Parallel Token Sliding and Approximate Reconfiguration for Independent Sets (Draft)

Aditya Jayaprakash
ajayapra@edu.uwaterloo.ca

Abstract

Suppose we are given two independent sets I_b and I_r of a graph such that $|I_b| = |I_r|$, and suppose a token is placed on each vertex in the independent set. The token sliding (TS) model is concerned if there is a sequence of steps which transforms I_b into I_r so that each intermediate step, where a token is slid along an edge, is also an independent set. We introduce a model where multiple tokens can be moved along their edges in one step. We investigate how much of an advantage this gives us over the token sliding model. We prove results for trees and claw-free graphs. We also introduce the notion of approximate reconfiguration, where I_b is partially transformed into I_r. We prove lowerbounds for the values of this approximation.

1 Introduction

The reconfiguration version of an algorithmic problem asks whether there is a sequence of reconfiguration steps between two feasible solutions to the problem such that all intermediate solutions are also feasible and each solution in the sequence is obtained from the previous one according to a reconfiguration rule. Reconfiguration has been applied to a number of problems: vertex coloring [INZ15], matching [Müh15], spanning tree [Ito+11], satisfiability [Ito+11] and independent sets [KMM12]. A recent survey [Nis18] gives a good introduction to this area of research.

We will turn our focus to reconfiguration of independent sets. An independent set in a graph is a set of pairwise non-adjacent vertices. Three different reconfiguration rules have been studied in the literature: token sliding (TS) was introduced by Hearn and Demaine [HD05], token addition/removal(TAR) was introduced by Ito et al. [Ito+11], and token jumping (TJ) by Kaminski et al [KMM12]. We will focus on a new rule of reconfiguration called parallel token sliding.

1.1 Notation

We will use $G = (V, E)$ to denote a graph G where $V(G)$ is the set of vertices and $E(G)$ is the set of edges. Let IND-SET(G) be the set of all independent sets of G. We use $I_b \longleftrightarrow I_r$ to denote the existence of a reconfiguration sequence between I_b and I_r for some specific model. We will formally define what we mean by reconfiguration sequence in the next section. For a vertex $v \in G$, let $N(G, v) = \{w \in V(G) : \{v, w\} \in E(G)\}$. Let $N[G, v] = N(G, v) \cup \{v\}$. Similarly, we will use $I_b \not\longleftrightarrow I_r$ to denote there does not exist a reconfiguration sequence between the two independent sets.

1.2 Token Sliding

The token sliding problem was introduced by Hearn and Demaine [HD05] in the context of reconfiguration problems for independent sets. Given a graph $G = (V, E)$ and two independent sets I_b and I_r such that $|I_b| = |I_r|$ where a token is placed on vertices in the independent set. The token sliding problem for independent set configurations is to determine if there exists a sequence $\langle I_1, I_2, \ldots, I_l\rangle$ of independent sets of G such that

1. $I_1 = I_b, I_l = I_r$, and $|I_i| = |I_b| = |I_r|$ for all i, $1 \leq i \leq l$; and
2. For each i, $2 \leq i \leq l$, there is an edge $\{u, v\}$ in G such that $I_{i-1} \setminus I_i = \{u\}$ and $I_i \setminus I_{i-1} = \{v\}$, that is I_i can be obtained from I_{i-1} by sliding exactly one token on a vertex $u \in I_{i-1}$ to an adjacent vertex v along $\{u, v\} \in E$.

1.2.1 Parallel Token Sliding

We now introduce the parallel token sliding model. We refer to this by k-PTS for $k \in \mathbb{N}$. Given a graph $G = (V, E)$ and two independent sets, I_b and I_r such that $|I_b| = |I_r|$ where a token is placed on vertices in the independent set. The k-PTS problem for independent set configurations is to determine if there exists a sequence $\langle I_1, I_2, \ldots, I_l \rangle$ of independent sets of G such that

1. $I_1 = I_b, I_l = I_r$, and $|I_i| = |I_b| = |I_r|$ for all i, $1 \leq i \leq l$; and
2. For each i, $2 \leq i \leq l$, there exists $F \subseteq E$, $|F| \leq k$ such that $I_i \setminus I_{i-1} = \{u_1, \ldots, u_{|F|}\}$ and $I_{i-1} \setminus I_i = \{v_1, \ldots, v_{|F|}\}$. I_i can be obtained from I_{i-1} by sliding at most k tokens in parallel in one single step from vertex u_j to an adjacent vertex v_j along edge $u_j v_j$ for $1 \leq j \leq |F|$.

