Approximation algorithms for forests augmentation ensuring two disjoint paths of bounded length[1]

Victor Chepoi, Bertrand Estellon, and Yann Vaxès

LIF-Laboratoire d'Informatique Fondamentale de Marseille,
UMR 6166, Faculté des Sciences de Luminy,
Université de la Méditerranée, F-13288 Marseille Cedex 9, France
{*chepoi,estellon,vaxes*}*@lif.univ-mrs.fr*

Abstract

Given a forest $F = (V, E)$ and a positive integer D, we consider the problem of finding a minimum number of new edges E' such that in the augmented graph $H = (V, E \cup E')$ any pair of vertices can be connected by two vertex-disjoint paths of length $\leq D$. We show that this problem and some of its variants are NP-hard, and we present approximation algorithms with worst-case bounds 6 and 4. These algorithms can be implemented in $O(|V| \log |V|)$ time.

1 Introduction and preliminaries

Biconnectivity is a fundamental requirement to the topology of communication networks: a biconnected network survives any single link or node failure (the probability of two or several simultaneous failures in most networks is reasonably small). On the other hand, the communication performances of a network depend of the communication delay between any two nodes of the network. Since the delay of sending a message from one node to another is roughly proportional to the number of nodes (or links) the message has to traverse, it is desirable to route the messages along short paths or paths of bounded length. Therefore a network in which any pair of nodes can be connected by two disjoint paths of bounded length ensure a low communication delay even in case of a single link or node failure. In this paper, we consider the problem of optimal augmentation of networks (more precisely, of their underlying graphs) so that the resulting networks satisfy this connectivity requirement. We show that this augmentation problem is NP-hard even in the case of forests. On the other hand, we provide efficient approximation algorithms for this problem and its variants if the input graph is a forest. Our work continues the research started in [4, 3, 11].

[1] An extended abstract of this paper appeared in the proceedings of the 9th Workshop on Algorithms and Data Structures, WADS 2005. The authors were partly supported by the ANR grant BLAN06-1-138894 (projet OPTICOMB).

Several other models have been proposed in the literature to study fault-tolerant networks whose reliability and communication performances survive node or edge failures. For instance, Farley and Proskurowski [9] study the class of self-repairing graphs which consists of 2-connected graphs such that the removal of any single vertex results in no increasing in distance between any pair of vertices in the graph. Another interesting model has been proposed by Dolev et al. [7]. Given a graph G, a fixed routing and a set of faults F, they define a surviving route graph consisting of all non-faulty nodes in the network with two nodes being connected by an edge if and only if the route between them avoid F. Then, the problem is to obtain a routing such that for any set of faults of a given cardinality, the surviving route graph has a small diameter. Note that, since this diameter represents the number of routes along which a message must travel between any two non-faulty nodes, it can be viewed as an estimate of the fault-tolerance of the routing.

The problem of augmenting a graph to reach biconnectivity by adding a minimum number of new edges is an important graph-algorithmic problem with applications to network reliability and fault-tolerant computing. Eswaran and Tarjan [8] introduced this problem and established that its basic version can be solved efficiently. Subject to additional constraints, the biconnectivity augmentation problem becomes difficult: for example, both the weighted augmentation problem and the optimal augmentation of a planar graph to a biconnected planar graph are NP-hard [8, 12]; for both problems there exist constant factor approximation polynomial algorithms.

