A Polynomial Time Algorithm for Longest Paths in Convex Graphs

1 Convex Graphs

A bipartite graph $G = (S \cup T, E)$ is convex if all its vertices from the set S can be numbered as $1, 2, \cdots, |S|$ and all vertices from T can be numbered as $1, 2, \cdots, |T|$ in such a way that for every vertex of S or every vertex of T (but not both), all neighbors of the vertex are numbered consecutively (is an interval of numbers).

2 Orderings and Monotone Paths

2.1 Ordering of Vertices

Let $\pi_1 = (s_1, s_2, \cdots, s_p)$ be the labelled vertices of partition S, and $\pi_2 = (t_1, t_2, \cdots, tq)$ be that of partition T. If T is a convex partition, we can obtain an ordering σ_S on S vertices as follows:
Initially set $\sigma_S = \pi_1$.

1. Traverse the vertices of π_2 from left to right.
2. Find the leftmost endpoint and rightmost endpoint of the neighborhood interval of every t_i . Let s_{li} and s_{ri} be the leftmost and rightmost endpoints of the neighborhood interval of some t_i respectively.
3. Update σ_S as follows: Insert t_i immediately after its rightmost neighbor, i.e after s_{ri} $\forall 1 \leq i \leq q$.

We denote the new ordering as: $\sigma_S = (u_1, u_2, \cdots, u_n)$. We denote by $u_{f(u_i)}$ and $u_{h(u_i)}$ the leftmost and rightmost neighbour of u_i ($u_i \in T$) in σ_S respectively, which appear before u_i in σ_S.
Similarly for σ_T ordering.

2.2 Monotone Path

Definition 1. *A S-Monotone path of a convex graph $G = (S \cup T, E)$ is a simple path $P = \{s_{\alpha_1}, t_{\beta_1}, s_{\alpha_2}, \cdots, t_{\beta_{j-1}}, s_{\alpha_j}, t_{\beta_j}\}$ such that $s_k \prec_{\sigma_S} s_{k+1}$ $\forall k$, $1 \leq k \leq j-1$.*
Similarly for T-Monotone path.

2.3 Non-monotone Path

A simple path of a convex graph G is called a non-monotone path if it does not follow ordering on S and T vertices.

Lemma 1. *Let $G(S, T; E)$ be a convex graph in which S is a convex partition. For every non-monotonic path of length k in G, the first or last k-1 vertices can be reordered to get a T-monotonic path of length k-1.*

Proof. We prove the lemma by induction on $|S|$ and $|T|$.
Basis: To prove the basis, let P=$\{t_a, s_\alpha, t_b, s_\gamma, t_c, s_\beta\}$ be a non-monotonic path of length 6 in graph G. We construct a path P' on first 5 vertices of path P such that P' satisfies T-monotonicity property. There are two possible cases in which we can order T vertices to get a T-monotone path of length 5.

1. $b \prec a \prec c$. Since S is a convex partition, $L(s_\gamma) = t_b$ and $R(s_\gamma) = t_c$, this implies that s_γ is also adjacent to t_a. Hence, we have T-monotone path P'=$\{t_b, s_\alpha, t_a, s_\gamma, t_c\}$. (Figure 1) [1]

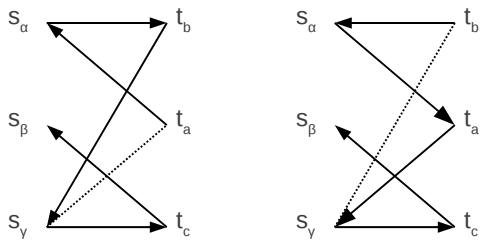

Fig. 1. (a) Path P violating monotonicity property (b) *T-Monotone* path P'

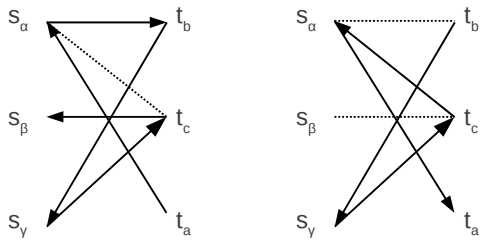

Fig. 2. (a) Path P violating monotonicity property (b) *T-Monotone* path P'

2. $b \prec c \prec a$. As $L(s_\alpha) = t_b$ and $R(s_\alpha) = t_a$, s_α is adjacent to t_c. Hence, we get a T-monotone path P'=$\{t_b, s_\gamma, t_c, s_\alpha, t_a\}$. (Figure 2)

Thus the lemma works when the path contains atleast three T vertices.

