Distributed Approximation Allocation Resources Algorithm for Connecting Groups

Fabien Baille[1], Lelia Blin[2,⋆], and Christian Laforest[2]

[1] LIAFA, Univ. Denis Diderot- Case 7014 2, place Jussieu, F-75251 Paris Cedex 05
[2] Tour Evry II, IBISC, Univ. d'Evry, 523 place des terrasses, 91000 EVRY, France
lelia.blin@lami.univ-evry.fr

Abstract. This paper presents a distributed algorithm to allocate resources (links of a network) for interconnecting machines (forming a group) spread in a network. This is what we call a *connection structure* for this *group* of machines. An important innovative feature of our construction method is that we *prove* (not just simulate on particular and restricted cases) the fact that this structure has good properties in terms of, *simultaneously*, induced distances (for latency considerations) and cost (for cost considerations). Hence, we propose a *distributed multicriteria approximation* algorithm.

In applications like video-conferences or net-meetings, *members* of a *group* spread in a network, have to communicate with high QoS requirements. A possibility for a *provider* selling this service on his network is to allocate/rent resources (links) for the exclusive use of the members. We call this a *connection structure* for the group. We propose here a distributed protocol to construct connection structure with *high guaranty* of quality.

Minimizing induced distances in the structure. To optimize QoS requirements on latency between members, we want to construct a structure in which distances are minimized. However, these distances cannot be smaller than those of the underlying network. Hence, we want to design a structure in which the induced distances between members are as close as possible to those of the original graph. To capture this desired property, we focus in this paper on the minimization of two criteria, namely the *maximum* and *average* distances between members.

The cost of the structure. Steiner tree. As a connection structure is a definitive allocation of links, exclusively reserved for the group, these resources are not available for others applications during the existence of the group. Hence, the provider allocating the structure must *minimize* the total number of links

⋆ Corresponding author.

used in the structure in order to minimize its exploitation costs and to keep the maximum number of available links for others services. In this paper, the number of edges in a graph or in a structure is called its *cost.* The minimum cost (weight) tree spanning a given set M of vertices is called the *Steiner tree* problem and is NP-complete.

Conflicting parameters. Simultaneous approximation. In the situation described above, the provider must minimize the distance parameters (to satisfy the customers) and, *simultaneously*, the cost of the structure. It was shown in [1] that it is impossible; the criteria cost and diameter are conflicting. To solve this conflict we do not strictly minimize the parameters but we *approximate* them.

A distributed algorithm. In spite of its interest in the guarantees that can be offered, this kind of approach generally suffers from a weakness: The methods to construct the structure are almost all centralized. To exploit them, the provider needs a global view of the state of its system before applying the algorithms. In general this is not possible in practice. To help him, we propose in this paper a *distributed* algorithm. The connection structure is then constructed by exchanging messages between members, with no centralized node doing all the job. Members just know local information to process. In this paper, we make reasonable hypothesis; we suppose that a routing table and an allocating mechanism is available in the system (see more details on the distributed model in section 1).

Another interesting parameter to investigate for our asynchronous protocol is then the *number of exchanged messages* (or messages complexity) during its execution to avoid to overload the network during the construction.

Representation by a graph. We model the network by a graph $G = (V, E)$ where V is the set of *vertices* representing the nodes of the network, E is the set of *edges* modeling its bidirectional physical links. Graphs considered here are unweighted, undirected and connected.

Let $M \subseteq V$ be the *group* of $m = |M|$ *members* that must be connected by allocating/renting links. The set of these links form what we call a *connection structure*; As the links are reserved for intra-group communications there is no external traffic disturbing the communications between members. Note that in terms of graphs a connection structure $S = (V_S, E_S)$ in a connected subgraph of $G = (V, E)$ spanning M ($M \subseteq V_S \subseteq V$, $E_S \subseteq E$).

Technically, the latency is represented by distances. For each pair u, v of vertices of V, we denote by $d_G(u, v)$ the *distance* between u and v. This is the minimum number of edges to cross to go from u to v in G.

Notation 1. *Let $G = (V, E)$ an unweighted, undirected graph and $M \subseteq V$ be the group.*

- *The* ***diameter*** *of M in G is:* $D_G(M) = \max\{d_G(u, v) : u, v \in M\}$
- *The* ***sum of distances*** *of M in G is:* $C_G(M) = \sum_{u,v \in M} d_G(u, v)$.