If two independent sets I_b and I_r can be reconfigured under the k-PTS model as defined above, we denote it by $I_b \longleftrightarrow_k I_r$. We are interested in answering the following questions:

Question 1 *Let $\mathbb{G}$ be the set of all graphs where $\forall G \in \mathbb{G}, V(G)$ and $E(G)$ are finite. For $k, j \in \mathbb{N}$ where $k > j$, let*

$$S = \{(I_b, I_r) : I_b \longleftrightarrow_k I_b, I_b, I_r \in \text{IND-SET}(G), G \in \mathbb{G}\}$$

$$T = \{(I_b, I_r) : I_b \longleftrightarrow_j I_b, I_b, I_r \in \text{IND-SET}(G), G \in \mathbb{G}\}$$

S is the set of all pairs of independent sets that can be reconfigured using k-PTS while T is the set of all pairs of independent sets that can be reconfigured using j-PTS. Is $|S| > |T|$? Loosely, we want to know if we can reconfigure more independent sets if we allow more tokens to be moved at the same time.

Question 2 *Given a graph G, how to compute the minimum k such that $I_b \longleftrightarrow_k I_r$ assuming $I_b \longleftrightarrow_k I_r$ for some k? What is the complexity of computing this k?*

Question 3 *How much of an advantage in terms of the length of reconfiguration does k-PTS offer over the existing token sliding model (1-PTS)?*

2 Question 1

We will show that if we increase the value of k, we can overall reconfigure more independent sets.

Claim 4 *Let $\mathbb{G}$ be the set of all graphs where $\forall G \in \mathbb{G}, V(G)$ and $E(G)$ are finite. For $k, j \in \mathbb{N}$ where $k > j$, let*

$$S = \{(I_b, I_r) : I_b \longleftrightarrow_k I_b, I_b, I_r \in \text{IND-SET}(G), G \in \mathbb{G}\}$$

$$T = \{(I_b, I_r) : I_b \longleftrightarrow_j I_b, I_b, I_r \in \text{IND-SET}(G), G \in \mathbb{G}\}$$

S is the set of all pairs of independent sets that can be reconfigured using k-PTS while T is the set of all pairs of independent sets that can be reconfigured using j-PTS, then $|S| > |T|$.

Proof: A trivial example is an even cycle C where $|V(C)| = 2k$. There are two possible maximum independent sets where there are tokens on each alternate vertex or every alternate vertex is part of an independent set. We will call them I_b and I_r. Since we can move all k token at the same time, $I_b \longleftrightarrow_k I_r$. However, if we choose to just move $j < k$ tokens, there will be two tokens adjacent to each other, violating the property of an independent set, proving $I_b \not\longleftrightarrow_j I_r$. This example showcases a separation between k-PTS and j-PTS. ∎

The above example also shows there is no single k such that $I_b \not\longleftrightarrow_k I_r$ for any graph G and $I_b, I_r \in \text{IND-SET}(G)$.

3 Question 2

Given a graph $G = (V, E)$ and let $I_b, I_r \in \text{IND-SET}(G)$. We want to find the minimum value of k such that $I_b \longleftrightarrow_k I_r$. If such a k exists, we will call it the *PTS-threshold* .

