Dominating Set問題を解く自己安定アルゴリズム

2M−1

亀井 清華　角川 裕次
広島大学大学院 工学研究科

1　はじめに

分散システムとは、通信リンクを介して通信するプロセスの集合である。初期状態に何も仮定しない分散アルゴリズムを自己安定アルゴリズムという。自己安定アルゴリズムの概念は1974年にDijkstraによって分散システムに導入された[1]。自己安定アルゴリズムではシステムは起こり得る任意の初期状態から有限時間内に論理的な安定状態に到達する。自己安定アルゴリズムは任意の一時故障より自動回復する性質を持つという特徴を持つ。

グラフ $G=(V,E)$ (Vは節点の集合、Eは枝の集合) において、SをVの部分集合とする。$V-S$内の任意のメンバーxが、必ずあるSのメンバーに隣接しているとき、Sはdominating setであるという。一般的にはVの真部分集合であるdominating setを求める問題をdominating set問題という。

本稿ではこのグラフ理論の問題を、節点をグラフに、枝をリンクにとネットワークに対応させて考え、極小、あるいは最小のdominating setを求める問題について考える。この問題は局所的なリーダーを選挙する問題とも見倣せる。独立点集合問題の観点より局所リーダー選挙を調べた研究については文献[2]がある。他のグラフ理論の問題を解く自己安定アルゴリズムとして、[3]を参照されたい。

2　モデル

分散システムはn個のプロセス集合 $V=\{p_i|i=1,2,3,\ldots,n\}$ と双方向の通信リンクの集合 $E=\{\ldots,(p_i,p_j),\ldots\}$ から構成される。$(p_i,p_j)\in E$のとき、p_i,p_jは互いに隣接プロセスと呼ぶ。また、N_iをp_iの隣接プロセス集合、即ち$N_i=\{p_j|(p_i,p_j)\in E\}$とする。

本稿では、 状態通信モデル (全ての隣接プロセスの状態を遅延なく知ることができるような通信モデル) を採用する。プロセスp_iの取り得る状態の集合をQ_iとする。n個の全プロセスの状態の組 $\gamma\in Q_1\times Q_2\times\cdots\times Q_n$ をシステムSの状況という。全ての状況の集合をΓと表す。

スケジュールとはプロセスの部分集合の無限系列をいう。ネットワーク状況の無限系列は任意の初期状況 ($\in\Gamma$) とこのスケジュールに対するアルゴリズムの実行といえる。本稿では以下の2つのスケジューラを仮定する。

- Dデーモン:複数のプロセスが同時に動作することを許すモデル
- Cデーモン:1つのプロセスしか同時には動作しないという制限を加えたモデル

また、ここでは全てのプロセッサを識別子なしの同一の状態機械にしたネットワークと、識別子付きのネットワークについて考える。

任意の状況$\gamma\in\Gamma$に対して1ステップの実行により状況がγ'となるとき、$\gamma\to\gamma'$と表す。

定義 2.1 (自己安定)　システムSが次の2つの条件を満たすとき、Sは正当な状況の集合$\Lambda\in\Gamma$に関して自己安定であるという。。

- 任意の$\lambda\in\Lambda$に対して、$\lambda\to\gamma$ならば$\gamma\in\Lambda$(閉包性)。
- 任意の$\gamma_0\in\Gamma$と任意の実行$\gamma_0\to\gamma_1\to\cdots$に対して、ある$k$が存在して$\gamma_k\in\Lambda$(収束性)。□

3　Cデーモンの下での場合

極大独立点集合とdominating setの関係を示す。極大独立点集合とはそのメンバー同士は隣接しないような節点集合の部分集合である。

補題 3.1 無向グラフ $G=(V,E)$ における任意の独立点集合を$I\subseteq V$とするとき、$D=V-I$は*dominating set*である。□

Self-stabilizing Distributed Algorithms
for the dominating set problem.
Sayaka Kamei,Hirotsugu Kakugawa
Graduate school of Engineering, Hiroshima University