Inductive Hypothesis: For the non-monotonic path P=$\{t_{\alpha_1}, s_{\beta_1}, t_{\alpha_2}, s_{\beta_2}, \cdots, s_{\beta_{j-1}}, t_{\alpha_j}, s_{\beta_j}\}$, $|P| = k$ there exist a path P' on first $k-1$ vertices of path P which satisfies T-monotonicity property.
P'=$\{t_{\gamma_1}, s_*, t_{\gamma_2}, s_*, \cdots, s_*, t_{\gamma_j}\}$.

Inductive Step: Let $P_1 = P.t_{\alpha_{j+1}}, s_{\beta_{j+1}}$. Hence $|P_1| = k+2$.

1. If $\gamma_j \prec \alpha_{j+1}$, since s_{β_j} is adjacent to $t_{\alpha_j} = t_{\gamma_i}$; $1 \leq i \leq j$ we get a T-monotone path P'_1=$P'.s_{\beta_j}, t_{\alpha_{j+1}}, s_{\beta_{j+1}}$ of length $k+2$.
2. Similarly if $\alpha_{j+1} \prec \gamma_1$, we get a T-monotone path P'_1=$s_{\beta_{j+1}}, t_{\alpha_{j+1}}, s_{\beta_j}.P'$ of length $k+2$
3. If $\gamma_m \prec \alpha_{j+1} \prec \gamma_{m+1}$ then in the path P'=$\{t_{\gamma_1}, s_*, t_{\gamma_2}, \cdots, t_{\gamma_m}, s_{\beta_m}, t_{\gamma_{m+1}}, \cdots, s_*, t_{\gamma_j}\}$ we can see that s_{β_m} is also adjacent to $t_{\alpha_{j+1}}$. Now we know that s_{β_j} is adjacent to $t_{\alpha_{j+1}}$ and $t_{\alpha_j} = t_{\gamma_i}$; $1 \leq i \leq j$.

   - If $1 \leq i \leq m$, we get a T-monotone path P'_1=$\{t_{\gamma_1}, s_*, t_{\gamma_2}, \cdots, t_{\gamma_m}, s_{\beta_j}, t_{\alpha_{j+1}}, s_{\beta_m}, t_{\gamma_{m+1}}, \cdots, s_*, t_{\gamma_j}\}$. Here $|P'_1| = k+1$ and $V(P'_1) = V(P_1) \setminus s_{\beta_{j+1}}$.

[1] Graphs considered here are undirected. In the diagram, the arrowheads simply represent the vertex-traversal sequence in the path.

- If m+1$\leq i \leq j$, we get a T-monotone path P_1'=$\{t_{\gamma_1}, s_*, t_{\gamma_2}, \cdots, t_{\gamma_m}, s_{\beta_m}, t_{\alpha_{j+1}}, s_{\beta_j}, t_{\gamma_{m+1}}, \cdots, s_*, t_{\gamma_j}\}$. Here $|P_1'| = k+1$ and $V(P_1') = V(P_1) \setminus s_{\beta_{j+1}}$.

Lemma 2. *If the longest path in G is non-monotonic, the longest T-monotone path P' can be extended, by shifting a T vertex, u_m (belonging to the path P') who has a neighbour S vertex u_s (not belonging to P'), along with its adjacent left or right S vertex in the path P', either to the beginning or the end of the path and extending it by one (u_s).*

Proof. The S partition is convex and the T partition is non-convex. Let P'' denotes the longest non-monotone path. The longest T-monotone path obtained from the algorithm *(Phase 3)* is $P' = \{u_1, u_2, \cdots, u_n\}$.
$|P''| = |P'| + 1$.

- From the longest T-monotone path P' obtained, find out the T vertex in P', u_m that has an unvisited neighbour S vertex, u_s, i.e. u_s doesn't belong to P'.
- If possible, shift u_m and it's adjacent S vertex u_{m-1} to the front i.e. before u_1 or shift u_m and adjacent S vertex u_{m+1} to the back i.e. after u_n.
- Add u_s to u_m to extend the path.

