Journal of Mathematical Modeling
Vol. 3, No. 2, 2016, pp. 129-144

JMM

An efficient algorithm for finding the semi-obnoxious (k, l)-core of a tree

Samane Motevalli[a], Jafar Fathali[b*] and Mehdi Zaferanieh[c]

[a]*Faculty of Mathematics, Shahrood University, Shahrood, Iran*
email: samane.motevalli@gmail.com

[b]*Faculty of Mathematics, Shahrood University, Shahrood, Iran*
email: fathali@shahroodut.ac.ir

[c]*Department of Mathematics, Hakim Sabzevari University, Sabzevar, Iran*
email: mehdi.zaferanieh@gmail.com

Abstract. In this paper we study finding the (k, l)-core problem on a tree which the vertices have positive or negative weights. Let $T = (V, E)$ be a tree. The (k, l)-core of T is a subtree with at most k leaves and with a diameter of at most l which the sum of the weighted distances from all vertices to this subtree is minimized. We show that, when the sum of the weights of vertices is negative, the (k, l)-core must be a single vertex. Then we propose an algorithm with time complexity of $O(n^2 logn)$ for finding the (k, l)-core of a tree with pos/neg weight, which is in fact a modification of the one proposed by Becker et al. [Networks 40 (2002) 208].

Keywords: Core, Facility location, Median subtree, Semi-obnoxious.
AMS Subject Classification: 90B90, 90B06.

1 Introduction

Classical location theory is concerned with the finding optimal location of a set of single points on a given network $G = (V, E)$. An important location problem of this kind is the p-median problem. In this problem we want to find a subset $X \subseteq V$ of cardinality p such that the sum of the weighted distances from X to all other vertices is minimized. The p-median problem

*Corresponding author.

Received: 20 April 2015 / Revised: 19 June 2015 / Accepted: 5 July 2015.

has been known to be NP-hard [15]. When the underlying network is a tree Kariv and Hakimi [15] showed that this problem can be solved in $O(p^2n^2)$ time (also see [18]). Tamir [25] improved the time complexity to $O(pn^2)$. For the case $p = 1$ Goldman [11] presented a linear time algorithm. For $p = 2$ an $O(n \log n)$ algorithm was given by Gavish and Sridhar [10]. Hassin and Tamir [13] proposed an $O(pn)$ algorithm for the p-median problem on a path with positive weights. Maimani [16] and Fathali et al. [9] considered the median location problem on special graphs.

In many real applications, the facility to be located is too large to be modeled as a point. These kinds of facilities called extensive facilities which have the form of a path or of a tree (see [12]). A path-center on a tree is a path which the maximum distance from all vertices to this path is minimized. Hedetniemi et al. [14] and Slater [23] considered the problem of finding a path-center on a tree. A core of a tree is a path of tree so that the sum of the weighted distances from all vertices to this path is minimized. For the case in which weight of all vertices is positive, Morgan and Slater [19] and Becker [1] presented linear time algorithms for finding a core of a tree. After that a number of authors considered problem with a constraint on length of path so that length can be at most l [2, 17, 20]. Peng et al. [21] extended the problem to finding k-tree core on the tree. A k-tree core is a subtree that minimizes the sum of the distances and which contains exactly k leaves. A linear time algorithm is gave for it by Shioura and Uno [22]. Zaferanieh and Fathali [28] applied meta-heuristic methods, ant colony and simulated annealing algorithm, for finding the core of a graph.

Problem of finding a core of a tree is a special case of problem of finding the (k, l)-core which in that $k = 2$ and there isn't any constraints on l. A (k, l)-core is a subtree of tree with at most k leaves and with a diameter of at most l so that the sum of the weighted distances from all vertices to the subtree is minimized. Becker et al. [2] presented an efficient algorithm for finding a (k, l)-core of a tree with time complexity of $O(n^2 logn)$. Their idea is that, by starting from the tree T, construct new rooted trees where the maximum length of a path is at most l. Then, for each new tree, apply a greedy-type procedure to find a subtree containing the root with at most k leaves and which minimizes the sum of the distances.

If all vertices have negative weights, the problem is the obnoxious location problem, and the median problem of this kind called maxian problem. Zelinka [29] show that the solution of 1-maxian problem, is obtained at a leaf of the tree and Ting [26] presented a linear algorithm for this problem. An algorithm for 1-maxian on general networks is provided by Church and

Garfinkel [8] with time complexity of $O(mnlogn)$, where m is the number of edges and n is the number of vertices of the network. The complexity of this algorithm was improved to $O(mn)$ by Tamir [24].