We will define PTS-Threshold(G, I_b, I_r) to be the function that computes *PTS-threshold* . If such a value doesn't exist, the *PTS-threshold* is undefined. We will use PTS-Threshold(G, I_b, I_r) and *PTS-threshold* interchangeably

In the original paper introducing token sliding by Hearn and Demaine [HD05], it was shown that for any graph G and $I_b, I_r \in \text{IND-SET}(G)$, computing if $I_b \longleftrightarrow I_r$ is PSPACE-COMPLETE. Suppose there is some oracle that could compute PTS-Threshold(G, I_b, I_r). Let $k =$ PTS-Threshold(G, I_b, I_r). If $k > 1$, then the answer to whether or not $I_b \longleftrightarrow I_r$ is false, since token sliding cannot reconfigure these independent sets. If $k = 1$, then token sliding can reconfigure them. Hence, the problem of computing PTS-Threshold(G, I_b, I_r)

is at least as hard as finding if token sliding could reconfigure them.

Let $\mathcal{F}$ be a set of graphs. *PTS-threshold* ($\mathcal{F}$) is the smallest integer such that for every $G \in \mathcal{F}$ and every $I_b, I_r \in$ IND-SET(G), either PTS-THRESHOLD$(G, I_b, I_r) \leq$ *PTS-threshold* ($\mathcal{F}$) or PTS-THRESHOLD(G, I_b, I_r) is undefined. We are now interested in looking at different graph classes and bounding the value of *PTS-threshold* ($\mathcal{F}$).

Claim 5 *Let $\mathcal{F}$ be the set of all simple cycles, then PTS-threshold ($\mathcal{F}$) $= \infty$.*

Proof: Take any cycle of $C = (V, E)$ of even length i.e., $|V| = 2k$ and there are exactly two maximum independent sets, I_b and I_r. As we have seen before, $I_b \longleftrightarrow_k I_r$. For the sake of contradiction, let us assume there exists some $l < \infty$ such that under l-PTS, we could reconfigure any two independent sets for some cycle $C' \in \mathcal{F}$. Let $|V(C')| = 2(l+1)$ and let I_b, I_r be its maximum independent sets where $|I_b| = |I_r| = l + 1$. $I_b \not\longleftrightarrow_l I_r$ since moving at most l tokens would violate the property of independent sets, but $I_b \longleftrightarrow_{l+1} I_r$ contradicting the fact that such a finite l exists. Hence, PTS-THRESHOLD$(\mathcal{F}) = \infty$. ∎

Theorem 6 *Let $\mathcal{T}$ be the set of all trees. PTS-threshold ($\mathcal{T}$) = 1.*

Proof: We wish to prove for any $G \in \mathcal{T}$ and $I_b, I_r \in$ IND-SET(G), either PTS-THRESHOLD(G, I_b, I_r) is undefined or PTS-THRESHOLD$(G, I_b, I_r) = 1$. If $I_b \not\longleftrightarrow_k I_r$ for any k, then PTS-THRESHOLD(G, I_b, I_r) is undefined. We will prove the other case.

For the sake of contradiction, assume there is a tree $G \in \mathcal{T}$ and $I_b, I_r \in$ IND-SET(G) such that PTS-THRESHOLD$(G, I_b, I_r) = k$ where $k > 1$. This implies there exists two independent sets $I_j, I_{j+1} \in$ IND-SET(G) in the reconfiguration sequence such that

$$I_j \setminus I_{j+1} = \{u_1, u_2, \ldots, u_k\} = A$$

$$I_{j+1} \setminus I_j = \{v_1, v_2, \ldots, v_k\} = B$$

$I_j \longleftrightarrow_k I_{j+1}$ involves k tokens sliding along edges $u_1v_1, u_2v_2, \ldots, u_kv_k$ in one single step. There needs to be at least one such step where this has to happen, otherwise it will contradict the definition of PTS-THRESHOLD(G, I_b, I_r).

The token u_1 cannot be moved to v_1 while keeping all other tokens constant, since it would violate the definition of PTS-Threshold(G, I_b, I_r). This implies v_1 is adjacent to some vertex in $A \setminus \{u_1\}$. Without loss of generality, let u_2 be the vertex v_1 is adjacent to. This implies v_1 and u_2 are adjacent. Similarly, since u_2 cannot be moved to v_2, there must be an edge adjacent to v_2 in $A \setminus \{u_1, u_2\}$. Since $I_j \longleftrightarrow I_{j+1}$ under k-PTS, we know k tokens must be moved simultaneously and we can show v_i is connected to u_{i-1} for all $2 \le i \le k-1$. As illustrated in the graph drawn below, since $k =$ PTS-Threshold(G, I_b, I_r), v_k needs to be adjacent to a vertex with a token.

