Lecture 9 (7.12.2012) Author: Michał Włodarczyk

Separators in graphs

In this lecture we introduce the theory of separators and its applications in parameterized complexity. We focus on undirected graphs but similar results can be obtained for directed ones.

1 Preliminaries

1.1 Definitions

Let $G = (V, E)$ be an undirected graph including two special vertices: s, t. Let $S \subseteq V(G) \backslash \{s, t\}$.

Definition 1 (Reachable set)**.** $R(S) = \{v \in V(G) :$ there is a path $s \to v$ in $G \backslash S\}$. To dispel doubts: $s \in R(S)$.

Definition 2 (Separator)**.** S is called *st*-**separator** or *st*-**cut** (or just **separator / cut** if it does not lead into confusion) in G if $t \notin R(S)$

We will need three different orders over the family of *st*-separators. If we say S is a **minimal separator** then it means that there are no cuts in proper subsets of S. Next, S is called a **smallest separator** if any other cut S' satisfies $|S'| \geq |S|$. Third order is defined as follows:

Definition 3. $S_1 \prec S_2$ means $R(S_1) \subseteq R(S_2)$.

Definition 4 (Neighbours set)**.** Let $A \subseteq V(G)$. Then

$$N(A) = \{v : \exists_{a \in A}(a, v) \in E(G)\} \backslash A$$

1.2 Basic properties

Lemma 5. *If separator S is minimal then $S = N(R(S))$.*

Proof: For any separator $N(R(S)) \subseteq S$. Moreover $N(R(S))$ is also a cut so if $N(R(S)) \neq S$ then S cannot be minimal. □

Lemma 6 (Submodularity)**.** *For $X, Y \subseteq V(G)$*

$$|N(X)| + |N(Y)| \geq |N(X \cap Y)| + |N(X \cup Y)|$$

Proof: We will check for every vertex how many times it is counted on the left and right side. There are 4 cases:

- $v \notin N(X \cap Y), v \notin N(X \cup Y)$ – Then v does not appear on the right side.
- $v \in N(X \cap Y), v \notin N(X \cup Y)$ – Of course $v \in N(X \cup Y) \cup X \cup Y$ but it cannot lie neither in $N(X \cup Y)$ nor in $X \cap Y$. Hence v can be found in just one set from X, Y and in just one set from $N(X), N(Y)$. It adds one to the left and one to the right side of the formula.

- $v \notin N(X \cap Y), v \in N(X \cup Y)$ – $N(X \cup Y) \subseteq N(X) \cup N(Y)$ so at least one from those two sets contains v and exactly one from $N(X \cap Y), N(X \cup Y)$ does.
- $v \in N(X \cap Y), v \in N(X \cup Y)$ – There is $w \in X \cap Y$ that $(w, v) \in E(G)$ but $v \notin X \cup Y$ so $v \in N(X), v \in N(Y)$. This time v is counted twice on both sides.

Splitting the sum on these 4 cases leads to the thesis. □

Lemma 7. *The set of smallest separators in G ordered with $\prec$ contains the first element and the last element.*

Proof: Let us consider smallest cuts S_1, S_2. We combine Lemma 5 and Lemma 6 applied to $R(S_1), R(S_2)$:

$$|S_1| + |S_2| = |N(R(S_1))| + |N(R(S_2))| \geq |N(R(S_1) \cap R(S_2))| + |N(R(S_1) \cup R(S_2))|$$

As $S_- = N(R(S_1) \cap R(S_2))$ and $S_+ = N(R(S_1) \cup R(S_2))$ also form cuts then they have to be smallest, too. In other words, we have just constructed elements S_-, S_+ so $S_1, S_2 \prec S_+$ and $S_- \prec S_1, S_2$. As the set of smallest separators in G is finite, this gives thesis. □

1.3 Important separators

Definition 8. $S \subseteq V(G) \backslash \{s, t\}$ is called **important separator** if S is minimal and for any <u>other</u> separator S':

$$S \prec S' \Longrightarrow |S'| > |S|$$

One can regard that every separator behind the important one must be bigger. The Lemma 7 leads to the following

Remark 9. In every graph there is exactly one smallest separator which is important.

Lemma 10. *The important smallest separator can be found in polynomial time.*

Proof: The problem of finding the smallest separator can be reduced to the Maximum Flow problem. If we found a smallest st-cut S_1 we need to check if there exists another smallest cut S_2 satisfying $S_1 \prec S_2$. If yes then there is a vertex $v \in R(S_2) \backslash R(S_1)$. We can check all possibles vertices v by contracting $R(S_1) \cup \{v\}$ into one vertex s' and checking if there is a $s't$-separator of size equal to $|S_1|$.

