A Polynomial Time Algorithm for Longest Paths in Biconvex Graphs

Esha Ghosh, N.S. Narayanaswamy, and C. Pandu Rangan
Dept. of Computer Science and Engineering
IIT Madras, Chennai 600036, India

Abstract. The longest path problem is the problem of finding a simple path of maximum length in a graph. Polynomial solutions for this problem are known only for special classes of graphs, while it is NP-hard on general graphs. In this paper we are proposing 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.

Keywords: Longest path problem, biconvex graphs, polynomial algorithm, complexity, dynamic programming.

1 Introduction

One of the most well studied problem in graph theoretic literature is the Hamiltonian Path problem. The Hamiltonian Path problem, is to determine whether a graph admits a Hamiltonian Path, i.e., a simple path in which every vertex of the graph appears exactly once. The Hamiltonian Path problem is actually a special case of the well known Longest Path problem. The Longest path problem is to find the vertex-simple longest path in a graph. Longest path algorithms find various applications across diverse fields. 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. Longest path algorithm is required to calculate critical paths. The well-known Travelling Salesman problem is also a special case of Longest Path problem. As Hamiltonian path problem is NP-Hard on general graphs, so obviously solving the longest path problem is also intractable. In fact, it has been shown that there is no polynomial-time constant-factor approximation algorithm for the longest path problem unless P=NP [6]. Therefore, it is meaningful to investigate the Longest Path problem on special graph classes, for which Hamiltonian Path problem is polynomial time solvable.

The Hamiltonian Path problem remains NP-complete even when restricted to some small classes of graphs such as split graphs, chordal bipartite graphs, strongly chordal graphs, circle graphs, planar graphs, and grid graphs [7, 8]. However it becomes polynomial time solvable on interval graphs [2], co-comparability graphs, circular-arc graphs and convex bipartite graphs [6]. But surprisingly, there are a very few known polynomial time solutions for the Longest Path problems. The
small graph classes on which Longest path problem has a polynomial time solution includes weighted trees, block graphs, cacti \[8\] \[10\] bipartite permutation graphs \[10\] and ptolmaic graphs. Recently polynomial time solutions have been proposed by Ioannidou et al. on interval graphs \[6\] and co-comparability graphs \[4\]. However, status of the Longest path problem was left open for convex and biconvex graphs in \[3\].

Biconvex (and hence, Convex) graphs are superclass of bipartite permutation graphs \[3\] on which the Longest Path problem is polynomial time solvable, and subclass of chordal bipartite graphs \[7\], on which, this problem is NP-Complete. Naturally it is interesting to investigate the status of the problem on Biconvex graph class. Biconvex graphs have various industrial and other practical applications also. A bipartite graph $G=(S, T, E)$ is convex on the vertex set $S$ if $S$ can be ordered so that for each element $t$ in the vertex set $T$ the elements of $S$ connected to $t$ form an interval of $S$; this property is called the adjacency  property. $G$ is biconvex if it is convex on both $S$ and $T$ \[1\].

In this paper, we have proposed a $O(n^3)$ time algorithm for the Longest Path problem extending the approach presented by Ioannidou et al. in \[6\]. We have used the standard Dynamic Programming paradigm.

We will discuss about the preliminaries in Section 2. Then we will briefly introduce the vertex ordering and some notations used in the algorithm in section 3. Then we will present the algorithm and prove its correctness in section 4. We will discuss the complexity analysis in section 5 and then we will conclude in section 6.

2 Preliminaries

A graph $G = (V,E)$ consists of a finite set $V(G)$ of vertices and a collection $E(G)$ of 2-element subsets of $V(G)$ called edges. An undirected edge is a pair of distinct vertices $u,v \in V(G)$, and is denoted by $uv$. We say that the vertex $u$ is adjacent to the vertex $v$ or, equivalently, the vertex $u$ sees the vertex $v$, if there is an edge $uv \in E(G)$. Let $S$ be a set of vertices of a graph $G$. Then, the cardinality of the set $S$ is denoted by $|S|$ and the subgraph of $G$ induced by $S$ is denoted by $G[S]$. The set $N(v) = \{ u \in V(G) : uv \in E(G) \}$ is called the neighborhood of the vertex $v \in V$ in $G$, the set $N[v] = N(v) \cup \{ v \}$ is called the closed neighborhood of the vertex $v \in V(G)$.