If weights of some of the vertices are positive and the others are negative or zero the problem is referred to as the semi-obnoxious location problem. Burkard and Krarup [7] showed that the pos/neg 1-median problem on a cactus can be solved in linear time. For the p-median problem on a network with pos/neg weight, Burkard et al. [4] introduced two different models to minimize: (1) the sum of the minimum weighted distances (MWD); and (2) the sum of the weighted minimum distances (WMD). For the case p = 2, they presented $O(n^2)$ and $O(n^3)$ time algorithms for MWD and WMD, respectively. Benkoczia et al. [3] improved the time complexity of MWD to $O(nlogn)$. Burkard and Fathali [5] extended the results for WMD in the case $p = 3$. Burkard et al. [6] presented a linear time algorithm for p-maxian problem with a first kind of objective function.

The problem of finding a core of a tree with pos/neg weights is considered in [27] by Zaferanieh and Fathali. They proved that, when the sum of the weights of vertices is negative, the core must be a single vertex and that, when the sum of the weights of vertices is zero, there exists a core that is a vertex. They also showed that the algorithm of Morgan and Slater [19] can be used to find a core of a tree with pos/neg weights.

In this paper, we consider the problem of finding a (k, l)-core of a tree with pos/neg weights. In what follows first we discuss some properties of the optimal solution for the (k, l)-core of a tree with pos/neg weights. In the case which the weight of tree is negative we show that the (k, l)-core is the solution of 1-median problem. Based on these observations a modification of the algorithm of Beker et al. [2] is proposed to solve the (k, l)-core problem with pos/neg weights on a tree. This algorithm has time complexity of $O(n^2logn)$.

2 Problem definition

Let $T = (V, E)$ be a tree, that $|V| = n$, $w(v_i)$ be the weight of vertex $v_i \in V$ (for simplicity we write w_i) and $a(i, j)$ be the length of edge (i, j). Then $w(T) = \sum_{i=1}^{n} w_i$ is the weight of the tree T. Also let $d(v_i, v_j)$ be the length of path from v_i to v_j, then the length of shortest path between path p and vertex v is given by

$$d(p, v) = \min_{u \in p} \; d(u, v).$$

The diameter d_T of T is the maximum distance between two vertices of T and any path whose length equals d_T is a diameter path.

Suppose $T' = (V', E')$ be a subtree of T. Let $d(v, T')$ be the minimum distance from $v \notin V'$ to a vertex in T'. We show the sum of distances from T' to all the vertices that they are not in V' by $d(T')$, that is called DISTSUM of T'. A (k, l)-core of T is a subtree T' with at most k leaves and with a diameter of at most l that the following function is minimized:

$$d(T') = \sum_{v \notin V'} d(v, T')$$

Applications of the semi-obnoxious (k, l)-core can found in the design of high speed communication networks such as railroad lines, high ways, subways, pipelines and transit routes, so that optimal subtree can be a composition of some paths, that is number of leaves and because of exits of negative and positive weights, lines are designed between the desirable(urban regions) and undesirable(military depots) vertices. The existing facilities with negative weights may represent depots of obnoxious materials. Clearly, the core should be located as far as possible from such facilities with negative weights. The other facilities (e.g. population centers) may have positive weights. The core should be located as close as possible to these facilities. For more information on and applications of the semi-obnoxious problems the reader is referred to [4].

3 Some properties of the problem

In this section, we extend the results which obtained by Zaferanieh and Fathali [27] for the core of a tree with pos/neg weight to the (k, l)-core case. Suppose weight of the tree T is non-positive, that is $w(T) \leq 0$. We investigate two cases separately: $w(T) < 0$ and $w(T) = 0$.

Theorem 1. *Let T be a tree with $w(T) < 0$. Then, the solution of 1-median problem is a (k, l)-core of the tree T and vice versa.*

Proof. Zaferanieh and Fathali [27] proved this theorem for the core of the tree. With attention to the proof of Theorem 1 in [27], since in the (k, l)-core problem we want to find a core with at most k leaves and with the diameter of at most l, if we choose a 1-median as the core, then by adding any other vertex to the core, The amount of objective function will be increased. So the theorem holds. □

For the case $w(T) = 0$ in [27] is shown that there exist a core which is a 1-median, but in this case we may not have a (k, l)-core which is 1-median. In fact it may have more than one vertex, see Example 3.