If shifted to the beginning we will get $P'' = \{u_s, u_m, u_{m-1}, u_1, u_2, \cdots u_{m-2}, u_{m+1}, \cdots u_n\}$.
If shifted to the end, we will get $P'' = \{u_1, u_2, \cdots, u_{m-1}, u_{m+2}, \cdots u_n, u_{m+1}, u_m, u_s\}$.

To obtain a non-monotonic path, we need to consider atleast 3 vertices on each partition. Hence $|S| = |T| = 3$. We consider only hamiltonian non-monotonic paths, as paths of length lesser than 6 will always be monotonic. For e.g., consider a path of length 5 having 3 T vertices and 2 S vertices or vice versa . It will always be monotonic with respect to the smaller partition. Similarly, paths of lesser lengths, will be monotonic with respect to both partitions. Hence, we consider hamiltonian paths.

Considering all hamiltonian paths, the possible non-monotonic paths are:
Paths where S and T can occur in the order:
132
213
231
312
eg. *T1 S1 T3 S3 T2 S2.*
This results in 32 paths.

Construction. We construct each graph by first drawing the non-monotonic longest path (one of the 32), then we make the S partition convex (shown by the dotted edges in *Figure 3*). No extra edges are added since it increases the chance of getting a monotonic longest path or a biconvex graph. We get a biconvex graph if all the T vertices are adjacent to $S2$.
On analyzing all these graphs, we obtain 4 graphs with non-monotonic longest paths. The rest are either biconvex or have monotonic longest paths.
These four graphs are shown in *Figure 3.*

Considering the first graph, the longest monotonic path obtained from the algorithm is *T1 S1 T2 S3 T3.*
Now applying the shifting algorithm:

1. $T2$ has an unvisited S vertex $S2$, which is not in the path.
2. Shift $S3$ and $T2$ after $T3$. This is possible because an edge exists between $T3$ and $S3$, and between $S1$ and $T3$.
3. Thus we get the path: *T1 S1 T3 S3 T2 S2* on adding $S2$ after $T2$.

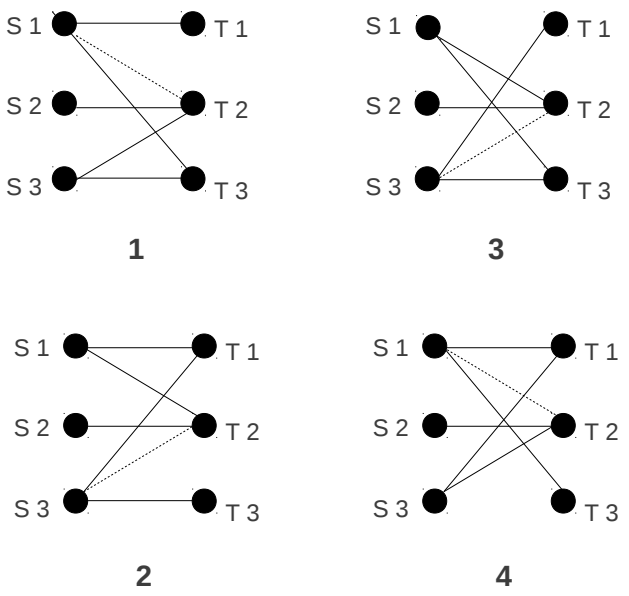

Fig. 3.

The same algorithm holds good for other 3 graphs.

Observation. We observe that in all these four graphs, $S2$ is adjacent only to $T2$. The longest T monotone path can either be:
Path 1: *T1 S1 T2 S3 T3* (which we observe in graphs 1 and 2)
Path 2: *T1 S3 T2 S1 T3* (which we observe in graphs 3 and 4)

By drawing Path 1 and the edge connecting $S2$ and $T2$, we get a biconvex graph. But, the graph has to be convex, hence there must exist an edge between either $T1$ and $S3$, or between $T3$ and $S1$. If an edge exists between $T1$ and $S3$, we can shift $T2$ and $S1$ to the front. Else, if an edge exists between $T3$ and $S1$, we can shift $T2$ and $S3$ to the end.