If v_k was connected to some vertex $v \notin A$ with a token and if v could move to its adjacent vertex, v_k closer to v and $I_j \longleftrightarrow I_{j+1}$. This would contradict our assumption $I_j \longleftrightarrow_k I_{j+1}$. If the previous case is not true, then v is connected to some vertex w with a token that cannot be moved, contradicting the fact that $I_b \longleftrightarrow_k I_r$. Hence, the only way this is possible is if there is an edge between v_k and some vertex in A. The vertex in A would have to be u_1, else $I_b \longleftrightarrow_{k-1} I_{j+1}$. But since $v_k u_1$ is an edge, we have a cycle, contradicting the fact that G is a tree. Hence, such a k cannot exist, proving PTS-Threshold$(G, I_b, I_r) = 1$.

■

4 Computing PTS-Threshold(G, I_b, I_r)

Given a graph G and $I_b, I_r \in$ IND-SET(G), we want to determine if PTS-Threshold(G, I_b, I_r) is undefined or if there exists some k such that $I_b \longleftrightarrow_k I_r$. A natural way to determine if PTS-Threshold(G, I_b, I_r) is undefined is to check if $I_b \not\longleftrightarrow_i I_r$ for $1 \le i \le |I_b|$.

Question 7 *Given a graph G and independent set configurations I_b and I_r, is there a general algorithm to check if $I_b \longleftrightarrow_k I_r$? If so, what is its complexity?*

Question 8 *What if we restrict it to some graph class? Can we compute it efficiently?*

Claim 9 *If $G = (V, E)$ is a simple cycle with independent set configurations I_b and I_r, then the following are true,*

1. *If G is an odd cycle, then* PTS-THRESHOLD$(G, I_b, I_r) = 1$.
2. *If G is an even cycle where $|I_b| < \frac{|V|}{2}$, then* PTS-THRESHOLD$(G, I_b, I_r) = 1$.
3. *If G is an even cycle where $|I_b| = \frac{|V|}{2}$, then* PTS-THRESHOLD$(G, I_b, I_r) = \frac{|V|}{2}$.

Proof: For (1) and (2), there are two adjacent vertices without tokens, so we can always move our token in our cycle without violating the independent set property.

For (3), as we saw earlier, $I_b \longleftrightarrow I_r$ only under $\frac{|V|}{2} - \text{PTS}$ ■

Claim 10 *If G is a tree, then we can compute* PTS-THRESHOLD(G, I_b, I_r) *in linear time.*

Proof: In [Dem+15], Demaine et al. proposed a linear time algorithm to compute if $I_b \longleftrightarrow I_r$ when G is a tree and $I_b, I_r \in \text{IND-SET}(G)$. We will use our result about trees, PTS-THRESHOLD(G, I_b, I_r) is undefined or PTS-THRESHOLD$(G, I_b, I_r) = 1$ when G is a tree. If we treat the algorithm to compute $I_b \longleftrightarrow I_r$ as a blackbox, then
PTS-THRESHOLD$(G, I_b, I_r) = 1$ if $I_b \longleftrightarrow I_r$ and is undefined otherwise. ■

We will now prove there exists a non-trivial class of graphs $\mathcal{F}$ such that for every $G \in \mathcal{F}$ and for every $I_b, I_r \in \text{IND-SET}(G)$, there exists some k such that $I_b \longleftrightarrow_k I_r$, but $I_b \not\longleftrightarrow I_r$. A *claw* is a tree with four vertices and three leaves. A graph if *claw-free* if it does not contain a claw as an induced subgraph.