– *The* ***cost*** *of any graph (or structure)* $G = (V, E)$ *is its number of edges:* $W(G) = |E|$.

We focus here on sum of distance since it is easy to get average distance from that. The minimum cost (weight) spanning structure of a group M is the well known *Steiner tree* denoted by $T^*(M)$.

We want to construct a structure S, spanning members of M such that its diameter $D_S(M)$ (resp. its sum of distance $C_S(M)$, resp. its cost $W(S)$) is no more than $\rho_D D_G(M)$ (resp. $\rho_a C_G(M)$ resp. $\rho_W W(T^*(M))$). The parameter ρ_D (resp. ρ_a resp. ρ_W) is the *approximation ratio* for the diameter (resp. sum of distances, resp. weight).

Known results. Due to space limitation we focus here on the main related works to ours. We underline the difference with our own contribution.

In [2] for example, reader can find treatment of approximation algorithms and of the NP-hard Steiner tree problem. However, classical approximation methods just deal with the optimization of *one* criterion: Only the weight is treated for the Steiner tree for example; the induced distances in such trees are not considered.

At the opposite, works on *spanners* investigate the problem to construct a structure of minimum weight, spanning *all the vertices* and inducing distances between *each pair* of vertices at most a given multiplicative factor of the one in the underlying graph. This approach is very interesting but unfortunately there are many non approximability results (see [3]). A variant for groups has been investigated in [4] and was also shown to be hard. Moreover all the existing methods are centralized.

We can see that the main constraint in spanner is to give guarantees for the distances between *each pair* of vertices. In other works, authors relax this constraints and investigate the construction of structures in which the maximum and/or average distance are minimized. For example, one can find in [5,1,6] approximation algorithms to construct trees spanning a given group with the objective to approximate simultaneously these parameters. In particular, [6] investigates exactly the three parameters considered in the present paper. However the algorithm given in [6] is not distributed. In the present paper we propose an alternative construction to [6] that allows us to obtain a distributed protocol. Indeed this version exhibits original local properties that are exploited. We show in this paper that this new approach leads to an efficient algorithm (in terms of message complexity) and good approximation ratios. For completeness, we present here the whole construction and all the proofs.

To finish we can cite [7] surveying several recent works on approximation distributed algorithms. However, there is no reference on our own subject, works mentioned focus on the optimization of only one criterion (we are multicriteria) and the distributed systems under consideration are synchronous (we deal here with asynchronous systems).

Outline of the paper. In Section 1, we describe our distributed algorithm. We prove its approximation ratios and number of exchanged messages in Section 2.

1 Our Tricriteria Approximation Distributed Algorithm

Our tricriteria algorithm proceeds in two phases: In the first phase, we construct the tree T_p, a ρ-approximation of a Steiner tree $T^*(M)$ of the group M. In a second phase, we reduce the induced distances between members in this tree; this is done by adding appropriated shortest paths without increasing significantly the weight of the initial tree.

Distributed model: A shortest path *routing function* is available as in many current networks. Each node u can use its local copy of the routing table to determine the distance $d_G(u, v)$ between itself and any node v. This routing function can also be used by node u to send messages by a shortest path of G to any node v. It can also be used to *allocate* a shortest path between u and v (reserve the links of this path). We suppose that these two operations induces a number of messages equal to the number of links crossed, that is $d_G(u, v)$. Moreover, we consider that each member is awake and knows the m distinct identities of the members.

1.1 Phase 1: Construction of an Approximated Steiner Tree

Phase 1 constructs a tree T_P satisfying $W(T_P) = O(\log(m)W(T^*(M)))$. Moreover, the first phase finds a member with particular properties, called the *Median* r of the group M. This particular member is essential for beginning the second phase.

As the identifiers of members of M are known by each node of M, each node can create an order. W.l.o.g. we suppose that the members are numbered $M = \{u_1, u_2, \ldots, u_m\}$ and that each member knows this order.

The construction of tree T_p is the following: Each member u_i connects itself to the *nearest* member in the set $\{u_1, \ldots, u_{i-1}\}$ (u_i can select this nearest member using its local routing table). These $m-1$ connections can be done in any order, even in parallel. At the end, the (intermediate) obtained structure is composed of $m-1$ shortest paths and is connected. It is not necessarily a tree (it may contain cycles), an appropriated DFS is done twice from node u_1 to cut potential cycles and to prune the structure to obtain the desired *tree* T_P. After this operation, each node $u \in M$ is in the tree T_P.