We can iterate this procedure until no such v can be found – then constructed separator is important. □

Lemma 11. *Let S be the important smallest separator. If S_1 is important then $S \prec S_1$.*

Proof: We again take advantage of Lemma 6

$$|S| + |S_1| = |N(R(S))| + |N(R(S_1))| \geq |N(R(S) \cap R(S_1))| + |N(R(S) \cup R(S_1))|$$

As S is smallest then $|N(R(S) \cap R(S_1))| \geq |S|$. By combining these 2 inequalities, we obtain $|N(R(S) \cup R(S_1))| \leq |S_1|$. But S_1 is important and $S_1 \prec N(R(S) \cup R(S_1))$ – this is possible only if $S_1 = N(R(S) \cup R(S_1))$ which implies $S \prec S_1$. □

Lemma 12. *Let S be an important st-separator in G and $sv \in E(G)$.*

1. *If $v \in S$ then $S \backslash \{v\}$ forms an important separator in $G \backslash \{v\}$.*
2. *If $v \notin S$ then S is important in G/sv (edge contraction).*

Proof:

1. Suppose it is not true and $S_1 = S \backslash \{v\}$ is not important in $G_1 = G \backslash \{v\}$. Then there exists a cut $S_2 \subset G_1$ that $|S_2| \leq |S_1|$ and $S_1 \prec S_2$. $S_2 \cup \{v\}$ is therefore a cut in G which proves S was not important.
2. Straightforward as above.

□

Theorem 13. *There is an algorithm with running time $\mathcal{O}^\star(4^k)$ which enumerates all important st-separators in G which size does not exceed k.*

Proof: We take advantage of Lemmas 11 and 12. Firstly we look for the important smallest st-cut S and define $\lambda(G) = |S|$. As all important separators are behind S we can contract $R(S)$ into new s. Then for any $v \in S$ – it is now a neighbour of s – we branch:

1. if it lies in some separator we can remember v and erase it from G,
2. otherwise we contract edge sv.

We present pseudocode of the algorithm (in the beginning $Z = \emptyset$).

```
function GEN-CUTS(G, s, t, Z, k)
    S ← important smallest st-separator in G
    if |S| > k then
        return
    else if k = 0 or S = ∅ then
        output Z
    else
        G ← G with R(S) contracted into s
        v ← some vertex from S
        GEN-CUTS(G − v, s, t, Z + v, k − 1)
        GEN-CUTS(G/sv, s, t, Z, k)
    end if
end function
```

Due to Lemma 10 every node in this recursion takes polynomial time and we must only estimate the number of leaves in the recursion tree. Let us define a potential as $2k - \lambda(G)$. In the first branch k decreases by one and $\lambda(G)$ decreases by at most one. In the second one k stays the same whereas $\lambda(G)$ increases because S was important. As potential must be positive to produce separators and it always decreases, the algorithm will output at most $2^{2k} = 4^k$ cuts. □

Construction of the algorithm proves the following

Remark 14. The number of all important st-separators in G which size does not exceed k is at most 4^k.

What is interesting, the constant 4 is optimal.

Remark 15. There is a family of graphs in which the number of important separators of size $\leq k$ is $\Omega^*(4^k)$.

Draft of proof: Let us consider a full binary tree of height $k+2$ rooted in s. We connect all leaves with new vertex t. The number of important st-separators of size exactly k equals $(k-1)$-th Catalan number which is $\Omega^*(4^k)$. □

The idea of important separators has been introduced by Marx [6]. Presented boundary for them and the enumerating algorithm originate from Chen et al. [2].

2 Multiway Cut problem

In this section we will need $S-T$ separators where S, T are sets but we can contract S and T into two vertices s, t and work with st-separators instead. If one of S, T is singleton (e.g. $S = \{s\}$) we will use simplified notation $s - T$.

Multiway Cut
Input: undirected graph G, set of terminals $T \subseteq V(G)$, k
Question: is there a set $X \subseteq V(G) \backslash T, \lvert X\rvert \leq k$ that every connected component of $G \backslash X$ contains at most one vertex from T

We will present a FPT algorithm with parameter k whereas in general the problem is difficult.

Theorem 16 (Dahlhaus et al. [4])**.** Multiway Cut *is NP-HARD even for* $|T| = 3$.

Let us order T, no matter how. We start with observation that we can consider only important separators between subsets of T.