Theorem 11 *Let $\mathcal{F}$ be the set of all claw-free graphs. For every $G \in \mathcal{F}$ and $I_b, I_r \in \text{IND-SET}(G)$ such that $|I_b| = |I_r|$, there is some k such that $I_b \longleftrightarrow_k I_r$. Moreover, we can also compute such a k in polynomial time.*

Proof: Bonsma et al. in [BKW14] introduced a polynomial time algorithm to check if $I_b \longleftrightarrow I_r$. We will first introduce definitions and lemmas from [BKW14]. We will use $I_b \Delta I_r = (I_b \setminus I_r) \cup (I_r \setminus I_b)$ to denote the symmetric difference between two sets.

Lemma 12 *Let I_b and I_r be independent sets in a connected claw-free graph G, with $|I_b| = |I_r|$. If $G[I_b \Delta I_r]$ contains no cycles, then $I_b \longleftrightarrow I_r$.*

Lemma 13 *Let I_b be a non-maximum independent set in a claw-free connected graph G. Then for any independent set I_r with $|I_b| = |I_r|$, $I_b \longleftrightarrow I_r$ holds.*

We only need to consider the case when $G[I_b \Delta I_r]$ contains (even) cycles, and both I_b and I_r are maximum independent sets. The authors show $I_b \longleftrightarrow I_r$ by showing that that this is equivalent with stating that every cycle in $G[I_b \Delta I_r]$ can be reconfigured. That is, we want to show that for every cycle C, let I_b^C and I_r^C be its maximum independent sets. We want to show there exists a sequence of moves such that $I_b^C \longleftrightarrow I_r^C$.

Let $G[I_b \Delta I_r] = \{C_1, \ldots, C_d\}$ be the set of even cycles. Let I_b^i, I_r^i for $1 \leq i \leq d$ be the corresponding maximum independent sets for each of the cycles.

$$k = \frac{\max_i |V(C_i|}{2}$$

Since k is at least half the vertices in each of the cycles, $I_b^i \longleftrightarrow_k I_r^i$. We can individually reconfigure each of the cycles using k-PTS, there by showing that $I_b \longleftrightarrow_k I_r$.

The original algorithm runs in polynomial time. There are only a polynomial amount of cycles possible when we compute the symmetric difference, and computing the number of vertices in each cycle also takes polynomial hence. Hence, computing such a k takes polynomial time as well since polynomials are closed under composition. ■

Question 14 *Let l be the number such that using the algorithm we defined above, for two independent set $I_b, I_r \in$ IND-SET(G), $I_b \longleftrightarrow_l I_r$ and let $k =$ PTS-THRESHOLD(G, I_b, I_r). Is $k = l$? We want to know if we can find the optimal value of k using the algorithm we defined. If not, we want to know two things:*

1. *A counter-example showing $I_b \longleftrightarrow_k I_r$ where $k < l$.*
2. *Is there a polynomial time algorithm for finding* PTS-THRESHOLD(G, I_b, I_r) *for claw-free graphs?*

5 Approximate Reconfiguration

We have seen examples of graphs G and $I_b, I_r \in \text{IND-SET}(G)$ such that PTS-THRESHOLD(G, I_b, I_r) is undefined, i.e., $I_b \not\longleftrightarrow_k I_r$ for any $k \in \mathbb{N}$. An interesting question to ask if we can approximate I_b to be as similar as I_r? For G, I_b, I_r, we define a reconfiguration approximator to be $I_{\text{approx}} \in \text{IND-SET}(G)$ such that under k-PTS,

$$I_{\text{approx}} = \underset{I \in \text{IND-SET}(G): I_b \longleftrightarrow_k I}{\text{argmax}} \frac{|V(I) \cap V(I_r)|}{|V(I_r)|}$$

$I_{\text{approx}} \in \text{IND-SET}(G)$ is an independent set such that $I_b \longleftrightarrow_k I_{\text{approx}}$ and $|V(I_{\text{approx}}) \cap V(I_r)|$ is maximum. If $I_b \longleftrightarrow I_r$, then $I_{\text{approx}} = I_r$. However, if $I_b \not\longleftrightarrow I_r$, then $I_b \longleftrightarrow I_{\text{approx}}$, but $I_{\text{approx}} \not\longleftrightarrow I_r$. We can also define ϵ where $0 \leq \epsilon \leq 1$ to denote how well I_{approx} approximates I_b.

$$\epsilon = \frac{|V(I_{\text{approx}}) \cap V(I_r)|}{|V(I_r)|}$$

If $\epsilon = 1$, then $I_b \longleftrightarrow I_r$. However, if $I_b \not\longleftrightarrow I_r$, then ϵ denotes how close of an approximate reconfiguration we can achieve.

