A Polynomial Time Algorithm for Longest Paths in Biconvex Graphs

ESHA GHOSH

Joint work with

N. S. Narayanaswamy and C. Pandu Rangan

Dept of Computer Science and Engineering
IIT Madras, Chennai 600036, India

February 20^{th}, 2011

Outline

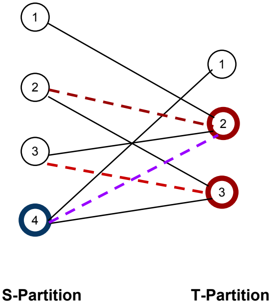
Outline

Doubly Convex or Biconvex Graphs

Definition (Glover-1967)

A bipartite graph $G = (S \cup T, E)$ is *doubly convex* or *biconvex* if $\exists$ a numbering $1, 2, \ldots, |S|$ of the vertices in S and a numbering of the vertices $1, 2, \ldots, |T|$ in T such that $\forall v \in S \cup T$, $N(v)$ is a set of consecutive integers.

Illustration



Outline

The Longest Path problem

The problem

The longest path problem is the problem of finding a simple path of maximum length in a given graph.

Importance

- The well-known Travelling Salesman problem and Hamiltonian Path problem are special cases of Longest Path problem.
- The longest path in program activity graph is known as critical path, which represents the sequence of program activities that take the longest time to execute.

Outline

Previous Work done

Polynomial Solution for longest path problem on some graph classes

- Polynomial time algorithms for longest path exist for graph classes like Trees, Cacti, Ptolemic graphs, Bipartite Permutation graphs, proposed by Uehera et al.
- Recently polynomial time solutions for the longest path problem is proposed by Ioannidou et al. on interval graphs and cocomparability graphs.

Outline

Our Contribution

Workdone

- We have proposed a $O(n^6)$ time algorithm to find the longest path on biconvex graphs, where n is the number of vertices of the input graph.
- We have used Dynamic Programming approach.

The Ordering

Ordering the vertices

- Let $\pi_1 = (s_1, s_2, \cdots, s_{|S|})$ be the labelled vertices of partition S, and $\pi_2 = (t_1, t_2, \cdots, t_{|T|})$ be that of partition T
- Initiliaze $\sigma_S = \pi_1$.
- Update σ_S as follows: For all t_i in π_2, insert t_i immediately after its rightmost neighbor.

Illustration

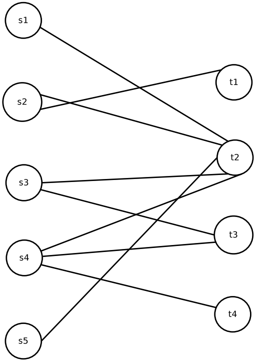


Figure: $\pi_1 = (s_1, s_2, s_3, s_4, s_5)$ and $\pi_2 = (t_1, t_2, t_3, t_4)$
$\sigma_s = (s_1, s_2, t_1, s_3, s_4, t_3, t_4, s_5, t_2)$

Monotonic Path

Definition

A *S-Monotone* path of a Biconvex graph $G = (S \cup T, E)$ is a simple path $P = \{s_{\alpha_1}, t_{\beta_1}, s_{\alpha_2}, \ldots, t_{\beta_{j-1}}, s_{\alpha_j}, t_{\beta_j}\}$ such that $s_{\alpha_k} \prec_{\sigma_S} s_{\alpha_{k+1}} \forall$ k $\ni 1 \leq$k$\leq j$.

Symmetrically, we define T-Monotone path.

Lemma

Lemma

Let

- $P = \{s_{\alpha_1}, t_{\beta_1}, s_{\alpha_2}, t_{\beta_2} \dots s_{\alpha_{j-1}}, t_{\beta_{j-1}}, s_{\alpha_j}, t_{\beta_j}\}$ *be a simple path of a biconvex graph* $G = (S \cup T, E)$.
- *Let* P_{max} *denote the longest S-S sub path of* P.

