Algorithm for Finding Linkage in a Totally ϕ-decomposable Digraph

Gabriella Juhl Jensen

Jørgen Bang-Jensen

2021

Decomposable Digraph

A digraph $D = S[H_1, \ldots, H_s]$ is called Decomposable where $|S| = s$ and each H_i are disjoint digraphs replacing every vertex in S. The digraphs H_i will in this presentation be called **houses**. For a set of digraphs ϕ we can talk about ϕ-decomposable digraphs.
For a digraph $D = S[H_1, \ldots, H_s]$ to be ϕ-decomposable either $D \in \phi$ or $S \in \phi$.
For a digraph $D = S[H_1, \ldots, H_s]$ to be totally ϕ-decomposable it has to be ϕ-decomposable and if $D \notin \phi$, then $S \in \phi$ and each H_i has to be totally ϕ-decomposable for $i = 1, \ldots, s$.

The weak linkage Problem

Given two pair of vertices s_1, t_1 and s_2, t_2 finding arc-disjoint paths between each pair is 2-weak linkage problem. Then the k-weak linkage problem is finding k arc-disjoint paths between k pair of terminals. where a terminal pair is a source and a zink in the paths of the solution of the linkage problem. When talking about linkage problem for decomposable digraph, we can have houses with terminals in and some without any terminals. The houses with no terminals in is called **clean houses**. Then a terminal pair can either be inside the same houses or in different houses. If a terminal pair is contained inside the same house it is called an internal pair other wise we call the pair external. The same with a path if it is fully containd inside a house it is called an internal path other wise it is external. you can have an external path for a internal piar if the path go out of the house and in agian.

Notation for the weak linkage problem

Every thing on this list is denoted like this unless it is specifed noot to be like that.

- a natruel number k is going to repricent the number of terminal pairs there in the linkage problem.
- Π denote the set of terminal pairs $(s_1, t_1), \ldots, (s_k, t_k)$.
- D is the totally ϕ-decomposable digraph where each house is denoted H_i for $i = 1, \ldots, s$.

Bombproof sets

Definition

[1] We say that a class of digraphs ϕ is Bombproof is there exsists a polynomial algorithm $\mathcal{A}_\phi$ to find a totally ϕ-decomposition of every totally ϕ-decomposable digraph and, for every integer c, there exists a polynomial algorithm $\mathcal{B}_\phi$ to decide the weak k-linkage problem for the class

$$\phi(c) := \bigcup_{D \in \phi} D(c) \tag{1}$$

for a integer c, the class denoted $D(c)$ is the for a digraph D you can first add as many parallel arcs to arcs that already exists in D you can blow up b vertices where $0 \leq b \leq c$ the digraph that is blown up to has a size $\leq c$.

Contraction clean vertices

Lemma

[1] Let D be a digraph, Π a list of k terminal pairs and $H \subset D$ a clean house with respect to Π. Let D' be the contraction of H into a single vertex h. Then D has a waek Π-linkage if and only if D' has a weak Π-linkage.

Max inside House External Path

Lemma

Let $D = S[H_1, \dots H_s]$ be a decomposable digraph, let Π' be a list of h terminal pair and let F be a set of arcs in D satisfying that $d_F^-(v), d_F^+(v) \leq r$ for all $v \in V(D)$. If $(D \backslash F, \Pi')$ has a weak linkage, then it has a weak linkage $P_1, \dots, P_h$ such that, we have $|V(\bigcup_{i \in \mathcal{E}} P_i \cap H_j)| \leq 2h(h + r)$, for every $j \in [1, \dots, s]$, where $\mathcal{E}$ denotes the set of indeces i for wich P_i is external.

Algorithm for Digraph with Deleted Arcs

Lemma

Let $\mathcal{C}$ be a class of digraphs for which there exists an algorithm $\mathcal{A}$ to decide the weak k-linkage problem, whose running time is bounded by $f(n,k)$. Let $D=(V,A)$ be a digraph, Π a list of k pairs of terminals and $F \subseteq V \times V$ such that $D' := (V, A \cup F)$ is a member of $\mathcal{C}$. There exists an algorithm $\mathcal{A}^-$, whose running time is bounded by $f(n, k+|F|)$, to decide whether D has a weak Π-linkage.

