CS6130: Paper Presentation Report

Niranjan and Mano Prakash P

IIT Madras ic37229@imail.iitm.ac.in ma22c024@smail.iitm.ac.in

1 Introduction

Given two matchings M_s and M_t of a graph G, the authors gave an algorithm to find the shortest transformation sequence to reach M_t from M_s, provided such a transformation exists. The transformation sequence, in simple terms, is a sequence of steps, such that resultant of each step is still a matching of graph G. The *distance* of a transformation is the length of shortest transformation sequence. Computing *distance* is referred to as MATCHING DISTANCE problem.

Reconfiguration problems, in general appear across wide variety of use case. For Example, Given two configurations of Rubik's cube, say R_1 and R_2, the problem of computing the shortest transformation sequence from R_1 to get R_2 can be modelled in a similar fashion to our reconfiguration of matchings.

Previous Works: Nishimura, N obtained an algorithm for the reachability variant (Is there a reconfiguration sequence from M_s to M_t ?). Ito, T., et al gave a bound of $O(n^2)$ to compute the distance.

2 Intuition / Examples / Your Observations

From a given instance I of bipartite MATCHING DISTANCE where M_s and M_t are maximum, the authors construct an instance I' of DIRECTED STEINER TREE which has at most $\frac{d}{2}$ terminals, where $d = |M_s \oplus M_t|$. Given a Steiner Tree F for I' which has cost $c(F)$, they present a polynomial time algorithm which constructs a reconfiguration sequence from M_s to M_t with length at most $c(F)$. The algorithm makes use of the following proposition.

Proposition 1. *Let F be a directed Steiner Tree for the instance I' with cost $c(F)$. Using F, we can construct in polynomial time a directed Steiner tree F' with $c(F') \leq c(F)$ and which satisfies the following properties:*

1. *For each $P \in \mathcal{P}$, F' contains all the arcs corresponding to edges of P.*
2. *For each $C \in \mathcal{C}$, F' misses exactly one arc of C.*
3. *There is an arc in F' joining root r to the M_s-exposed vertex of P, for each $P \in \mathcal{P}$.*

Here $\mathcal{P}$ and $\mathcal{C}$ denote the set of paths and cycles respectively in the symmetric difference $M_s \oplus M_t$.

The algorithm runs a DFS traversal of the Steiner Tree F' starting from the root r by giving preference to the higher weight arcs. We can choose arbitrarily between two arcs of the same cost. The key idea is that each arc of cost 2 corresponds to two steps in the matching transformation, and each arc of cost 1 corresponds to one step. Thus, we will obtain a reconfiguration sequence with length $c(F')$.

When we traverse down an arc (u, v) of cost 2, u has to be a free vertex in our current matching. If we traverse down an arc of cost 1 starting from u first, u will not be free when we traverse down (u, v) and we would be unable to perform its corresponding step in the reconfiguration sequence. This is why we prioritize arcs of higher cost.

One key observation is that if M_s and M_t are maximum, then each cycle $C \in \mathcal{C}$ has to be reconfigured by sliding tokens along an M_s-alternating path which starts from an M_s-exposed vertex and ends at a vertex of C. This is because in the case of maximum matchings, TOKEN JUMPING is equivalent to TOKEN SLIDING. Thus, in order to reconfigure the cycle C, we must first slide tokens along such an M_s-alternating path. This brings us to the following lemma.

Lemma 1. *Suppose we are given an instance (G, M_s, M_t) of* MATCHING RECONFIGURATION *where M_s and M_t are maximum. A transformation from M_s to M_t exists if and only if for each cycle $C \in \mathcal{C}$, there is an M_s-alternating path attached to C in G which starts from an M_s-exposed vertex.*

Note that TOKEN JUMPING is equivalent to TOKEN SLIDING in any maximal matching, not just a maximum matching. Thus, to obtain a reconfiguration sequence from M_s to M_t which does not visit any non-maximal matching, we have to use the technique from the constructive proof of Lemma 1. This is the intuition behind the algorithm for Case 2. For the rest of this section, we assume that the given instance of MATCHING DISTANCE falls into Case 2, i.e, each step in any shortest transformation from M_s to M_t is a maximal matching.

Let (U, W) be the color classes of the bipartite graph G. For each cycle $C \in \mathcal{C}$, we have at most two choices: C can be reconfigured using a vertex in U or in W. As there can be at most $\frac{d}{4}$ cycles in $\mathcal{C}$, we have at most $2 \times 2^{\frac{d}{4}}$ choices. The algorithm branches over each of these choices.

Once we have fixed a choice for each cycle $C \in \mathcal{C}$, we create two sub-instances of MATCHING DISTANCE in maximum matchings corresponding to this set of choices, one for the cycles reconfigured using a vertex from U, and one for those reconfigured using a vertex from W. The first sub-instance is obtained by deleting the M_s-exposed vertices in W from G. For this sub-instance, the target matching M is the matching obtained from M_s by reconfiguring only those components in $M_s \oplus M_t$ which are to be reconfigured using a vertex in U. For the second sub-instance, the input graph is obtained by deleting M_s-exposed vertices in U from G, and we need to find the shortest transformation from M to M_t. It is easy to see that both of these sub-intances are reconfiguring maximum matchings, and can be solved by solving the corresponding DIRECTED STEINER TREE instances.

Note 1. If a component of $M_s \oplus M_t$ is an odd path, then it will not have a unique M_s-exposed vertex. It is easy to prove that if such a component exists in $M_s \oplus M_t$, then we have a shortest transformation which visits a non-maximal matching and the above algorithm does not apply.

3 Results and Main Techniques

The key idea is an approximation preserving distance from MATCHING DISTANCE in bipartite graphs to DIRECTED STEINER TREE, which is known to be FPT.

If one of M_s or M_t is maximal, then it is easy to see that the shortest reconfiguration sequence has length $\frac{d}{2}$ if G has no cycles, and $\frac{d}{2} + 1$ otherwise. If M_s is not maximum, we can transform it into a non-maximal matching by sliding tokens along an M_s-augmenting path. We can use this idea to find the shortest reconfiguration sequence from M_s to M_t which visits a non-maximal matching by reducing this case to finding the shortest s-t path in an appropriate weighted digraph. Thus, we have a polynomial time exact algorithm for this case.

If no shortest reconfiguration sequence visits a non-maximal matching, then we have an FPT algorithm which gives the shortest transformation by reducing the problem to DIRECTED STEINER TREE.

4 Key take away from the paper

The key take away is that there is an approximation preserving reduction from MATCHING DISTANCE in bipartite graphs to DIRECTED STEINER TREE. This reduction has been used to give an FPT algorithm for MATCHING DISTANCE in bipartite graphs by using the FPT algorithm for DIRECTED STEINER TREE. It is also possible to design an approximation algorithm for the problem using this reduction.

5 Frame a question

Which parts of the algorithm require the assumption that G is bipartite? What are the ideas which apply to non-bipartite graphs also?

We only make use of the assumption that G is bipartite in the algorithm for Case 2. The algorithm for Case 1, i.e., when there is a shortest reconfiguration sequence from M_s to M_t which visits a non-maximal matching, is applicable to general graphs also.