Then, the vertices on the path P_{max} *can be reordered to get a path* P'_{max} *on the same set of vertices, which is S-Monotone.*

Illustration

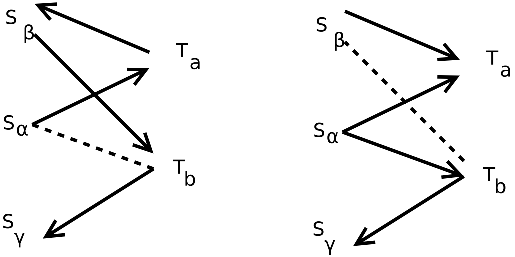


Figure: Case $\beta \prec \alpha \prec \gamma$
(a)*Non S-Monotone* Path (b)*S-Monotone* Path

Outline

Algorithm Overview

First Part of the Algorithm

Given the ordering ordering $\sigma_S = (u_1, u_2, \cdots, u_n)$, for all s_i, s_j, where, $1 \leq i < j \leq |S|$, do the following:

- Choose the subsequence $\sigma_{Sij} = (u_k, u_{k+1}, \cdots, u_m)$ such that:
  - u_k is the vertex s_i
  - u_m is s_j, if s_{j+1} immediately suceeds s_j in σ_S.
  - Otherwise, u_m is the rightmost T-vertex that lies between s_j and s_{j+1} in σ_S.
- Run the "Longest Path" routine and remember the maximum path length obtained over these iterations and all the paths of that maximum length.

Illustration

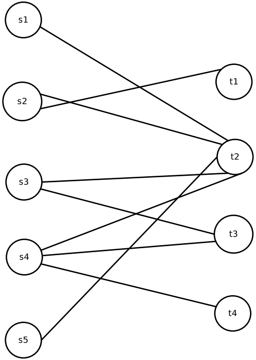


Figure: $\sigma_S = (s_1, s_2, t_1, s_3, s_4, t_3, t_4, s_5, t_2)$

Illustration

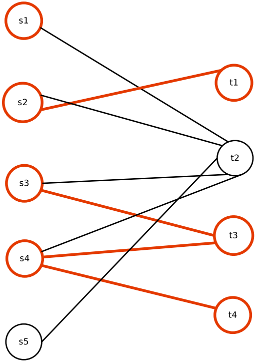


Figure: $\sigma_S = (s_1, s_2, t_1, s_3, s_4, t_3, t_4, s_5, t_2)$ and $\sigma_{S14} = (s_1, s_2, t_1, s_3, s_4, t_3, t_4)$
Longest S-Bimonotone Path is $P = \{s_3, t_3, s_4, t_4\}$

Algorithm Overview(Cont.)

Second Part of the Algorithm

Symmetric to Part 1, this part is executed for vertices of T-partition with the initial ordering $\sigma_T = (u_1, u_2, \cdots, u_n)$.

Algorithm Overview(Cont.)

Third Part of the Algorithm

- Choose the maximum of the two lengths obtained as output from the first and second part of the algorithms. Let us denote this length as max
- For all paths of length max, check if any of the end vertices have unvisited neighbor.
- If such a neighbor exists, extend the path till that neighbor and declare it as a longest path and $max + 1$ as longest path length.
- Else declare max as the longest path length and all paths of this lenth as longest paths.

Some Constructs and Notations

Notations

- For every pair of indices i, j such that $1 \leq i \leq j \leq n$ the graph $G(i, j)$ denotes the subgraph of G induced by the set $\{u_i, u_{i+1}, ..., u_j\}$.
- A path P is called S-bimonotone if P is a S-Monotone path with both endpoints in S-partiton.
- $P(u_k; i, j)$ denotes the longest S-bimonotone path of $G(i, j)$ with u_k as its right endpoint and $l(u_k; i, j)$ denotes the number of vertices on $P(u_k; i, j)$.