A simple path of a graph $G$ is a sequence of distinct vertices $v_1, v_2, \ldots, v_k$ such that $v_i v_{i+1} \in E(G) \forall 1 \leq i \leq k-1$ and is denoted by $(v_1, v_2, \ldots, v_k)$. Throughout the paper all paths considered are simple. We define the length of the path $P$ to be the number of edges in $P$ and $V(P)$ to be the set of vertices in the path $P$.

$v_k$ is referred to as the right endpoint of the path $P = (v_1, v_2, \ldots, v_k)$. Moreover, let $P = (v_1, v_2, \ldots, v_{i-1}, v_i, v_{i+1}, \ldots, v_j, v_{j+1}, \ldots, v_k)$ and $P_0 = (v_i, v_{i+1}, \ldots, v_j)$ be two paths of a graph. Sometimes, we shall denote the path $P$ by $P=(v_1, v_2, \ldots, v_{i-1}, P_0, v_{j+1}, \ldots, v_k)$. We define the sub path of $P$ to be a path $P'$ such that $V(P') \subseteq V(P)$ and $E(P')$ is the same as $E(P)$.

Let $\pi$ be some ordering given on the indices of the vertices. We use the notation $v_i \prec_\pi v_j$ to denote that index $i$ occurs before index $j$ in the given ordering $\pi$. 


3 Biconvex Orderings and Monotone Paths

In this section, we first define two orderings \( \sigma_S \) and \( \sigma_T \) on the indices of the vertices of a biconvex graph and then define Monotonicity of a path. We assume that \( \pi_1 \) and \( \pi_2 \) are the labellings of the vertices of \( S \) and \( T \) partition respectively (in increasing order of indices), preserving the adjacency property. Using these two orderings, \( \pi_1 \) and \( \pi_2 \), we create \( \sigma_S \) and \( \sigma_T \) as follows.

3.1 Ordering of Vertices

Let \( \pi_1 = (s_1, s_2, ..., s_p) \) be the vertices of partition \( S \), and similarly, \( \pi_2 = (t_1, t_2, ..., t_q) \) be the vertices of partition \( T \), where \( |S| = p \) and \( |T| = q \) and total number of vertices \( n = p + q \). We introduce an ordering \( \sigma_S \) as follows:

Initially set \( \sigma_S = \pi_1 \).

1. Traverse the vertices of \( \pi_2 \) from left to right.
2. Find the leftmost endpoint and rightmost endpoint of the neighborhood interval of every \( t_i \). Let \( s_{l_i} \) and \( s_{r_i} \) be the leftmost and rightmost endpoints of the neighborhood interval of some \( t_i \) respectively.
3. Update \( \sigma_S \) as follows: Insert \( t_i \) immediately after its rightmost neighbor, i.e. after \( s_{r_i} \) for \( 1 \leq i \leq q \).

Hereafter we will denote by \( \sigma_S = (u_1, u_2, ..., u_n) \) the above ordering of the vertices of graph \( G \).

We denote by \( u_{f(u_i)} \) and \( u_{h(u_i)} \) the leftmost and rightmost neighbor of \( u_i \) in \( \sigma_S \) respectively, which appear before \( u_i \) in \( \sigma_S \).

In a similar fashion, we can start from the ordering \( \pi_2 \) of \( T \) and relabel \( S \) and obtain another ordering denoted by \( \sigma_T \).

3.2 Monotonic Path

We define \( S \)-Monotone and \( S \)-S path in the section. \( T \)-Monotone and \( T \)-T paths are defined symmetrically.

**Definition 1.** A \( S \)-Monotone path of a Biconvex graph \( G = (S \cup T, E) \) is a simple path

\[ P = \{s_{\alpha_1}, t_{\beta_1}, s_{\alpha_2}, t_{\beta_2}, ..., s_{\alpha_j}, t_{\beta_j}\} \]

such that \( s_{\alpha_k} \prec_{\sigma_S} s_{\alpha_{k+1}} \forall k \geq 1 \leq k \leq j \).