Definition 15 *I_{approx} is an ϵ-approximator for G, I_b, I_r, k if the following holds true under k-*PTS*,*

1. $$I_b \longleftrightarrow_k I_{approx}$$

2. $$\frac{|V(I_{approx}) \cap V(I_r)|}{|V(I_r)|} \geq \epsilon$$

Claim 16 *Given a graph G and $I_b, I_r \in \text{IND-SET}(G)$, let ϵ_1 be the ϵ-approximator under 1-*PTS *and let ϵ_i be the ϵ-approximator under i-*PTS *for all $1 \leq i \leq |I_b|$. Then the following holds,*

1. *If $I_b \not\longleftrightarrow_{|I_b|} I_r$, then*

$$\epsilon_1 \leq \epsilon_2 \leq \ldots \leq \epsilon_{|I_b|-1} \leq \epsilon_{|I_b|} < 1$$

2. *If* PTS-THRESHOLD$(G, I_b, I_r) = k$, *then*

$$\epsilon_1 \leq \epsilon_2 \leq \ldots \leq \epsilon_{k-1} < \epsilon_k = \epsilon_{k+1} = \ldots = \epsilon_{|I_b|-1} = \epsilon_{|I_b|} = 1$$

Proof: $(i+1)$-PTS can reconfigure I_b to the same independent set that i-PTS can do to I_b, hence $\epsilon_{i+1} \geq \epsilon_i$. We also know $I_b \not\longleftrightarrow_{|I_b|} I_r$, hence $\epsilon_{|I_b|} < 1$. Combining the two, we get $\epsilon_1 \leq \epsilon_2 \leq \ldots \leq \epsilon_{|I_b|-1} \leq \epsilon_{|I_b|} < 1$, proving (1).

Since PTS-THRESHOLD$(G, I_b, I_r) = k$, we know for any $j < k$, $I_b \not\longleftrightarrow_l I_r$. Using part (1), we have $\epsilon_1 \leq \epsilon_2 \leq \ldots \epsilon_{k-1}$. We also know $\epsilon_{k-1} < \epsilon_k$ since $I_b \not\longleftrightarrow_{k-1} I_r$, but $I_b \longleftrightarrow_k I_r$. This is using the definition of PTS-THRESHOLD(G, I_b, I_r).

Hence, $\epsilon_k = 1$. Since $\epsilon_{k+i} \leq 1$ and $\epsilon_{k+i} \geq \epsilon_k$, we have $\epsilon_k \leq \epsilon_{k+1} \leq \ldots \leq \epsilon_{|I_b|}$. Combining the two, we get $\epsilon_1 \leq \epsilon_2 \leq \ldots \leq \epsilon_{k-1} < \epsilon_k = \epsilon_{k+1} = \ldots = \epsilon_{|I_b|-1} = \epsilon_{|I_b|} = 1$. This proves (2). ■

In some sense, the concept of ϵ-approximator generalizes everything we have done so far. Asking if there is a 1-approximator for G, I_B, I_r, k is equivalent to asking if $I_b \longleftrightarrow_k I_r$. Also, given some G, I_b, I_r, k, computing the value of ϵ could tell us if $I_b \longleftrightarrow_k I_r$. Computing the value of ϵ gives us a lot of information about reconfiguration using token sliding. It also tells us how close we can get. This leads to a few questions,

Question 17 *Given a graph G and $I_b, I_r \in$ IND-SET(G) and some k, how do we compute I_{approx} under k-PTS?*

Question 18 *Suppose we relax certain rules of reconfiguration such that instead of having I_b reconfigure to I_r, what if we require them to be similar in the sense that they both have tokens on at least some number of vertices? We will define the problem as follows, given a graph G, $I_b, I_r \in$ IND-SET(G) and ϵ where $0 \leq \epsilon \leq 1$. We are also given that $I_b \longleftrightarrow I_r$ under k-PTS for some k. We want to find an I_{approx} under k-PTS such that*

$$I_b \longleftrightarrow I_{approx}$$

$$\frac{|I_{approx} \cap I_r|}{|I_r|} \geq \epsilon$$

Can we do this efficiently for certain values of ϵ?