Similarly, by drawing Path 2 and the edge connecting $S2$ and $T2$, we get a biconvex graph. But, the graph has to be convex, hence there must exist an edge between either $T1$ and $S1$, or between $T3$ and $S3$. If an edge exists between $T1$ and $S1$, we can shift $T2$ and $S3$ to the front. Else, if an edge exists between $T3$ and $S3$, we can shift $T2$ and $S1$ to the end.

Thus, for all the possible cases, we can perform the shift on the longest monotone path to get the longest non-monotonic path.

3 The Modified Algorithm and Correctness

3.1 Some constructs and notations used in the algorithm

Definition 2. *For every 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* $A = \{u_i, u_{i+1}, \cdots, u_j\} \setminus \{u_k \in S(G) : u_{f(u_k)} \prec_{\sigma_T} u_i\}$.

Definition 3. *Let* P *be a path of* $G(i,j)$, $1 \leq i \leq j \leq n$. *The path* P *is called* T*-bimonotone if* P *is a* T*-Monotone path of* $G(i,j)$ *and both endpoints of* P *belong to* T*-partition.*

Similarly we define S-bimonotone path symmetrically.

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

3.2 Algorithm

In this section, we will present how the existing algorithm for finding the longest path in a biconvex graph can be modified and extended to solve the longest path problem on convex graphs. We have to make some modifications in the loops and subroutine of the main algorithm to get all the T and S-bimonotone paths. This is because all the extended bimonotone paths must be considered on which we will apply shifting algorithm to obtain non-monotone longest paths (if exists). Phase 1 and 3 of the algorithm are similar to that of algorithm for longest path problem on biconvex graph. Let $G(S \cup T, E)$ be a convex graph in which S is the convex partition and T is non-convex partition.

- **Phase 1:**
  It takes the convex graph G and generates ordering $\sigma_T = (u_1, u_2, \cdots, u_n)$. Now, for all ti, tj, where, $1 \leq i < j \leq |T|$, do the following:
  1. Choose the subsequence $\sigma_{tij} = (u_k, u_{k+1}, \cdots, u_m)$ such that u_k is the vertex t_i and u_m is either t_j or, $um \in S(G)$ and it lies between t_j and t_{j+1} in the ordering σ_T. If there are multiple S vertices lying between t_j and t_{j+1} in σ_T, then u_m is the rightmost of them.
  2. Run the first phase of the algorithm for all σ_{tij} 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 S-partition with the initial ordering $\sigma_S = (u_1, u_2, \cdots, u_n)$. As T is the non-convex partition, we dont have proper ordering on S vertices. Thus we may get some longest S-bimonotone paths in which some edges do not belong to the given convex graph. Remove all such non-existing SS paths.

- **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 x be the maximum.
  2. Consider all the T-bimonotone paths of length x obtained from Phase1. Check if the end vertices of the paths have any unvisited neighbour, i.e., neighbour which does not occur on that path.
  3. If such a neighbour exists, extend the path till that neighbour. Let P' denotes this extended path.
  4. Output $x + 1$ as the maximum monotone path length and P' as the longest monotone path.
  5. Else, output x as the maximum monotone path length and the corresponding path as the longest monotone path.

– **Phase 4:**
  1. Apply shifting algorithm on each of the longest monotone path P' obtained from phase 3.
  2. If the longest path is non-monotonic, the shifting algorithm will obtain that path from path P'. Let P'' denotes this longest non-monotonic path.
  3. Output $|P'|+1$ as the maximum path length and P'' as the longest path which is non-monotonic.
  4. Else, output $|P'|$ as the maximum path length and $|P'|$ as the longest path which is monotonic.

Algorithm 1 Longest Path (Phase 1)

Input: The convex graph G and input ordering $\sigma_{t1n} = (u_1, u_2, ..., u_n)$.
Output: Longest T-bimonotone paths of G and the longest path length.

