Finding Large Subgraphs

Hemanshu Kaul
joint work with G. Calinescu and C. Fernandes

Illinois Institute of Technology

The maximum subgraph problem

The maximum subgraph problem for a Graph property Π asks:

Given a graph G, find a subgraph H of G satisfying property Π that has the maximum number of edges.

The maximum induced subgraph problem

The maximum induced subgraph problem for a Graph property Π asks:

Given a graph G, find an induced subgraph H of G satisfying property Π that has the maximum number of vertices.

In other words, find the minimum number of vertices to remove from G such that the remaining subgraph satisfies the property Π.

Graph Properties

The following Graph properties are commonly considered:

- Forest (cycles are forbidden)
- Bipartite subgraph (odd cycles are forbidden)
- Planar subgraph ($\{K_5, K_{3,3}\}$-minors are forbidden)
- Complete subgraph

  There is no difference between induced and non-induced versions for this.
- Independent set

  This is meaningful only for the induced version.

All these properties are hereditary, every subgraph of a graph with property Π also has property Π.

Connectedness is an example of a property that is not hereditary.

Finding Large Subgraphs

Except for the largest Forest subgraph problem,
all these largest subgraph problems are NP-hard.

In case of the largest induced subgraph problem,
Lewis and Yannakakis (1980) showed that:

The largest induced subgraph problem is NP-hard for every non-trivial hereditary property.

What about approximate solutions?

Approximation Algorithms

Algorithm $\mathcal{A}$ for a maximization problem MAX achieves an approximation factor α if

for all inputs G, we have: $\frac{\mathcal{A}(G)}{OPT(G)} \leq \alpha$,
where $\mathcal{A}(G)$ is the value of the output generated by the algorithm $\mathcal{A}$,
and $OPT(G)$ is the optimal value.

A α-approximation algorithm for MAX is a polynomial time algorithm that achieves the approximation factor α.

To show $\mathcal{A}$ achieves approximation factor α, we typically show that: $\mathcal{A}(G) \geq L$ and $OPT(G) \leq U$, so $\alpha \geq L/U$.

Approximation Algorithms for Large Subgraphs

For example
for the largest bipartite subgraph problem:

Goemans and Williamson (1995): 0.878-approximation algorithm

Hastad (1997): If $P \neq NP$ then there is no α-approximation algorithm for any $\alpha > 0.941$.

for the largest clique subgraph problem:
Feige (2005): $O(n(loglogn)^2/(logn)^3)$-approximation algorithm

Feige et al. (1996): It is hard to approximate MAX-Clique for any constant factor.

Hastad (1999): It is hard to approximate MAX Clique within a factor $O(1/n^{\epsilon})$ for any $\epsilon > 0$

Approximation Algorithms for Large Subgraphs

For example
for the largest bipartite subgraph problem:

Goemans and Williamson (1995): 0.878-approximation algorithm

Hastad (1997): If $P \neq NP$ then there is no α-approximation algorithm for any $\alpha > 0.941$.

for the largest clique subgraph problem:
Feige (2005): $O(n(loglogn)^2/(logn)^3)$-approximation algorithm

Feige et al. (1996): It is hard to approximate MAX-Clique for any constant factor.

Hastad (1999): It is hard to approximate MAX Clique within a factor $O(1/n^{\epsilon})$ for any $\epsilon > 0$

Approximation Algorithms for Large Induced Subgraphs

Lund and Yannakakis (1993): It is hard to approximate the largest induced subgraph problem for any hereditary property.

Comparatively, very little research has been done on approximation algorithms for these problems.

For example,

For the maximum induced bipartite subgraph problem: Some results for very special classes of graphs -

Zhu (2009): $5/7$ approximation factor algorithm over triangle-free subcubic graphs.

Addario-Berry (2006): Some results for i-triangulated graphs and clique-separable graphs.

Approximation Algorithms for Large Induced Subgraphs

Lund and Yannakakis (1993): It is hard to approximate the largest induced subgraph problem for any hereditary property.

Comparatively, very little research has been done on approximation algorithms for these problems.