Symmetrically, we define \( T \)-Monotone path.

**Definition 2.** A path with starts from a vertex on \( S \)-partition and end ends in \( S \)-partition is called a \( S \)-S path.

Symmetrically, we define \( T \)-T path. We state and prove the lemma for \( S \)-S paths. The same argument will hold good for \( T \)-T paths just by replacing the \( S \)-vertices with \( T \)-vertices and vice versa.

First, we informally introduce the lemma. Informally, we claim that given any simple path \( P = \{s_{\alpha_1}, t_{\beta_1}, s_{\alpha_2}, t_{\beta_2}, ..., s_{\alpha_j}, t_{\beta_j}\} \) of a biconvex graph, for the longest \( S \)-S sub path of \( P \) we can construct an equivalent path \( P' \) such that, \( P' = \{s_{\gamma_1}, t_* s_{\gamma_2}, t_* s_{\gamma_3}, ..., s_{\gamma_j}\} \) where \( \gamma_1 \prec_{\sigma_S} \gamma_2 \prec_{\sigma_S} \gamma_3 \prec_{\sigma_S} \gamma_j \), where \( \gamma_i \in \{\alpha_1, \alpha_2, ..., \alpha_j\} \) and \( t_* \) denotes \( T \)-vertex of some index belonging to \( \{\beta_1, \beta_2, ..., \beta_j\} \).
Lemma 1. Let $P = \{s_{\alpha_1}, t_{\beta_1}, s_{\alpha_2}, t_{\beta_2}, \ldots, s_{\alpha_{j-1}}, t_{\beta_{j-1}}, s_{\alpha_j}, t_{\beta_j}\}$ 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 in $V(P_{max})$ can be reordered to get a path $P_{max}'$, which is $S$-Monotone.

Proof. We prove the lemma by induction on $|S|$.

Basis: To prove the basis, let $P = \{s_\alpha, t_a, s_\beta, t_b, s_\gamma\}$ be a simple path which violates the monotonicity property. We construct $P'$ on the set of vertices of $V(P)$ such that $P'$ satisfies $S$-Monotonicity property. There are three possible cases.

1. $\alpha > \beta > \gamma$. Then $P' = P_R$.

2. $\alpha > \beta$, $\beta < \gamma$ and $\alpha < \gamma$. It follows that $\beta < \alpha < \gamma$. Now $t_b$ is adjacent to $s_\beta$ and $s_\gamma$. Owing to the adjacency property, $t_b$ is also adjacent to $s_\alpha$. Hence we have $P' = \{s_\beta, t_a, s_\alpha, t_b, s_\gamma\}$. The other case $\beta > \alpha > \gamma$ can be dealt with similarity to (1). (Fig 1)

3. $\alpha > \beta, \beta < \gamma$ and $\alpha < \gamma$. It follows that $\beta < \gamma < \alpha$. Now $t_a$ is adjacent to $s_\alpha$ and $s_\beta$. Owing to the adjacency property, $s_\alpha$ is also adjacent to $s_\gamma$. Hence we have $P' = \{s_\beta, t_b, s_\gamma, t_a, s_\alpha\}$. The other case $\beta > \gamma > \alpha$ can be dealt in a similar way. Thus the lemma works when the path contains at least three $S$ vertices. (Fig 2)

Inductive hypothesis: For the longest $S$-$S$ sub path of a simple path $P = \{s_{\alpha_1}, t_{\beta_1}, s_{\beta_2}, \ldots, t_{\beta_{j-1}}, s_{\alpha_{j-1}}, t_{\beta_j}\}$, there exists a path $P' = \{s_{\gamma_1}, t_*, s_{\gamma_2}, \ldots, t_*, s_{\gamma_j}\}$ satisfying the $S$-Monotonicity Property.

