Disjunctive Minimal Separators Enumeration

Benjamin Bergougnoux[1], Mamadou Moustapha Kanté[2], and Kunihiro Wasa[3]

[1] *University of Bergen, Norway, benjamin.bergougnoux@uib.no*
[2] *Université Clermont Auvergne, LIMOS, CNRS, France, mamadou.kante@uca.fr*
[3] *National Institute of Informatics, Japan, wasa@nii.ac.jp*

Abstract

In this paper, we study the minimal disjunctive separator enumeration problem, which asks given a graph $G = (V, E)$ and a set of vertex pairs $P \subseteq V \times V$, to output all inclusion-wise minimal vertex subsets $C \subseteq V$ such that for some pair $(s, t) \in P$, there is no path between s and t in $G[V \setminus C]$. As main result, we propose the first polynomial delay polynomial space enumeration algorithm for minimal disjunctive separator problem.

1 Introduction

Given a graph $G = (V, E)$ and a vertex pair $(s, t) \in V \times V$, A *minimal separator enumeration problem* asks, given a graph $G = (V, E)$ and a pair of vertices $(s, t) \in V \times V$, to output all inclusion-wise minimal separators without duplicates. It is known that this problem can be solved in polynomial delay [3], that is, the maximum interval between two consecutive solutions, preprocessing time, and post-processing time can be bounded by polynomial in the input size. See [1] for the definition of the complexity analysis of enumeration algorithms. However, if the problem admits to receive a set P of vertex pairs $\{(s_1, t_1), \ldots, (s_k, t_k)\}$, then the problem becomes harder. There can be two variants of the minimal separator problem for vertex pairs. One is the conjunction, that is, asked to output all minimal vertex subsets such that each subset is a separator of all pairs in P. This problem is solved in [2] if one considers edge cuts. The other is the disjunction, that asks to output all minimal vertex subsets such that each subset is a separator of some pairs in P. We can easily obtain an enumeration algorithm for minimal disjunctive separators which runs in polynomial amortized time with exponential space by using the previous result [3]. However, there is no known algorithm whose complexity is polynomial delay and polynomial space for the minimal disjunctive separator enumeration problem. As the main result of this paper, we propose a novel idea for developing efficient algorithms for disjunctive enumeration problems.

2 Preliminaries

Let $G = (V, E)$ be a graph with n vertices and m edges. Vertices a and b are *connected* if there is a vertex sequence $(a = v_1, v_2, \ldots, v_\ell = b)$ such that for each $i = 1, \ldots, \ell - 1$, $(v_i, v_{i+1}) \in E$. For a vertex set C, let $G[V \setminus C] = (V \setminus C, E[V \setminus C])$, where $E[V \setminus C] = \{(x, y) \in E \mid x \notin C \wedge y \notin C\}$. A vertex set A induces a *connected component* if any two vertices in A are connected in $G[A]$ and any vertex v in $V \setminus A$ is adjacent to vertex in A. We call a vertex set $C \subseteq V$ an (a, b)-*separator* , for two distinct vertices a and b, if a and b are not connected in $G[V \setminus C]$. Let P be a set of pairs of vertices $\{(s_1, t_1), \ldots, (s_k, t_k)\}$. C is a *disjunctive separator for* P if C cuts s_j and t_j for some $j = 1, \ldots, k$. Note that some other pairs might be connected in $G[V \setminus C]$. A (disjunctive) separator C is *minimal* if there is no proposer subset C' of C such that C' is also a (disjunctive) separator.

3 Proposed algorithm

In this paper, we consider the following problem: Given a graph G and a set P of pairs of vertices $P = \{(s_1, t_1), \ldots, (s_k, t_k)\}$, enumerate all minimal disjunctive separators for P. Let $\mathcal{S}(G, P)$ be the set of solutions. Let `MC(`G, s_i, t_i`)` be an enumeration algorithm for the minimal (s_i, t_i)-separators. For simplicity, we write `MC(`G, i`)` instead of `MC(`G, s_i, t_i`)`.

