単体法の反復適用による不完全CSPの高速近似解法

2J－3

斎藤 逸郎　石塚 満
東京大学工学部電子情報工学科
e-mail: saito@miv.t.u-tokyo.ac.jp

1 はじめに

制約充足問題 (CSP) は広い範囲に応用可能であるが，実際問題に適用するとすべての制約が満たせず不完全CSP(PCSP) となる場合が多々ある．このような問題に対して，PCSP を 0–1 整数線形計画問題 (ILP) に帰着させ解く手法について発表を行ってきた [1].

ここでは得られた ILP をどの様に解くかについて発表を行う．

2 PCSPからILPへの帰着

2.1 ILPへの帰着

PCSP のうちアーク制約要素に重みがないものについては，アーク制約要素に対応する変数を用いずに制約式・目的関数を記述することができる．このようにすることで ILP の簡素化がはかれ，解の高速化が望める．

そこでアーク制約要素に重みがある問題については「アークに基づく帰着法」とし，アーク制約要素重みがない問題については「ノードに基づく帰着法」とする．

変換後の 0–1 変数・制約式・目的関数は以下のようになる．

2.2 アークに基づく帰着法

0–1 変数

x_{ik} ノード i の要素 d_{ik} に対応する変数．$x_{ik}=1$ の時，ノード i が値 k をとる．

$y_{ik:jl}$ アーク制約要素 (d_{ik}, d_{jl}) に対応する変数．$y_{ik:jl}=1$ の時，アーク $d_i:d_j$ はアーク制約要素 (d_{ik}, d_{jl}) により満たされる．

$z_{i:j}$ アーク $d_i:d_j$ の成立・不成立に対応する変数．$z_{i:j}=1$ の時，制約は不成立となる．

制約式

$$\sum_k \sum_l y_{ik:jl} + z_{i:j} = 1$$

$$x_{ik} \le \sum_l y_{ik:jl} + z_{i:j}$$

$$\sum_k x_{ik} = 1$$

目的関数

$$\begin{aligned}\min a \;=\; & \sum_i \sum_k W(d_{ik}) x_{ik} \\ & + \sum_i \sum_k \sum_j \sum_l W(d_{ik}:d_{jl}) y_{ik:jl} \\ & + \sum_i \sum_j W(d_i:d_j) z_{i:j}\end{aligned}$$

2.3 ノードに基づく帰着法

0–1 変数

x_{ik} ノード i の要素 d_{ik} に対応する変数．$x_{ik}=1$ の時，ノード i が値 k をとる．

$z_{i:j}$ アーク $d_i:d_j$ の成立・不成立に対応する変数．$z_{i:j}=1$ の時，制約は不成立となる．

制約式 一つのアークのアーク制約要素のうち，一方のノードの値の変数が x_{ik} となるアーク制約要素の他方のノードの値の変数を $x_{jl'}$ とする．つまり $(x_{ik}, x_{jl'})$ を列挙したものがアーク制約要素となるようにする．

$$x_{ik} \le \sum_{l'} x_{jl'} + z_{i:j}$$

$$\sum_k x_{ik} = 1$$

目的関数

$$\begin{aligned}\min a \;=\; & \sum_i \sum_k W(d_{ik}) x_{ik} \\ & + \sum_i \sum_j W(d_i:d_j) z_{i:j}\end{aligned}$$

Fast solution method of patial CSP
by repeat use of simplex method
Itsuro SAITO, Mitsuru ISHIZUKA
Courses of Info. & Commun. Eng., Univ. of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo, 113, JAPAN

3 単体法の反復適用

上記の方法によりPCSPをILPに置換えることが出来る．ILPの解法としては，ILPの緩和問題(LP)の実数最適解を求め，実数最適解の周りの局所探索を行う手法があげられる[2].

しかしながら，PCSPをILPに置換えたものは，元のPCSPが制約が強いためすべての制約を満たせないにも関わらず，局所制約伝播による整合化が困難である特徴を持つ．そのためILPの最適解周辺でのコスト平面は凸凹が多く局所最適が数多くあり，単純に局所探索を行うことでは近似最適解を得ることは困難であると考えられる．

そこでILPの緩和問題の実数最適解を求め，実数解のうち適当なものを0-1に丸めることにより，問題規模を縮小させたILPをつくる．縮小させたILPも同様に実数最適解を求め，さらに縮小させたILPをつくる．この手法を適当回数繰り返すことにより，最終的なILPの解を得ることとした．ここで緩和問題の解法として単体法を用い，高速化を行った．

ここで問題となるのが，どのノードの値についてどの様に丸めを行うかである．

3.1 丸め先の選択

まず後者について考える．どの値に丸めるかは，丸めによる制約の状態への影響が最も少ない点について丸めを行うのが良いと考えられる．そこで，ノードの各要素の値に対応する変数を x_{ik} とし，単体法の解のうち最も大きな値をもつ変数を $x_{ik_{max}}$ とする．この時ノード i について丸めを行う場合，丸め先は $x_{ik_{max}}$ を1とし，それ以外を0とする点が最も良いと考えられる．

3.2 丸めるノードの選択

どのノードについて丸めを行うかは，丸めによる制約の状態への影響が少ないノードに対して行うのが良いと考えられる．そこで，ノードの各要素の値に対応する変数を x_{ik} とし，単体法の解のうち最も大きな値をもつ変数を $x_{ik_{max}}$ とする．この時ノード i について，実数解と丸め先との距離 NDL_i は以下の式で求まる．この NDL_i を各ノードについて求め，小さいものに対して丸めを行うのが良いと考えられる．

$$NDL_i = \sqrt{\sum_{k \neq k_{max}} x_{ik}^2 + (1 - x_{ik_{max}})^2}$$

3.3 丸めの手法

そこで以下のような丸めの手法を用いることとした。

1. ノードの値に対応する変数についてのみ丸めを行う．
2. 各ノードに関して NDL_i を求める．
3. $NDL_i = 0$ となったノードについては，実数最適解が整数解であるので，実数最適解を用いる．
4. $0 < NDL_i < 0.5$ となったノードについては，すでに述べた手法により丸めを行う。
5. 4. の処理が行われなかったとき， NDL_i が最も小さな値となるノードに対して，すでに述べた手法により丸めを行う．
6. 5. の処理で， NDL_i が最も小さな値となるノードが複数存在した場合は，それぞれのノードについて，そのノードについて丸めを行ったILPの緩和問題であるLPについて単体法を用いて解きその目的関数値を求め，目的関数値が最小となった時に丸めを行ったノードを，反復過程の丸めとして用いて，丸めを行う．目的関数値が同じ値をとる丸めが複数存在したときはいずれかを選択する．

4 評価結果

本手法を実装し評価を行った．解コストは各問題について解コストと最適解コストとの比をとり，その平均を求めた．平均は1.16となった．

5 まとめ

PCSPの解法は大部分は分枝限定法を基にしたものであり，最適解が得られる代わりに探索時間が長くなる欠点を持つ．一方ILPを用いた手法は近似解ではあるが高速に解が得られる特徴を持つ．さらにPCSPの特徴に応じた手法として，単体法の反復適用の手法を提唱した．

参考文献

[1] 斎藤 逸郎, 石塚 満: 数理計画法を用いた不完全CSPの高速近似解法, 第58回情報処理学会 全大, No.2, pp.233–234, March, 1999.

[2] 蒸 東風, 石塚満: 線形計画法を利用した離散制約最適化問題の効率的近似解法, 人工知能学会誌, Vol.14, No.2, pp.334–342, 1999.