次に極小 dominating set を求めるアルゴリズム DSC を提案する。補題で述べた関係を利用し、非独立点を dominating set の要素の初期候補とする。そして、候補を減らしてゆき、極小 dominating set を求める。

アルゴリズム **DSC(概要)**

各プロセス p_i は以下の動作を繰り返す。

- 極大独立点集合問題を解くアルゴリズムを実行する。独立点となった場合は、dominating set の候補から外れ、以下の動作は行われない。
- p_i に隣接する dominating set の要素の数を数えて局所変数 P_i に代入する。
- もし、全ての隣接プロセスが dominating set の要素でない、つまり $\forall p_j \in N_i[P_j = 0]$ ならば、p_i は dominating set の要素になる。
- p_i が dominating set の要素で、他の dominating set の要素に隣接していて、どの隣接プロセスも p_i 以外の dominating set の要素に隣接していれば、p_i は dominating set の要素ではなくなる。
- 隣接プロセスのなかに独立点が存在し、そのプロセスが dominating set の要素に隣接していないとき p_i は dominating set の要素になる。□

定理 1 アルゴリズム *DSC* は極小 *dominating set* 問題を解く。□

定理 2 アルゴリズム *DSC* は n 個のプロセスがあるネットワークにおいて $O(n^2)$ ステップで安定する。□

4　D デーモンの下での場合

4.1　識別子を与えない場合

定理 3 *D* デーモンの下でプロセス *ID* を持たない場合、正則グラフの形状をしたネットワーク上で、極小 *dominating set* 問題を解くアルゴリズムは存在しない。□

4.2　識別子を与えた場合

全てのプロセスがネットワークトポロジーを認識することができれば、逐次アルゴリズムを適用できるという点に着目した。トポロジーを認識するためのアルゴリズム RTD を説明する。

アルゴリズム **RTD(概要)**

各プロセスは自身の識別子を根に、隣接プロセスの識別子を子に、隣接プロセスの隣接プロセスの識別子を孫に持つような木構造の変数を持つとする。つまり、このアルゴリズムで得られる木は節がプロセスの識別子、枝がリンクと置き換えることのできる木となる。アルゴリズム実行開始後各プロセスは自身の識別子を根にいれる。そして、隣接プロセスの持つ木の高さを $n-1$ に切りそろえたものを自身の子以下の部分木とする。この動作を繰り返すことによって、いずれはトポロジーの情報を全て含んだ木を全てのプロセスがもつことができる。□

このアルゴリズムを使ってトポロジーを認識すれば、逐次アルゴリズムを利用して最小の dominating set を求めることも可能である。

定理 4 トポロジー認識アルゴリズム *RTD* は n 個のプロセスがあるネットワークにおいて $O(n^3)$ で安定する。□

5　終わりに

dominating set 問題を解く自己安定アルゴリズムを D デーモン、C デーモンのそれぞれの下について挙げた。また、D デーモンでの識別子が与えられていない場合でのアルゴリズムは存在しないことも示した。しかし、D デーモンの下での識別子を与えた場合のアルゴリズム RTD は、単純にネットワークトポロジーを求める方法に基づいている。より小さな複雑度のアルゴリズムが存在するかはまだ未解決である。

参考文献

[1] Edsger W. Dijkstra; "Self-stabilizing Systems in Spite of Distributed Control", Commun. ACM, Vol. 17, No. 11, pp. 643–644 (1974)

[2] 川本 幸司, 角川 裕次, 阿江 忠; "拡張独立点集合問題を解く自己安定分散アルゴリズム", 情報処理学会第 61 回秋の全国大会 (2000)

[3] Sandeep K.Shukla, Daniel J.Rosenkrantz, S.S.Ravi; "Observation on Self-Stabilizing Graph Algorithms for Anonymous Networks", Proceedings of the second workshop on self-stabilizing systems (1995)