The "Longest Path" Routine

Description

1. Iterate through all the vertices of σ_{S1n} from indices 1 to n
2. Initialize $P(u_i; i, i)$ with u_i and $l(u_i; i, i)$ with 1.
3. For all u_k , $i \leq k \leq j-1$, which are S-vertices, initialize $P(u_k; i, j)$ with $P(u_k; i, j-1)$ and $l(u_k; i, j)$ with $l(u_k; i, j-1)$
4. If u_j is also a S-vertex , intialize $l(u_j; i, j)$ with 1 and $P(u_k; i, j)$ with u_j
5. If u_j is a T-vertex and $i \leq f(u_j)$ then execute $process(G(i, j))$
6. Compute the $\max\{(u_k; 1; n) : u_k \in S(G)\}$ and the corresponding path $P(u_k; 1; n)$.

The $process(G(i,j))$ subroutine

$process(G(i,j))$

- For $y = f(u_j)+1$ to $j-1$, iterate through $x = f(u_j)$ to $y-1$ if both u_x and u_y are S-vertices, then update the path as follows:
  - $w_1 \leftarrow l(u_x; i, j-1)$; $P_1' \leftarrow P(u_x; i, j-1)$
  - $w_2 \leftarrow l(u_y; x+1, j-1)$; $P_2' \leftarrow P(u_y; x+1, j-1)$

  If $w1 + w2 + 1 > l(u_y; i, j)$, then
  - $l(u_y; i, j) \leftarrow w1 + w2 + 1;$
  - $P(u_y; i, j) = (P_1', u_j, P_2');$

Return the value $\{l(u_k; i, j)\}$ and the path $\{P(u_k; i, j), \forall u_k \in S(G(f(u_j)+1, j-1))\}$

Illustration

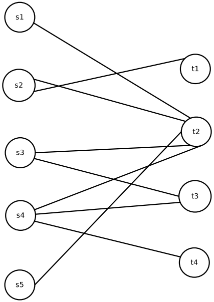


Figure: $\sigma_s = (s_1, s_2, t_1, s_3, s_4, t_3, t_4, s_5, t_2)$

Illustration

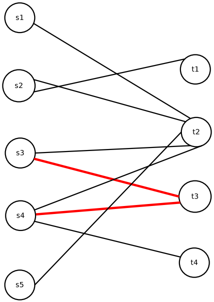


Figure: Path getting updated at call $process(G(4,6))$ and $x = 4, y = 5$

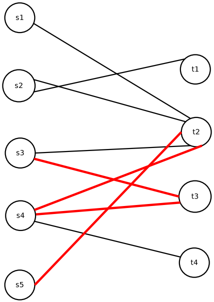


Figure: Path getting updated at call $process(G(1,9))$ with $x = 5, y = 8$

Algorithm Correctness

Claim

At least one of the paths, obtained from First and Second part of the Algorithm is extendible if and only if a longer path exists.

Proof.

- If possible, let a longer path P_{new} of length x exist.
- This path is either a ST or a TS path.
- Due to the Lemma, we know for the longest S-S subpath of every path, we get a S-Monotone path on the same set of vertices.
- Length of the longest S-S subpath of P_{new} is $x - 1$.
  - $max < x - 1 \Rightarrow$ contradiction to the correctness of the "Longest path" routine.
  - $max > x - 1 \Rightarrow P_{new}$ is not the longest path.

$\square$

Time Complexity

Complexity Analysis

- Generating the ordering σ_S and σ_T will take $O(|S||T|)$ time.
- The subroutine process() takes O(n^2) time and is executed at most once for each subgraph G(i, j) of G. Hence takes O(n^4) time.
- Since the routine "Longest Path" is called for each ordered pair s_i, s_j (and t_i, t_j), and there can be $O(n^2)$ such ordered pairs, so the total running time is $O(n^6)$.

THANK YOU