Bounded Diameter Minimum Spanning Tree

by

Giorgi Nadiradze

Submitted to

Central European University

Department of Mathematics And Its Applications

In partial fulfilment of the requirements for the degree of
Master of Science

Supervisor: Professor Ervin Győri

Budapest, Hungary

2013

Contents

Chapter 1

Introduction

In this thesis we deal with the classical problem of minimum cost spanning tree. Given a connected, undirected graph G with non-negative edge costs, we want to find the minimum cost connected and acyclic subgraph of G which contains all vertices of G. There are many known polynomial time algorithms to find the minimum cost spanning tree in a graph, for example Prims Algorithm[9] and Kruskals Algorithm [8]. The general minimum spanning tree problem becomes harder if we add more constraints to the minimum spanning tree, for example we can add an upper bound to the degree of vertices in the minimum spanning tree or to the diameter of the minimum spanning tree. In both cases problem is $NP - hard$. We are interested in the Bounded Diameter Minimum Spanning Tree Problem.

Problem 1. *The Bounded Diameter Minimum Spanning Tree Problem : We are given a connected Graph $G = (V, E)$, where V is a node set, $\mid V \mid = n$, a cost function $cost : E \to \mathbb{Z}_+$ and an integer $2 \leq D \leq n - 2$. We want to find a spanning tree (connected, acyclic subgraph which contains all vertices of G) with the minimal total cost of edges and with the diameter D (for every pair of vertices of the spanning tree u and v, the number of edges in the path in the spanning tree from u to v is not more than D)*

The Bounded Diameter Minimum spanning Tree Problem has applications in communication network design. It is used in Distributed Mutual Exclusion algorithms [10]. We have a Computer Network of n nodes, which communicate by sending messages to each other along a tree. Only one node has a priviledge to enter a critical section. If some node wants to have a priviledge it must request it by sending a message to the node which has a priviledge. Time this request takes depends on the number of edges in the path to the vertex with a priviledge. We want to build a communication tree with the minimal cost and bounded time of communication, that

is, Bounded Diameter Minimum Spanning Tree. The Bounded Diameter Minimum Spanning Tree problem is also used in Information Retrieval [4].

If $D = 2$ or $D = 3$ or the costs of edges of the graph are equal, then the Bounded Diameter Minimum Spanning Tree problem is solvable in polynomial time [5]. We are interested in the case when $4 \leq D < n - 1$ and the costs of edges of the graph are not equal because in this case the Bounded Diameter Minimum Spanning Tree Problem is $NP - hard$ [5]. There are exact solutions for Bounded Diameter Minimum Spanning Tree Problem, for example the Mixed Integer Linear Programming aproach [2], but they require exponential time. Common way to deal with $NP - hard$ problems is to find approximate solutions for them. There are Greedy Heuristical algorithms developed, some of them use Prims algorithm [9], for example [1]. Another way is to put some constraints on the graph G so that maybe the problem will be solvable in polynomial time for the constrained G. For example let us consider the Euclidean instance of the Bounded Diameter Minimum Spanning Tree Problem, where vertices of G are points in the plane and the cost function is the Euclidean Distance between two points. The Euclidean instance of the Bounded Diameter Minimum Spanning Tree Problem is probably also $NP - hard$. There is no proof of $NP - hardness$ available, but approximation algorithms are better because the triangle inequality holds for it [6]. Let us consider a more constrained version of the Euclidean instance of the Bounded Diameter Minimum Spanning Tree Problem, where vertices of G are points on the real line and the cost function is the Euclidean Distance between them.

Problem 2. *The Bounded Diameter Minimum Spanning Tree Problem for points on the real line: We are given a connected Graph $G = (V, E)$, where V is a node set, $\mid V \mid = n$ and an integer $4 \leq D \leq n - 2$. Vertices of G are points on the real line with coordinates $x_1, x_2, ...x_n$, a cost of the edge between vertices u and v is $\mid x_u - x_v \mid$. We want to find a spanning tree(connected, acyclic subgraph which contains all vertices of G) with the minimal total cost of edges and with the diameter D (for every pair of vertices of the spanning tree u and v, the number of edges in the path in the spanning tree from u to v is not more than D)*

We proved that the Bounded Diameter Minimum Spanning Tree Problem for points on the real line is solvable in polynomial time (Theorem 3.5). We found $O(n^4 \cdot D)$ time algorithm which uses dynamic programming [3]. We can also consider case of the Bounded Diameter Minimum Spanning Tree when vertices of G are points on a circle, cost function is a length of an arc between two points.

Problem 3. *The Bounded Diameter Minimum Spanning Tree Problem for points on a circle : We are given a connected Graph $G = (V, E)$, where V is a node set, $\mid V \mid = n$ and integer $4 \leq D \leq n - 2$. Vertices of G are points on a circle, the cost of the edge between vertices u and v is the length of the arc between u and v. We want to find a spanning tree (connected, acyclic subgraph which contains all vertices of G) with the minimal total cost of edges and with the diameter D(for every pair of vertices in the spanning tree u and v the number of edges in the path in the spanning tree from u to v is not more than D)*

We proved that the Bounded Diameter Minimum Spanning Tree Problem for points on a circle can be also solved in polynomial time using algorithm for points on the real line case (Theorem 3.7). We found algorithm which runs in $O(n^5 \cdot D)$ time if D is even and in $O(n^6 \cdot D)$ time when D is odd.

More general instance of the Bounded Diameter Minimum Spanning Tree Problem for points on the real line is, when we are given a tree T with n vertices and edge lengths. Vertices of the graph G are vertices of T and the cost of edge between vertices u and v in G is the length of the path between u and v in T. If tree T is a path(there are vertices u and v such that every vertex of T belongs to the path from u to v) we have points on the real line case.

Problem 4. *The Bounded Diameter Minimum Spanning Tree Problem for vertices in a tree : given a tree T with n vertices and edge lengths and a connected Graph $G = (V, E)$, where V is a node set, $\mid V \mid = n$ and an integer $4 \leq D \leq n - 2$. The vertices of G correspond to the vertices of T, a cost of the edge between vertices u and v in G is the length of the path between u and v in T. We want to find a spanning tree(connected, acyclic subgraph which contains all vertices of G) with the minimal total cost of edges and with the diameter D (for every pair of vertices in the spanning tree u and v, the number of edges in the path in the spanning tree from u to v is not more than D)*

If T is a Caterpillar Tree (there are vertices u and v such that every vertex of T either lies on the path from u to v or is connected with the edge to the vertex which belongs to the path from u to v), we can modify algorithm for points on the real line case and use it. We get algorithm with running time $O(n^4 \cdot D)$, so we solved the Bounded Diameter Minimum Spanning Tree Problem for vertices in a tree in polynomial time, if T is a Caterpillar Tree (Theorem 3.9).

Open Question is what happens for the general T ? Is the Bounded Diameter Minimum Spanning Tree Problem for vertices in a tree $NP - hard$ or is there polynomial time algorithm?

Chapter 2

General Bounded Diameter Minimum Spanning Tree

First We prove $NP-hardness$ of the general Bounded Diameter Minimum Spanning Tree problem.