Inductive step: Let $P_i = P \cdot s_{\alpha_{j+1}}$.

\[\text{Fig. 1. (a) } SS \text{ path violating monotonicity property (b) } S \text{ Monotone path}\]
Fig. 2. (a) SS path violating monotonicity property (b) S Monotone path

Fig. 3. (a) Non S Monotone path $P_1 = P \cdot S_{\alpha_j+1}$ (b) S Monotone path $P$

Fig. 4. (a) Non S Monotone path $P$ (b) Corresponding S Monotone path $P_1 = P \cdot S_{\alpha_{j+1}}$
Similarly we define Notation 1. Let $S$ belong to $G$ such that $|S| \geq 2$. Then it follows that $P_i = s_{\alpha_{i+1}}, t_{\beta_i}, \{s_{\gamma_1}, t_{\beta_1}, \ldots, s_{\gamma_1}\}$. (Fig 3)

- If $\alpha_{j+1} \prec \gamma_j$, then it follows that the last T vertex $t_{\beta_j}$ is also adjacent to $s_{\gamma_j}$.
- If $\alpha_{j+1} \prec \gamma_1$. This implies that the last T vertex $t_{\beta_j}$ is also adjacent to $s_{\gamma_1}$.

In this section, we present our algorithm for solving the longest path problem by $l$. It takes the biconvex graph $G$ as input and generates ordering $\sigma_S = (u_1, u_2, \ldots, u_n)$ such that $u_k$ is the vertex $s_i$ and $u_m$ is the rightmost of them.

4.1 Some Constructs and Notations Used in the Algorithm

Definition 3. For each pair of indices $i, j$ such that $1 \leq i \leq j \leq n$ we define the graph $G(i, j)$ to be the subgraph $G[A]$ of $G$ induced by the set $A = \{u_i, u_{i+1}, \ldots, u_j\} \setminus \{u_k \in T(G) : u_f(u_k) < \sigma_S u_i\}$.

Definition 4. Let $P$ be a path of $G(i, j)$, $1 \leq i \leq j \leq n$. The path $P$ is called $S$-bimonotone if $P$ is a $S$-Monotone path of $G(i, j)$ and both endpoints of $P$ belong to $S$-partition.

Similarly we define $T$-bimonotone path symmetrically.

Notation 1. Let $\sigma_S = (u_1, u_2, \ldots, u_n)$ be the ordering on $G$ or $\sigma_T = (u_1, u_2, \ldots, u_n)$ \forall $u_k \in S(G)$ or $\forall u_k \in T(G)$ we denote by $P(u_k; i, j)$ the longest S-bimonotone path of $G(i, j)$ with $u_k$ as its right endpoint and by $l(u_k; i, j)$ the number of vertices of $P(u_k; i, j)$.

4.2 Algorithm

In this section, we present our algorithm for solving the longest path problem on biconvex graphs; it consists of three phases. Let $S$ denote the partition with higher cardinality, i.e., $|S| \geq |T|$. Therefore the length of the longest path is less than or equal to $2 \ast |T|$.

- Phase1:
  It takes the biconvex graph $G$ and generates ordering $\sigma_S = (u_1, u_2, \ldots, u_n)$.
  Now, for all $s_i, s_j$, where $1 \leq i < j \leq |S|$, do the following:
  1. Choose the subsequence $\sigma_{s_{ij}} = (u_k, u_{k+1}, \ldots, u_m)$ such that $u_k$ is the vertex $s_j$ and $u_m$ is either $u_i$ or, $u_m \in T(G)$ and it lies between $s_j$ and $s_{j+1}$ in the ordering $\sigma_S$. If there are multiple $T$ vertices lying between $s_j$ and $s_{j+1}$ in $\sigma_S$, then $u_m$ is the rightmost of them.
  2. Run the first phase of the algorithm for all $\sigma_{s_{ij}}$ as the input ordering.
  There $k$ and $m$ will replace indices 1 and $n$ respectively.
3. Remember the maximum path length obtained over these iterations and all the paths of that maximum length

- **Phase 2:**
  Symmetric to phase 1, this phase is executed for vertices of T-partition with the initial ordering \( \sigma_T = (u_1, u_2, \ldots, u_n) \)

- **Phase 3:**
  1. Let the path lengths obtained as output from Phase 1 and Phase 2 be \( x \) and \( z \) respectively. Compute \( \max\{x, z\} \). Without loss of generality, let \( z \) be the maximum.
  2. Consider all the T-bimonotone paths of length \( z \) obtained from Phase 2. check if the end vertices of the paths have any unvisited neighbor, i.e., neighbor which does not occur on that path.
  3. If such a neighbor exists, extend the path till that neighbor. Let \( P' \) denote this extended path.
  4. Output \( z + 1 \) as the maximum path length and \( P' \) as the longest path.
  5. Else, output \( z \) as the maximum path length and the corresponding path as the longest path.