The problem of augmenting a graph $G = (V, E)$ to a graph $H = (V, E \cup E')$ of a given diameter D by adding a minimum number of edges is NP-hard for any $D \geq 2$ [4, 13, 14] (and is at least as difficult to approximate as SET COVER). The complexity status of this problem is unknown if the input graph is a forest (or a tree). In this case, [4] presents a factor 2 algorithm for even D and [11] presents a factor 8 algorithm for odd D. More recently, a factor $2 + \frac{1}{\delta}$ for any $\delta > 0$ approximation algorithm in the case of odd D was proposed in [3]. Chung and Garey [5] established that if G is a path with n vertices, then the minimum number of added edges is at least $(n - D - 1)/(D + 1)$ and at most $(n - D + 2)/(D - 2)$; for some other related bounds see [1]. If, additionally to be of diameter D, the resulting graph H must be biconnected, then the resulting augmentation problem is NP-hard even if the input graph is a tree [4]. For forests, [4] presents a factor 3 approximation algorithm in case of even D and a factor 6 approximation algorithm for odd D. The last result has been improved by Ishii, Yamamoto, and Nagamochi [11] to a factor 4 (plus 2 edges) approximation algorithm. Notice that for trees the performance guarantees of all mentioned algorithms should be much better, however the bottleneck in analyzing them is the difficulty of establishing better lower bounds for the minimum number of added edges; for example, the proof of the above mentioned lower bound for paths [5] is already quite involved.

In this note, we consider three variants of the augmentation problem with additional distance constraints:

Problem A2VDBP (Augmentation with 2 Vertex-Disjoint Bounded length Paths): *given a graph $G = (V, E)$ and a positive integer D, add a minimum number of new edges E' such that any pair of vertices can be connected in the augmented graph $H = (V, E \cup E')$ by two*

vertex-disjoint paths of length $\leq D$.

Problem A2EDBP (Augmentation with 2 Edge-Disjoint Bounded length Paths): *given a graph* $G = (V, E)$ *and a positive integer* D*, add a minimum number of new edges* E' *such that any pair of vertices can be connected in the augmented graph* $H = (V, E \cup E')$ *by two edge-disjoint paths of length* $\leq D$.

Problem ADCE (Augmentation with Diameter Constraints in the augmented graph minus an Edge): *given a graph* $G = (V, E)$ *and a positive integer* D*, add a minimum number of new edges* E' *such that for any edge* $e \in E \cup E'$ *the diameter of the augmented graph* H *minus* e *is at most* D.

The problems A2VDBP, A2EDBP, and ADCE are not equivalent: any feasible augmentation for A2VDBP is a feasible solution to A2EDBP and any feasible solution to A2EDBP is a feasible solution of ADCE but not vice versa, as the following example shows (see Fig. 1). Let H be a graph consisting of a cycle of length $2R + 2$ plus a diagonal cc' connecting two opposite vertices c and c' of this cycle. Removing any edge from H results into a graph of diameter at most $2R - 1$, however the neighbors a and b of c different from c' cannot be connected in H by two vertex- or edge-disjoint paths of length $\leq 2R - 1$.

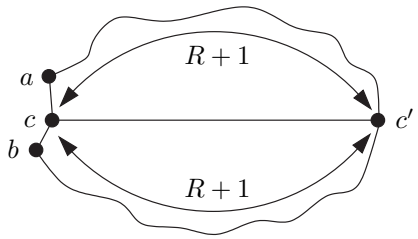


Figure 1: Non-equivalence of problems A2EP and ADCE.

In Section 2 we will prove that the problems A2VDBP, A2EDBP, and ADCE are NP-hard already when the input graph is a forest. Based on lower bounds established in Section 3, in Section 4 we present a factor 6 approximation algorithm for all three problems. In Section 5, this algorithm is improved to a factor 4 approximation algorithm for problem ADCE. In Section 6 we show how to implement these algorithms in $O(|V| \log |V|)$ time.

We conclude this introductory section with a few necessary definitions. A polynomial algorithm is called an *α-factor approximation* algorithm for a minimization problem Π if for each instance I of Π, it returns a solution whose value is at most α times the optimal value $\mathrm{OPT}_\Pi(I)$ of Π on I plus a constant not depending of I; see [15]. For a graph $G = (V, E)$, the *length* of a path between two vertices is the number of edges in this path. The *distance* $d(u, v) := d_G(u, v)$ between two vertices u, v of G is the length of the shortest path between these vertices (if u and v are in distinct connected components of G we will set $d(u, v) = \infty$). The *diameter* $\mathrm{diam}(G)$ of G is the largest distance between two vertices of G. For a positive integer k and a vertex $u \in V$ let $B(u, k) = \{v \in V : d(u, v) \leq k\}$ denote the *ball* of radius k centered at u. (For other graph-theoretical notions and notations used but not defined in this text, see [16].) For two vertices u, v belonging to the same tree of a forest F, denote by $P(u, v)$