Let u be a vertex of tree T and v be an adjacent vertex to it, then let T_{uv} be the sub-tree of T obtained by deleting edge $[u\ v]$ so that $v \in T_{uv}$. In the case $w(T) < 0$, since both core and (k,l)-core are 1-median, then similar to properties which proved in [27] for the core case, we can conclude the following lemmas.

Lemma 1. *Let T be a tree with a negative weight. Two cases may occur:*

1. *The (k,l)-core(1-median) is a leaf.*
2. *The (k,l)-core(1-median) is an inner vertex u so that $w(u) \geq 0$ and $w(T_{uv}) \leq 0$ for each vertex v adjacent to u.*

Lemma 2. *Let T be a tree with $w(T) \leq 0$. If u is a leaf such that $w(T) - 2w(u) > 0$, then u is not a (k,l)-core.*

Proof. Suppose that v is a vertex adjacent to u, then $d(v) = d(u) + (2w(u) - w(T))d(u,v)$. So $d(v) < d(u)$, and therefore u cannot be a (k,l)-core of the tree T. □

Now consider the tree with positive weight. Again since core is a special case of (k,l)-core, the same as property of core in [27], the following lemma can be stated.

Lemma 3. *Let T be a tree with $w(T) > 0$ and u be a 1-median. If there exists a vertex v adjacent to u such that $w(T_{uv}) > 0$ and $d(u,v) \leq l$, then each (k,l)-core has more than one vertex.*

Proof. In the proof of Lemma 3 in [27] has been shown that the objective function of core for edge $[u,v]$ is less than the single vertex u. Therefore since $d(u,v) \leq l$ then by adding v to the 1-median u the value of the objective function for the (k,l)-core is decreased. So the (k,l)-core should have more than one vertex. □

Before express the next theorem we need some definitions. Let q be an edge of tree and T_1 and T_2 be the two subtrees of T which obtained by removing q. Also let T_c be a subtree in T_2 and x the closest vertex of T_1 to T_c . We divide the vertices of T_2 to three parts V_{21}, V_{22} and V_c, so that V_{21} is the set of vertices of T_2 that the shortest path from them to x does not pass through T_c and V_{22} is the set of vertices of T_2 that the shortest path from them to x passes through T_c. Also let V_c be the set vertices of T_c (see Fig. 1), and let s be the nearest vertex of T_c to x. We indicate by l_{vi} the length of path from v_i in T_c to s and $u(v_i)$ the closest vertex in T_c to v_i.

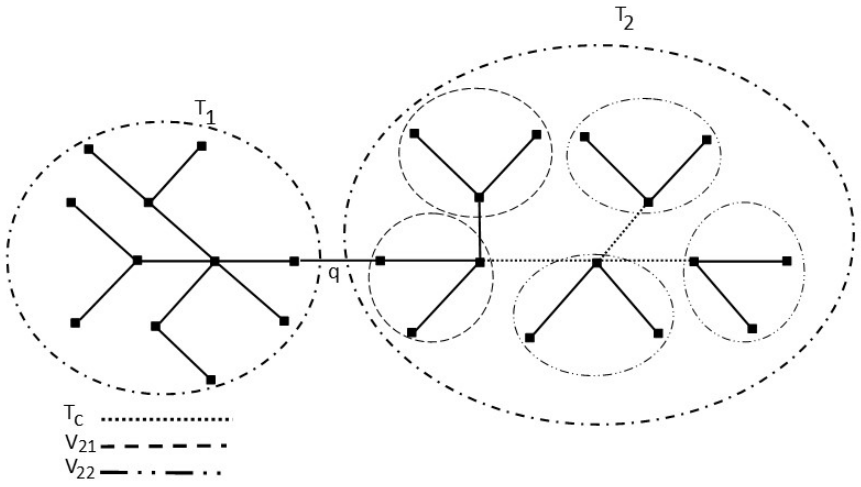


Figure 1: Dividing tree T.

Theorem 2. *Let $T = (V, E)$ be a tree with $W(T) > 0$ and q be an edge of T. Suppose by removing q two subtrees T_1 and T_2 are obtained so that $W(T_1) > W(T_2)$. If T_c be a subtree in T_2 and x the nearest vertex of T_1 to T_c such that*

$$d(x, T_c) \geq \frac{\sum_{v_i \in T_c} w(v_i) l_{vi} + \sum_{v_i \in V_{21}} w(v_i) l_{u(v_i)}}{W(V_1) - W(V_{22}) - W(c)}$$

and $W(V_{21}) > 0$ then T_c is not a (k, l)-core of the tree T.