### 4.3 Proof of Correctness

**Candidates for Longest Path.** There are four candidates for a longest path \( P \), which starts in a vertex of \( S \) or \( T \) and ends in a vertex of \( S \) or \( T \). We denote these candidates as \( P_{SS}, P_{ST}, P_{TS}, P_{TT} \), where \( P_{XY} \) denotes a longest path among the set of paths starting from a vertex in \( X \) and ending in a vertex in \( Y \).

Now the outline of our algorithm is as follows:

1. Compute the length of \( P_{SS} \) and \( P_{TT} \) from Phase 1 and Phase 2 of the algorithm respectively.
2. Let \( x = |P_{SS}| \) and \( z = |P_{TT}| \) and let \( y = \max(x, z) \).
3. As is evident from the Phase 3, the length of a longest path possible on the graph is either \( y \) (If \( P_{SS} \) or \( P_{TT} \) is the candidate) OR \( y + 1 \) (If \( P_{ST} \) or \( P_{TS} \) is the candidate).

### 4.4 Correctness Argument

We shall prove two claims in order to prove the correctness.

*Claim.* The Algorithm (Phase 1) correctly computes a longest S-bimonotone path(i.e, \( P_{SS} \) path) of the graph \( G \) and Phase 2 correctly computes a longest T-bimonotone path(i.e, \( P_{TT} \) path) of the graph \( G \).

The following observations hold for every induced subgraph \( G(i, j), 1 \leq i \leq j \leq n \) and is used for proving the correctness of phase 1 of the algorithm. Similarly we can state the observations by replacing \( \sigma_S \) with \( \sigma_T \), partition \( S \) with \( T \) and vice versa and prove the correctness of phase 2 of the algorithm. The following properties hold trivially:
Algorithm 1. Longest Path (Phase 1)

Input: The biconvex graph \(G\) and input ordering \(\sigma_{1:n} = (u_1, u_2, ..., u_n)\).

Output: A longest S-bimonotone path of \(G\) and the longest path length.

for \(j = 1\) to \(n\) do
  for \(i = j\) down to \(1\) do
    if \(i = j\) and \(u_i \in S(G)\) then
      \(l(u_i; i, i) \leftarrow 1; P(u_i; i, i) \leftarrow (u_i)\)
    end if
    if \(i \neq j\) then
      for all \(u_k \in S(G), i \leq k \leq j - 1\) do
        \(l(u_k; i, j) \leftarrow l(u_k; i, j - 1); P(u_k; i, j) \leftarrow P(u_k; i, j - 1);\)
      end for
      if \(u_j \in S(G)\) then
        \(l(u_j; i, j) \leftarrow 1; P(u_j; i, j) \leftarrow (u_j)\)
      end if
      if \(u_j\) is a T vertex i.e \(u_j \in T(G)\) and \(i \leq f(u_j)\) then
        execute process \(G(i, j)\)
      end if
    end if
  end for
end for
compute the max \(\{l(u_k; 1; n) : u_k \in S(G)\}\) and the corresponding path \(P(u_k; 1; n)\). Return \((\max \{l(u_k; 1; n) : u_k \in S(G)\} - 1)\) as the maximum path length and \(P(u_k; 1; n)\) also along this path.

We carry out the second phase by re-running the algorithm with \(\sigma_{1:n} = (u_1, u_2, ..., u_n)\). By replacing vertex set T with S and vice versa. The output of second phase is a longest T-bimonotone path of \(G\) and the longest path length.