the unique path of F connecting u and v. For a vertex x in a rooted tree T with root r of a forest F, any vertex $y \neq x$ on the path $P(r,x)$ is called an *ancestor* of x. If y is an ancestor of x, then x is a *descendant* of y. For a subset S of vertices of T, the set of *direct descendants* of a vertex $v \in V$ consists of all descendants $u \in S$ of v such that the path $P(u,v)$ does not contain any other vertex from S.

2 NP-completeness

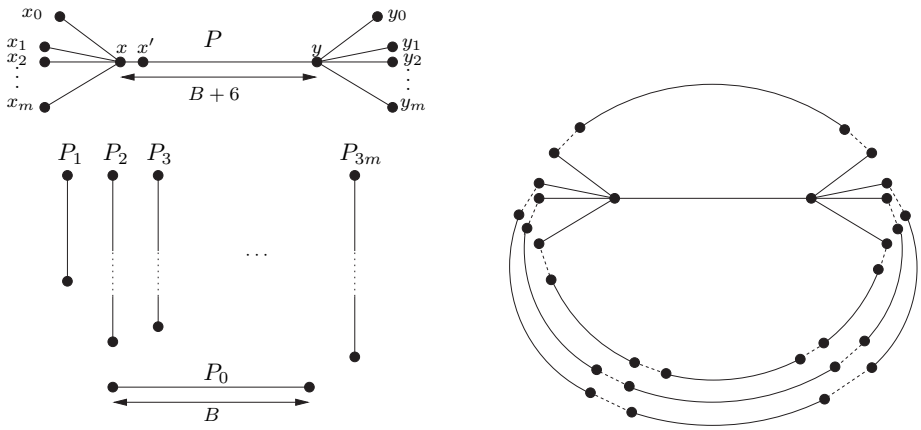
The decision variants of problems A2VDBP, A2EDBP, and ADCE belong to the class NP. To establish that these problems are NP-complete on forests, we present pseudo-polynomial transformations from the strongly NP-complete problem 3-PARTITION. Then Lemma 4.1 of [10] implies that the augmentation problems are NP-complete as well. Our construction is similar to that used in [4] with one difference: we use an additional path in order to force the structure of the augmented graph.

Theorem 2.1 *The problems A2VDBP, A2EDBP, and ADCE are NP-complete on forests.*

Proof. We will describe a pseudo-polynomial transformation from 3-PARTITION [10] to A2VDBP. Let an instance of 3-PARTITION be given, i.e., a set A of $3m$ elements $a_1, \ldots, a_{3m}$, a bound $B \in \mathbb{Z}^+$ ($B \geq 8$), and a size $s(a_i) \in \mathbb{Z}^+$ for each $a_i \in A$ such that $B/4 < s(a_i) < B/2$ and $\sum_{a_i \in A} s(a) = mB$. We construct a forest $F = (V, E)$ as follows: for each $a_i \in A$ introduce a path P_i of length $s(a_i)(> 2)$, additionally consider a path P_0 of length B, and a "bistar" formed by a path P of length $B+6$ plus $m+1$ leaves at each end of this path; see Fig. 2. We assert that the set A can be partitioned into m disjoint sets $A_1, \ldots, A_m$ such that $\sum_{a_i \in A_j} s(a_i) = B$ for every $1 \leq j \leq m$ if and only if there exists a feasible solution to problem A2VDBP with $D := 2B + 10$ using at most $4m + 2$ edges.