Theorem 2.1. *The Bounded Diameter Minimum Spanning Tree Problem (Problem 1) is $NP-hard$ if $D \geq 4$ and not all edges of the graph have equal cost [5]*

Proof. To prove $NP-hardness$ of the Bounded Diameter Minimum Spanning Tree problem we need to reduce it to the known NP-Complete problem. We will use Exact Cover by 3-sets problem for this purpose [5]
Exact Cover by 3-sets : Given a set $S = \{1, 2, 3, ..., 3n\}$ and F - set consisting of 3-element subsets of S, is there $F' \subseteq F$ such that the union of the elements of F' is S and the intersection of any two elements of F' is an empty set [5]
To reduce Exact Cover by 3-sets problem to the Bounded Diameter Minimum Spanning Tree problem we construct a graph G. G has two vertices x and y such that the path between them contains $D-2$ edges. It has m vertices called a subset vertices , $\mid F \mid = m$ and it has $3n$ vertices called an element vertices. The elements of F correspond to the subset vertices and the elements of S correspond to the element vertices. Each edge in the path from x to y has cost 0. Edges between x and the subset vertices have cost 1. Each edge between two subset vertices has cost 0. There is an edge of cost 0 between subset vertex and element vertex if element of F corresponding to the subset vertex contains the element of S corresponding to the element vertex. All the other edges have constant cost $C > n$.

The graph G has the minimum spanning tree with the cost not more than n and with the diameter D if and only if Exact Cover by 3-sets has a solution:

If F' is a solution, $\mid F' \mid = n$, let us consider a graph which does not contain any edge of cost C. It contains edges in the path from x to y. There is an edge between x and subset vertex if element of F corresponding to the subset vertex is also in F' - n edges of cost 1. All other $m-n$ subset vertices are connected to some other subset vertex which is connected to x and there are edges between the subset vertices connected to x and the element vertices. Because each element of S is contained in the exactly one element of F' and there is an edge of cost 0 between every two subset vertices, the graph is connected and acyclic so it is a tree. It has the cost n and the diameter is not more than D, because every path from x to any subset vertex or element vertex does not have more than two edges and there are $D-2$ edges in the path from x to y.

If there is the minimum spannning of cost not more than n and with the bounded diameter D, there are no edges of cost C in it because $C > n$. It contains the path from x to y consisting of $D-2$ edges. Each path from the element vertex to x must contain exacty two edges, it can not contain more than two edges because then the path from y to the element vertex will have more than D edges. It can not contain one edge because there is no edge of cost less than C between x and the element vertices. So each path from the element vertex to x has cost 1, because it contains edge between x and the subset vertex. Each subset vertex is connected with edge of cost less than C to the exactly three element vertices so there are at least $3n/3 = n$ such paths. But because the spanning tree does not have cost more than n, this means that there are exactly n such paths. If we consider the elements of F corresponding to the subset vertices on these paths they will give us solution to the Exact Cover by 3-sets problem.

In Figure 2.1, if $S = \{1,2,3,4,5,6\}$ and $F = \{\{1,2,3\},\{2,3,4\},\{4,5,6\}\}$ $n = 2$, $m = 3$, Exact Cover by 3-sets has a solution $F' = \{\{1,2,3\},\{4,5,6\}\}$ The Minimum Spanning Tree with the Bounded Diameter D has cost 2 : it consists of all edges in the path between x and y, edges from x to the subset vertices $\{1,2,3\}$ and $\{4,5,6\}$, edges from the subset vertex $\{1,2,3\}$ and the element vertices 1, 2 and 3, edges from the subset vertex $\{4,5,6\}$ to the element vertices 4, 5 and 6 and also the edge between the subset vertex $\{1,2,3\}$ and the subset vertex $\{2,3,4\}$

Note that the graph we constructed for the proof does not have edges of the equal cost. The edges connecting x with the subset vertices have cost 1 and the edges in the path between x and y and the edges connecting two subset vertices or the subset vertex and the element vertex have cost 0. All other edges have some big constant cost. We can consider that if there is the edge of big constant cost between the vertices u and v then there is no edge at all between u and v. □

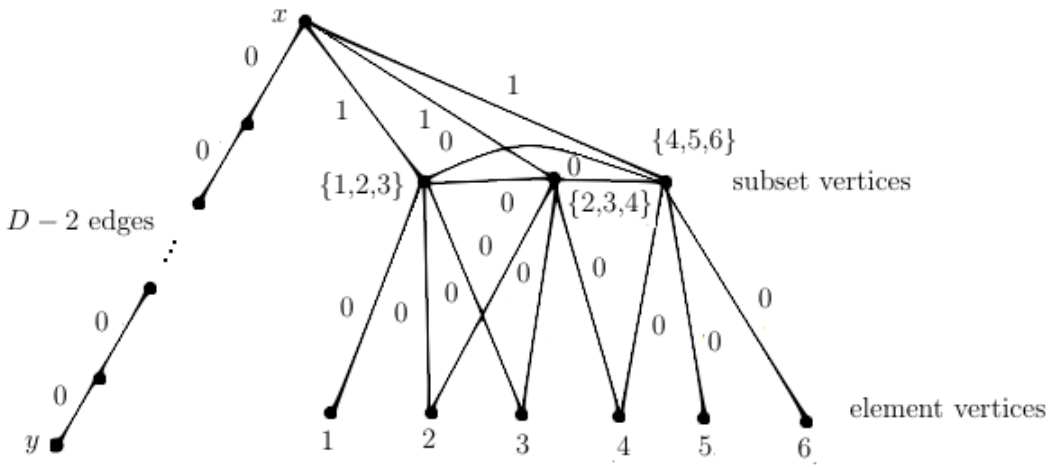


Figure 2.1

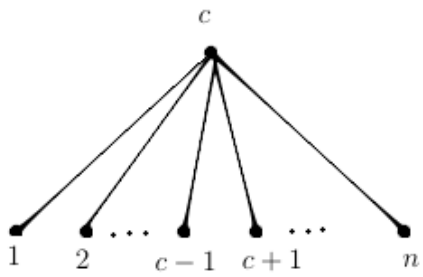


Figure 2.2

Theorem 2.2. *The Bounded Diameter Minimum Spanning Tree problem (Problem 1) is solvable in polynomial time if $D = 2$ or $D = 3$ or all edges of the graph have the equal cost (the graph is not necessarily complete) [5]*

Proof. If $D = 2$ let us take the path with the maximal number of edges in the minimum spanning tree, there will be no more than 2 edges. Let its middle vertex be c. We will call c center of the spanning tree. All other vertices should be connected to c with the edge because otherwise the spanning tree will have the diameter more than 2, see Figure 2.2, so the cost of the minimum spanning tree with the center vertex c is $\sum_{v=1}^{n}$ the cost of edge between v and c (we consider that cost of the edge between c and itself has value 0). So we find c for which this value is minimal and the spanning tree will be just

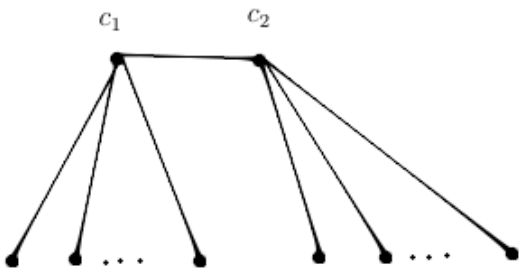


Figure 2.3

all the vertices(except c) connected with the edge to c. Running time of this algorithm is $O(n^2)$, because we have n choices for the center and we need $O(n)$ time to calculate the cost of connecting all other vertices to c. So the Problem 1 is in P when $D = 2$.