Proof. Let T_c be the (k, l)-core and x the nearest vertex of T_1 to T_c, then

$$F(T_c) = \sum_{v_i \in V} w(v_i) d(v_i, T_c) = \sum_{v_i \in T_1} w(v_i) d(v_i, T_c) + \sum_{v_i \in T_2} w(v_i) d(v_i, T_c)$$

$$= \sum_{v_i \in T_1} w(v_i)[d(v_i, x) + d(x, T_c)] + \sum_{v_i \in T_2} w(v_i) d(v_i, T_c) = W(V_1) d(x, T_c)$$

$$+ \sum_{v_i \in T_1} w(v_i) d(v_i, x) + \sum_{v_i \in V_{21}} w(v_i) d(v_i, T_c) + \sum_{v_i \in V_{22}} w(v_i) d(v_i, T_c)$$

and,

$$F(x) = \sum_{v_i \in V} w(v_i) d(v_i, x) = \sum_{v_i \in T_1} w(v_i) d(v_i, x) + \sum_{v_i \in T_2} w(v_i) d(v_i, x)$$

$$= \sum_{v_i \in T_1} w(v_i) d(v_i, x) + \sum_{v_i \in V_{21}} w(v_i) d(v_i, x) + \sum_{v_i \in T_c} w(v_i)[l_{vi} + d(T_c, x)] +$$

$$\sum_{v_i\in V_{22}} w(v_i)[d(v_i,T_c)+l_{u(v_i)}+d(T_c,x)] = \sum_{v_i\in T_1} w(v_i)d(v_i,x)+\sum_{v_i\in V_{21}} w(v_i)d(v_i,x)$$

$$+[W(c)+W(V_{22})]d(T_c,x)+\sum_{v_i\in T_c} w(v_i)l_{vi}+\sum_{v_i\in V_{22}} w(v_i)l_{u(v_i)}$$

Since

$$d(x,T_c)\geq \frac{\sum_{v_i\in T_c} w(v_i)l_i+\sum_{v_i\in V_{21}} w(v_i)l_{u(v_i)}}{W(V_1)-W(V_{22})-W(c)}$$

so,

$$[W(V_1)-W(V_{22})-W(c)]d(x,T_c)\geq \sum_{v_i\in T_c} w(v_i)l_i+\sum_{v_i\in V_{21}} w(v_i)l_{u(v_i)}$$

and

$$W(V_1)d(x,T_c)\geq \sum_{v_i\in T_c} w(v_i)l_{vi}+\sum_{v_i\in V_{21}} w(v_i)l_{u(v_i)}+[W(V_{22})+W(c)]d(x,P_c).$$

Therefore $F(T_c)\geq F(x)$ which is in contradiction with the definition of the subtree T_c as a (k,l)-core. □

Note that since a core is a special case of a (k,l)-core, therefore Theorem 2 is also true for the core.

4 Algorithm

An algorithm with complexity of $O(n^2 logn)$ for finding a (k,l)-core of a tree with only positive weights is presented by Becker et al. [2]. We modify their algorithm to make it applicable to the trees with pos/neg weights.

Let P_{uv} be the path between vertices u and v in the tree T and T^u be the tree T rooted at u. We also denote by T^u_v the rooted subtree of T^u containing v and all the descendants of v.

To find the (k,l)-core of a tree, as in the papers of Peng and Lo [20] and Becker et al. [2], we use a recursive algorithm by starting a central vertex. A central vertex v of the tree T is the centroid of the corresponding unweighted tree of T, which is, a vertex that minimizes the maximum number of vertices of the subtrees obtained by removing it.

After finding the central vertex v, the algorithm finds a (k,l)-core containing v in the tree T^v. If it is not the (k,l)-core of the tree T, then the (k,l)-core must lie entirely in one of the subtrees rooted at the adjacent vertices of v. The algorithm is recursively applied to these subtrees.