Algorithm 2. The subroutine \(\text{process}(G(i, j))\)

for \(y = f(u_j) + 1\) to \(j - 1\) do
  for \(x = f(u_j)\) to \(y - 1\) do
    if \(u_x, u_y \in S(G)\) then
      \(w_1 \leftarrow l(u_x; i, j - 1); P'_1 \leftarrow P(u_x; i, j - 1);\)
      \(w_2 \leftarrow l(u_y; x + 1, j - 1); P'_2 \leftarrow P(u_y; x + 1, j - 1);\)
      if \(w_1 + w_2 + 1 > l(u_y; i, j)\) then
        \(l(u_y; i, j) \leftarrow w_1 + w_2 + 1; P(u_y; i, j) = (P'_1, u_y, P'_2);\)
      end if
    end if
  end for
end for
return the value \(\{l(u_k; i, j)\}\) and the path \(\{P(u_k; i, j), \forall u_k \in S(G(f(u_j) + 1, j - 1))\}\)
Hence the algorithm sets path $P$.

Observation 1. Let $G(i, j)$ be the induced subgraph of $S$-bimonotone graph $G$ and $\sigma_S$ be the input ordering of $G$. Let $P = (P_1, u_2, P_2)$ be a $S$-bimonotone path of $G(i, j)$, and let $u_j$ be a $S$ vertex of $G(i, j)$. Then, $P_1$ and $P_2$ are $S$-bimonotone paths of $G(i, j)$.

Observation 2. Let $G(i, j)$ be the induced subgraph of biconvex graph $G$ and $\sigma_S$ be the input ordering of $G$. Let $P_1$ be a $S$-bimonotone path of $G(i, j)$ with $u_x$ as its right endpoint, and let $P_2$ be a $S$-bimonotone path of $G(x + 1; j - 1)$ with $u_y$ as its right endpoint, such that $V(P_1) \cap V(P_2) = \emptyset$. Suppose that $u_j$ is a $T$ vertex of $G$ and that $u_1 \prec_{\sigma_S} u_{(u_2)} \prec_{\sigma_S} u_x$. Then, $P = (P_1, u_j, P_2)$ is a $S$-bimonotone path of $G(i, j)$ with $u_y$ as its right endpoint.

Now we shall prove our claim.

Proof. Let $P$ be the longest $S$-bimonotone path of the subgraph $G(i, j)$ with $u_y \in S(G(i, j))$ as its rightmost endpoint. Consider the case when $T(G(i, j)) = \phi$. Hence the algorithm sets path $P(u_y; i, j) = u_y$ and $l(u_y; i, j) = 1$. Therefore, the lemma holds for every induced subgraph $G(i, j)$ for which $T(G(i, j)) = \phi$.

Now consider the case where $T(G(i, j)) \neq \phi$.

We will prove the correctness by induction on number of $T$-vertices.

Basis: Consider $\sigma_S = (u_1, u_2, u_3)$ where $u_1, u_2 \in S(G(i, j))$ and $u_3 \in (G(i, j)$ and $u_1$ and $u_2$ are the leftmost and rightmost neighbors of $u_3$ respectively. $process(G(i, j))$ will be called and according to the algorithm, $P(u_2; i, j) = (u_1, u_3, u_2)$ and $S(u_2; i, j) = 3$ will be given as output which is indeed a longest $S$-bimonotone path.

Inductive Hypothesis: Let the algorithm generate the longest $S$-bimonotone path when $G(i, j)$ contain $k$ number of $T$-vertices.

Induction Step: Now we consider the scenario where we $G(i, j)$ contains $k + 1$ $T$-vertices. Now according to the algorithm, all the neighbors of the newly added $T$-vertex are considered and for all possible combinations of the neighbors say $u_x$ and $u_y$ the length of path $P(u_y; i, j)$ is computed by considering the sub paths $P_1$, a longest $S$-bimonotone path in $G(i, j - 1)$ with $u_x$ as the rightmost vertex and $P_2$, a longest $S$-bimonotone path in $G(x + 1, j - 1)$ with $u_y$ as the rightmost vertex. (By the principle of strong induction, $P_1$ and $P_2$ are longest $S$-Bimonotone paths, given as output by the algorithm as in both the subgraphs $G(i, j - 1) and G(x + 1, j - 1)$ the number of $T$-vertices $\leq k$.)