If $D = 3$, let us take the path with the maximal number of edges in the minimum spanning tree. There will be no more than 3 edges. Let us consider the middle edge of this path, and let c_1 and c_2 be the vertices this edge connects to each other. Every vertex in the spanning tree should be connected with the edge to c_1 or c_2 because otherwise the spanning tree will have the diameter more than 3, see Figure 2.3. Out of two possibilities we take one with the minimal cost, so cost of the minimum spanning tree with the diameter 3 and the center vertices c_1 and c_2 is $\sum_{v=1}^{n} \min$ (cost of the edge between c_1 and v, cost of the edge between c_2 and v). We choose c_1 and c_2 such that this value is minimal and to construct the minimum spanning tree we add the edge between c_1 and c_2 and for every vertex v except c_1 and c_2 we add the edge between c_1 and v if the cost of the edge between c_1 and v $<$ the cost of the edge between c_2 and v and we add the edge between v and c_2 otherwise. Running time of this algorithm is $O(n^3)$ because we have n^2 choices for the center vertices and we need $O(n)$ time to calculate the cost of connecting all other vertices with the edge to c_1 or c_2. So the Problem 1 is in P when $D = 3$.

If all edges of the graph have the same cost (the graph is not necessarily complete), this means that we just have to find the spanning tree with the diameter D (if it exists) and for every spanning tree cost is $(n-1) \cdot$ the cost of one edge. Let us take the path with the maximal number of edges in it. There will be no more than D edges. If D is even let the middle vertex of this path be c. c will be the center vertex of the spanning tree. If D is odd let us

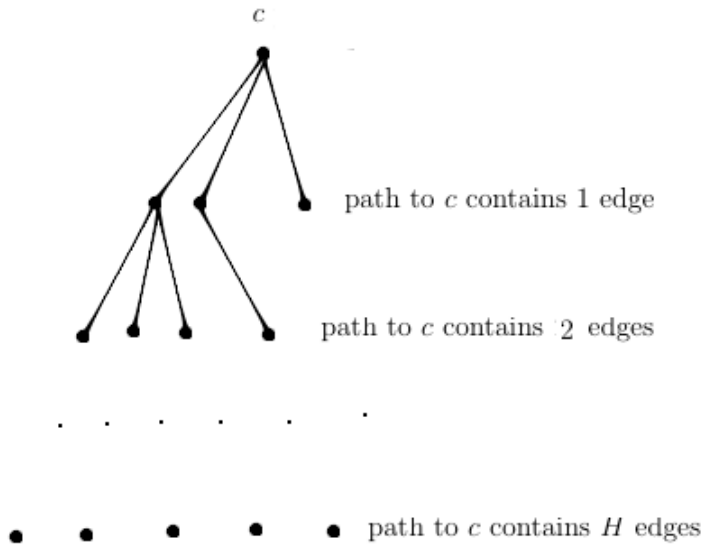


Figure 2.4

consider the middle edge and let c and c' be the vertices this edge connects to each other. Let $H = \lfloor D/2 \rfloor$. For every vertex v the path between v and c does not contain more than H edges if D is even and one of the paths between v and c or between v and c' does not contain more than H edges if D is odd, because otherwise the diameter will be more than D.

To find the spanning tree with the diameter D, we fix c and in the odd diameter case we fix c' also. Then we start from c and add all neighbouring vertices to the spanning tree(along with the edges between them and c). Then for each of the new added vertices we do the same as we did for c (we do not add the vertices we have already added and if D is odd we do not add c') . We repeat the same procedure for the new vertices until we add the vertices such that the path between them and c in the spanning tree has H edges, this means we can not add any more vertices, see Figure 2.4. In the case of odd D we do the same for c' (we do not add vertices we already added for c). If there are all vertices of the graph in the tree we get, then we have found the minimum spanning tree with the diameter D. If there are some vertices missing then there is no spanning tree of diameter D and the center vertices we have chosen.

We have n choices for c, in the odd D case we have n choices for c' also. The algorithm we described above for the fixed center vertices is called Breadth-first search and it takes $O(\mid E \mid +n)$ time [11]. So running time of

our algorithm is $O((\mid E \mid +n) \cdot n)$ if D is even and $O((\mid E \mid +n) \cdot n^2)$ if D is odd. So the Problem 1 is in P if all edges of the graph have the same cost. $\square$

Chapter 3

Bounded Diameter Minimum Spanning Tree for points on the real line, points on a circle and vertices in a tree

In this Chapter we will prove that the Bounded Diameter Minimum Spanning Tree for points on the real line is solvable in polynomial time for $D = 4$, $D = 5$ and then for $D > 5$. We will show that Points on a circle is also solvable in polynomial time and vertices in a tree case is solvable in polynomial time if we have a Caterpillar Tree.

Theorem 3.1. *The Bounded Diameter Minimum Spanning Tree Problem (Problem 1) where each vertex of the graph is a point on the real line and the cost function is the Euclidean distance between two points (Problem 2) is solvable in polynomial time if* $D = 4$.

Proof. If for some i and j $x_i = x_j$, we can add a very small number ϵ to x_i so that it does not change anything in the proof. So we can assume that the coordinates of the vertices are $x_1 < x_2 < .....x_n$. Let us take the path in the spanning tree containing the maximal number edges. Number of the edges in this path is not more than 4, so if we take the middle vertex of this path and call it center vertex of the spanning tree c, for every vertex the path to c contains no more than 2 edges.

Claim 3.2. *If vertex* u *is the child of* c *in the minimum spanning tree or* u *is* c*, let* l *be the vertex with the least coordinate -* x_l *in the subtree of the spanning tree with the root* u *and let* r *be the vertex with the biggest coordinate -* x_r *in the subtree of the spanning tree with the root* u*. Then there is no vertex* v

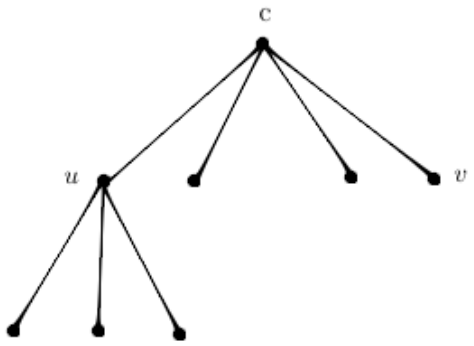


Figure 3.1

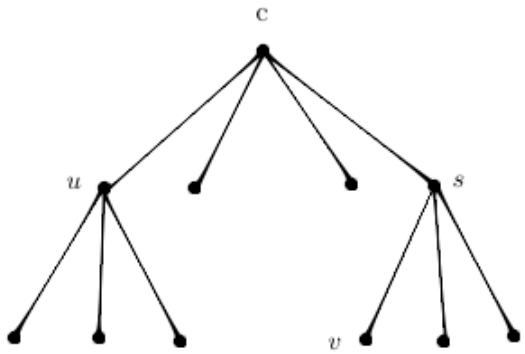
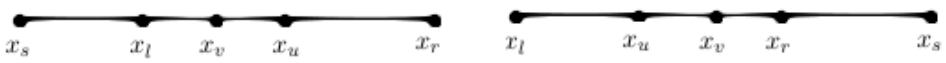


Figure 3.2

(a) (b)

Figure 3.3



(a) (b)

Figure 3.4

such that v does not belong to the subtree of the spanning tree with the root u and $x_l < x_v < x_r$

Proof. If u is c all vertices from the interval $[1, n]$ are in the subtree of the spannig tree with the root u, so there is no such vertex v. If u is the child of c then let us assume there is such vertex v. We have two cases:

Case 1. v is the child of c or v is c, see Figure 3.1
If $x_v \in [x_l, x_u]$, see Figure 3.3a. Then if we make v parent of l, we will get the spanning tree with the less cost than the minimum spanning tree and with the diameter 4, so we have a contradiction. If $x_v \in [x_u, x_r]$, see Figure 3.3b. Then if we make v parent of r we will get the spanning tree with the cost less than the minimum spanning tree and with the diameter 4, so we have a contradiction again.