For example,

For the maximum induced bipartite subgraph problem: Some results for very special classes of graphs -

Zhu (2009): $5/7$ approximation factor algorithm over triangle-free subcubic graphs.

Addario-Berry (2006): Some results for i-triangulated graphs and clique-separable graphs.

Approximation Algorithms for Large Induced Subgraphs

For the maximum induced Planar subgraph problem:

Calinescu et al. (1998): There exists an $\epsilon > 0$ such that there is no $1 - \epsilon$- approximation algorithm unless $P = NP$.

Edward and Farr (2007): $3/(d+1)$-approximation algorithm on graphs of average degree at most $d \geq 4$, [in fact they find an induced series-parallel subgraph (more about these later)].

Approximation Algorithms for Large Subgraphs

For the maximum Planar subgraph problem:

Calinescu et al. (1998): There exists an $\epsilon > 0$ such that there is no $1 - \epsilon$- approximation algorithm unless $P = NP$.

Faria et al. (2004): This is true even if the input is a cubic graph.

Till 1990's a number of algorithms were studied but none gave an approximation ratio better than $1/3$, which can be trivially achieved by the Spanning Tree algorithm.

$ST(G) = n - 1$ and $OPT(G) \leq 3n - 6$

Calinescu et al. (1998): $4/9$-approximation algorithm, which is still the best known.
In fact, this algorithm generates an outerplanar subgraph (which gives a $2/3$-approximation algorithm for the maximum outerplanar graph problem).

Approximation Algorithms for Large Subgraphs

For the maximum Planar subgraph problem:

Calinescu et al. (1998): There exists an $\epsilon > 0$ such that there is no $1 - \epsilon$- approximation algorithm unless $P = NP$.

Faria et al. (2004): This is true even if the input is a cubic graph.

Till 1990's a number of algorithms were studied but none gave an approximation ratio better than $1/3$, which can be trivially achieved by the Spanning Tree algorithm.

$ST(G) = n - 1$ and $OPT(G) \leq 3n - 6$

Calinescu et al. (1998): $4/9$-approximation algorithm, which is still the best known.
In fact, this algorithm generates an outerplanar subgraph (which gives a $2/3$-approximation algorithm for the maximum outerplanar graph problem).

Approximation Algorithms for Large Subgraphs

For the maximum Planar subgraph problem:

Calinescu et al. (1998): There exists an $\epsilon > 0$ such that there is no $1 - \epsilon$- approximation algorithm unless $P = NP$.

Faria et al. (2004): This is true even if the input is a cubic graph.

Till 1990's a number of algorithms were studied but none gave an approximation ratio better than $1/3$, which can be trivially achieved by the Spanning Tree algorithm.

$ST(G) = n - 1$ and $OPT(G) \leq 3n - 6$

Calinescu et al. (1998): $4/9$-approximation algorithm, which is still the best known.
In fact, this algorithm generates an outerplanar subgraph (which gives a $2/3$-approximation algorithm for the maximum outerplanar graph problem).

Large Series-Parallel Subgraphs

Planar graphs are characterized as having no $\{K_5, K_{3,3}\}$ minors or subdivisions.

Outerplanar graphs are characterized as having no $\{K_4, K_{2,3}\}$ minors or subdivisions.

How about subgraphs with no K_4 minors or subdivisions? These will be planar but not outerplanar.

These are Series-Parallel graphs.

H is a minor of G if a graph isomorphic to H can be obtained from G by contracting some edges, deleting some edges, and deleting some isolated vertices.

Large Series-Parallel Subgraphs

Planar graphs are characterized as having no $\{K_5, K_{3,3}\}$ minors or subdivisions.

Outerplanar graphs are characterized as having no $\{K_4, K_{2,3}\}$ minors or subdivisions.

How about subgraphs with no K_4 minors or subdivisions? These will be planar but not outerplanar.

These are Series-Parallel graphs.

H is a minor of G if a graph isomorphic to H can be obtained from G by contracting some edges, deleting some edges, and deleting some isolated vertices.

