木におけるスライディングトークン問題の最短手順を求める多項式時間アルゴリズム

数理情報学専攻 48166234 杉森 健
指導教員 定兼 邦彦 教授

1 はじめに

近年，理論計算機科学の分野で遷移問題の研究が盛んに行われている．遷移問題とは，ある問題の初期解と目標解が与えられたとき，初期解から目標解まで解を段階的に遷移させられるか判定する問題である．独立集合問題に対する遷移問題として，スライディングトークン問題が提案された [2]．この問題は，無向グラフ上の独立集合 I_s および I_t が与えられたとき，I_s から I_t まで独立集合を段階的に遷移させられるか判定する問題である．本研究では，木におけるスライディングトークン問題に対して，遷移の最短手順を求める多項式時間アルゴリズムを提案する．

2 既存研究と本研究の成果

スライディングトークン問題は，Hearn と Demaine によって導入された [2]，次のような問題である．G を無向グラフとし，I_s および I_t を G 上の独立集合とする．I_s 上に置かれたトークンの集合を，I_t 上まで遷移させるのが目標である．ただし，可能な操作は「トークンをひとつ選んで隣接頂点へスライドする」というものに限る．さらに，遷移の間トークンの集合は独立集合を保たなければならない．この設定のもとで，トークンの集合を I_s 上から I_t 上まで遷移させられるか判定する問題が，スライディングトークン問題である．

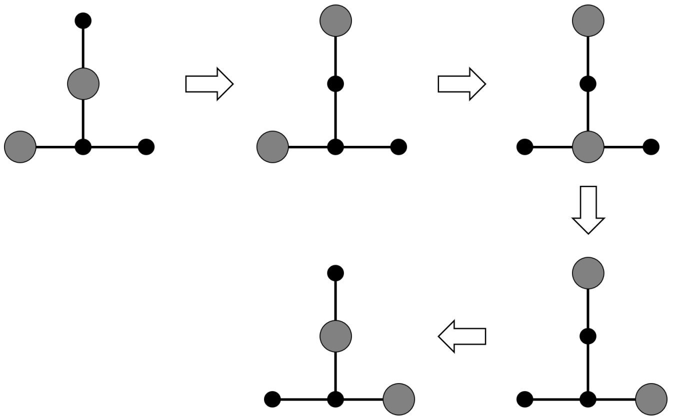

図 1. スライディングトークン問題の遷移の例

一般のグラフにおけるスライディングトークン問題は PSPACE 完全に属する [2]．さらに，平面グラフに限定してもこの問題は PSPACE 完全に属する [3]．一方で，いくつかのグラフクラスにおけるスライディングトークン問題には，多項式時間アルゴリズムが発見されている．例えば，木においては線形時間アルゴリズムが発見されている [1]．また，木よりも広いグラフクラスである cactus tree [5] および clique tree [4] においても多項式時間アルゴリズムが発見されている．

以上は遷移可能性の判定に対する結果であるが，遷移の最短手順を構成するという問題設定もまた自然である．この問題設定に対しては，木の部分クラスである caterpillar tree において，遷移の最短手順を構成する多項式時間アルゴリズムが発見されている [6]．しかし，木において遷移の最短手順を構成する多項式時間アルゴリズムが存在するかは分かっていなかった．[1] では遷移の手順を構成するアルゴリズムも提案されているが，最短手順ではない．

本研究では，木におけるスライディングトークン問題に対して，遷移の最短手数を求める多項式時間アルゴリズムを与えた．さらに，遷移の最短手順を構成する多項式時間アルゴリズムを与えた．

3 最短手数の計算

本研究で扱う問題は以下のものである．以降, $n = |T|$ とおく．

> **インスタンス**: 木 T と，T 上の独立集合 I_s と I_t．
> **タスク**: T 上でトークンの集合を I_s から I_t まで遷移させられるか判定せよ．遷移可能な場合，最短手順を構成せよ．

本研究ではまず，インスタンスの情報から最短手数を陽に計算するための式を与えることを目指した．しかし，元の問題設定において最短手数を陽に表すのは難しい．そこで代わりに，トークンの移動方法に関する制約 C も入力として与えられるという問題設定のもとで，C に従う最短手数を陽に表す式を与えた．

