An Efficient Approach to Finding Dense Temporal Subgraphs

Shuai Ma, Renjun Hu, Luoshu Wang, Xuelian Lin, Jinpeng Huai

(Supplementary Material)

APPENDIX: PROOFS

Proof of Proposition 2: Assume without loss of generality, the time interval $[i, j]$ has a dense subgraph. We only need to consider the case when there are no local maxima between i and j. Let i' be the largest point with $i' \leq i$ and j' be the least point with $j' \geq j$, if there exist, such that the cohesive density curve has a local maximum at points i' and j'. Otherwise, we simply let $i' = i$ or $j' = j$. Note that there must exist at least one of i' and j' in this case.

By the evolving convergence phenomenon and the definition of dense subgraphs, time interval $[i', j']$ has a subgraph whose cohesive density is no less than the dense subgraph of interval $[i, j]$. We thus have the conclusion. □

Proof of Theorem 3: We first present the definition of AFP-reduction and then show that there exists an AFP-reduction (h, g) from the NWM problem to the FDS problem. The conclusion follows since the NWM problem is NP-hard to approximate within any constant factor [13] [23].

(1) Let Π_1 and Π_2 be two maximization optimization problems. An AFP-reduction from Π_1 to Π_2 is a pair of PTIME functions $(h,\ g)$ that satisfies the following:

(a) for any instance I_1 of Π_1, $I_2 = h(I_1)$ is an instance of Π_2 such that $\mathsf{opt}_2(I_2) \geq \mathsf{opt}_1(I_1)$, where $\mathsf{opt}_1(I_1)$ (respectively $\mathsf{opt}_2(I_2)$) is the value of an optimal solution to I_1 (respectively I_2), and

(b) for any feasible solution s_2 to I_2, $s_1 = g(s_2)$ is a feasible solution to I_1 such that $\mathsf{obj}_1(s_1) \geq \mathsf{obj}_2(s_2)$, where $\mathsf{obj}_1()$ (respectively $\mathsf{obj}_2()$) is a function measuring the value of a solution to I_1 (respectively I_2).

(2) We next construct algorithm h. Given an instance I_1 of NWM as its input, algorithm h outputs an instance I_2 of FDS on an aggregate graph. The instance I_1 consists of a graph $G(V, E)$ with a non-negative node weight $p(v)$ for each node $v \in V$ and a non-negative edge weight $w(e)$ for each edge $e \in E$. Algorithm h constructs the instance I_2 consisting of an aggregate graph $\widehat{G}(V \cup V', E \cup E', W_a)$ such that (a) for each node $v \in V$ of G, it adds a new node v' and a new edge $e_v = (v, v')$ to $\widehat{G}$ with $W_a(e_v) = p(v)$, and (b) for each edge $e \in E$ of G, it sets $W_a(e) = -w(e)$ in $\widehat{G}$. That is, aggregate graph $\widehat{G}$ has $2 \times |V|$ nodes and $|E| + |V|$ edges, and, hence, algorithm h runs in PTIME.

(3) We then construct algorithm g. Given a feasible solution s_2 to I_2, *i.e.,* a subgraph $\widehat{H}(V_s, E_s, W_a)$ of $\widehat{G}$, algorithm g outputs a solution s_1 of the NWM instance I_1. Algorithm g first produces a subgraph $H(V'_s, E'_s)$ of G from $\widehat{H}(V_s, E_s, W_a)$ by (a) removing all the nodes v' not in G and their incident edges (v, v'), (b) assigning all the remaining nodes with the same node weights as in G, and (c) setting $w(e) = -W_a(e)$ for each remaining edge e. Then it finds and returns a minimum spanning tree of H, which is a feasible solution to I_1. Given these, algorithm g is in PTIME as well.

(4) By the constructions of algorithms h and g above, it is easy to see that $\mathsf{opt}_2(I_2) = \mathsf{opt}_1(I_1)$ and $\mathsf{obj}_1(s_1) \geq \mathsf{obj}_2(s_2)$. Hence, (h, g) is indeed an AFP-reduction from the NWM problem to the FDS problem.

