 松岡聡, 米澤明憲: HiRise: GUI 構築のためのインクリメンタルな制約解消系, コンピュータソフトウェア, Vol. 16, No. 6 (1999) (印刷中).

[8] 小国力, 村田健郎, 三好俊郎, J.J. ドンガラ, 長谷川秀

表 1 実験 1 の結果 (単位：ミリ秒)

	初期配置	移動開始	移動繰返し	移動終了	ノード追加	ノード削除
従来の HiRise 制約解消系	7390	710	140	290	1320	700
疎性の利用	2260	120	10	50	180	90
疎性と不連結性の利用	1500	110	10	40	140	80

表 2 実験 2 の結果 (単位：ミリ秒)

	初期配置	移動開始	移動繰返し	移動終了	ノード追加	ノード削除	木の接続
従来の HiRise 制約解消系	27780	1440	270	580	2730	1470	2080
不連結性の利用	14230	850	260	540	1220	960	2460
疎性と不連結性の利用	2530	180	20	70	180	150	1100

彦: 行列計算ソフトウェア, 丸善, 1991.

[9] Sannella, M.: SkyBlue: A Multi-Way Local Propagation Constraint Solver for User Interface Construction, *Proc. ACM UIST*, 1994, pp. 137–146.

[10] Suzuki, T., Kakinuma, N. and Tokuda, T.: An Experimental Comparison of Three Modified DeltaBlue Algorithms, *Principles and Practice of Constraint Programming—CP'96* (Freuder, E. C., ed.), Lecture Notes in Computer Science, Vol. 1118, Springer-Verlag, 1996, pp. 425–435.

[11] Vander Zanden, B.: An Incremental Algorithm for Satisfying Hierarchies of Multi-Way Dataflow Constraints, *ACM Trans. Prog. Lang. Syst.*, Vol. 18, No. 1 (1996), pp. 30–72.