数理計画法を用いた不完全CSPの高速近似解法

1N－2

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．一方CSPは様々な高速解法が提唱されており，そのうちの一つに数理計画法 (ILP) を用いて解く手法がある [1]．そこでPCSPをCSPに置換え，ILPを用いることにより高速に解く手法について提唱を行う．

2　PCSPからILPへの帰着

2.1　基本的な考え方

ノード i の変数 $p_i(u)$ とノード j の変数 $p_j(v)$ との間のアーク $p_{i-j}(uv)$ について，すべての組合せでアークを満たすようにアーク制約を追加する．追加した制約の重みは制約の重み wp_{i-j} と等しくする．

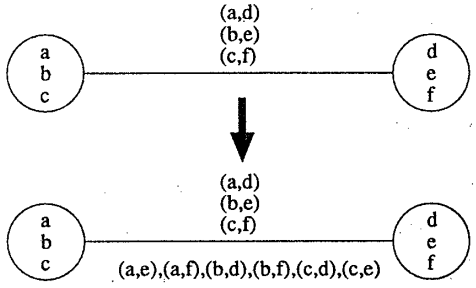

図 1: 基本的な考え方による帰着法

図 1 ではそれぞれのノードの値 a,b,c と d,e,f の間に元々のアーク制約 $(a,d),(b,e),(c,f)$ 以外に $(a,e),(a,f),(b,d),(b,f),(c,d),(c,e)$ なるアーク制約を付け加える．

この様にする事でどのような場合でも常にアークを成立させる事ができ，PCSP をCSPに帰着する事ができる．

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

2.2　実際の帰着方法

すべての組合せを列挙すると制約が多くなり問題規模が大きくなる．そこで p_{i-j} に $False$ なるアーク制約を設ける．$False$ はノード i とノード j が如何なる値をとっても成立するアーク制約である．一般のCSPの解法ではこの様な制約は許されないが，CSPをILPに帰着する場合には用いても解く事ができる．

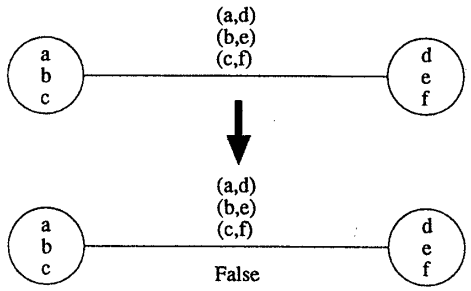

図 2: 実際の帰着法

変換後の 0–1 変数・制約式・目的関数は以下のようになる．

2.2.1　0–1 変数

x_{iu}　$p_i(u)$ に対応する変数．$x_{iu}=1$ の時，ノード i が値 u をとる．

y_{iu-jv}　$p_{i-j}(u,v)$ に対応する変数．$y_{iu-jv}=1$ の時，アーク p_{i-j} は (u,v) により満たされる．

z_{i-j}　p_{i-j} の制約の成立·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．図 2 $False$ に相当する．

2.2.2　制約式

制約が成立する場合は，アーク制約は一つのアークにおいて一つしか成立しない．制約が成立しない場合

は，すべてのアーク制約は不成立となる．よって一つのアークにおいて，アーク制約の変数の和と制約の変数との和は 1 となる．

$$\sum_u \sum_v y_{iu-jv} + z_{i-j} = 1$$

一つのアークにおいて，ノードがある値をとった時，そのノードを含むいずれかのアーク制約が成立もしくは制約が不成立となる．ノードがある値をとらなかった時，そのノードを含むすべてのアーク制約は不成立もしくは制約が不成立となる．よって一つのアークにおいて，あるノードの値に関して，ノードの値の変数はそれに関連したアーク制約の変数の和と制約の変数との和より小さくなる．

$$x_{iu} \leq \sum_v y_{iu-jv} + z_{i-j}$$

あるノードに関してとり得る値は一つだけである．よってあるノードにおいてノードの値の変数の和は 1 となる．

$$\sum_u x_{iu} = 1$$

2.2.3 目的関数

ノードの値の変数とノードの値の重みとの積と，アーク制約の変数とアークの重みの積と，制約の変数と制約の重みの積との和が目的関数となる．

$$\begin{aligned} \min a \quad = \quad & \sum_i \sum_u w_{iu} x_{iu} \\ & + \sum_i \sum_u \sum_j \sum_v w_{iu-jv} y_{iu-jv} \\ & + \sum_i \sum_j w p_{i-j} z_{i-j} \end{aligned}$$

2.3 複数の制約パス

以上の方法により PCSP を CSP に変換できるが，*False* を設けた事により複数の制約パスが存在し得る問題がある．例をあげると図 3 の様な場合，アーク制約が成立する場合として a と d の間に (a,d) が成立する場合と *False* が成立する場合の二つの場合がある事である．

(a,d) と *False* は一つのアークについてのみ影響するため，成立するアーク制約が入れ替わったとしても ILP の解は変わらず，目的関数のみが影響を受ける．そのため，(a,d) の重み w_{ia-jd} が *False* の重み wp_{i-j} より小さければ，常に (a,d) が選ばれる．

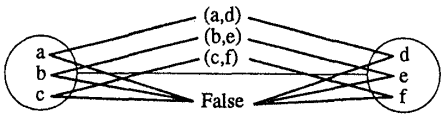

図 3: 変換後の制約パス

よって複数の制約パスが存在する問題は，制約の重みがアーク制約の重みより小さい場合にのみ発生する．この場合は最大 CSP の解法を用いることで解決される．

2.4 最大 CSP

最大 CSP はもっとも多くのアークを満たす解を求める問題である．そのため制約の重みとアーク制約の重みの和を目的関数とした手法では正しい解が得られない．そこでまず最初に，ノードの値の重み・アーク制約の重みをすべて 0 とし，制約の重みをすべて 1 とした PCSP を解く．この解は制約の重みの和が最小，すなわち満たされないアークが最も少なくなる解である．この解のうちアーク制約を満たすアークについては最大 CSP のアーク制約を，満たさないアークについては *False* をアーク制約とし，PCSP を解く．この様にする事で，最大 CSP を PCSP 帰着させる事ができ，ILP を用いて解く事が可能となる．

3 まとめ

以上により PCSP を ILP に置き換える事ができる．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 は様々な高速解法が存在するため，ILP に変換する事により PCSP も高速に解けると考えられる．

PCSP の応用としては最適解が必要とされるものばかりではなく，少々解の精度を犠牲にしても一定時間内で解く必要があるものも存在する．この様な応用にとって本稿で述べた手法は有効であると考えられる．

参考文献

[1] 蔡 東風, 石塚満: 局所探索に基づく重み付き制約充足問題の効率的解法, 第 54 回情報処理学会 全大, No.2, pp.327–328, 1997.