Large Series-Parallel Subgraphs

Series-Parallel graphs are characterized as:

- No K_4 minor or subdivision.
- Arises from a forest by adding parallel edges, subdividing edges, and at the end removing any parallel edges to keep the graph simple.
- tree width ≤ 2 (subgraph of 2-tree).

Large Series-Parallel Subgraphs

The maximum Series-Parallel subgraph problem is *NP*-hard.

Since, the number of edges of a Series-Parallel graph on n vertices is bounded above by $2n - 3$,
the spanning tree algorithm gives a $1/2$-approximation algorithm.

Can we do better?

Large Series-Parallel Subgraphs

The maximum Series-Parallel subgraph problem is *NP*-hard.

Since, the number of edges of a Series-Parallel graph on n vertices is bounded above by $2n - 3$,
the spanning tree algorithm gives a $1/2$-approximation algorithm.

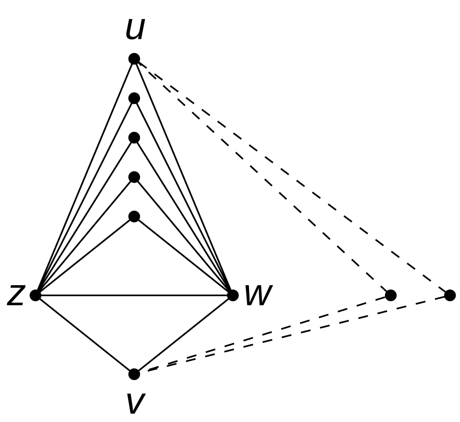
Can we do better?

New Results

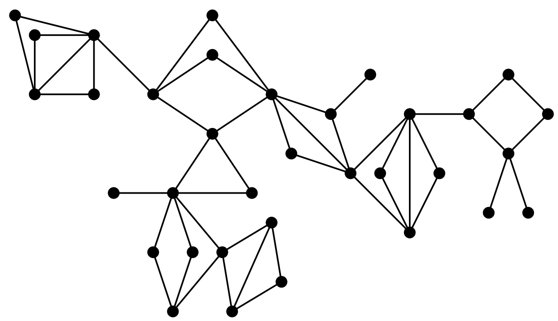
Calinescu, Fernandes, K. (2009): $7/12$ approximation algorithm for the maximum Series-Parallel subgraph problem.

The output is a spruce structure: a graph each of whose blocks is either a spruce or an edge.
A spruce consists of two *base* vertices and at least one *tip* vertex, in which each tip vertex is adjacent to exactly the two base vertices.

(a)



(b)



New Results

Calinescu, Fernandes, K. (2009): The maximum spruce structure would give a $2/3$ approximation for the maximum Series-Parallel subgraph.

Calinescu, Fernandes, K. (2009): The maximum spruce structure subgraph problem is *NP*-hard.

New Results

Calinescu, Fernandes, K. (2009): The maximum spruce structure would give a $2/3$ approximation for the maximum Series-Parallel subgraph.

Calinescu, Fernandes, K. (2009): The maximum spruce structure subgraph problem is NP-hard.

New Ideas

Comparison with previous algorithms for Planar subgraphs:

- Unlike earlier algorithms, the subgraph we generate is not a tree or an outerplanar graph.
- Unlike earlier algorithms, we have to allow blocks of unbounded size in our subgraph.
- Unlike earlier algorithms, we sometimes have to shrink or throw away previously selected blocks.

New Ideas

Comparison with previous algorithms for Planar subgraphs:

- Unlike earlier algorithms, the subgraph we generate is not a tree or an outerplanar graph.
- Unlike earlier algorithms, we have to allow blocks of unbounded size in our subgraph.
- Unlike earlier algorithms, we sometimes have to shrink or throw away previously selected blocks.

New Ideas

Comparison with previous algorithms for Planar subgraphs:

- Unlike earlier algorithms, the subgraph we generate is not a tree or an outerplanar graph.
- Unlike earlier algorithms, we have to allow blocks of unbounded size in our subgraph.
- Unlike earlier algorithms, we sometimes have to shrink or throw away previously selected blocks.