Lemma 17 (Pushing Lemma)**.** *If there exists a solution X then it can be modified into other solution Y that contains some important $t_1 - (T \backslash t_1)$ separator.*

Proof: As X is in particular $t_1 - (T \backslash t_1)$ separator, it contains some minimal cut, let us call it S_1. If S_1 is not important then we can find an important one S_2 so $S_1 \prec S_2$ and $|S_2| \leq |S_1|$. We define Y as $(X \cup S_2) \backslash R(S_2)$. It is clear that $|Y| \leq |X| \leq k$ but we have to prove that Y separates any t_i from t_j.

Let us suppose by contradiction that there is a $t_i - t_j$ path in $G \backslash Y$ for $i, j > 1$ but there was not such a path in $G \backslash X$. This may be possible only if the path goes through some $v \in R(S_2)$ – these are only vertices we have erased. However there is a $t_1 - v$ path in $G \backslash Y$ which of course implies existence of $t_1 - t_i$ path. This is impossible because Y contains $t_1 - (T \backslash t_1)$ separator S_2. □

Theorem 18. Multiway Cut *can be solved in* $\mathcal{O}^\star(8^k)$.

Proof: Lemma 17 ensures the following algorithm finds a solution (returns at least one 'Success') if there is any.

function MULTIWAY-CUT(G, T, k)
  if $T = \emptyset$ **then**
    return Success
  else if $k < 0$ **then**
    return Failure
  else
    $t \leftarrow$ first element of T
    for all $S \leftarrow (t - (T \backslash t))$ important separator **do**
      MULTIWAY-CUT($G - S, T - t, k - |S|$)
    end for
  end if
end function

We can iterate through all important separators in time $\mathcal{O}^\star(4^k)$ due to Theorem 13. Then we branch at most 4^k times. Moreover if we found an empty separator then we can just forget about currently analyzed t and jump to another one without branching. This allows us to estimate the running time of the algorithm:

$$T(k) \geq \sum_{l=1}^{k} 4^l T(k-l)$$

If we substitute $T(k) = 8^k$ then

$$\sum_{l=1}^{k} 4^l 8^{k-l} = 4^k \sum_{l=0}^{k-1} 2^l < 8^k$$

which gives desired complexity. □

Theorem 19 (Chen et al. [1])**.** *More careful analysis of the presented algorithm gives complexity* $\mathcal{O}^\star(4^k)$.

3 R-Way Cut problem

R-Way Cut
Input: undirected connected graph G, k, r
Question: is there a set $X \subseteq E(G), \|X\| \leq k$ that G without X has exactly r connected components

We can assume $r \leq k + 1$ as deleting one edge increases the number of connected components by at most one.

3.1 Border R-Way Cut

It turns out it is easier to think about more general problem with additional set of terminals $T \subseteq V(G), |T| \leq 2k$. The second condition will make sense soon.

Definition 20. Let **type** be a pair $(m, (T_i)_{i=1}^m)$ where $0 \leq m \leq r$ and $(T_i)_{i=1}^m$ is some partition of T (allowing empty sets).

Type $(m, (T_i)_{i=1}^m)$ means a way to split G so m is the number of connected components and we know how the terminals are divided.

Definition 21. We say that $X \subseteq E(G), |X| \leq k$ is a **realization** of type t if G without X satisfies conditions above and size of X is minimal. Type t can be **realized** in (G, T) if it has some realization.

We can now define a new problem.

Border R-Way Cut **Input:** undirected connected graph G, k, r, $T \subseteq V(G), \|T\| \leq 2k$ **Output:** function which takes type and returns any realization of it or $\perp$ if the type cannot be realized

By returning function we mean filling some table indexed with types.

In order to solve R-Way Cut we start with Border R-Way Cut having $T = \emptyset$. Next, we just check if there is a realization for $(r, \mathit{empty\_partition})$ – if yes answer *yes*, otherwise *no*.

3.2 Intuition

Let us imagine we can split $V(G)$ into V_1 and V_2 so there are no more than k edges between them, V_1 is connected and $|V_1| > q$ where $q = (2k)^{2k}k(k+1)+1$. It is important that q is greater than number of all types times k.

Let us suppose we solved Border R-Way Cut for $(G[V_1], k, T = N(V_2) \cap V_1)$ and got some function f. Every solution in G induces some type t over $(G[V_1], T)$ and contains a realization of t. However when we substitute that realization with another realization of t (e.g. $f(t)$) the solution stays valid. One may treat f as an 'interface' of V_1 for the rest of G.

If edge e satisfy

$$e \in E(G[V_1])$$

$$e \notin \bigcup_{t \in types(G[V_1], T)} f(t)$$

then in every solution we can replace e with some other edge. Hence we can contract e and work with a smaller instance of the problem. $|V_1| > q$ guarantees that $E(G[V_1]) \geq q > \sum_t |f(t)|$ and we will be able to contract at least one edge.