Note that if $w(T)<0$ then according to the Theorem 1 the (k,l)-core is a vertex. So the algorithm should find a vertex as a (k,l)-core. Also in this case, according to Lemma 1, the target vertex is either a leaf or an inner vertex u so that $w(u) \geq 0$ and $w(T_{uv}) \leq 0$ for each vertex v adjacent to u. Therefor in the algorithm after finding a central vertex, if central vertex isn't a leaf and $w(u) \geq 0$, subtrees that are adjacent to it should be checked. If all of them had negative weight, so the central vertex is the (k, l)-core too. Otherwise, the central vertex is removed and the subtrees obtained by deleting them one by one so checked.

Before introducing the algorithm, we give the following notations as in [2].

let $f(u)$ be the father of u. The distance saving $sav(v, P_{vu})$ obtained by adding P_{vu} to the root v in T^c_v, is given by

$$sav(v, P_{vu}) = sav(v, P_{vf(u)}) + a(f(u), u)sum_c(u)$$

where if $v = f(u)$ then $sav(v, P_{vf(u)}) = sav(v, v) = 0$ and $sum_c(u)$ is calculated by

$$sum_c(v) = \begin{cases} w(v) & if\ v\ is\ a\ leaf\ of\ T_c \\ w(v) + \sum\limits_{u\ a\ child\ of\ v} sum_c(u) & Otherwise \end{cases} \tag{1}$$

Let the path $P = P_{vu}$ be given and let B be the set of children of v. We denote by $T^v_{\bar{b}}$ with $b \neq \bar{b}$, the subtree in which P lies. In the algorithm, the tree T^v is pruned as follows:

1. Prune the paths that belong to subtrees T^v_b, with $b \neq \bar{b}$, in order to obtain paths with length at most $min\{l - L(P), L(P)\}$.
2. For the paths that lie in the same subtree as $P = P_{vu}$ (i.e.,in $T^v_{\bar{b}}$), prune those paths P_{xw}, with $x \in P$, $x \neq v$ and $w \in T^v_{\bar{b}} \setminus P$, such that $L(P_{xw}) > min\{L(P) - L(P_{vx}), l - L(P_{xu})\}$

The pruned tree is called $\hat{T^v}$, and the weights of tree $\hat{T^v}$ is calculated by the following function:

$$w'(u) = \begin{cases} w(u) & for\ each\ vertex\ u\ of\ \hat{T^v}\ that\ is\ not\ a\ leaf \\ sum_v(u)a(u, f(u)) & for\ each\ vertex\ u\ that\ is\ a\ leaf\ of\ \hat{T^v} \end{cases} \tag{2}$$

Algorithm (Beker et al. [2])
Input: a tree T with pos/neg weight

```
Output: a $(k,l)$-core $S^*$ of $T$ and its DISTSUM $d^*$
begin
    $d^* := +\infty$
    SUBTREE(T)
end
```

```
    Procedure $SUBTREE(T')$
Input: a subtree $T' = (V', E')$ of T with $\mid V' \mid = n'$ and the best current
DISTSUM $d^*$
Output: if the best subtree in $T'$ has DISTSUM less than the previous
value of $d^*$, the best subtree $S^*$ in $T'$ , having at most $k$ leaves and with a
diameter of at most $l$ and its DISTSUM as the new value of $d^*$
begin
    if $T'$ consists of one vertex then
        $S' = T'$ and let $d(S')$ be its DISTSUM
    else
        find a central vertex $v$ of $T'$
        if $w(T) < 0$
          if $v$ be an inner vertex and $w(v) \geq 0$ and for each vertex $u$ adja-
cent             to $v$, $w(T_{vu}) \leq 0$
          then $v$ is the $(k,l)$-core
           let $S' = v$ and let $d(S')$ be its DISTSUM
        else let $S'$ := BEST-TREE$(T', v)$
     if $d(S') < d^*$
         $d^* := d(S')$
         $S^* := S'$
    for each subtre $T^i$ obtained from $T'$ by removing $v$
        do
        SUBTREE$(T^i)$
end
```

```
    Procedure $BEST - TREE(T', v)$ (Beker et al. [2])
Input: a subtree $T' = (V', E')$ of $T$ with $n'$ vertices rooted at the central
vertex $v$
Output: the best subtree $S'$ containing $v$ in $T'^v$ having at most $k$ leaves
and
          a diameter of at most $l$ and its DISTSUM $d(S')$
begin
    find the paths starting from $v$ with length at most $l$
    for each path $P$ starting from $v$ with length at most $l$
```

```
        do
        prune the tree $T'^v$ and let $\hat{T'^v}$ be the new tree (see Prune)
        find the distance savings of the paths from v to each vertex $u \neq v$
in $\hat{T}'^v$
        if $deg(v) \geq 2$ then
          find the path $P'$
          find $P''^{=p \cup p\prime}$
        else
          $P'' = P$
          find the set $LS$ with respect to $P''$
        if $\hat{T}'^v$has more than $k$ leaves then
          select in $LS$ the $k-2$ paths to be added to $P''$
          let $S'$ be the best subtree containing $v$ and $d(S')$ be its DISTSUM
        else
          select all the paths in $LS$
          let $S' = \hat{T'^v}$ and let $d(S')$ be its DISTSUM
    end for
end
```