Putting these together, we have the conclusion. □

Proof of Proposition 4: The equivalence can be proved by showing that the dense subgraph in an aggregate graph can be used to construct the maximum net worth subtree in the corresponding converted graph, and vice versa.

Observe that the dense subgraph SG of an aggregate graph $\widehat{H}$ contains the entire CC_i (line 2, Fig. 5) if any edge of CC_i is included in SG, and these components are further linked by edges of $\mathsf{convertAG}(\widehat{H})$ (lines 6–7, Fig. 5). Hence, $\mathsf{convertAG}(SG)$ is a subtree of $\mathsf{convertAG}(\widehat{H})$ whose net worth equals to $\mathsf{cden}(SG)$. Moreover, any subtree ST of $\mathsf{convertAG}(\widehat{H})$ can construct a subgraph of $\widehat{H}$ whose cohesive density equals to $NW(ST)$.

Given the dense subgraph SG of an aggregate graph $\widehat{H}$, subtree $\mathsf{convertAG}(SG)$ must be the maximum net worth subtree of $\mathsf{convertAG}(\widehat{H})$. Otherwise, a subtree ST^* with a higher net worth exists and can construct subgraph SG^* of $\widehat{H}$ with a higher cohesive density. Similarly we can prove that given the maximum net worth subtree ST of $\mathsf{convertAG}(\widehat{H})$, subgraph SG corresponding to ST must be the dense subgraph of $\widehat{H}$. We thus have the conclusion. □

Proof of Proposition 5: Let T be the merged MST of two MSTs $T_1(V_1, E_1)$ and $T_2(V_2, E_2)$ in the process of $\mathsf{strongMerging}$. We first show that it suffices to consider a T consisting of edges of T_1 and T_2 and those between T_1 and

• *S. Ma, R. Hu, X. Lin and J. Huai are with the SKLSDE Lab, School of Computer Science and Engineering, Beihang University, Beijing 100083, China. E-mail: {mashuai, hurenjun, linxl, huaijp}@buaa.edu.cn.*

• *L. Wang is with Google Inc., Mountain View, CA 94043. This work was done when the author was at Beihang University. E-mail: luoshu@google.com.*

Manuscript received 17 Oct. 2017; revised 8 Nov. 2018; accepted 31 Dec. 2018. (Corresponding author: Shuai Ma)

T_2 only. Without loss of generality, assume that T contains an edge $e = (u, v)$ with $u, v \in V_1$ and $e \notin E_1$. Adding e into T_1 forms a cycle $u/\cdots/v/u$ in T_1. Given the fact that T_1 is an MST, the weight of e must be the maximum among all edges in the cycle. And, hence, edge e of T can be replaced by an edge in the cycle without increasing the total weight of T. Similarly, there is no need to consider edges $e = (u, v)$ with $u, v \in V_2$ and $e \notin E_2$.

From above, it is easy to see that merging T_1 and T_2 is equivalent to maintaining an MST in a graph $T_1 \cup T_2$ after inserting the edges between V_1 and V_2. Using dynamic trees, it takes $O(\log(|V_1| + |V_2|))$ time to deal with an edge [34]. Hence, the extra cost is bounded by $O(|E'| \log |V'|)$. □

Proof of Corollary 6: Procedure strongPruning is essentially the same as the BEST-SUBTREE pruning procedure (Fig. 2-5) in [28], except that the latter recursively finds the subtree. Note that Theorem 2.8 in [28] proves that the returned subtree ST obtains the optimal net worth among all possible subtrees of T. We next show that procedure strongPruning runs in $O(|V_T|)$ time. It first traverses the tree from the root in a breath-first manner, and updates $nw(u)$ in the reverse traversal order. Finally, strongPruning extracts and returns the subtree rooted at the node with the maximun $nw(u)$. All these operations together can be done in $O(|V_T|)$ time. □