for $j = 1$ to n **do**
  for $i = j$ down to 1 **do**
    if $i = j$ and $u_i \in T(G)$ **then**
      $l(u_i; i, i) \leftarrow 1$;
      $np(u_i; i, i) \leftarrow 1$;
      Add path (u_i) to $P(u_i; i, i)$
    end if
    if $i \neq j$ **then**
      for all $u_k \in T(G)$, $i \leq k \leq j-1$ **do**
        $l(u_k; i, j) \leftarrow l(u_k; i, j-1)$;
        $np(u_k; i, j) \leftarrow np(u_k; i, j-1)$;
        Copy all the paths in $P(u_k; i, j-1)$ to $P(u_k; i, j)$;
      end for
      if $u_j \in T(G)$ **then**
        $l(u_j; i, j) \leftarrow 1$;
        $np(u_j; i, j) \leftarrow 1$;
        Add path (u_j) to $P(u_j; i, j)$
      end if
      if u_j is a S vertex i.e $u_j \in S(G)$ and $i \leq f(u_j)$ **then**
        execute process *G(i,j)*
      end if
    end if
  end for
end for

compute the max$\{l(u_k; 1; n) : u_k \in T(G)\}$ and the corresponding paths in $P(u_k; 1, n)$. Return (max $\{l(u_k; 1, n) : u_k \in T(G)\}$) as the maximum path length and paths in $P(u_k; 1, n)$ as longest T-bimonotone paths.
We carry out the second phase by re-running the algorithm with $\sigma_{s1n} = (u_1, u_2, ..., u_n)$. By replacing vertex set T with S and vice versa. The output of second phase is a longest S-bimonotone path of G and the longest path length.

Algorithm 2 The subroutine **process(G(i,j))**

for $y = f(u_j) + 1$ to $j - 1$ **do**
  for $x = f(u_j)$ to $y - 1$ **do**
    if $u_x, u_y \in T(G)$ **then**
      $w_1 \leftarrow l(u_x; i, j-1)$;
      $w_2 \leftarrow l(u_y; x+1, j-1)$;
      if $w1 + w2 + 1 > l(u_y; i, j)$ **then**
        $l(u_y; i, j) \leftarrow w1 + w2 + 1$;
        $np(u_y; i, j) = 0$
        Remove all the paths in $p(u_y; i, j)$
        for all paths P_1' in $P(u_x; i, j-1)$ **do**
          for all paths P_2' in $P(u_y; x+1, j-1)$ **do**
            Add path (P_1', u_j, P_2') to $P(u_y; i, j)$;
            $np(u_y; i, j) + +$;
          end for
        end for
      end if
      if $w1 + w2 + 1 = l(u_y; i, j)$ **then**
        for all paths P_1' in $P(u_x; i, j-1)$ **do**
          for all paths P_2' in $P(u_y; x+1, j-1)$ **do**
            Add path (P_1', u_j, P_2') to $P(u_y; i, j)$;
            $np(u_y; i, j) + +$;
          end for
        end for
      end if
    end if
  end for
end for
return the value $\{l(u_k; i, j)\}$ and the path $\{P(u_k; i, j), \forall u_k \in T(G(f(u_j) + 1, j - 1))\}$

Algorithm 3 The shifting (Phase 4)

Input: Longest T-monotone paths $P' = \{u_1, u_2, \cdots u_{m-2}, u_{m-1}, u_m, u_{m+1}, u_{m+2}, \cdots u_n\}$ of G obtained from phase 3.
Output: Longest non-monotone paths of G (if exist) and its length $|P'| + 1$.

for all paths $P' = \{u_1, u_2, \cdots u_{m-2}, u_{m-1}, u_m, u_{m+1}, u_{m+2}, \cdots u_n\}$ **do**
  for $m = 2$ to $n - 1$ **do**
    if $u_m \in T(G)$ and $\{\exists u_s \in N(u_m) : u_s \notin P'\}$ **then**
      if $u_1 \in T(G)$ **then**
        if $(u_{m-1}, u1) \in E(G)$ and $(u_{m-2}, u_{m+1}) \in E(G)$ **then**
          Output $P'' = \{u_s, u_m, u_{m-1}, u_1, u_2, \cdots u_{m-2}, u_{m+1}, u_{m+2}, \cdots u_n\}$ as the longest path and $|P''|$ as maximum path length
        end if
      end if
      if $u_n \in T(G)$ **then**
        if $(u_n, u_{m+1}) \in E(G)$ and $(u_{m-1}, u_{m+2}) \in E(G)$ **then**
          Output $P'' = \{u_1, u_2, \cdots u_{m-2}, u_{m-1}, u_{m+2}, \cdots u_n, u_{m+1}, u_m, u_s\}$ as the longest path and $|P''|$ as maximum path length
        end if
      end if
    end if
  end for
end for

3.3 Proof Of Correctness

Refer to Lemma 1 and 2.

4 Time Complexity