New Ideas

Comparison with previous algorithms for Planar subgraphs:

- Unlike earlier algorithms, the subgraph we generate is not a tree or an outerplanar graph.
- Unlike earlier algorithms, we have to allow blocks of unbounded size in our subgraph.
- Unlike earlier algorithms, we sometimes have to shrink or throw away previously selected blocks.

New Ideas

Unlike earlier algorithms, we have to allow blocks of unbounded size in our subgraph.

If the input graph is a *complete spruce* (spruce with an edge between the base vertices) with $n-2$ tips, then any algorithm that only generates blocks of size at most k would result in an output with a total $n+k-3$ edges.

With large n and fixed k, this is only a $1/2$-approximation.

New Ideas

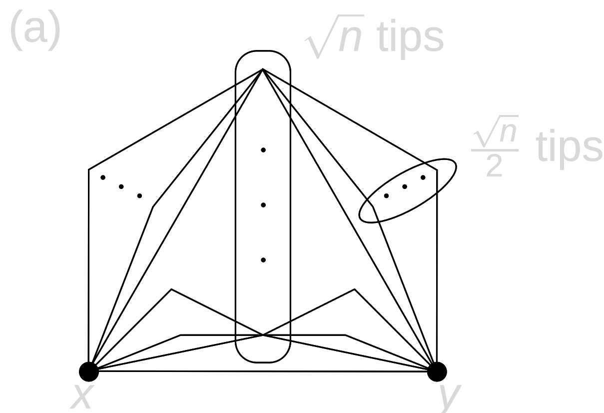
Unlike earlier algorithms, we have to allow blocks of unbounded size in our subgraph.

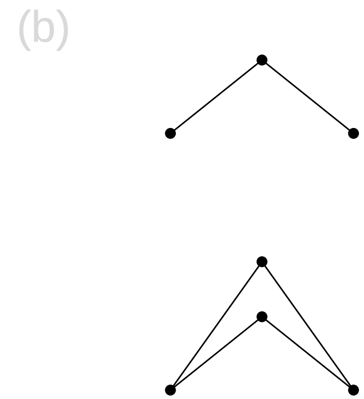
If the input graph is a *complete spruce* (spruce with an edge between the base vertices) with $n-2$ tips, then any algorithm that only generates blocks of size at most k would result in an output with a total $n+k-3$ edges.

With large n and fixed k, this is only a $1/2$-approximation.

New Ideas

Unlike earlier algorithms, we sometimes have to shrink or throw away previously selected blocks.





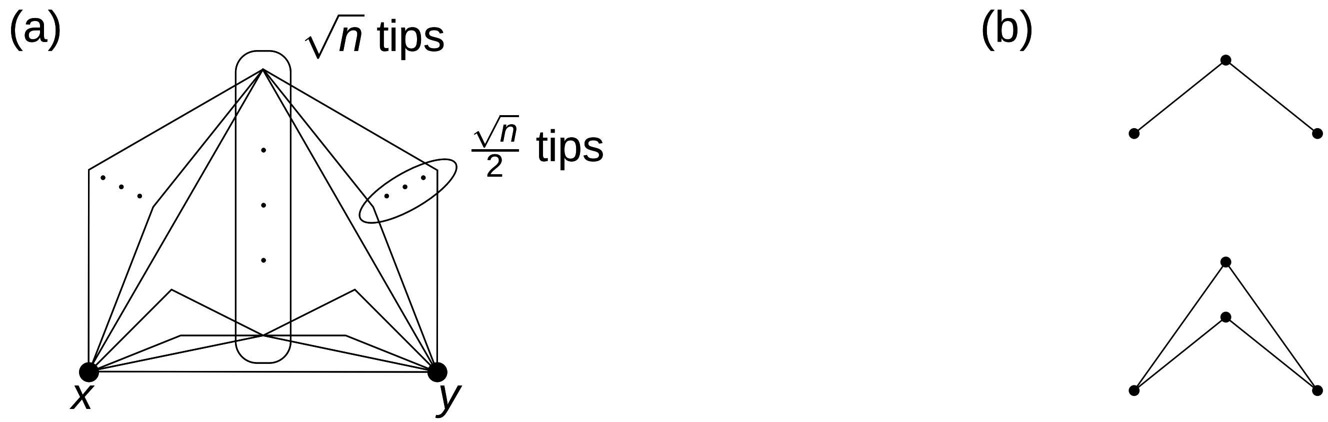
The optimum has n vertices and $2n-3$ edges.