We give now the definition of the median and the way to construct it.

Definition 1 (Median of a group). *Let* $G = (V, E)$ *and* $M \subseteq V$. *A vertex* $r \in M$ *is a* median *of* M *(in* M*) if it satisfies:*

$$\sum_{u \in M} d_G(u, r) = \min \left\{ \sum_{u \in M} d_G(u, v) : v \in M \right\}$$

To do that, u_1 starts the process by sending to u_2 the pair (S_1, u_1) where $S_1 = \sum_{v \in M} d_G(u_1, v)$. When it receives S_1, node u_2 can compute its own value

$S_2 = \sum_{v \in M} d_G(u_2, v)$. If $S_1 < S_2$ then u_2 sends (S_1, u_1) to u_3, otherwise it sends (S_2, u_2). This process can be continued from node to node, following the order $\{u_1, \ldots, u_m\}$. The last node u_m sends its result to u_1 that now knows the median r (since the minimum sum of distances has been filtered in successive transfers). It then broadcasts the result to the whole group.

1.2 Phase 2: Distributed Median-Control

The general idea of this phase 2 is to make a DFS from the $Median$ r in T_P (constructed in previous phase). Each time the DFS reaches a member, it makes a particular test on distances: If the test is positive, it roughly means that the member is "too far" from the median in T_p *compared* to its distance in G (read in the routing table). In this case, we say that this member v_i is added in set S. This member connects to the $Median$ by allocating a shortest path of the graph (using the routing mechanism). Note that this set S is *not* transported in the message. We only need its cardinality in the protocol; However, in the proof we will use this implicit set S. At the beginning, when $S = \emptyset$, we set $v_i = r$.

Message: For the protocol, we need to transport several information in the messages of type: $< DFS, Dist_{T_P}, Dist_G, cardS >$

- DFS is the name of the message.
- $Dist_{T_P}$ is the distance in tree T_P between the last node v_i inserted in set S and the node sending the DFS message. This parameter must be updated at each node (member or not).
- $Dist_G$ is the distance in G between v_i and the $Median$ node r. This distance is read by v_i in its local routing table. Note that $Dist_G$ $(= d_G(r, v_i))$ does not change during the construction while v_{i+1} is not detected.
- Counter $cardS$ is the number of nodes in the current set S.

Local variables: For each node u in T_P, the main variables are:

- N_{T_P}: set of neighbors identifiers of u in tree T_P.
- TBS: (To Be Sent) set of neighbors of u in T_p to which node u has to forward the DFS message.
- $Visited$: boolean with value $True$ iff node u has received a DFS message. Init. at $False$.
- $Parent$: node identifier by which node u receives for the first time message DFS. Init. at NIL.
- $Last\_Card\_S$: integer counter containing the knowledge node u has of the cardinality of the current set S. Init. at 0.

We suppose in the following that a parameter α is already known by all the members of M.

Pseudo-code of the algorithm

```
PROCEDURE: Executed only by Node $Median$ (node $r$), at the beginning
$Last\_Card\_S := 0$; $Dist_{T_P} := 0$; $Dist_G := 0$;
$NumConnections := 0$; $TBS := N_{T_P}$;
Do
    {Choose any $v \in TBS$; $TBS := TBS - \{v\}$;
    Send $< DFS, Dist_{T_P}, Dist_G, cardS >$ to $v$;
    Wait for a message $< DFS, Dist_{T_P}, Dist_G, cardS >$ from $v$;
    $Last\_Card\_S := cardS$;
    If $cardS = 0$ then $Dist_{T_P} := 0$; else $Dist_{T_P} := Dist_{T_P} + 1$;}
While $TBS \neq \emptyset$

/* When $Last\_Card\_S = NumConnections$, the algorithm is finished
and the final connection structure $G_f$ is allocated. */
```

```
PROCEDURE: Each time Node $Median$ $r$ receives a connection from a
node $v$
$NumConnections := NumConnections + 1$;
```

```
PROCEDURE: When a node $u \neq Median$ receives
$< DFS, Dist_{T_P}, Dist_G, cardS >$ from node $v \in N_{T_P}$
If $Visited = False$ then { /* First visit of the DFS */
    $Visited := True$; $Parent := v$; $TBS := N_{T_P} - \{Parent\}$;
    $Dist_{T_P} := Dist_{T_P} + 1$; $Last\_Card\_S := CardS$;
    If $u \in M$ and $Dist_{T_P} + Dist_G > \alpha d_G(r, u)$ Then {
        /* A new $v_i$ is detected and added in set $S$ */
        $Dist_{T_P} := 0$; $Dist_G := d_G(r, u)$; $cardS := cardS + 1$;
        $Last\_Card\_S := CardS$;
        Connect to $Median$ $r$ by a shortest path;}
    }
Else {/* $Visited = True$ */
    If $Last\_Card\_S < CardS$ then {
/* New $v_i$'s have been discover in the exploration of the subtree of $u$ */
        $Dist_{T_P} := Dist_{T_P} + 1$; $Last\_Card\_S = CardS$;}
    else $Dist_{T_P} := Dist_{T_P} - 1$;
    If $TBS = \emptyset$ then $a = Parent$; /* It is time to backtrack */
    else {Choose any $a \in TBS$; $TBS := TBS - \{a\}$;}
    Send $< DFS, Dist_{T_P}, Dist_G, cardS >$ to $a$;
}
```