The theorem of the main Algorithm

Theorem

Let ϕ be a bombproof class of digraph. There is a polynomial algorithm $\mathcal{M}$ that takes as input a 5tuple $[D, k, k', \Pi, F]$ where D is a totally ϕ-decomposable digraph, k, k' are natural numbers with $k' \leq k$,Π is a list of k' terminal pairs and $F \subseteq A(D)$ is a set of arcs satiesfying

$$d_F^-(v), d_F^+(v) \leq k - k' \text{ for alle } v \in V(D) \tag{2}$$

$$|F| \leq (k - k')2k$$

and decides wheter $D \backslash F$ contains a weak Π-linkage.

The main algortihm

The main algorithm for this part is a proof of a theorem that tells that there exists a algorithm for finding the k-linkage in special kind of graphs. This result is found by ... in .... .

Theorem

Let ϕ be a bombproof class of digraphs. There is a polynomial algorithm $\mathcal{M}$ that takes as input a 5 tuple $[D, k, k', \Pi, F]$, where D is a totally ϕ-decomposable digraph, k,k' are natrual numbers with $k' \leq k$, Π is a list of k' terminal pairs and $F \subseteq A(D)$ is a set of arcs satisfying

$$d_F^-(v), d_F^+(v) \leq k - k' \ \forall v \in V(D) \qquad (3)$$
$$|F| \leq (k - k')2k$$

and decides wheter $D \backslash F$ contains a waek Π-linkage.

The main algortihm I

To proof 0.2 we first state the algorithm $\mathcal{M}$ then we prove that it works, and last that the time for the algorithm is polynomial.

Procedure 1: The main algorithm $\mathcal{M}$

Input: Digraph D, two natural numbers k and k' where $k' \leq k$, a list of k' terminal pairs Π, A set of arcs $F \subseteq A(D)$ satiesfying:

$$d_F^-(v), d_F^+(v) \leq k - k' \ \forall v \in V(D)$$
$$|F| \leq (k - k')2k$$

Output: Either "No weak-linkage exists" or "there exists a weak-linkage in (D, Π) with arc set F."

1: If $\Pi = \emptyset$ output that a solution exists and return
2: Run $\mathcal{A}_\phi$ to find a total ϕ-decomposition of $D = S[H_1, \ldots, H_s]$.
3: **if** this decomposition is trivial that is $D = S$ **then**

The main algortihm II

4: $D \in \phi \subset \phi(1)$, so run $\mathcal{B}_{\phi}^{-}$ on $(D \backslash F, \Pi)$ to decide the problem and return.
5: **end if**
6: Find among $H_1, \ldots, H_s$ those houses $K_1, \ldots, K_l$ that contain at least one terminal. Let D' be obtaint by contracting all the clean houses. Let F' be the set of arcs obtaint from F after the contraction.
7: Let $\Pi^e \subset \Pi$ ($\Pi^i \subset \Pi$) be the list of external (internal) pairs $(s_q, t_q) \in \Pi$.
8: **for** every partion of $\Pi^i = \Pi_1 \cup \Pi_2$ look for external paths linking the pairs in $\Pi^e \cup \Pi_1$ and internal pairs in Π_2 **do**
9: **if** $\Pi^e \cup \Pi_1 = \emptyset$, then for $i = 1, \ldots, l$: **then**
10: run $\mathcal{M}$ recursively on input $[K_i, k, k_i', \Pi \cap K_i, F \cap A(K_i)]$, where $\Pi \cap K_i$ denotes the list of terminal pairs that lie inside K_i and k_i' is the number of those pairs.
11: **end if**
12: **if** $\Pi^e \cup \Pi_1 \neq \emptyset$ **then**

The main algortihm III

13: let k_i' be the number of pairs in $\Pi_2 \cap K_i$

14: **for** each possible choice of l vertex sets $W_i \subseteq V(K_i), i = 1, \ldots, l$ of size $\min\{|V(K_i)|, 2(k' - k_i')(k - k')\}$ and arc sets $F_i \subseteq A(K_i \langle W_i \rangle) \backslash F$, $i = 1, \ldots, l$ with F_i satisfying

$$d^-_{F_i \cup (F \cap A(K_i))}(v), d^+_{F_i \cup (F \cap A(K_i))}(v) \leq k' - k_i'. \quad (4)$$

$$|F_i| \leq 2(k' - k_i')(k - k') \quad (5)$$

do

15: **for** every K_i **do**

16: remove all the vertices of $V(K_i) \backslash W_i$ and then all remaining arcs except those in F_i.

17: **end for**

18: Define D'' to be the digraph obtaint from D' with this procedure.

19: Run B^-_ϕ on $(D'' \backslash F', \Pi^e \cup \Pi_1)$.