Note that the main difference between our algorithm and algorithm with positive weight is the case that $W(T) < 0$. However the time complexity for this case is no more than the other cases therefore the time complexity will be the same as Beker et al. [2] algorithm and we can state the following theorem.

Theorem 3. *The time complexity of presented algorithm is $O(n^2 logn)$.*

5 Numerical examples

In this section we use the algorithm for a different case of the total weight of a tree.

Example 1. Consider the tree depicted in Fig. 2. The edge lengths are written on the edges. Let the weights of vertices be as showed in the Table 1. The total weight of tree is $w(T) = 1 > 0$. Let we want to find the (3,4)-core on this tree. According to algorithm we should find a central vertex. We start with v_{12} as a central vertex. Then we should find all of the paths starting from v_{12} with length at most 4, this paths are showed in Fig. 3.

Now for each path prune the tree T and continue steps of algorithm. For example consider the path number 2, new tree after prune is presented in fig. 4.

w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}	w_{11}	w_{12}
-2	-1	4	1	3	-1	-2	1	-2	-3	2	1

Table 1: The weights of vertices of tree in Fig. 2 for Example 1.

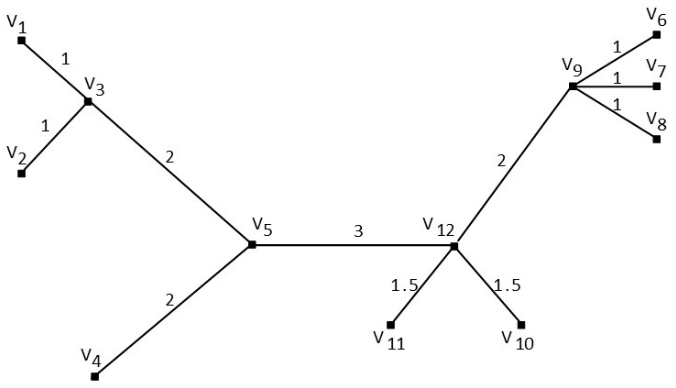


Figure 2: A tree with 12 vertices.

The distance savings of $p : v_{12}\text{-}v_{10}$ and $p : v_{12}\text{-}v_{11}$ is $sav(v_{12}, p_{v_{12}v_{10}}) = -4/5$ and $sav(v_{12}, p_{v_{12}v_{11}}) = +3$. Since $deg(v_{12}) = 2$ we consider $p' = v_{12} - v_{10}$ so $p'' = p \cup p'$ that is showed in Fig. 5. Also since $T^{\wedge' v}$ has 2 leaves so we select all the paths in LS so $S' = T^{\wedge' v}$ and $d(S') = 6$. After performing the above steps for each path, we consider subtrees obtained by removing of v_{12} one by one (see Fig. 6) and find a central vertex for each subtree and continue above steps for each subtree.

Finally the subtree that is showed in Fig. 7, is (3,4)-core and its distsum is equal to -26.5.

Example 2. Now consider the Fig. 2 with the weights of vertices are as showed in Table 2. In this case we have $w(T) = -1 < 0$. Therefore the (3,4)-core should be a vertex. We obtain that the vertex v_4 is (3,4)-core and $d(v_4) = -33.5$.

w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}	w_{11}	w_{12}
-2	-2	3	1	3	-2	-3	2	1	-1	-2	1

Table 2: The weights of vertices of tree in Fig. 2 for Example 2.