The forest F has $8m + 4$ leaves. Since a feasible augmentation of F results into a biconnected graph, any leaf of F must be incident to a new edge, therefore this augmentation must contain at least $4m + 2$ edges. If the optimal solution E' of A2VDBP uses exactly $4m + 2$ edges, then both ends of added edges are leaves of F. It can be easily seen that the graph $H = (V, E \cup E')$ obtained from F by adding $4m+2$ edges E' consists of the path P and $m+1$ ears. An *ear* Ear_i is a path of H between x and y consisting of the edges xx_i, yy_i and 0,1, or several paths P_j ($j \in \{0, 1, \ldots, 3m\}$) and some new edges connecting either the end-vertices of two paths or an end-vertex of a path with x_i or y_i.

Let Ear_0 be the ear containing the path P_0. We assert that Ear_0 does not contain other paths of F. By using the feasibility of E', one can prove that Ear_0 does not contain other paths of F, and thus, Ear_0 has length $B+4$. We will show now that each of the ears $\mathrm{Ear}_1, \ldots, \mathrm{Ear}_m$ has length $B+6$. Suppose by way of contradiction that Ear_i has length at least $B+7$. Due to the structure of H, there exists a unique path between x_i and x' not passing via x : it consists of the subpath of P between x' and y, and the subpath of Ear_i between y and x_i. The length of this path is $\geq B + 5 + B + 6 = 2B + 11 > D$, therefore all paths of length at most D between x_i and x' pass via x, contrary to the admissibility of H. Thus every Ear_i has length



(a) Forest F (b) Augmented graph H

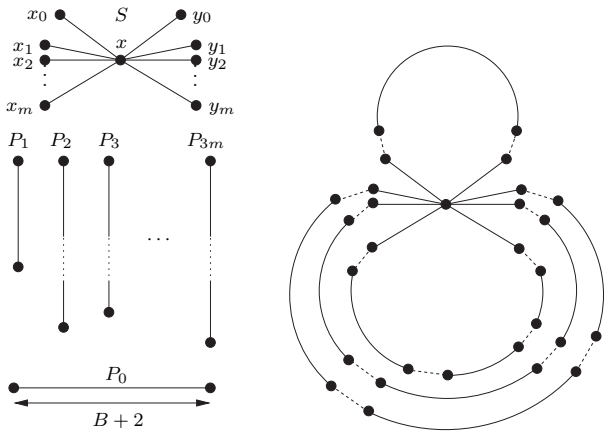
Figure 2: To the NP-completeness of problem A2VDBP.

$B + 6$, i.e., Ear_i is composed of exactly three paths of F of total length B (recall that the length of every path of the forest F is comprised between $B/4$ and $B/2$). The corresponding triplets of A yield a feasible 3-partition.

Conversely, given a feasible solution to 3-PARTITION, we can biconnect F by adding $4m + 2$ edges in such a way that in the augmented graph H, Ear_0 has length $B + 4$ and every other ear Ear_i $(i = 1, \ldots, m)$ has length $B + 6$. We claim that H is a feasible solution of problem A2VDBP. This establishes the NP-completeness of A2VDBP.

The NP-completeness of ADCE and A2EDBP is established by using a slightly different pseudo-polynomial transformation from 3-PARTITION. Suppose we are given an instance of 3-PARTITION in which the integer B is divisible by 4. The forest $F = (V, E)$ is constructed in the following way: F consists of a star S with $2m+2$ leaves, a path P_0 of length $B+2$, and, for each element a_i $(i = 1, \ldots, 3m)$, a path P_i of length $s(a_i)$; see Fig. 3 for this construction. We assert that there exists a 3-partition if and only if there exists a feasible solution of problem ADCE (or A2EP) for F and $D := 3B/2 + 8$ having exactly $4m + 2$ edges.