2 Proofs and Correctness

We express and prove in Section 2.1 a generic result giving the three approximation ratios obtained by our algorithm. In Section 2.2 we use this result to give the real approximation ratios obtained by our method and an upper bound on the number of exchanged messages.

2.1 Three Simultaneous Approximation Ratios

Theorem 1. *Let $G = (V, E)$, $M \subseteq V$ $(m = |M|)$, G_f be the connect structure of M returned (in the second phase* MEDIAN-CONTROL*) with T_P a ρ-approximation of the Steiner tree $T^*(M)$ of M, constructed in phase 1. The three simultaneous properties of our algorithm are the following:*

1. *A 2α-approximation for the sum of distances: $C_{G_f}(M) \leq 2\alpha C_G(M)$.*
2. *A 2α-approximation for the diameter of M: $D_{G_f}(M) \leq 2\alpha D_G(M)$.*
3. *A $\rho\left(1 + \frac{2}{\alpha-1}\right)$-approximation for the weight:*

$$W(G_f) \leq \rho\left(1 + \frac{2}{\alpha - 1}\right) W(T^*(M)).$$

For the clearness of the proof we will use Lemmas.

Lemma 1. *Let $G = (V, E)$, $M \subseteq V$ $(|M| = m)$ and r be any vertex of V. Let A be any subgraph of G, spanning M and r, satisfying $d_A(u, r) \leq \alpha d_G(u, r)$ for all $u \in M$ for some $\alpha \geq 1$. We have:*

$$C_A(M) \leq 2\alpha m \sum_{u \in M} d_G(u, r)$$

Lemma 1 is an extension of a result of [8].

Proof. We upper bound $C_A(M)$ by using triangular inequality.

$$\begin{aligned}
C_A(M) &= \sum_{u,v \in M} d_A(u, v) \leq \sum_{u,v \in M} (d_A(u, r) + d_A(r, v)) \\
&= \sum_{u \in M} \sum_{v \in M} (d_A(u, r) + d_A(r, v)) \\
&= \sum_{u \in M} \left(m d_A(u, r) + \sum_{v \in M} d_A(r, v) \right) \\
&= 2m \sum_{u \in M} d_A(u, r) \leq 2\alpha m \sum_{u \in M} d_G(u, r)
\end{aligned}$$

The last inequality follows from the property of distance in A. □

Lemma 2. *Let $G = (V, E)$, $M \subseteq V$ and $r \in M$ a median of M. We have:*

$$C_G(M) \geq m \sum_{u \in M} d_G(u, r)$$

Proof. $C_G(M) = \sum_{v \in M} \left(\sum_{u \in M} d_G(u, v) \right) \geq m \sum_{u \in M} d_G(u, r)$ □

We suppose here that a tree T_P spanning M is given, constructed in Phase 1. This is an approximation of the Steiner tree $T^*(M)$. Graph G_f in the following is the final connection structure constructed by our algorithm. The techniques used here and related proofs are adaptations and modifications of the ones of [5,6] for the distributed context.

Lemma 3. *For all $u \in M$ we have:* $d_{G_f}(r,u) \leq \alpha d_G(r,u)$

Proof. Let us consider the step where the DFS reaches vertex $u \in M$ for the first time. At this particular moment, let v_i be either the last vertex put in set S or vertex r (if set S is still empty). Two cases must be examined, depending on the test treating u in the algorithm. Note that $d_{T_P}(v_i,u)$ (resp. $d_G(r,v_i)$) is transported by the incoming DFS message in parameter $Dist_{T_P}$ (resp. $Dist_G$).