Example 3. Again consider the Fig. 2, where the weights of vertices are showed in the Table 3. The sum of weights of all vertices in this case is

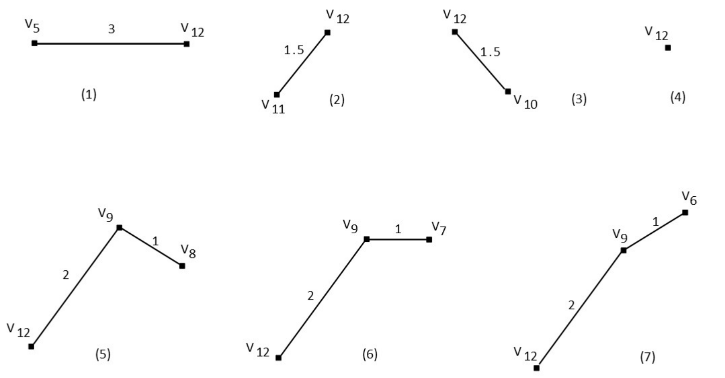


Figure 3: The paths starting from v_{12} with length at most 4.

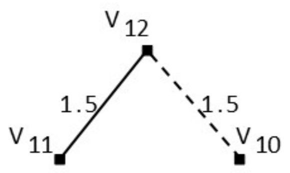


Figure 4: The path number 2 after prune.

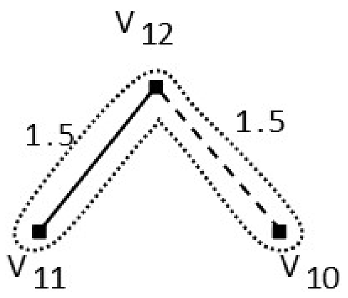


Figure 5: The path p''.

$w(T) = 0$. The (3,4)-core is the presented in Fig. 8. Its distsum is equal to -12. Note that in this case the weight of tree is zero, however the (3,4)-core is not a vertex.

w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}	w_{11}	w_{12}
-1	1	-1	-1	2	1	1	1	-3	1	1	-2

Table 3: The weights of vertices of tree in Fig. 2 for Example 3.

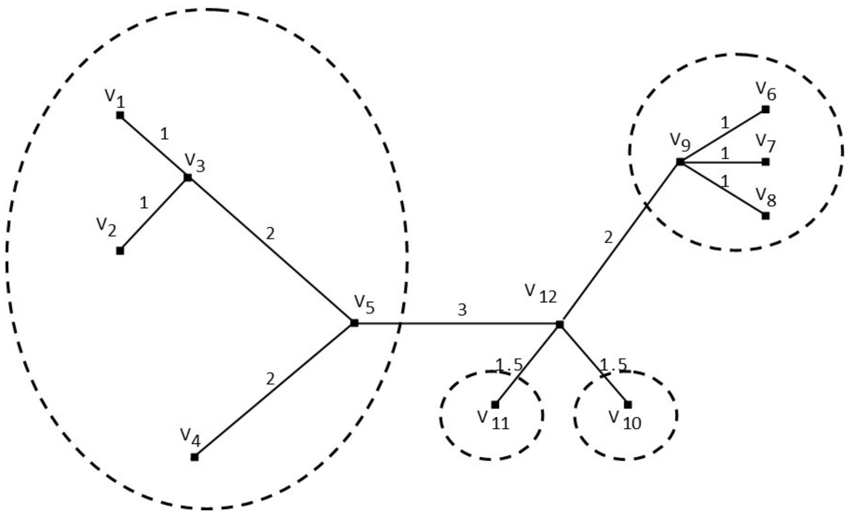
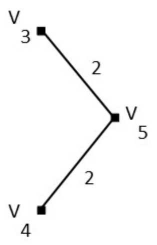


Figure 6: Subtrees obtained by removing of v_{12}.



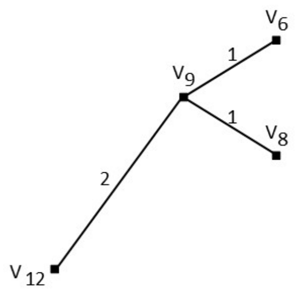


Figure 7: The (3,4)-core. Figure 8: The (3,4)-core.

6 Summary and conclusion

In this paper we considered the (k,l)-core problem on a tree with pos/neg weights. We showed some properties for this problem and modified the previously presented polynomial time algorithm to find the solutions of this problem. The time complexity of modified algorithm is the same as previous one, i.e. $O(n^2 logn)$.

References

[1] R. I. Becker, *Inductive algorithms on finite trees*, Quaest. Math. **13** (1990) 165–181.

[2] R. I. Becker and I. Lari and G. Storchi and A. Scozzari, *Efficient algorithms for finding the (k, l)-core of tree networks*, Networks **40** (2002) 208–215.