Case 2. v is the child of the child of c, see Figure 3.2
Let s be the parent of v. If $x_l < x_s < x_r$ then we have Case 1 for vertex s and we have a contradiction again. If $x_v \in [x_l, x_u]$ then $x_s < x_l$, because otherwise we can make u parent of v and we will get the spanning tree with the less cost than the minimum spanning tree and with the diameter 4, so we have a contradiction again, see Figure 3.4a. $\mid x_s - x_l \mid \geq \mid x_u - x_l \mid$, because otherwise we can make s parent of l and we will get the spanning with the less cost than the minimum spanning tree and with the diameter 4, a contradiction $\Rightarrow \mid x_s - x_v \mid > \mid x_u - x_v \mid$, so if we make u parent of v we will have the spanning tree with the less cost than the minimum spanning tree and with the diameter 4, so we have a contradiction again. If $x_v \in [x_u, x_r]$ then $x_s > x_r$, because otherwise we can make u parent of v and we will get the spanning tree with the less cost than the minimum spanning tree and with the diameter 4, so we have a contradiction again, see Figure 3.4b. $\mid x_s - x_r \mid \geq \mid x_u - x_r \mid$, because otherwise we can make s parent of r and we will get the spanning with the less cost than the minimum spanning tree and with the diameter 4, a contradiction $\Rightarrow \mid x_s - x_v \mid > \mid x_u - x_v \mid$. So if we make u parent of v we will have the spanning tree with the less cost than the minimum spanning tree and with the diameter 4, so we have a contradiction again. □

Now using this property of the minimum spanning tree we can describe polynomial time algorithm: First we calculate $B[l, r, k]$ the cost of connecting k with the edge to the vertices from the interval $[l, r]$,
$B[l, r, k] = \sum_{i=l}^{r} \mid x_k - x_i \mid$. For all possible center vertices c, calculate $S[l, r, c]$ - the cost of connecting all vertices from the interval $[l, r]$ to c by choosing

one vertex $k \in [l, r]$ making k child of c, and making all other vertices from the interval $[l, r]$ children of k. $S[l, r, c] = \min_{l \le k \le r}(B[l, r, k] + \mid x_k - x_c \mid)$. Let

also $A[l, r, c]$ be the cost of connecting the vertices from the interval $[l, r]$ to c so that each vertex is the child or the child of the child of c. Let us take vertex k which is the child of c and x_k is maximal. The subtree of the minimum spanning tree with the root k contains vertices from the interval $[m, r]$ for some m and k is such that $S[m, r, c]$ is minimal. We need to connect the remaining vertices from the interval $[l, m-1]$ to c so this gives us the reccurence relation $A[l, r, c] = \min_{l \le m \le r}(A[l, m-1, c] + S[m, r, c])$. It can be that $m = l$
then we consider that $A[l, m-1, c]$ has value 0. First we calculate $A[1, 1, c]$, $A[2, 2, c]$, ..., $A[c-1, c-1, c]$, $A[c+1, c+1, c]$, ..., $A[n, n, c]$ - intervals containing one vertex (but not c). Then we calculate $A[1, 2, c]$, $A[2, 3, c]$, ..., $A[c-2, c-1, c]$, $A[c+1, c+2, c]$, ..., $A[n-1, n, c]$ - intervals containing two vertices (but not containing c). Then we calculate intervals containing 3 vertices (but not containing c) and so on we calculate intervals in the ascending number of vertices in it (intervals not containing c). For every c we need to connect vertices from the intervals $[1, c-1]$ and $[c+1, n]$ to c, so cost of the minimum spanning tree with root c and with the diameter 4 will be $A[1, c-1, c] + A[c+1, n, c]$. We choose c such that the minimum spanning tree with the root c and with the diameter 4 has the minimal cost.

This algorithm gives us the minimal cost of the spanning tree with the diameter 4, but not the spanning tree itself. To construct the spanning tree we find c for which $A[1, c-1, c] + A[c+1, n, c]$ was minimal and make it the center of the spanning tree. Then for the interval $[1, c-1]$ we find m such that $A[1, m-1, c] + S[m, c-1, c]$ was minimal , also we find k such that $S[m, c-1, c]$ was minimal and we add the edge connecting c and k. We also add the edges connecting vertices from the interval $[m, c-1]$ (except vertex k) and k. After that we repeat the same procedure with the vertices from the interval $[1, m-1]$ until all vertices from the interval $[1, c-1]$ are in the spanning tree. Then we use this procedure with the vertices from the interval $[c+1, n]$.

Running time of the algorithm is : To calculate B we have n choices for l, r, and k and $r-l+1$ choices for i, so it takes $O(n^4)$ time. We have n choices for c. For the fixed c to calculate $S[l, r, c]$ we have maximum n choices for l, also maximum n choices for r and we have $r-l+1$ choices for k. So it takes $O(n^3)$ time to calculate the values of S. For the fixed c to calculate $A[l, r, c]$ we have n choices for l also maximum n choices for r and we have $r-l+1$ choices for m. So it takes $O(n^3)$ time to calculate A also. So algorithm to find the cost of the minimum spanning tree with the diameter 4 takes $O(n^4)$

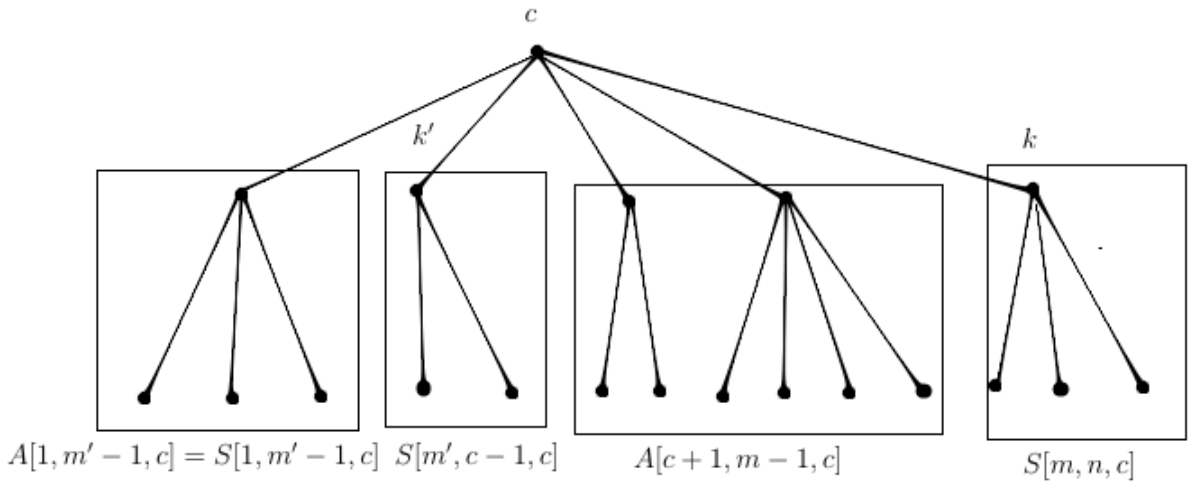


Figure 3.5

time. To find the minimum spanning tree itself, we need $O(n)$ time to find c for which $A[1, c-1, c] + A[c+1, n, c]$ was minimal. After we found c we do the same operations as when we where finding the cost of the minimum spanning tree but the number of vertices in the intervals is decreasing instead of increasing. Time taken is $O(n^3)$ again. Time of our algorithm is $O(n^4)$, so the Problem 2 is in P if $D = 4$

Example of the minimum spanning tree with the diameter 4, center vertex c and where c has five children can be seen in Figure 3.5

□

Now we consider the case when $D = 5$.