Now $P$ is updated as follows: $P(u_y; i, j) = (P_1, u_j, P_2)$ if $l(P_1) + l(P_2) + 1 > l(P)$, where $V(P_1) \cap V(P_2) \neq \emptyset$. This is evident from Observation 2.

Hence the path computed by phase 1 of the Algorithm is indeed a longest $S$-bimonotone path on $G$ with $u_y$ as its right endpoint.

Claim. At least one of the paths, obtained as output from Phase 1 and Phase 2 is extendible if and only if a longer path exists.

Proof. Forward Implication:

This follows from the correctness of the Algorithm (Phase 1), proved above.
Backward Implication:

Here we need to prove the following implication: A longer path exists implies it is the extension of one of the paths obtained in Phase 1 and Phase 2.

As we have already discussed, there are 4 candidates for longest path, \( P_{SS}, P_{TT}, P_{ST} \) and \( P_{TS} \). In Phase 1 and Phase 2 of the algorithm, all the longest S-bimonotone and T-bimonotone paths (respectively) are generated.

So the only case, where a longer path might exist, is the case where the candidate longest path is either a ST or a TS path.

Without Loss of Generality, let the longest path be a ST path. Now, from lemma 1, we know, for the longest S-S sub path of \( P \), we can reorder the vertices, so that the S-vertices follow the S-Monotonicity property. Let length of this ST path be \( x \). Then the length of its longest S-S sub path has to be \( x - 1 \).

Now, let the longest path length, given as output of Phase1 (Generating longest S-bimonotone path) be \( m \).

If possible, let \( m \neq x - 1 \).

Then there are two possibilities:

- If \( m < x - 1 \), this contradicts the correctness of Algorithm(Phase 1). Therefore, \( m > y \).
- If \( m > x - 1 \), this implies ST is not the longest path. Hence a contradiction again.

Hence \( m = x - 1 \). Since the Phase 1 of the algorithm has generated all possible S-bimonotone paths of length \( m \), hence the longest ST path has to be an extension of any one of them.

Similar argument will follow if the longest path is a TS path.

5 Time Complexity

- The ordering \( \sigma_S \) (and similarly \( \sigma_T \)) will take \( O(|S||T|) \) time, where \( O(|S|) \) time required to compute the specific neighborhood of \( u_i \in T \) for all \( u_i \). So total time required for ordering \( \sigma_S \) (and similarly \( \sigma_T \)) is \( O(|S||T|) \).
- The subroutine \( \text{process}() \) takes \( O(|S|^2) \) due to \( |S|^2 \) pairs of neighbors \( u_x \) and \( u_y \) of the T vertex in the graph \( G(i, j) \).

Additionally, the subroutine \( \text{process}() \) is executed at most once for each subgraph \( G(i, j) \) of \( G \), \( 1 \leq i \leq j \leq n \), i.e. it is executed \( O(n^2) \) times. Thus, time complexity is \( O(|S|^2n^2) \) time for phase 1 of the algorithm. Similarly for phase 2 of the algorithm, time complexity is \( O(|T|^2n^2) \). Hence total complexity is \( \max \{ O(|S|^2n^2), O(|T|^2n^2) \} \), which is \( O(n^4) \) where \( n \) is the total number of vertices of the biconvex graph.

- Generating \( \sigma_{si} = [\sigma_{tij}] \) from \( \sigma_S \) (and similarly \( \sigma_T \)) will take linear time.
- Phase 1 of the algorithm will be executed for each ordered pair \( S_i, S_j \). There can be \( O(|S|^2) \) such ordered pairs. Similarly, Phase 2 can be executed for \( O(|T|^2) \) such ordered pairs. So the total time complexity is \( \max \{ O(|S|^2n^4), O(|T|^2n^4) \} \), which is \( O(n^6) \) where \( n \) is the total number of vertices of the biconvex graph.
6 Conclusion and Further Work

Here we have presented a polynomial time algorithm to find the longest path on a biconvex graph. The obvious further step is to extend this direction to look for a polynomial time solution of longest path problem on Convex graphs.

References