Since any feasible augmentation of F results in a biconnected graph, any leaf of F must be incident to a new edge. As F contains $8m + 4$ leaves, any feasible augmentation must contain at least $4m + 2$ new edges. Suppose that there exists an optimal augmentation E' of F containing exactly $4m + 2$ edges. Then necessarily all end-vertices of edges of E' are leaves of F, any leaf of F is incident to exactly one new edge, and, finally, any edge of E' is connecting two different trees from the forest F. This shows that the augmented graph $H = (V', E \cup E')$ consists of $m + 1$ cycles $C_0, \ldots, C_m$ sharing a unique common vertex x (see Fig. 3b). Suppose without loss of generality that the cycle C_i consists of two edges xx_i and xy_i and 0,1, or several paths of F, and some added edges connecting either the end-vertices of two such paths or an end-vertex of a path with a leaf of the star S.



(a) (b)

Figure 3: To the NP-completeness of problems A2EP and ADCE.

Let C_0 be the cycle containing the path P_0. We assert that C_0 does not contain any other path of F. Suppose this is not the case. Then the length ℓ_0 of the cycle C_0 is at least $B + 2 + B/4 + 5 = 5B/4 + 7$. Let ℓ_i be the length of the cycle C_i with $i > 0$. Since E' is a feasible augmentation, if the edge between x and x_0 is removed, the resulting graph must contain a path of length at most D running between x_0 and a furthest from x vertex of C_i. This is possible only if the inequality $5B/4+7-1+\ell_i/2-1 \leq D$ holds. Since $D = 3B/2+8$, we infer that the length ℓ_i of each such cycle C_i is at most $B/2+6$. Hence, each C_i contains at most two paths of F. Thus the cycle C_0 must contain, additionally to P_0, at least m paths of the forest F, yielding $\ell_0 \geq B+2+mB/4+2+m+2 = (m+4)B/4+m+6 \geq 2B+10$ (because $m \geq 4$). Now, if we remove the edge between x and x_0, the length of any path between x and x_0 in the resulting graph is at least $\ell_0 - 1 \geq 2B + 9 > D$, yielding to a contradiction with the feasibility of the augmentation E'. This shows that indeed the cycle C_0 contains only the path P_0 of F and that $\ell_0 = B + 6$.

Now, we will show that any other cycle C_i ($i > 0$) contains exactly 3 paths of F whose total length is equal to B. Since the length of every path of the forest F is strictly comprised between $B/4$ and $B/2$, it suffices to establish that the length of any such cycle C_i is exactly $B + 6$. Suppose by way of contradiction that the length ℓ_i of some C_i is at least $B + 7$. If the edge $\{x, x_i\}$ is removed, then, since $\ell_0 = B+6$, the distance in the resulting graph between x_i and the furthest from x vertex of the cycle C_0 is at least $B+7-1+(B+6)/2 = 3B/2+9 > D$, contrary to the feasibility of the augmentation E'. This contradiction shows that indeed any cycle C_i is composed of exactly three paths of F of total length B . Then the corresponding triplets of A yield a feasible 3-partition.

Conversely, we establish that any augmentation E' of this form derived from a feasible 3-partition is a feasible augmentation for problem A2EP (and thus for ADCE). Namely, for any edge $e = v_1v_2$ of the augmented graph H we will show that v_1 and v_2 can be connected

in the graph $H - e$ by two edge-disjoint paths of length at most D. If v_1 and v_2 belong to the same cycle of H, then they are connected by two paths (along the cycle) of length at most $B + 6 - 1 = B + 5 < D$ each. Suppose now that v_1 and v_2 belong to different cycles, say $v_1 \in C_i$ and $v_2 \in C_j$. The vertices v_1 and x are connected along the cycle C_i by two paths P_1 and P_1' of total length at most $B + 6$. Thus one of these paths, say P_1 has length at most $B/2 + 3$. Additionally, since $v_1 \neq x$, the length of the path P_1' is at most $B + 5$. Analogously, the two paths P_2 and P_2' connecting the vertices v_2 and x along the cycle C_j have length at most $B/2 + 3$ and $B + 5$, respectively. From the paths P_1, P_1', P_2, and P_2' we can compose in the following way two edge-disjoint paths of length at most D connecting the vertices v_1 and v_2 : the first path is composed of P_1 and P_2' and the second path is composed by P_2 and P_1'. The length of each of these paths is at most $B/2 + 3 + B + 5 = 3B/2 + 8 = D$, establishing our assertion. □