Theorem 3.3. *The Bounded Diameter Minimum Spanning Tree Problem (Problem 1), where each vertex of the graph is point on the real line and the cost function is the Euclidean distance between two points (Problem 2) is solvable in polynomial time if $D = 5$.*

Proof. let us consider the path with the maximal number of edges in it, not more than 5 edges. Let c_1 and c_2 be the vertices of the middle edge of the path. Then in the tree every vertex except c_1 and c_2 is the child of c_1 or c_2 or the child of the child of c_1 or c_2, because otherwise the minimum spanning tree will have the diameter more than 5. We will call c_1 and c_2 centers of the tree. We can use Claim 3.2 for every vertex which is the child of one of the center vertices(we have two center vertices but proof is the same), but

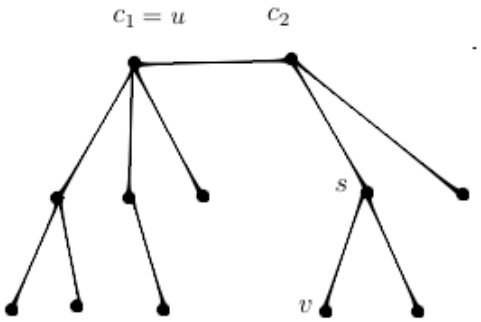


Figure 3.6

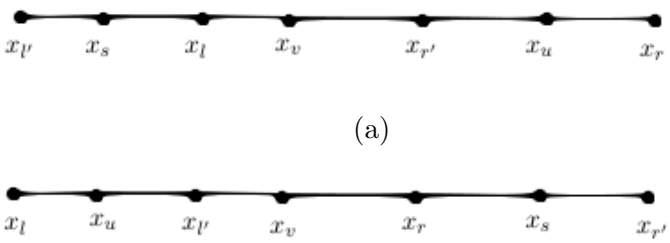


Figure 3.7

we can not use Claim 3.2 for the center vertices in this case because there the minimum spanning tree had just one center vertex and here we have two center vertices.

Claim 3.4. *If u is the center vertex of the minimum spanning tree, let l be the vertex with the least coordinate - x_l in the subtree of the spanning tree with the root u and let r be the vertex with the biggest coordinate - x_r in the subtree of the spanning tree with the root u. Then there is no vertex v such that v does not belong to the subtree of the spanning tree with the root u and $x_l < x_v < x_r$*

Proof. We will prove claim when u is c_1, it will be the same if u is c_2. Let us assume that there is such vertex v, let u_l be the child of u with the least coordinate - x_{u_l}, and let u_r be the child of u with the biggest coordinate x_{u_r}. We know that Claim 3.2 holds for u_l and u_r and all children of u. So if $l \neq u$ then $x_u \notin [x_l, x_{u_l}]$ because in this case l belongs to the subtree of

the minimum spanning tree with the root u_l. If $r \neq u$ then $x_u \notin [x_{u_r}, x_r]$, because in this case r belongs to the subtree of the minimum spanning with root u_r.

Case 1. v is c_2 itself. This case is the same as Case 1 in Claim 3.2, but if $l \neq u$ we use u_l instead of l and if $r \neq u$ we use u_r instead of r

Case 2. v is the child of c_2.This case is the same as Case 2 in Claim 3.2 with the difference we mentioned above.

Case 3. v is the child of the child of c_2, see Figure 3.6.
If $l \neq u$ we use u_l instead of l and if $r \neq u$ we use u_r instead of r. Let s be the parent of v. If $x_l < x_s < x_r$ then we have Case 2 for s and we have a contradiction. Let l' be the vertex with the smallest coordinate in the subtree of the minimum spanning tree with the root s - $x_{l'}$ and let r' be the vertex with the biggest coordinate in the subtree of the minimum spanning tree with the root s - $x_{r'}$, $x_{l'} \leq x_s, x_v \leq x_{r'}$. We know that we can use Claim 3.2 for s so there is no v' such that $x_{l'} < x_{v'} < x_{r'}$ and v' is not in the subtree of the minimum spanning tree with the root s. If $x_v \in [x_l, x_u]$ then $x_s < x_l$, because otherwise we can make u parent of v and we will have the spanning tree with the cost less than cost of the minimum spanning tree and with the diameter 5, a contradiction, see Figure 3.7a. Then we have $x_{l'} < x_l < x_{r'}$, so we have a contradiction again. If $x_v \in [x_u, x_r]$ then $x_s > x_r$ because otherwise we can make u parent of v and we will have the spanning tree with the cost less than cost of the minimum spanning tree and with the diameter 5, a contradiction, see Figure 3.7b. Then we have $x_{l'} < x_r < x_{r'}$, so we have a contradiction again. □

Now using this property of the minimum spanning tree we can describe polynomial time algorithm. We calculate $A[l, r, c]$ and $S[l, r, c]$ as in the proof of Theorem 3.1. For the center vertices c_1 and c_2, $c_1 < c_2$ and for some $c_1 \leq m \leq c_2 - 1$, the subtree of the spanning tree with the root vertex c_1 contains the vertices from the interval $[1, m]$ and the subtree of the spanning tree with the root c_2 contains the vertices from the interval $[m + 1, n]$. The cost of the minimum spanning tree with the center vertices c_1 and c_2 is
$\min_{c_1 \leq m \leq c_2 - 1}(A[1, c_1 - 1, c_1] + A[c_1 + 1, m, c_1] + A[m + 1, c_2 - 1, c_2] +$

$A[c_2 + 1, n, c_2]) + \mid x_{c_1} - x_{c_2} \mid$ and we choose the vertices c_1 and c_2 for which this value is minimal. We construct the minimum spanning tree itself by constructing the subtree of the spanning tree with the root c_1 using values of $A[1, c_1 - 1, c_1]$ and $A[c_1 + 1, m, c_1]$ and using the method described in the proof of Theorem 3.1, we do the same for c_2 and we also add the edge from c_1

to c_2 to the spanning tree. To calculate A and S we need $O(n^4)$ time. Then we have n choices for c_1, n choices for c_2 and $c_2 - c_1$ choices for m, so it takes $O(n^3)$ time to get the cost of the minimum spanning tree. To construct it takes $O(n^3)$ time also, so running time of our algorithm is $O(n^4)$. This shows that the Problem 2 is in P if $D = 5$ □

Theorem 3.5. *The Bounded Diameter Minimum Spanning Tree Problem (Problem 1), where each vertex of the graph is a point on the real line and the cost function is the euclidean distance between two points (Problem 2) is solvable in polynomial time.*

Proof. We have proved that the problem is in P if $D = 4$ and $D = 5$, so we have to prove that it is in P when $D > 5$. Let c be the center vertex of the minimum spanning tree with the diameter D if D is even and let c_1 and c_2 be the center vertices of the spanning tree with the diameter D if D is odd. Let $H = \lfloor D/2 \rfloor$ and let for vertex v, $h(v)$ be the number of edges in the path from v to c if D is even and if D is odd then let $h(v)$ be the number of edges in the path from v to c_1 if v belongs to the subtree of the minimum spanning tree with the root c_1 and the number of edges in the path from v to c_2 otherwise. $h(v) \leq H$ because otherwise the diameter of the minimum spanning tree will be more than D.

As in the case when $D = 4$ or $D = 5$, we will use the following claim to show that the minimum spanning tree has the property we can use.