A spruce with base vertices x and y and $\sqrt{n}$ tips. For each of its tips v, there are two complete spruces, one with base vertices x and v, and the other with base vertices v and y, each with $\sqrt{n}/2$ tips.

If an algorithm mistakenly (or greedily) selects the spruce with base vertices x and y, then it cannot add any more spruces and it ends up with about $n+\sqrt{n}$ edges — asymptotically not better than a $1/2$-approximation.

New Ideas

Unlike earlier algorithms, we sometimes have to shrink or throw away previously selected blocks.



The optimum has n vertices and $2n-3$ edges.

A spruce with base vertices x and y and $\sqrt{n}$ tips. For each of its tips v, there are two complete spruces, one with base vertices x and v, and the other with base vertices v and y, each with $\sqrt{n}/2$ tips.

If an algorithm mistakenly (or greedily) selects the spruce with base vertices x and y, then it cannot add any more spruces and it ends up with about $n+\sqrt{n}$ edges — asymptotically not better than a $1/2$ -approximation.

The Algorithm - Preliminaries

$gain(S)$:= cyclomatic number

For complete spruces, *gain* is the number of tips; adjusted gain $\widehat{gain} := gain$.

For incomplete spruces, *gain* is one less than the number of tips; adjusted gain $\widehat{gain} := gain - 1$.

The Algorithm - Underlying Idea

We maintain Q, a collection of spruces.

What we add: Spruces with tips that are isolated vertices.
Let $v_1, v_2, \ldots, v_k$ be all vertices isolated in Q that are adjacent in G to both x and y.

If $k \geq 1$, let $S_Q(x,y)$ be the spruce with base vertices x and y, tips $v_1, v_2, \ldots, v_k$, and the edge xy if it exists in G. *Add* $S_Q(x,y)$ to Q.

What do we remove: For each component C of Q, the algorithm keeps a *weighted tree* T_C whose vertex set is $V(C)$ and edge set is as follows: For each spruce S in C with base vertices x and y, and tips $v_1, v_2, \ldots, v_k$, there is an edge xy with weight $\widehat{gain}$ in T_C and edges xv_i with weight 1 for $i = 1, \ldots, k$.

$index_Q(x,y)$ is an edge in T_C of minimum weight in the path in T_C from x to y. Let x' and y' be the endpoints of $index_Q(x,y)$, and C be the component of Q containing x, x', y, and y'. Let S' be the spruce in Q containing x' and y'. *If* x' and y' are the base vertices of S', then remove S' from Q.

The Algorithm - Underlying Idea

We maintain Q, a collection of spruces.

What we add: Spruces with tips that are isolated vertices.
Let $v_1, v_2, \dots, v_k$ be all vertices isolated in Q that are adjacent in G to both x and y.

If $k \geq 1$, let $S_Q(x, y)$ be the spruce with base vertices x and y, tips $v_1, v_2, \dots, v_k$, and the edge xy if it exists in G. *Add* $S_Q(x, y)$ to Q.

What do we remove: For each component C of Q, the algorithm keeps a *weighted tree* T_C whose vertex set is $V(C)$ and edge set is as follows: For each spruce S in C with base vertices x and y, and tips $v_1, v_2, \dots, v_k$, there is an edge xy with weight $\widehat{gain}$ in T_C and edges xv_i with weight 1 for $i = 1, \dots, k$.