Question 19 *Given a graph G, $I_b, I_r \in$ IND-SET(G) and ϵ where $0 \leq \epsilon \leq 1$. We want to answer the following,*

1. *Does there exist an I_{approx} such that $\frac{|V(I_{approx}) \cap V(I_r)|}{|V(I_r)|} \geq \epsilon$?*
2. *What is the minimum value of k such that an I_{approx} exists under k-PTS?*

5.0.1 Approximation for Trees

We will now show how to find a lowerbound for ϵ_1 for trees under some conditions. First, we will introduce some definitions and algorithm from [Dem+15].

Definition 20 *We say that a token on a vertex $v \in I$ is (T, I)-rigid if $v \in I'$ holds for any independent set I of T such that $I \longleftrightarrow I'$. For an independent set I of T, we denote by $R(I)$ the set of all vertices in I on which (T, I)-rigid*

The algorithm to check if $I_b \longleftrightarrow I_r$ when G is a tree is as follows,

1. Compute $R(I_b)$ and $R(I_r)$. Return "no" if $R(I_b) \neq R(I_r)$; otherwise go to Step 2.
2. Delete the vertices in $N[T, R(I_b)] = N[T, R(I_r)]$ from T and obtain a forest F consisting of q trees $T_1, T_2, \ldots, T_q$. Return "yes" if $|I_b \cap T_j| = |I_r \cap T_j|$ holds for every $j \in \{1, 2, \ldots, q\}$; otherwise return "no".

Suppose $R(I_b) = R(I_r)$, then we delete the vertices in $N[T, R(I_b)] = N[T, R(I_r)]$ from T and obtain a forest F consisting of q trees $T_1, T_2, \ldots, T_q$. An important observation here is that every vertex that still remains can move since all rigid tokens have been removed. Suppose $I_b \not\longleftrightarrow I_r$, then there is at least one tree T_i such that $|I_b \cap T_i| \neq |I_r \cap T_i|$.

We can still reconfigure $|I_b \cap T_i|$ tokens in T_i so that they have the same position as those in I_r since tokens in T_i are movable. From each such T_i, there are $\Big||I_b \cap T_i| - |I_r \cap T_i|\Big|$ tokens that cannot be reconfigured. Let $\hat{\epsilon_1}$ be our lower bound. Hence,

$$\hat{\epsilon_1} = \frac{|I_b| - \sum_{i=1}^{q} \Big||I_b \cap T_i| - |I_r \cap T_i|\Big|}{|I_b|}$$

Naturally, if $I_b \longleftrightarrow I_r$, then $\sum_{i=1}^{q} \Big||I_b \cap T_i| - |I_r \cap T_i|\Big| = 0$ and $\hat{\epsilon_1} = 1$. This algorithm is a modification of the original algorithm, checking set intersection takes polynomial time. Hence, our proposed algorithm takes polynomial time as well.

Question 21 *Is $\hat{\epsilon_1} = \epsilon_1$? Can we modify the above algorithm to also handle the case when $R(I_b) \neq R(I_r)$ and compute ϵ_1?*

5.0.2 Approximation for Claw-free graphs

We will show how to also get a lowerbound for claw-free graphs. Suppose $I_b \not\longleftrightarrow I_r$, since G is claw-free and combining our previous result, we know there is some k such that $I_b \longleftrightarrow_k I_r$. We will show to find a lowerbound for $\epsilon_1, \ldots, \epsilon_k$.