The main algortihm IV

20: **for** $i = 1, \ldots, l$ **do**
21: run $\mathcal{M}$ recursively on input $[K_i, k, k_i', \Pi_2 \cap K_i, F_i \cup (F \cap A(K_i))]$.
22: **end for**
23: **end for**
24: **end if**
25: **if** the if statement in 9 all intances examined are linked **or** at the if statement in 12 there is a choice of $W_i, F_i, i = 1, \ldots, l$ such that all instances examined are linked **then**
26: output that a weak linkage exists and return.
27: **end if**
28: **end for**
29: **if** all choices of Π_1, Π_2 have been consideredwithout verifying the existens of any weak linkage **then**
30: output that no weak linkage exists.
31: **end if**

Step by step what $\mathcal{M}$ does

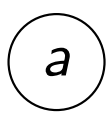
First we deskribe wiht words what the input and output of the algorithm is. The output is already written in words and is very undestandeble.
The input can be elaborated somewhat more, first $\mathcal{M}$ is polynomial but also recursively defined. It decides whether (D, Π) has a weak-linkage on overall k terminals. Since the algorithm is recursive it does not find all the solutions in one go therfore we define k' as the number of terminals that we still need to find a weak linkage for, and F is a part of the solution of at most $k - k'$ already found weak-linkages of D, Π is the set of terminals that we what to find the weak linkage for.
So you can in the begining have $F = \emptyset$ and $k = k'$. This will help on the undestaniding of the algorithm.

Step by step what $\mathcal{M}$ does I

Step 1: "$\Pi = \emptyset$" makes sure that if we call the algorithm $\mathcal{M}$ with no pairs then there exists the solution with zero acrs to solve the weak-linkage problem.
Step 2: Recall that the digraph is totally ϕ-decomposable and ϕ is bomproof and from 1 we know tha the digraph has a algortihm $\mathcal{A}_\phi$ that gives the ϕ-decomposition of the digraph.
Step 3-5: From 1 we know that $\mathcal{B}_\phi$ decides a weak-linkage for $D \in \phi$, since we cant guarentee that $D \backslash F \in \phi$ we use 4 that tells us that $\mathcal{B}_\phi^-$ can decide a weak-linkage in $D \backslash F = (V, A \backslash F)$ if $D' \in phi$ $D' = (V, (A \backslash F) \cup F) = (V, A) = D \in \phi$.
Step 6: Here we find all the non-clean houses from $H_1, .., H_s$ and contract all the clean houses w.r.t. Π we make a new nummeration of all the non-clean houses $K_1, \ldots, K_l$ of D w.r.t. Π Since by 2 we know that contracting one clean house in D if it has a linkage so does our new digraph, then use this lemma agian and agian until there is no more clean houses. This is our new digraph D' with

Step by step what $\mathcal{M}$ does II

non-clean houses $K_1, \ldots, K_l$ and if we find a weak linkage w.r.t. Π in D' we know that D has a weak linkage from continueing using 2. We also let $F' = F \cap A'$ where A' is the arcset of D'.
Step 7: Recall an internal pair is where both vertices is in the same house and an external pair is where the vertices is two different houses.
Step 8: This for loop is looking for two different cinds of path between internal pairs since the path for an internal pair can be an internal path (fully cept in the house) or an external path going out of and later in the house. For simplyfication look at **??**



Step by step what $\mathcal{M}$ does III

Step 9-11: if $\Pi^e \cup \Pi_1 = \emptyset$, either we have already found all external path in this partition or there was none. Either way all terminal pairs left is internal so and $\Pi = \Pi_2$. So we are only interesten in finding the internal path of the internal pairs, which is why we can call $\mathcal{M}$ on each house for itself. Each K_i could be a big graph in itself that is decomposable with at least some house H_i where $|H_i| \geq 2$, if this is not the case the algorithm returns after step 3: and continue with the next. If $\mathcal{M}$ has already found some external paths F might not be empty and may use some arcs inside K_i therefore $F \cap A(K_i)$. $\Pi \cap K_i$ is becouse we are not interested in the terminal pairs that are not a part of the graph we are looking at (pairs inside K_j where $j \neq i$).

Step 12: Looking for external paths in a big graph is a bit more deficualt since we do not know which arcs and vertices not to use.