[3] R. Benkoczia, B.K. Bhattacharya, D. Breton, *Efficient computation of 2-medians in a tree network with positive/negative weights*, Discrete Math. **306** (2006) 1505–1516.

[4] R.E. Burkard, E. Çela, H. Dollani, *2-Median in trees with pos/neg weights*, Discrete Appl. Math. **105** (2000) 51–71.

[5] R.E. Burkard, J. Fathali, *A polnomial method for the pos/neg weighted 3-median problem on a tree*, Math. Meth. Oper. Res. **65** (2007) 229–238.

[6] R.E. Burkard, J. Fathali, H.T. Kakhki, *The p-maxian problem on a tree*, Oper. Res. Lett. **35** (2007) 331–335.

[7] R.E. Burkard, J. Krarup, *A linear algorithm for the pos/neg-weighted 1-median problem on a cactus*, Computing **60** (1998) 193–215.

[8] R.L. Church, R.S. Garfinkel, *Locating an obnoxious facility on a network*, Transport. Sci. **12** (1978) 107–118.

[9] J. Fathali, N. Jafari-Rad, S. R. Sherbaf, *The p-median and p-center problems on bipartite graphs*, Iranian Journal of Mathematical Sciences and Informatics **9** (2014) 37–43.

[10] B. Gavish, S. Sridhar, *Computing the 2-median on tree networks is $O(n \log n)$ time*, Networks **26** (1995) 305–317.

[11] A.J. Goldman, *Optimal center location in simple networks*, Transport. Sci. **5** (1971) 212–221.

[12] S.L. Hakimi, E.F. Schmeichel and M. Labbe, *On locating path-or tree-shaped facilities on networks*, Networks *23* (1993) 543–555.

[13] R. Hassin and A. Tamir, *Improved complexity bounds for location problems on the real line*, Oper. Res. Lett. **10** (1991) 395–402.

[14] S.M. Hedetniemi, E.J. Cockaine, S.T. Hedetniemi, *Linear algorithms for finding the jordan center and path center of a tree*, Transport. Sci. **15** (1981) 98–114.

[15] O. Kariv, S.L. Hakimi, *An algorithmic approach to network location problems. Part II: p-medians*, SIAM J. Appl. Math. **37** (1979) 539–560.

[16] H. R. Maimani, *Median and center of zero-divisor graph of commutative semigroups*, Iranian Journal of Mathematical Sciences and Informatics **3** (2008) 69–76.

[17] E. Minieka and N.H. Patel, *On finding the core of a tree with a specified length*, J. Algorithms **4** (1983) 345–352.

[18] P.B. Mirchandani, R. Francis, *Discrete Location Theory*, J.Wiley, 1990.

[19] C.A. Morgan, P.J. Slater, *A linear algorithm for a core of a tree*, J. Algorithms **1** (1980) 247–258.

[20] S. Peng, W. Lo , *Efficient algorithms for finding a core of a tree with a specified length*, J. Algorithms **20** (1996) 445–458.

[21] S. Peng, A.B. Stephens, and Y. Yesha, *Algorithms for a core and a k-tree core of a tree*, J. Algorithms **15** (1993) 143–159.

[22] A. Shioura and T. Uno, *A linear time algorithm for finding a k-tree core*, J. Algorithms **23** (1997) 281–290.

[23] PJ. Slater , *Locating central paths in a graph*, Transport. Sci. **16** (1982) 1–18.

[24] A. Tamir, *Obnoxious facility location on graphs*, SIAM J. Discrete Math. **4** (1991) 550–567.

[25] A. Tamir, *An $O(pn^2)$ algorithm for the p-median and related problems on tree graphs*, Oper. Res. Lett. **19** (1996) 59–64.

[26] S.S. Ting, *A linear-time algorithm for maxisum facility location on tree networks*, Transport. Sci. **18** (1984) 76–84.

[27] M. Zaferanieh and J. Fathali, *Finding a core of a tree with pos/neg weight*, Math. Meth. Oper. Res. **26** (2012) 147–160.

[28] M. Zaferanieh and J. Fathali, *Ant colony and simulated annealing algorithms for finding the core of a graph*, World. Appl. Sci. J. **7** (2009) 1335–1341.

[29] B. Zelinka, *Medians and peripherians of trees*, Archvium Mathematicum **4** (1968) 87–95.