Claim 3.6. *For every vertex u let l be the vertex with the least coordinate - x_l in the subtree of the spanning tree with the root u and let r be the vertex with the biggest coordinate - x_r in the subtree of the spanning tree with the root u. Then there is no vertex v such that v does not belong to the subtree of the spanning tree with the root u and $x_l < x_v < x_r$.*

Proof. We prove it by induction on $h(u)$. If u is a leaf Claim holds, if $h(u) = H$ then u is a leaf and Claim holds also. Now let the Claim hold for every vertex u' such that $h(u') > h(u)$ then it holds for u also. Let us assume that there is vertex v not in the subtree of the minimum spanning tree with root u such that $x_l < x_u < x_r$.

Case 1. $h(v) \leq h(u)$, this is the same as Case 1 in Claim 3.4

Case 2. $h(v) = h(u) + 1$, this is the same as Case 2 in Claim 3.4

Case 3. $h(v) > h(u) + 1$
Let s be the parent of v. We can assume that $x_s \notin [x_l, x_r]$ because otherwise we can consider s instead of v to get a contradiction. This Case is the same as the Case 3 in Claim 3.4, so we have a contradiction again. □

Now we can describe polynomial time algorithm. We calculate B as we did in the proof of Theorem 3.1. Let $A[l, r, p, h]$ be the cost of connecting the vertices from the interval $[l, r]$ to p, by choosing some vertex $k \in [l, r]$ making k the child of p, and making all the vertices from the interval $[l, r]$ members of the subtree of the minimum spanning tree with the root k, so that the subree with the root k we get does not have the height more than h (for every vertex $v \in [l, r]$ the path from v to k contains no more than h edges). If $l = r$ we have just one vertex and $A[l, r, p, h] =\mid x_p - x_l \mid$. If $h = 1$ this means that every vertex should be connected to k with the edge so $A[l, r, p, h] = \min_{l \le k \le r}(B[l, r, k] + \mid x_k - x_p \mid)$. Let $S[l, r, p, h]$ be the cost of

connecting vertices from the interval $[l, r]$ to p (making vertices from the interval $[l, r]$ members of the subtree of the spanning tree with the root p), So that for every vertex from the interval $[l, r]$ the path to p contains no more then h edges. If $h = 1$ this means that every vertex should be connected to p with the edge so $S[l, r, p, h] = B[l, r, p]$. If $l = r$ we have just one vertex and $S[l, r, p, h] =\mid x_l - x_p \mid$. If $h > 1$ when calculating $A[l, r, p, h]$ after we choose the vertex k we need to connect all other vertices from the interval $[l, r]$ to k the cost of it is $S[l, k-1, k, h] + S[k+1, r, k, h]$, so $A[l, r, p, h] = \min_{l \le k \le r}(S[l, k-1, k, h] + S[k+1, r, k, h] + \mid x_k - x_p \mid)$.

To calculate $S[l, r, p, h]$ when $h > 1$ let k be the child of p with the biggest coordinate. For some m the subtree of the spanning tree with the root k consists of the vertices from the interval $[m, r]$. These vertices are connected to p with the cost $A[m, r, p, h-1]$ and we need to connect the vertices from the interval $[l, m-1]$ to p, this has the cost $S[l, m-1, p, h]$ the total cost is $S[l, m-1, p, h] + A[m, r, p, h-1]$, so $S[l, r, p, h] = \min_{l \le m \le r}$ $(S[l, m-1, p, h] + A[m, r, p, h-1])$ (if $m = l$ we consider that $S[l, m-1, p, h]$ has value 0). We calculate A and S in the ascending order of the number of vertices in the interval $[l, r]$ as we did in the proof of theorem 3.1 and also in the ascending h order.

If D is even, we choose the center vertex c and the cost of the spanning tree with the center vertex c is $S[1, c-1, c, H] + S[c+1, n, c, H]$. We choose c such that this value is minimal. In the case of odd D we consider all possible center vertices c_1 and c_2. For some $m \in [c_1, c_2 - 1]$, the subtree of the spanning tree with the root vertex c_1 contains the vertices from the interval $[1, m]$ and the subtree of the spanning tree with the root vertex c_2 contains the vertices from the interval $[m+1, n]$, so the cost of the minimum spanning tree with the center vertices c_1 and c_2 is

$\min_{c_1 \le m \le c_2 - 1}(S[1, c_1 - 1, c_1, H] + S[c_1 + 1, m, c_1, H] + S[m+1, c_2 - 1, c_2, H] +$

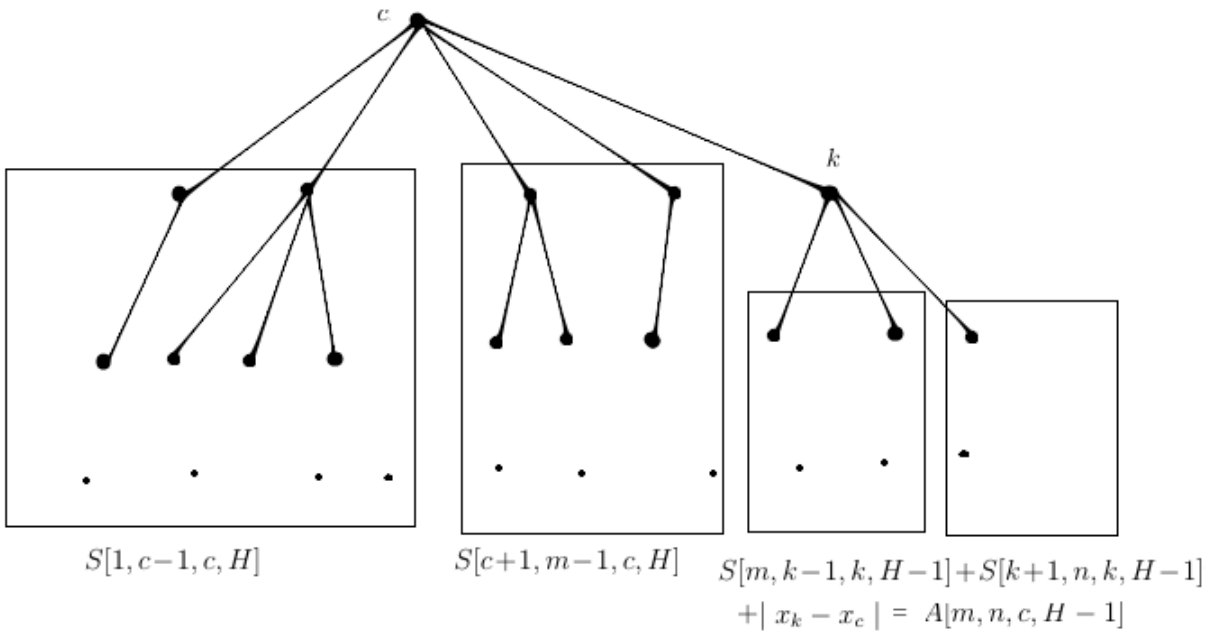


Figure 3.8

$S[c_2+1, n, c_2, H])+ \mid x_{c_1} - x_{c_2} \mid$. We choose c_1 and c_2 such that this value is minimal. To construct the minimum spanning tree we choose the center vertex c for which the spanning tree had the minimal cost. Then for the interval $[1, c-1]$ we choose m such that $S[1, m-1, c, H]+A[m, c-1, c, H-1]$ was minimal, also we choose k such that $S[m, k-1, k, H-1]+S[k+1, c-1, k, H-1]+ \mid x_k - x_c \mid$ was minimal and add the edge connecting c to k to the spanning tree. We do the same for the intervals $[1, m-1]$ and $[c+1, n]$. Then we do the same for the intervals $[1, k-1]$ and $[k+1, c-1]$ with the exception that we use k instead of c as the root vertex and H is decreased by 1. If D is even we choose c_1, c_2, and m such that the spanning tree had the minimal cost and repeat the same procedure for c_1 and vertices from the interval $[1, m]$, c_2 and the vertices from the interval $[m+1, n]$ as we did for c and vertices from the interval $[1, n]$.

