An exact exponential algorithm for trees to find tropical connected set

Eugene Vagin

[07.11.19]

1 Intro

Paper: **Exact exponential algorithms to find tropical connected sets of minimum size**

We consider the problem of finding minimal tropical connected sets on colored trees. The main result is a branching algorithm which computes a minimum tropical set on trees in time $\mathcal{O}^*(1.2721^n)$

2 Definitions

- $T = (V, E)$: tree, V - set of vertices, E - set of edges
- $\forall X \subseteq V \;\; G[X]$: subgraph induced by X
- $N(v)$: all neighbours of vertex v
- $N[v] = N(v) \cup \{v\}$
- $S \subseteq V$: **connected**, iff $G[S]$ - connected
- $C = \{c(v) : v \in V\}$: set of colors for G
- (T, c) : colored graph
- $c(S) = \{c(v) : v \in S\}$: set of colors of S
- S **is tropical**, if $c(S) = C$

3 Branching

Branching belongs to main tools to design exact exponential algorithms
Idea: solving instance of problem by recursively branching into instances of subproblems of smaller sizes via branching and reduction rules
Let $T(n)$ - upper bound for the running time of algorithm, when applied to any instance of the size n.
Branching rule: the rule which splits problem to subproblems (e.g. $n \to n - t_1, n - t_2, .., n - t_b$).
Branching vector $= (t_1, .., t_b) \sim \tau(t_1, .., t_b)$. Assumed that $T(n) \leq \sum_1^b T(n - t_i)$
Branching number: the largest $\lambda \in \mathbb{R}$ of equation $\lambda^n - \sum_1^b \lambda^{n-t_i} = 0$.
If there are different branching vectors suitable, λ_{max} is choosen
Finally, the upper bound of running time of algorithm is $\mathcal{O}^*((\lambda_{max})^n)$

4 Instances and subproblems

Again: we want to find minimal tropical connected set (**TCS**) on colored tree.
Let instance of subproblem is 3-partition of V: (S, F, D)

- S: selected vertices, which belong to any solution
- F: free vertices, later its will be moved to S or D
- D: discarded vertices, $\forall v \in D \;\; v \notin \{any\, solution\}$

Our task is to find TCS S^* of (T, c) (solution of (S, F, D)) with next properties:

- $S \subseteq S^*$
- $S^* \cap D = \emptyset$
- S^* is minumum size

Satisfied properties:

1. root r of T belongs to S
2. S and $S \cup F$ is a connected sets of T
3. $\forall v \in D \;\; T(v) \in D$

We will also use next notations:

- **T'** $:= G[S \cup F]$ - subgraph induced by $S \cup F$
- **C'** $:= C \setminus c(S)$ - colors which is not yet appear in S

Also we will consider size of F as size of subproblem instance

5 Description of algorithm

The main idea: we try to build the minimal TCS rooted at each vertex of graph:

```
for each v ∈ V do
  TCS-Tree(T=T(v), (S, F, D) = ({v}, V \{v}, ∅))
```

Also two routines used:

- **ADD(v)**: moves all nodes between r and v from F to S and updates C'.
- **RMV(v)**: moves all subtree of v from F to D

The main ideas of reduction rules:

- $C' = \emptyset \Rightarrow S$ is a solution
- $F = \emptyset \Rightarrow$ solution not found (exit) (note that we assume that all previous items was checked)
- vertices which colors $\notin C'$ can be eliminated (moved from F to D). Also vertices which color is unique in S should be added (moved from F to S)
- if for leaf vertex v exist vertex u with same color which is "cheaper"to add, we add it (e.g. $dist(S, u) = 1$)

There are 3 major rules, each rule have subrules:

- **B1**: $\exists v \in F : \, dist(v, S) \geq 4$
- **B2**: $\exists v \in F : \, dist(v, S) = 3$
- **B3**: $\exists v \in F : \, dist(v, S) = 2$

Subrules looks similar, so I'll consider only one subrule: **B1.(a)**.

if $\exists$ internal $v', v'' \in F \setminus \{v\} : c(v) = c(v') = c(v'')$ **then** $(\{ADD(v), RMV(\{v', v''\})\} \;||\; RMV(v))$
When $ADD(v)$ called, we should move $path(v, S)$ from F to S. Note that $path(v, S) \geq 4$ vertices.
For calling $RMV(\{v', v''\})$ exists two cases:

- v' is ancestor of v'' (or vice-versa): in this case at least **3** vertices should be moved from F (because v' is internal and has at least two children: v'' and child of v'')
- v' is not ancestor of v'' (and vice-versa): in this case at least **4** vertices should be moved from F (because v' and v'' has at least 1 child)

When $RMV(v)$ is called, we can move from F only one vertex (because v is leaf).
So, finally, instance with size n is branched to $(n-7, n-1)$ or $(n-8, n-1)$. Branching vector is $\tau(7, 1) \leq 1.2555$ (we consider the worst case).

Similarly can be considered other subrules. The worst obtained branching vector is $\tau(4, 2) \leq 1.2721$.
So, the running time of this algorithm is $\mathcal{O}^*(1.2721^n)$