We first explain the overview of our proposed algorithm `DMC` (See Algorithm 1). `DMC` starts from the first solution of `MC(`$G, 1$`)`. If the solution does not separate all other pairs $(s_2, t_2), \ldots, (s_k, t_k)$, then `DMC` outputs the solution and find the next solution of `MC(`$G, 1$`)`. Otherwise, without outputting the solution, `DMC` computes the first solution of `MC(`$G, 2$`)`. If the solution does not separate $(s_3, t_3), \ldots, (s_k, t_k)$, then `DMC` outputs the solution and find the next solution of `MC(`$G, 1$`)`. Otherwise, `DMC` computes the first solution of `MC(`$G, 3$`)` and re-

Algorithm 1: Enumeration algorithm for disjunctive minimal separator

```
1  Procedure DMC(G = (V, E), A = {(s_1, t_1), ..., (s_k, t_k)})
2      terminated ← a vector of 0s with length k;
3      DMCRec(G, 1);
4  Procedure DMCRec(G, i)
5      if there is no new solution of MC(G, i) then
6          terminated[i] ← 1;
7          if all the elements of terminated are 1 then
8              Stop the algorithm;
9          DMCRec(G, i + 1);
10     C ← a new solution of MC(G, i);
11     if C does not separate (s_{i+1}, t_{i+1}), ..., (s_k, t_k) then
12         Output C;
13         DMCRec(G, 1);
14     else
15         DMCRec(G, i + 1);
```

peats the above steps. Now, we first consider the correctness of DMC.

Theorem 1. DMC *outputs all solutions without duplicates.*

Proof. Every time a new solution is outputted, DMCRec makes a recursive call for (s_1, t_1) at line 13. In addition, after outputting all solutions, *terminated* becomes a vector consisting of k 1s. Thus, DMC stops after outputting all solutions. Let C be a solution and i_* be the maximum integer such that C separates (s_{i_*}, t_{i_*}). Then, C is outputted only by MC(G, i_*) otherwise this contradicts the construction of the algorithm. Thus, there is no duplicate output. □

Next we consider the time complexity. Let T be the delay of MC. We can easily see that DMC enumerates all solutions in $O(k(T + n + m)\,|\mathcal{S}(G, P)|)$ time in total since every solution separates at most k pairs and the connectivity check can be done in $O(n + m)$ time and $O(n)$ space by DFS traversing. More precisely, the connectivity check is done as follows: Suppose that DMC treats (s_i, t_i) and C is a current candidate solution. First DMC removes all vertices in C from G, and then executes DFS traversal for all connected components. If for some $j > i$ s_j and t_j are not connected, then DMC makes a recursive call for (s_{j+1}, t_{j+1}). Hence, the check can be done in $O(n+m)$ time. Moreover, after outputting a new solution for some pair (s_i, t_i), DMC makes a recursive call for (s_1, t_1). Therefore, since line 5 can be done in $O(T)$ time and MC(G, k) always outputs a solution, the algorithm needs $O(kT)$ delay and the same time bound for stopping after outputting the last solution. The state of the art algorithm for minimal $a-b$ separator enumeration runs in $O(nm)$ time per solution [3] with $O(n)$ space. Thus, we can obtain the following theorem:

Theorem 2. *Given a set P of k pairs of vertices,* DMC *enumerates all disjunctive minimal separators of P in G in $O(knm)$ delay with $O(kn + m)$ space.*

4 Conclusion

It should be noted that the basic idea can be applied to similar enumeration problems satisfying the following condition: (1) You have a list L of objects, (2) you want to enumerate all the objects that satisfy some property Π with at least one element of L, and (3) you have an algorithm to enumerate in polynomial delay the objects that satisfy Π with one element.

We conclude by pointing out that if one considers the conjunction of disjunctions, *i.e.*, given a set P of lists, each list consisting of pairs of vertices, and one wants to list all the minimal sets C that disconnects in each list at least one pair, then the problem is as hard as the Minimal Transversal Enumeration problem which is a long standing open problem asking for the existence of an output-polynomial algorithm for listing the minimal transversals of a hypergraph. However, the existence of an output-polynomial algorithm is open if one considers that the lists have bounded sizes. One can also consider cuts in matroids as done in [2] where hardness results were given, in particular for the conjunction of cuts in linear matroids. It is worth noticing that the conjunction of cuts in binary matroids is still open.

References

[1] David S. Johnson, Mihalis Yannakakis, and Christos H. Papadimitriou. On generating all

maximal independent sets. *Information Processing Letters*, 27(3):119 – 123, 1988.

[2] Leonid Khachiyan, Endre Boros, Konrad Borys, Khaled M. Elbassioni, Vladimir Gurvich, and Kazuhisa Makino. Generating cut conjunctions in graphs and related problems. *Algorithmica*, 51(3):239 – 263, 2008.

[3] Ken Takata. Space-optimal, backtracking algorithms to list the minimal vertex separators of a graph. *Discrete Applied Mathematics*, 158(15):1660 – 1667, 2010.