具体的には，制約 C は次のことを定めるものである．

- I_s 上の各トークンを，I^0 (移動しないトークン) または I_s^1 (1 回移動するトークン) または I_s^2 (2 回以上移動するトークン) に分類する．
- I_t 上の各トークンを，I^0 (移動しないトークン) または I_t^1 (1 回移動するトークン) または I_t^2 (2 回以上移動するトークン) に分類する．
- I_s^1 上のトークンから I_t^1 上のトークンへの一対一の対応を定める．
- I_s^2 上の各トークンについて，最初の移動先の隣接頂点を定める．
- I_t^2 上の各トークンについて，最後の移動元の隣接頂点を定める．

制約 C が与えられれば，C に従う最短手数は陽に計算できる．

結果 1. 木におけるスライディングトークン問題のインスタンスと制約 C が与えられたとき，C に従う最短手数は

$$\sum_{e\in E(T)} f_e(I_s, I_t; C)$$

という式で表される．ここで，$f_e(I_s, I_t; C)$ は辺 e におけるトークンの移動回数を表し，C が与えられれば容易に計算できることが示せる．

4 アルゴリズムの概要

以上の結果より，元の問題設定における最短手数は

$$\min_C \sum_{e\in E(T)} f_e(I_s, I_t; C)$$

と計算できるが，ありうる制約 C の個数は n の指数オーダーなので，全通り試すことはできない．そこで，木 T を根付き木とし，その上で動的計画法を用いて最適な制約 C^* を求めることを考える．このとき，$\sum_{e\in E(T)} f_e(I_s, I_t; C^*)$ が最短手数となる．

動的計画法のテーブルは $\mathrm{dp}(v, C_v)$ とする．ここで，v は T の頂点であり，C_v は v に接続する辺集合上の制約である．また，

$$\mathrm{dp}(v, C_v) := \min_{C'} \sum_{e\in E(T_v)} f_e(I_s, I_t; C')$$

と定義する．ここで，T_v は v 以下の部分木を指し，C' は v に接続する辺集合上で C_v に一致している必要がある．各 $\mathrm{dp}(\cdot,\cdot)$ は葉から順にボトムアップに計算していくことができる．

各 v について C_v は $\mathrm{O}((v\text{ の次数})^2)$ 通りなので，テーブル全体のサイズは $\mathrm{O}(n^2)$ である．また，各 $\mathrm{dp}(\cdot,\cdot)$ の計算は定数時間で行えることが示せるので，次の結果を得る．

結果 2. 木におけるスライディングトークンのインスタンスが与えられたとき，遷移可能性の判定および最短手数の計算は $\mathrm{O}(n^2)$ 時間で行える．

続いて，最短手順を構成する多項式時間アルゴリズムを与える．インスタンスが与えられたとき，最初に 1 回トークンを動かす方法は $\mathrm{O}(n)$ 通りである．よって，最初に 1 回トークンを動かす方法を全通り試し，残り最短手数がちょうど 1 だけ減るような操作を実際に行うことができる．この手続きを残り最短手数が 0 になるまで繰り返すことで，最短手順を構成できる．

このアルゴリズムの時間計算量は $\mathrm{O}(n \cdot n^2 \cdot m)$ である．ここで，m は最短手順の長さである．Demaine らによって $m = \mathrm{O}(n^2)$ が示されている [1] ので，次の結果を得る．

結果 3. 木におけるスライディングトークンのインスタンスが与えられたとき，最短手順の構成は $\mathrm{O}(n^5)$ 時間で行える．

参考文献

[1] E. D. Demaine, M. L. Demaine, E. Fox-Epstein, D. A. Hoang, T. Ito, H. Ono, Y. Otachi, R. Uehara and T. Yamada: Linear-time algorithm for sliding tokens on trees. *Theoretical Computer Science* 600, pp. 132–142, 2015.

[2] R. A. Hearn and E. D. Demaine: PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science* 343, pp. 72–96, 2005.

[3] R. A. Hearn and E. D. Demaine: *Games, Puzzles, and Computation.* A K Peters, 2009.

[4] D. A. Hoang, E. Fox-Epstein and R. Uehara: Sliding token on block graphs. in *Proc. WALCOM 2017*, pp. 460–471, 2017.

[5] D. A. Hoang and R. Uehara: Sliding tokens on a cactus. in *Proc. ISAAC 2016*, pp. 37:1–37:26, 2016.

[6] T. Yamada and R. Uehara: Shortest reconfiguration of sliding tokens on a caterpillar. in *Proc. WALCOM 2016*, pp. 236–248, 2016.