1. If $d_G(r,v_i) + d_{T_P}(v_i,u) > \alpha d_G(r,u)$ then u is put in set S and after the DFS a shortest path between r and u is added in G_f and in this case $d_{G_f}(r,u) = d_G(r,u)$.
2. Otherwise, $d_G(r,v_i) + d_{T_P}(v_i,u) \leq \alpha d_G(r,u)$. As G_f contains a shortest path between v_i and r in G and also contains the whole tree T_P, by using the triangular inequality we have:

$$d_{G_f}(r,u) \leq d_{G_f}(r,v_i) + d_{G_f}(v_i,u) \leq d_G(r,v_i) + d_{T_P}(v_i,u) \leq \alpha d_G(r,u) \qquad \square$$

Lemma 4. $W(G_f) \leq \left(1 + \frac{2}{\alpha - 1}\right) W(T_P)$

Proof. Let $S = \{v_1, \ldots, v_k\}$ be the k vertices added in set S during the algorithm. Let $v_0 = r$ (median of M). A new vertex v_i is added in S by the algorithm when: $d_G(r,v_{i-1}) + d_{T_P}(v_{i-1},v_i) > \alpha d_G(r,v_i)$. Making the sum on i, we obtain: $\alpha \sum_{i=1}^{k} d_G(r,v_i) < \sum_{i=1}^{k} d_G(r,v_{i-1}) + d_{T_P}(v_{i-1},v_i)$. But, as $v_0 = r$ we get $d_G(r,v_0) = 0$ and: $\sum_{i=1}^{k} d_G(r,v_{i-1}) \leq \sum_{i=1}^{k} d_G(r,v_i)$. By combining we have: $(\alpha - 1)\sum_{i=1}^{k} d_G(r,v_i) < \sum_{i=1}^{k} d_{T_P}(v_{i-1},v_i)$. As $v_1, \ldots, v_k$ is a prefix order of a subset of M visited in a DFS visiting exactly twice each edge of tree T_P we have: $\sum_{i=1}^{k} d_{T_P}(v_{i-1},v_i) \leq 2W(T_P)$. Hence, $\sum_{i=1}^{k} d_G(r,v_i) < \frac{2}{(\alpha-1)} W(T_P)$. But, $W(G_f) \leq W(T_P) + \sum_{i=1}^{k} d_G(r,v_i) \leq \left(1 + \frac{2}{(\alpha - 1)}\right) W(T_P)$. $\square$

Proof. (**Theorem 1**)

1. **Result for the sum of distances:** Lemma 2 shows: $m \sum_{u \in M} d_G(u,r) \leq C_G(M)$ with $r \in M$ a median of M. With Lemma 1, and Lemma 3 we obtain: $C_{G_f}(M) \leq 2\alpha m \sum_{u \in M} d_G(u,r)$. Hence: $C_{G_f}(M) \leq 2\alpha C_G(M)$.
2. **Result for the diameter.** With the triangle inequality, with Lemma 3 and with the fact that $r \in M$, for all u and v in M we have:
$$d_{G_f}(u,v) \leq d_{G_f}(u,r) + d_{G_f}(r,v) \leq \alpha(d_G(u,r) + d_G(r,v)) \leq 2\alpha D_G(M).$$
Hence $D_{G_f}(M) \leq 2\alpha D_G(M)$.
3. **Result for the weight.** Lemma 4 gives the result since $W(T_p) \leq \rho W(T^*(M))$. □

2.2 Total Number of Exchanged Messages and Induced Approximation Ratios

In this Section we use all the previous analysis to compute the performance of our algorithm in terms of approximation ratios (Theorem 2) and number of exchanged messages (Theorem 3).

Theorem 2. *If α is a constant then our algorithm is a distributed, constant approximation for diameter and sum of distances parameters and a $O(\log(m))$ approximation for cost.*

Theorem 3. *If α is a constant and $m = |M|$ then our algorithm uses at most $O(m \log(m) D_G(M))$ messages.*

To prove these final results we need some Lemmas.