Calculation of B takes $O(n^4)$ time. To calculate A we have n choices for l, r and p, H choices for h and $r-l+1$ choices for m (for k if $h=1$), so running time is $O(n^4 \cdot D)$. For S we have n choices for l, r and p, H choices for h and $r-l+1$ choices for m, so running time is $O(n^4 \cdot D)$ again. To find the centers takes even less time, also to construct the minimum spanning tree takes the same time as when we calculate A and S but the number of vertices in the intervals is decreasing in this case. So total running time of our algorithm is $O(n^4 \cdot D)$ and Problem 2 is in P. Example of the minimum spanning tree with the even diameter can be seen in Figure 3.8 □

Theorem 3.7. *The Bounded Diameter Minimum Spannng Tree Problem (Problem 1), where vertices are points located on a circle and the cost function is the length of the arc between two points (Problem 3) is solvable in polynomial time.*

Proof. If two points on the circle are the same. We can move one of them along the circle with a small distance ϵ so that it does not change anything in the proof. So we can assume that there are no two points which are the same.

We can not use the same algorithm as in the proof of Theorem 3.5, because there are no vertices with the least or the biggest coordinate. Let us assume that c is the center vertex of the minimum spanning tree if D is even and let c and c' be the centers of the spanning tree if D is odd.

Claim 3.8. *For every pair of vertices u, v, in the minimum spanning tree with the diameter D there is no edge between u and v, such that the arc we use to connect them with the edge contains the center vertex different from u and v (one of u and v can be the center vertex or both of them can be the center vertices if D is odd).*

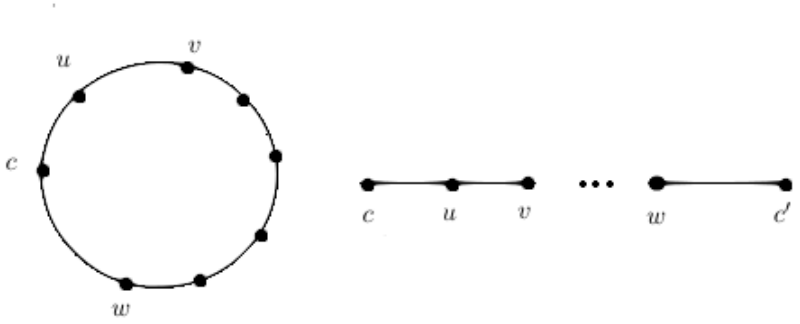


Figure 3.9

Proof. We will prove Claim for the center vertex c, it will be the same for c' in the odd diameter case. Let us assume that there are such vertices u and v , then after we remove the edge connecting u to v from the spanning tree, there will be no path connecting u to c or there will be no path connecting v to c, because otherwise there is a cycle in the spanning tree. Let us assume there is no path between u and c then we can add the edge connecting c to u. It has the cost less than the cost of the edge we removed, so we will get the spanning tree with the cost less than cost of the minimum spanning tree and with the diameter D, a contradiction. □

If D is even, we assign coordinates to the points to construct points on the real line case we will use. Let c have a coordinate 0. For each vertex v let the coordinate of v be the length of the arc between v and c. There are two arcs so we consider the one for which we have to travel clockwise from c to v, and we add additional vertex c'. The coordinate of c' is equal to the length of the circle, see Figure 3.9. For every pair of vertices u and v , which are not c or c', $\mid x_u - x_v \mid$ is the length of the arc between u and v which does not contain c, so by Claim 3.8 it is the arc used in the spanning tree if there is the edge between vertices u and v in the spanning tree. For every vertex u except c and c' $\mid x_c - x_u \mid$ is the length of the arc between c and u where we travel clockwise from c to u and $\mid x'_c - x_u \mid$ is the length of the arc between c and u where we travel counterclockwise from c to u. Let us consider the spanning tree T for points on a circle case with the even diameter D. We add the vertex c' to T, connect c' to c with the edge and for every child vertex u of c we make u child of c' if they are connected to c with the arc where we travel from c to u counterclockwise. The tree we get is the spanning tree with the center vertices c and c' and with the diameter $D + 1$ for points on the real line case we constructed and its cost will be the cost

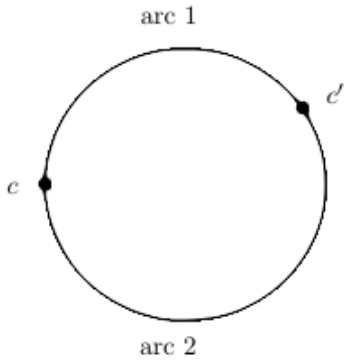


Figure 3.10

of $T+$ length of the circle(length of the circle is cost of the edge between c and c'). If we take the spanning tree T with the center vertices c and c' for points on the real line case we constructed, remove c' from T and for every child vertex u of c' make u child of c using the arc between c and u where we travel counterclockwise from c to u , we get the spanning tree with the diameter D for points on a circle case and its cost will be the cost of $T-$ length of the circle. So to find the minimum spanning tree with the diameter D and the center vertex c for points on a circle case, we find the minimum spanning tree with the diameter $D+1$ and the center vertices c and c' for points on the real line case we have constructed using the algorithm we used in the proof of Theorem 3.5 and then transform it as mentioned above. We choose c such that the minimum spanning tree has the minimal cost. We have n choices for c. Running time of points on the real line algorithm is $O(n^4 \cdot D)$, then running time of our algorithm is $O(n^5 \cdot D)$. So the problem 3 is in P if D is even

If D is odd , there are two arcs between c and c'. Let us call arc 1 the arc where we trevel clockwise from c to c' the other arc will be called arc 2. Let us assume that the length of arc1 $\leq$ the length of arc 2 (otherwise we can consider c instead of c' and c' instead of c), see Figure 3.10. For the vertices that lie on the arc 1, we construct points on the real line case as we did in the even D case, with the exception that we do not add additional c' as the point with the biggest coordinate because we already have it. If vertices u and v belong to the arc 1 and at least one of them is not the center vertex then $\mid x_u - x_v \mid$ is the length of the arc between u and v which does not contain any center vertex and by Claim 3.8 it is the arc used in the minimum spanning tree if there is the edge between u and v in it. $\mid x_c - x_{c'} \mid =$ the length of

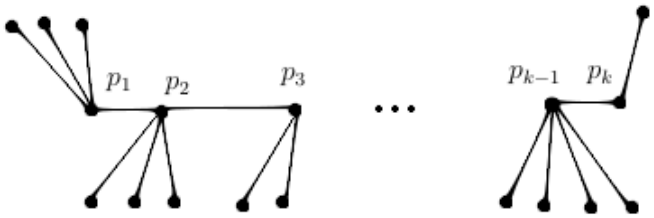


Figure 3.11

arc 1. For the arc 2 we do the same except that we use the counterclockwise direction instead of the clockwise. If vertices u and v belong to the arc 2 and at least one of them is not the center vertex then $\mid x_u - x_v \mid$ is the length of the arc between u and v which does not contain any center vertex, by Claim 3.8 it is the arc used in the minimum spanning tree if there is the edge between u and v in it. $\mid x_c - x_{c'} \mid =$ the length of arc 2. Let us take the minimum spanning tree for points on a circle case with the odd diameter D and the center vertices c and c'. Let T_1 be the subgraph of T induced by the vertices which belong to the arc 1 and let T_2 be the subgraph of T induced by the vertices which belong to the arc 2. Because there is no edge between vertex which belongs to the arc 1 and vertex which belongs to the arc 2(except for the center vertices), T_1 and T_2 are trees, $T_1 \cup T_2 = T$ and $T_1 \cap T_2$ is the edge connecting c and c'. Both T_1 and T_2 have the diameter D. The cost of $T =$ cost of T_1+ cost of T_2 $-$the length of arc 2 (cost of the edge connecting c and c' is the length of arc 1 in T_1 ,the length of arc 2 in T_2 and the length of arc 1 in T because arc 1 has the smaller length). The cost of T is minimal if the costs of T_1 and T_2 are minimal and vice versa. To find T_1 we use the algorithm we used in the proof of Theorem 3.5 for points on the real line case we constructed for arc 1, to find T_2 we use the algorithm we used in the proof of Theorem 3.5 for points on the real line case we constructed for arc 2. We have n choices for c, also n choices for c' and running time of points on the real line algorithm is $O(n^4 \cdot D)$, then running time of our algorithm is $O(n^6 \cdot D)$. So the Problem 3 is in P if D is odd. □