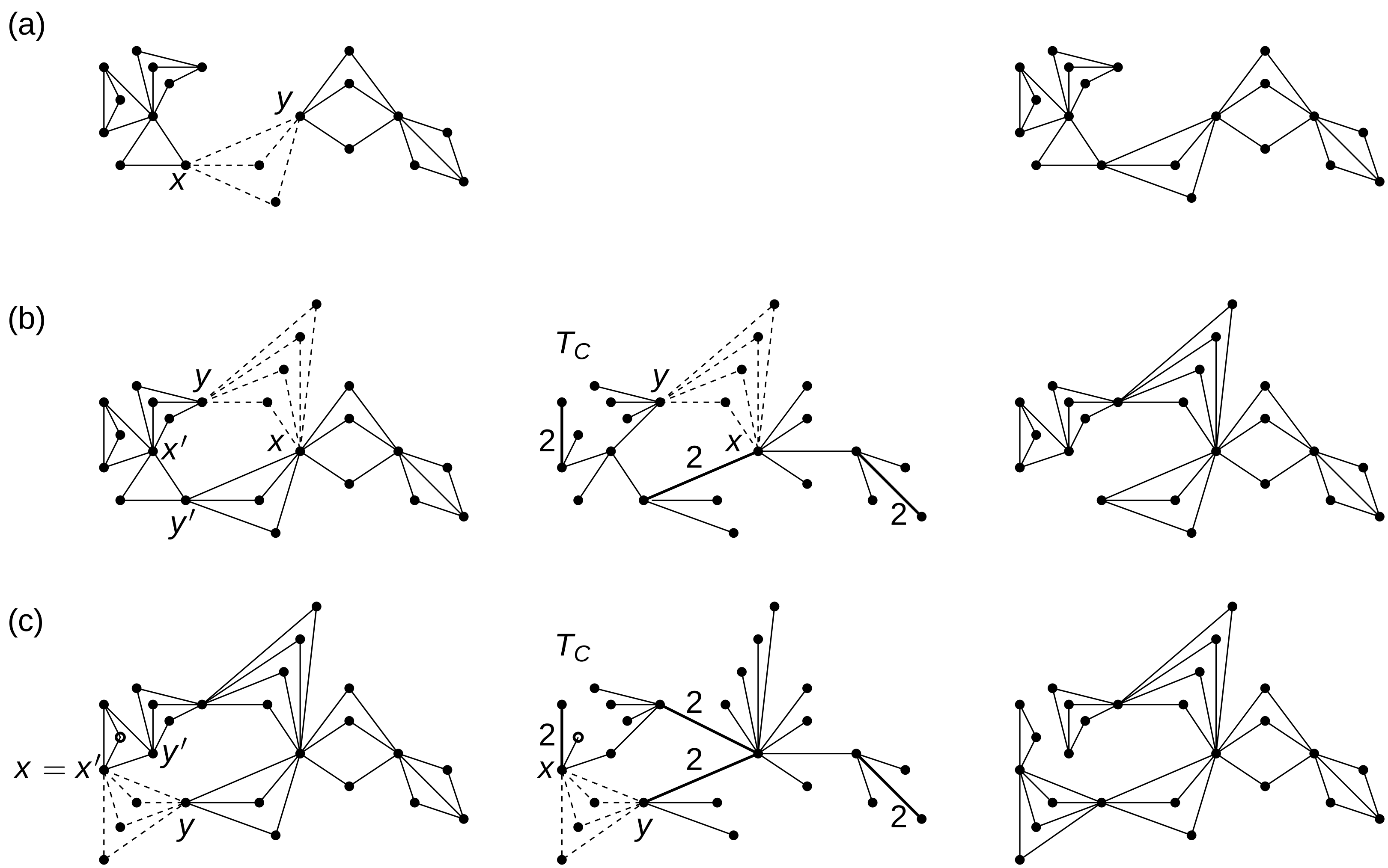
$index_Q(x, y)$ is an edge in T_C of minimum weight in the path in T_C from x to y. Let x' and y' be the endpoints of $index_Q(x, y)$, and C be the component of Q containing x, x', y, and y'. Let S' be the spruce in Q containing x' and y'. *If* x' and y' are the base vertices of S', then remove S' from Q.