Lemma 5. $W(T_P) = O(\log(m) W(T^*(M)))$

Proof. The process of construction of our tree T_P can be viewed as another equivalent version of the *online* algorithm proposed in [9]. The authors show that if a Steiner tree is constructed step by step by connecting a new member to the nearest member already connected, then, the final tree is a $O(\log(m))$-approximation of the Steiner tree. This is in fact what we do here, by "simulating" this process. □

Proof. (**Theorem 2**) Apply Lemma 5 and Theorem 1 with $\rho = O(\log(m))$ □

Lemma 6. *Phase* 1 *uses at most $O(m \log(m) D_G(M))$ messages ($m = |M|$).*

Proof. The construction of the initial substructure requires at most $(m-1)D_G(M)$ messages. The message complexity of the pruning process of this structure to obtain T_P is linear in the number of edges in T_P and, from Lemma 5, it is at most $O(\log(m) W(T^*(M)))$. As $W(T^*(M)) \leq (m-1)D_G(M)$, the construction of T_P requires at most $O(m \log(m) D_G(M))$ messages. Determining the median of M just requires at most $O(m D_G(M))$ messages by the presented process. □

Lemma 7. *Phase 2 uses at most* $2\rho\left(1+\frac{2}{\alpha-1}\right)W(T^*(M))$ *messages.*

Proof. During phase 2, a DFS of tree T_P is performed and each edge of T_P is crossed exaclty twice. Moreover, some direct connections (by the routing functions) are made between some elements of M and the $Median\ r$. All the edges crossed by messages is exactly the set of edges of the final structure G_f. Hence, at most $2|E(G_f)|$ messages are exchanged during phase 2. As $|E(G_f)| = W(G_f)$, approximation ratio on the cost of Theorem 1 shows the desired result. □

Proof. (**Theorem 3**) Lemma 6 shows that phase 1 uses $O(m\log(m)D_G(M))$ messages. Lemma 5 shows that $\rho = O(\log(m))$; combined with Lemma 7 and the fact that $W(T^*(M)) \leq mD_G(M)$ we get the result. □

Theorem 3 just gives an estimation, an upper bound on the number of messages. In particular, each time a member sends a message to another member, in the worst case the number of messages is equal of the diameter $D_G(M)$. In practice many pairs of members are closer; this reduces the expected complexity. Each message transports three data: two distances and one cardinal. Each value is smaller than the numbers of vertices of G, thus can be represented on $O(\log n)$ bits.

3 Conclusion

In this paper we have proposed a tricriteria distributed algorithm for the construction of a connection structure for interconnecting a group of machines spread in a network modelled by a graph. This connection structure is allocated for the group and is optimized in terms of induced delays and total cost.

We *proved* the quality of the structure by approximation ratios and we proposed and proved an upper bound on the number of exchanged messages. This last parameter was evaluated by a worst case analysis. It could be refined by simulations for example. Moreover, our construction and proofs are parametric: We used two separated phases and parameters α and ρ; Hence, if a better distributed algorithm [1] to construct the initial approximation Steiner tree exists then it can directly be "plugged" in our method and its impact can easily be evaluated by using our analysis.

References

1. Irlande, A., König, J.C., Laforest, C.: Construction of low-cost and low-diameter steiner trees for multipoint groups. In M. Flammini, E. Nardelli, G.P., Spirakis, P., eds.: Procedings of Sirocco 2000, Carleton Scientific (2000) 197–210
2. Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., Protasi, M.: Complexity and approximation. Springer (1999)
3. Elkin, M., Peleg, D.: The hardness of approximating spanner problems. In Reichel, H., S.Tison, eds.: STACS 2000. Number LNCS 1770, Springer Verlag (2000) 370–381

[1] Better in terms of approximation ratio **and** number of exchanged messages.

4. Laforest, C.: Construction of efficient communication sub-structures: Non-approximability results and polynomial sub-cases. In Springer, ed.: EUROPAR. Volume 2790 of LNCS. (2003) 903–910
5. Khuller, S., Raghavachari, B., Young, N.: Balancing minimum spanning trees and shortest-path trees. Algorithmica (14) (1995) 305–321
6. Laforest, C.: A good balance between weigth and distances for multipoint tre es. In: 6th International Conference On Principles Of DIstributed Systems (OPODIS). (2002)
7. Elkin, M.: Distributed approximation: a survey. SIGACT News **35**(4) (2004) 40–57
8. Wu, B., Chao, K., Tang, C.: Approximation algorithms for the shortest total path length spanning tree problem. Discrete applied mathematics (105) (2000) 273–239
9. Imase, M., Waxman, B.: Dynamic steiner tree problem. SIAM Journal on Discrete Mathematics **4**(3) (1991) 369–384