3.3 Algorithm

Definition 22. (V_1, V_2) for $V_1 \cap V_2 = \emptyset, V_1 \cup V_2 = V[G]$ is called **good partition** if

1. there are at most k edges between them,
2. $G[V_1], G[V_2]$ are connected,
3. $|V_1|, |V_2| > q$,

where $q = (2k)^{2k}k(k+1)+1$.

Theorem 23 (Kawarabayashi, Thorup [5])**.** *There exists a randomized FPT algorithm, parameterized with k, solving* Border R-Way Cut.

Proof: If $|V(G)| \leq q(k+1)$ then for all types we iterate through all subsets of $E(G)$ of size $\leq k$ and simply remember the smallest realization so far. The time complexity looks nasty but it of course depends only on k. Now we focus on bigger graphs.

If there is a good partition (V_1, V_2) in G we can find it in time $\mathcal{O}^\star(q^k)$ with constant probability (see Lemma 25). As $|T| \leq 2k$, one set from V_1, V_2 contains $\leq k$ terminals. Let it be V_1. Then we solve Border R-Way Cut recursively for $(G[V_1], k, r, (T \cap V_1) \cup (N(V_2) \cap V_1))$. Please note $|N(V_2) \cap V_1| \leq k$ from definition of a good partition and $|T \cap V_1| \leq k$ so the invariant '$|T| \leq 2k$' is kept. Next, we contract some edges as described in section 3.2 and work with a smaller instance.

The latter case is when there is no good partition in G and $|V(G)| > q(k+1)$. This time it is possible to generate a realization for every type (if such exists) with fixed probability in time $\mathcal{O}^\star(k(2kq)^k)$. Details are described in Lemma 26.

Remark 24. Both randomized algorithms can be derandomized using splitters with little complexity cost (see Lecture 3.) so we do not have to worry about the probability issues.

Let $h(n,k)$ estimate the asymptotic complexity of the above algorithm without recursive calls for fixed n, k ($n = |V(G)|$), and let $H(n,k)$ be the real complexity. Given h, we need to find H satisfying

$$\forall_{q<q'<n}\ H(n,k) \geq h(n,k) + H(q',k) + H(n-q'+q,k)$$

Substitution $H(n,k) = (n-q-1)h(n,k)$ gives desired FPT complexity.

$$H(n,k) = h(n,k)(n-q-1) = h(n,k)(1 + (q'-q-1) + (n-q'+q-q-1)) \geq$$

$$\geq h(n,k) + (q'-q-1)h(q',k) + (n-q'+q-q-1)h(n-q'+q) = h(n,k) + H(q',k) + H(n-q'+q,k)$$

□

In the end we will handle the randomized search for partitions.

3.4 Colour coding

Lemma 25. *If connected G contains good partition then it can be found in time $\mathcal{O}^\star(q^k)$ with constant probability.*

Proof: If some good partition (V_1, V_2) exists then there is an edge cut $E_1 \subseteq E(G), |E| \leq k$ (the edges which split V_1 and V_2) and sets $E_2, E_3 \subseteq E(G), |E_2| = |E_3| = q$ that form trees in V_1 and V_2 (they prove V_1, V_2 are large enough).

Let us consider randomized algorithm which colours every edge in $E(G)$ in red with probability $\frac{1}{q}$ or in blue with probability $1 - \frac{1}{q}$. If all E_1 became red and E_2, E_3 are blue then we can find them by contracting all blue connected components and checking for every pair of vertices b_1, b_2 which originate from components of size $> q$ if maximum flow between b_1 and b_2 is $\leq k$. It is easy to see that this is a sufficient condition to form a good partition.

What is the probability of getting such colouring?

- probability of painting whole E_1 in red is $\geq \frac{1}{q^k}$,
- probability of painting whole $E_2 \cup E_3$ in blue is $(1 - \frac{1}{q})^{2q} \approx e^{-2}$

If we repeat this procedure $> q^k e^2$ times we achieve the probability of success $\geq 1 - (1 - \frac{1}{q^k e^2})^{q^k e^2} \approx 1 - e^{-1}$ □

Lemma 26. *Let connected G contain no good partition and $|V(G)| > q(k+1)$. If realization of type t exists then it can be found in time $\mathcal{O}^\star(k(2kq)^k)$ with constant probability.*

Proof: Let $t = (m, (T_i)_{i=1}^m)$. If t can be realized then there is $X \subseteq E(G), |X| \leq k$ which splits G into exactly m connected components. As $|V(G)| > q(k+1), m \leq k+1$, there is a component V_m of size $> q$. There is only one such component because otherwise we would get a good partition.