Step 13-14: First we find all the pairs that is internal pairs, we want to link as internal paths, the number of these is k_i' for each $i = 1, \ldots l$. Then we choose a very specifik size of vertex sets W_i

Step by step what $\mathcal{M}$ does IV

and loop over every choiche of these. These vertex set induces a subdigraph, where we make a possible arc set where inside not containg what is inside F we call these F_i we make these set as big as possible linkage need for the rest of the terminal pairs (those we want to find external paths of$\Pi^e \cup \Pi_1$) the number of those is $k' - k'_i$ since every pair mabey has to go through the house we are looking at.

Step 15-19: For each house we remove all vertices not in the vertex set W_i after removing these vertices we remove all remaining arcs except those arcs in F_i. This is defined in the algorithm as D''. We can show that $D'' \in \phi(2k^2)$. First we know that since D is totally decomposable $S \in \phi$ and from 1 and the definition on $D(c)$ we can take S add as many parralelle arcs as we want we only need to blow up I vertices those houses of D that are not clean we know that there is k' terminal pairs and that $k' \leq k$ meaning $I \leq 2k$ these I vertices needs to be blown up and from lemma 3 lets say that we want to find $k'' \leq k'$ external paths in D ($|\Pi^e \cup \Pi_1| = k''$) then

Step by step what $\mathcal{M}$ does V

we are only looking at k'' terminals meaning in every blow up we need at most $2k''(k'' + (k - k'))$ since $k'' \leq k'$ we have $2k''(k'' + (k - k')) \leq 2k''k$ vertices in W_i and $2kk'' \leq 2kk' \leq 2k^2$ which is the biggest number we will need to blow up the l vertices meaning $c = 2k^2$ so $D'' \in \phi(2k^2)$.
Step 20-24: We need to make sure that the tuple $[K_i, k, k'_i, \Pi_2 \cap K_i, F_i \cup (F \cap A(K_i))]$ upholds every condition for every choice of that tuple. Since we are not focusing on loops we know that the max number of arcs is bounded by the max number of vertices $|F_i| \leq 2kk''$ the rest of the terminals is the number of internal pairs which we in the algorithm denote k'_i. we know that $k'_i \leq k' - k''$ meaning $k'' \leq k' - k'_i$. we start calculating the to demands of F in the tuple. Note that

Step by step what $\mathcal{M}$ does VI

$d_{(F\cap A(K_i))}(v) = d_F(v),\ \forall v \in V(K_i)$ and we also know $d_{F_i}(v) \leq k''$ so

$$d^+_{F\cup F_i}, d^-_{F\cup F_i} \leq k - k' + k'' = k - (k' - k'') \leq k - k'_i \quad (6)$$

$$|(F \cap A(K_i)) \cup F_i| \leq |F| + |F_i| \leq 2k(k - k') + 2kk'' \quad (7)$$

$$\leq 2k(k - k') + 2k(k' - k'_i) = 2k(k - k'_i). \quad (8)$$

Clearly the tuple for F holds forall its conditions.

induction on $\mathcal{M}$ I

We have now proved and explained each step in the algorithm, that it does what we think. Now we need to check whether given your favorite digraph, that upholds the conditions, the algorithm gives the right result. If the digraph do not terminate before examine list $\Pi^e \cap \Pi_1$ of k'' terminal pairs if $k'' = 0$ we enter step 9 and $F_i = \emptyset$, for $i = 1, \dots l$, and by the induction hypothesis we can assume that if there exists a weak linkage in each K_i the algorithm would find it. Now if this is not the case and $k'' > 0$ step 12 is then entered and we construct D'' which we have described belong to $\phi(2k^2)$ and then as described before we can use B_ϕ^- which is correct by 1, so the algorithm will find a weak Π''-linkage if it exists in $D'' \setminus F'$. After all this there is made a recursiv call on each K_i finding k_i' weak linkages and by the above proof we know it works. So since B_ϕ^- correctly finds the weak linkage inside $D'' \setminus F'$ using only arcs from F_i $\forall i \in [l]$, then each K_i is recursively called from D'' we can easily come back to D' since we find the weak k_i'-linkage

induction on $\mathcal{M}$ II

inside each K_i which is not using any of the arcs from F_i we know that together these stil form the seperated weak linkages. By 4 we know that we can find a weak linkage in $D \backslash F$ if we can find it in $D'' \backslash F'$ which just proved we can. given a perfekt weak Π-linkage.

this can be done in polynomial time


Jørgen Bang-Jensen, Tilde My Christiansen, and Alessandro Maddaloni.
Disjoint paths in decomposable digraphs.
Journal of Graph Theory, 85(2):545–567, 2017.