The Algorithm

CONSTRUCT-SPRUCE-STRUCTURE (G)

$Q \leftarrow \emptyset$
while there are x and y such that $S_Q(x,y)$ is defined
  and $\widehat{gain}(S_Q(x,y)) > w(index_Q(x,y))$ **do**
  if $index_Q(x,y)$ is undefined
    then $Q \leftarrow Q \cup \{S_Q(x,y)\}$
    else let x' and y' be the endpoints of $index_Q(x,y)$
      let S' be the spruce in Q containing x' and y'
      $Q \leftarrow Q \setminus \{S'\} \cup \{S_Q(x,y)\}$
      if x' or y' is a tip of S'
        then let z be between x', y', a tip of S'
          let $\{e,f\}$ be the edges of S' touching z
          $S \leftarrow S' - \{e,f\}$
          if S is not degenerate nor single edge
            then $Q \leftarrow Q \cup \{S\}$
add bridges to Q to obtain a connected spanning subgraph of G
return Q

Local improvement examples



Running Time Analysis

If $gain(Q)$ increased in every iteration, then it would have been easy to conclude that the algorithm runs a polynomial number of iterations.
The gain of Q never decreases and, in the iterations in which the gain of Q is same, the number of components increases.

Define $\Phi(Q) = 3\,gain(Q) + c(Q)$, where $c(Q)$ is the number of components of Q when Q is seen as a spanning subgraph of G.

We prove: Every iteration of the algorithm increases the parameter Φ.

$gain(Q) \le (2n-3) - (n-1) = n-2$,
so $\Phi(Q)$ is bounded by $3(n-2) + n = 4n-6$.

Each iteration can be easily implemented in polynomial time:
$O(n^2)$ pairs x, y for which $S_Q(x, y)$ must be computed and, if possible, used in updating Q.

Running Time Analysis

If $gain(Q)$ increased in every iteration, then it would have been easy to conclude that the algorithm runs a polynomial number of iterations.
The gain of Q never decreases and, in the iterations in which the gain of Q is same, the number of components increases.

Define $\Phi(Q) = 3\,gain(Q) + c(Q)$, where $c(Q)$ is the number of components of Q when Q is seen as a spanning subgraph of G.

We prove: Every iteration of the algorithm increases the parameter Φ.

$gain(Q) \leq (2n-3) - (n-1) = n-2$,
so $\Phi(Q)$ is bounded by $3(n-2) + n = 4n-6$.

Each iteration can be easily implemented in polynomial time:
$O(n^2)$ pairs x, y for which $S_Q(x, y)$ must be computed and, if possible, used in updating Q.

Running Time Analysis

If $gain(Q)$ increased in every iteration, then it would have been easy to conclude that the algorithm runs a polynomial number of iterations.
The gain of Q never decreases and, in the iterations in which the gain of Q is same, the number of components increases.

Define $\Phi(Q) = 3\,gain(Q) + c(Q)$, where $c(Q)$ is the number of components of Q when Q is seen as a spanning subgraph of G.

We prove: Every iteration of the algorithm increases the parameter Φ.

$gain(Q) \leq (2n-3) - (n-1) = n-2$,
so $\Phi(Q)$ is bounded by $3(n-2) + n = 4n-6$.

Each iteration can be easily implemented in polynomial time:
$O(n^2)$ pairs x, y for which $S_Q(x, y)$ must be computed and, if possible, used in updating Q.

Approximation ratio ideas - to beat $1/2$

- If significantly many vertices in our structure, we win.
- If OPT has significantly less than $2n$ edges, we win.
- If none of the above, the spruces of OPT have significant $\widehat{gain}$.

Approximation ratio ideas - to beat $1/2$

From OPT, construct *w*eighted Series Parallel graph with $\widehat{gain}$ on edges.

Compare to our weighted forest.

We have a maximum spanning forest in the union of the two graphs!

Thus our $\widehat{gain}$ is $1/2$ of what that of OPT.

Therefore we have significant $\widehat{gain}$.