We will use the same approach as we did for our previous theorem. Let $G[I_b \Delta I_r] = \{C_1, \ldots, C_d\}$ be the set of even cycles which can only be reconfigured if $k > 2$. Let I_b^i, I_r^i for $1 \leq i \leq d$ be the corresponding maximum independent sets for each of the cycles. Suppose we select some i, then $\hat{\epsilon_i}$ is the fraction of nodes that can be reconfigured under i-PTS. For each cycle C_j, if $V(C_j) \geq 2i$, then we can reconfigure it. Let $S_i = \{C_j : |V(C_j| \geq 2i\}$ for all $1 \leq i \leq k$.

$$\hat{\epsilon_i} = \frac{|I_b| - \sum_{C_i \in S_i} |V(C_i)|}{|I_b|}$$

Hence, we have shown a way to compute a lowerbound for ϵ_i. Our algorithm performs no more steps than the original algorithm, hence it is polynomial time as well.

Question 22 *Does the above approach give us tight bounds on ϵ_i that is, is $\hat{\epsilon_i} = \epsilon_i$?*

6 Conclusion

In this paper, we have introduced the definition of PTS. We have shown the equivalence of token sliding and parallel token sliding in the context of reconfiguring independent sets of trees. We have also shown that parallel token sliding can reconfigure any claw-free graph. A notion of approximate reconfiguration was introduced. We introduced efficient algorithms to compute lowerbounds on the value of such approximations for trees and for claw-free graphs.

7 Acknowledgement

I would like to thank Prof. Naomi Nishimura for giving me this opportunity, for her support and supervision.

References

[HD05] Robert A. Hearn and Erik D. Demaine. "PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation". In: *Theoretical Computer Science* 343.1 (2005). Game Theory Meets Theoretical Computer Science, pp. 72–96. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2005.05.008. URL: http://www.sciencedirect.com/science/article/pii/S0304397505003105.

[Ito+11] Takehiro Ito et al. "On the complexity of reconfiguration problems". In: *Theoretical Computer Science* 412.12 (2011), pp. 1054–1065. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2010.12.005. URL: http://www.sciencedirect.com/science/article/pii/S0304397510006961.

[KMM12] Marcin Kamfffdfffdnski, Paul Medvedev, and Martin Milanifffdfffd. "Complexity of independent set reconfigurability problems". In: *Theoretical Computer Science* 439 (2012), pp. 9–15. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2012.03.004. URL: http://www.sciencedirect.com/science/article/pii/S0304397512002241.

[BKW14] Paul Bonsma, Marcin Kamiński, and Marcin Wrochna. "Reconfiguring Independent Sets in Claw-Free Graphs". In: *Algorithm Theory – SWAT 2014*. Ed. by R. Ravi and Inge Li Gørtz. Cham: Springer International Publishing, 2014, pp. 86–97. ISBN: 978-3-319-08404-6.

[Dem+15] Erik D. Demaine et al. "Linear-time algorithm for sliding tokens on trees". In: *Theoretical Computer Science* 600 (2015), pp. 132–142. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2015.07.037. URL: http://www.sciencedirect.com/science/article/pii/S0304397515006829.

[INZ15] Takehiro Ito, Hiroyuki Nooka, and Xiao Zhou. "Reconfiguration of Vertex Covers in a Graph". In: *Combinatorial Algorithms*. Ed. by Kratochvíl Jan, Mirka Miller, and Dalibor Froncek. Cham: Springer International Publishing, 2015, pp. 164–175. ISBN: 978-3-319-19315-1.

[Müh15] Moritz Mühlenthaler. “Degree-Constrained Subgraph Reconfiguration is in P”. In: *Mathematical Foundations of Computer Science 2015*. Ed. by Giuseppe F. Italiano, Giovanni Pighizzini, and Donald T. Sannella. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 505–516. ISBN: 978-3-662-48054-0.

[Nis18] Naomi Nishimura. “Introduction to Reconfiguration”. In: *Algorithms* 11.4 (Apr. 2018), p. 52. ISSN: 1999-4893. DOI: 10.3390/a11040052. URL: http://dx.doi.org/10.3390/a11040052.