Tree is called a Caterpillar Tree if after we remove all leaf vertices from it we get a path, see Figure 3.11. Caterpillar trees were first studied by Harary and Schwenk [7].

Theorem 3.9. *Given a tree T with edge lengths, The Bounded Diamater Minimum Spanning Problem (Problem 1), where the vertices of the Graph are the vertices of T and the cost function is the length of the path between*

two vertices (Problem 4) is solvable in polynomial time if T is a Caterpillar Tree.

Proof. If one of the edges of T has the cost 0. We can add a very small number ϵ to the cost so that it does not change anything in the proof. So we can assume that there are no edges of cost 0 in T.

Let $p_1, p_2, ...p_k$ be vertices on the path. All the other vertices are leaf vertices, see Figure 3.11. Let h and H be the same as in the proof of Theorem 3.5, also let $Length(u, v)$ be the length of path between the vertices u and v in T.

Claim 3.10. *Leaf vertices do not have children in the minimum spanning tree. If leaf vertex l is connected with the edge to the vertex on the path p_i then in the minimum spanning tree l is the child of p_i if p_i is not a leaf in the minimum spanning tree, otherwise they have the same parent.*

Proof. Let l be a leaf vertex and let p be its parent. If l has at least one child in the minimum spanning tree. Let l be connected with the edge to the vertex on the path p_i. For every vertex q except l and p_i, $Length(l, p_i) < Length(l, q)$, because the path from q to l contains p_i. If $h(p_i) \leq h(l)$, we can make p_i the parent of the child of l , we will get the spanning tree with the cost less than cost of the minimum spanning tree and with the diameter D, so we have a contradiction. If $h(p_i) > h(l)$, we can make p_i child of p, l child of p_i and children of l children of p_i (if l is the center vertex we make p_i the center vertex, so if D is odd we add the edge connecting p_i to the the second center vertex), we will get the spanning tree with the cost less than cost of the minimum spanning tree and with the diameter D, we have a contradiction again. So leaf vertices do not have children in the minimum spanning tree. If $p_i \neq p$ then if $h(p_i) \neq H$ we can make p_i parent of l, we will get the spanning tree with the cost less than cost of the minimum spanning tree and with the diameter D, a contradiction. If $h(p_i) = H$, let p' be the parent of p_i, if $p' \neq p$, $Length(p', p_i) \leq Length(p, p_i)$, because otherwise we can make p parent of p_i and we will get the spanning tree with the cost less than cost of the minimum spanning tree and with the diameter D, a contradiction again. In case of equality after we make p parent of p_i we will get the spanning tree with the same cost as the minimum spanning tree and with the diameter D. We can consider it as the minimum spanning tree and claim will hold for l. If we do not have equality then $Length(p', p_i) + Length(p_i, l) < Length(p, p_i) + Length(p_i, l)$, so $Length(p', l) < Length(p, l)$ and after we make p' parent of l we will get the spanning tree with the cost less than cost of the minimum spanning tree and with the diameter D, a contradiction. □

We can assign coordinates to the path vertices to get points on the real line case we will use. Coordinate of p_i is $Length(p_i, p_1)$. So we can use the algorithm from the proof of Theorem 3.5 with a little modification : let $A[l, r, p, h]$ be the same as in the proof of Theorem 3.5. If $h > 1$ we add sum of the lengths of the edges connecting k to its neighbouring leaf vertices in T to it (because k is not a leaf) and if $h = 1$ or $l = r$, we add the sum of the lengths of the paths connecting k to the leaf vertices that are neighbours of the vertices from the interval $[l, r]$ in T to it (because all the vertices from the interval $[l, r]$ except k are leafs). Let $S[l, r, p, h]$ also be the same as in the proof of Theorem 3.5. If $h = 1$ or $l = r$ we connect vertices from the interval $[l, r]$ to p with the edges, so we add the sum of the lengths of the edges connecting p to its neighbouring leaf vertices in T to it (because p is not a leaf) and we also add the sum of the lengths of the paths connecting p to the leaf vertices that are neighbours of the vertices from the interval $[l, r]$ in T to it (because all the vertices from the interval $[l, r]$ are leafs). When we consider the center vertices we add the lengths of the edges connecting the center vertices to the neighbouring leaf vertices in T. Running Time of this algorithm is $O(n^4 \cdot D)$, because we use algorithm from the proof of 3.5 for the path vertices and during the algorithm for every leaf vertex we add it as the child of the neighbouring path vertex if it is not a leaf vertex in the spanning tree and we add it as the child of the parent of the neighbouring path vertex otherwise. So the Problem 4 is in P if T is a Caterpillar Tree. □

Chapter 4

Conclusion

In this thesis we showed that the Bounded Diameter Minimum Spanning Tree problem is $NP - hard$ when $D > 3$ and costs of all edges of the graph are not equal and otherwise it is solvable in polynomial time. We proved that the Bounded Diameter Minimum Spanning Tree problem is solvable in polynomial time when vertices of the graph are points on the real line, points on a circle or vertices in a Caterpillar Tree. The questions left to answer are : how to prove that the Euclidean instance of the Bounded Diameter Minimum Spanning Tree problem is also $NP - hard$ and what happens when the vertices of the graph are the vertices of a general tree not just a Caterpillar Tree, is there polynomial time algorithm to solve this problem or can we prove that it is $NP - hard$?

Bibliography

[1] Deo N. Abdalla A. and Gupta P. Random-tree diameter and the diameter constrained mst. *Congressus Numerantiun*, 144:161–182.

[2] Caccetta P. Achuthan N.R., Caccetta L. and J.F. Geelen. Computational methods for the diameter restricted minimum weight spanning tree problem. 1994.

[3] R. E. Bellman. *Dynamic Programming*. 1957.

[4] A. Bookstein and S. T. Klein. Compression of correlated bit-vectors. *Information Systems*, 16(4):387–400.

[5] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. 1979.

[6] Martin Gruber. *Exact and Heuristic Approaches for Solving the Bounded Diameter Minimum Spanning Tree Problem*. PhD thesis, Vienna University of Technology.

[7] F. Harary and A.J. Schwenk. The number of caterpillars. *Discrete Mathematics 6*, 4:359–365, 1973.

[8] J. B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proc. of the American Mathematics Society*, 7:48–50, 1956.

[9] R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technical Journal*, 36:1389–1401, 1957.

[10] K. Raymond. A tree-based algorithm for distributed mutual exclusion. *ACM Transactions on Computer Systems*, 7 (1):61–77.

[11] Ronald L. Rivest Clifford Stein Thomas H. Corman, Charles E. Leiserson. *Introduction to Algorithms*. Third edition.