As in previous lemma, we look for some witness of that solution which is possible to find by colouring edges. We will need:

1. set X – edges to be erased,
2. trees $E_1 \dots E_{m-1}$ that $E_i \subseteq E(G[V_i]), |E_i| < q$ spans V_i and if $v \in V_i, vw \in E(G)$ then $w \in V_i \vee vw \in X$
3. forest $E_m \subseteq E(G[V_m]), |E_m| \leq kq$ with condition: if $v \in V_m$ and some $vw \in X$ then component of E_m adjacent with v has at least q edges.

Observe that $\sum_{i=1}^m |E_i| \leq 2kq$.

Every edge from $E(G)$ is painted in red with probability $\frac{1}{2kq}$ or in blue with probability $1 - \frac{1}{2kq}$. What are the odds that X gets painted in red and $E_1 \dots E_m$ in blue?

- probability of painting whole X in red is $\geq \frac{1}{(2kq)^k}$,
- probability of painting all $E_1 \dots E_m$ in blue is $\geq (1 - \frac{1}{2kq})^{2kq} \approx e^{-1}$.

Like last time, after repeating this procedure $> e(2kq)^k$ times we achieve the probability of success $\geq 1 - (1 - \frac{1}{e(2kq)^k})^{e(2kq)^k} \approx 1 - e^{-1}$

The last thing to do is to check if obtained colouring represents $(m, (T_i)_{i=1}^m)$. We guess $T_m \in (T_i)_{i=1}^m$ – set of terminals included in V_m (the big component). There are a few rules:

1. all blue connected components containing some $v \in T_m$ must lie in V_m,
2. the ones containing any $v \in T \backslash T_m$ have to be in small components, so we need to check if sizes of such blue components do not exceed q,
3. every nonempty set from $(T_i)_{i=1}^{m-1}$ has to lie inside just one small component and vice versa – every small component may contain no more than one such set,
4. blue components of size $> q$ must be the parts of V_m,
5. if we already know some small component V_i and there are some red edges connecting it with other blue component of size $\leq q$, then the second one also have to form a small component.

What can be left? Only maximal connected subgraphs containing some blue components of size $\leq q$ with no terminals, connected by red edges with some part of the large component V_m and maybe to each other. For each such $W_j \subseteq V[G]$ we need to decide if it is a part of V_m or it represents some blue components. Let us remember W_j as a pair (b_j, r_j) where b_j is a number of possible small components in W_j and r_j is a number of red edges inside. If we found k' edges from X and m' small components during reductions $1-5$ we just need to check if there is subset J of pairs satisfying:

$$\sum_{j \in J} b_j = m - 1 - m'$$

$$\sum_{j \in J} r_j \to min$$

Finding J is an instance of the KNAPSACK problem which can be solved in polynomial time. If reductions $1-5$ can be made and $\sum_{j \in J} r_j + k' \leq k$ then we have found a realization of t. Otherwise given colouring does not represent any.

The overall complexity of the algorithm (including checking every possible T_m) is $\mathcal{O}^\star(k(2kq)^k)$. □

Methods shown in this section were introduced by Chitnis, Cygan, Hajiaghayi, Pilipczuk, Pilipczuk [3].

References

[1] Jianer Chen, Yang Liu, and Songjian Lu. An improved parameterized algorithm for the minimum node multiway cut problem. *Algorithmica*, 55(1):1–13, 2009.

[2] Jianer Chen, Yang Liu, Songjian Lu, Barry O'Sullivan, and Igor Razgon. A fixed-parameter algorithm for the directed feedback vertex set problem. *J. ACM*, 55(5), 2008.

[3] Rajesh Hemant Chitnis, Marek Cygan, MohammadTaghi Hajiaghayi, Marcin Pilipczuk, and Michal Pilipczuk. Designing fpt algorithms for cut problems using randomized contractions. *CoRR*, abs/1207.4079, 2012.

[4] Elias Dahlhaus, David S. Johnson, Christos H. Papadimitriou, Paul D. Seymour, and Mihalis Yannakakis. The complexity of multiterminal cuts. *SIAM J. Comput.*, 23(4):864–894, 1994.

[5] Ken ichi Kawarabayashi and Mikkel Thorup. The minimum k-way cut of bounded size is fixed-parameter tractable. In Rafail Ostrovsky, editor, *FOCS*, pages 160–169. IEEE, 2011.

[6] Dániel Marx. Parameterized graph separation problems. *Theor. Comput. Sci.*, 351(3):394–406, 2006.