New Questions

- Weighted maximum Series-Parallel subgraph problem.
- Maximum induced Series-parallel subgraph problem.
- For fixed r, maximum K_r-minor-free subgraph problem.
- In particular, maximum K_5-minor-free subgraph problem.
  Number of edges in such a graph are $\leq 3n - 6$.
  Also, the structural characterization is known - constructed from copies of planar graphs and Wagner's graph by gluing over k-cliques for $k \leq 3$.
- For fixed r, maximum subgraph of tree width $\leq r$.
- In particular, maximum subgraph of tree width ≤ 3.
  Number of edges in such a graph are $\leq 3n - 6$.
  Also, such graphs have no minors from $\{K_5, \mathit{Wagner}, \text{two other graphs}\}$.

New Questions

- Weighted maximum Series-Parallel subgraph problem.
- Maximum induced Series-parallel subgraph problem.
- For fixed r, maximum K_r-minor-free subgraph problem.
- In particular, maximum K_5-minor-free subgraph problem.
  Number of edges in such a graph are $\leq 3n - 6$.
  Also, the structural characterization is known - constructed from copies of planar graphs and Wagner's graph by gluing over k-cliques for $k \leq 3$.
- For fixed r, maximum subgraph of tree width $\leq r$.
- In particular, maximum subgraph of tree width ≤ 3.
  Number of edges in such a graph are $\leq 3n - 6$.
  Also, such graphs have no minors from $\{K_5, Wagner, \text{two other graphs}\}$.

New Questions

- Weighted maximum Series-Parallel subgraph problem.
- Maximum induced Series-parallel subgraph problem.
- For fixed r, maximum K_r-minor-free subgraph problem.
- In particular, maximum K_5-minor-free subgraph problem.
  Number of edges in such a graph are $\leq 3n - 6$.
  Also, the structural characterization is known - constructed from copies of planar graphs and Wagner's graph by gluing over k-cliques for $k \leq 3$.
- For fixed r, maximum subgraph of tree width $\leq r$.
- In particular, maximum subgraph of tree width ≤ 3.
  Number of edges in such a graph are $\leq 3n - 6$.
  Also, such graphs have no minors from $\{K_5, Wagner, \text{two other graphs}\}$.

New Questions

- Weighted maximum Series-Parallel subgraph problem.
- Maximum induced Series-parallel subgraph problem.
- For fixed r, maximum K_r-minor-free subgraph problem.
- In particular, maximum K_5-minor-free subgraph problem.
  Number of edges in such a graph are $\leq 3n - 6$.
  Also, the structural characterization is known - constructed from copies of planar graphs and Wagner's graph by gluing over k-cliques for $k \leq 3$.
- For fixed r, maximum subgraph of tree width $\leq r$.
- In particular, maximum subgraph of tree width ≤ 3.
  Number of edges in such a graph are $\leq 3n - 6$.
  Also, such graphs have no minors from $\{K_5, Wagner, \text{two other graphs}\}$.

New Questions

- Weighted maximum Series-Parallel subgraph problem.
- Maximum induced Series-parallel subgraph problem.
- For fixed r, maximum K_r-minor-free subgraph problem.
- In particular, maximum K_5-minor-free subgraph problem.
  Number of edges in such a graph are $\leq 3n - 6$.
  Also, the structural characterization is known - constructed from copies of planar graphs and Wagner's graph by gluing over k-cliques for $k \leq 3$.
- For fixed r, maximum subgraph of tree width $\leq r$.
- In particular, maximum subgraph of tree width ≤ 3.
  Number of edges in such a graph are $\leq 3n - 6$.
  Also, such graphs have no minors from $\{K_5, \mathit{Wagner}, \text{two other graphs}\}$.

New Questions

- Weighted maximum Series-Parallel subgraph problem.
- Maximum induced Series-parallel subgraph problem.
- For fixed r, maximum K_r-minor-free subgraph problem.
- In particular, maximum K_5-minor-free subgraph problem.
  Number of edges in such a graph are $\leq 3n - 6$.
  Also, the structural characterization is known - constructed from copies of planar graphs and Wagner's graph by gluing over k-cliques for $k \leq 3$.
- For fixed r, maximum subgraph of tree width $\leq r$.
- In particular, maximum subgraph of tree width ≤ 3.
  Number of edges in such a graph are $\leq 3n - 6$.
  Also, such graphs have no minors from $\{K_5, Wagner, \text{two other graphs}\}$.