3 Lower bounds

For a graph $G = (V, E)$ and a positive integer k, a *vertex k-dominating set* is a set of vertices $C \subseteq V$ such that $\cup_{c \in C} B(c, k) = V$. Analogously, $C \subseteq V$ is an *edge k-dominating set* if the two end-vertices of any edge of G belong to a common ball of radius k centered at a vertex of C. Finding a minimum vertex or edge k-dominating set in a graph is NP-hard, however for trees (and forests) these problems can be solved in linear time [2]. The algorithm can be easily modified to find in linear time a minimum vertex k-dominating set VC_k or a minimum edge k-dominating set EC_k of a forest F with the additional constraint that C contains the set L of all leaves of F. Let $vc_k(F) := |VC_k|$ and $ec_k(F) := |EC_k|$.

Proposition 3.1 *For a forest F,*

(i) $vc_{R-1}(F)/2 \leq OPT_{ADCE}(F) \leq OPT_{A2EDBP}(F) \leq OPT_{A2VDBP}(F)$ *if* $D = 2R - 1$.

(ii) $ec_R(F)/2 \leq OPT_{ADCE}(F) \leq OPT_{A2EDBP}(F) \leq OPT_{A2VDBP}(F)$ *if* $D = 2R$.

Proof. In both cases it suffices to establish only the leftmost inequality, because the inequalities $\mathrm{OPT}_{ADCE}(F) \leq \mathrm{OPT}_{A2EDBP}(F) \leq \mathrm{OPT}_{A2VDBP}(F)$ trivially hold for all graphs. Let E' be an optimal augmentation for problem ADCE, and let C denote the set of end-vertices of the edges from E'. Notice that any leaf x of the forest F belongs to C, otherwise the paths in the graph $H = (V, E \cup E')$ issued from x will use the unique edge e of H incident to x, contrary to the fact that H is a solution of ADCE. Thus $L \subseteq C$. We assert that if $D = 2R-1$, then every vertex of F is covered by a ball of radius $R-1$ centered at C, and if $D = 2R$, then the end-vertices of every edge of F are at distance $\leq R$ from a vertex of C.

Suppose by way of contradiction that in the case $D = 2R - 1$ there exists a vertex $u \notin \cup\{B_{R-1}(c) : c \in C\}$. Obviously, u is not a leaf of F. Pick a neighbor v of u in F and let $e = uv$. From the choice of u we conclude that the nearest to v vertex of C is at distance at least $R-1$ from v. Since u is at distance at least R from any vertex of C and any (u, v)-path

of the graph $H-e$ uses at least one added edge, we conclude that the distance between u and v in the graph $H-e$ is at least $R+1+(R-1)=2R$, contrary to the assumption that E' is a feasible augmentation for ADCE. Hence $\cup\{B_{R-1}(c) : c \in C\} = V$, yielding $vc_{R-1}(F) \leq |C|$. Since $|C| \leq 2|E'|$ (the worst case occurs when E' is a matching on C), we obtain the required inequality.

Now, let $D = 2R$ and assume that there exists an edge $e = uv$ which is not covered by any ball of radius R centered at vertices of C. This means that any vertex of C is located at distance at least $R+1$ from either u or v. Any (u,v)-path in the graph H employs at least one added edge. Hence the distance between u and v in the graph $H-e$ is at least $R+1+R = 2R+1$, contrary to the assumption that E' is a feasible augmentation for ADCE. □

An immediate consequence of Proposition 3.1 is that any feasible augmentation for A2VDBP using at most $3vc_{R-1}(F)$ edges for $D = 2R-1$ and at most $3ec_R(F)$ edges for $D = 2R$ would provide a factor 6 approximation algorithm for each of the problems A2VDBP, A2EDBP, and ADCE. Next section is devoted to the description and analysis of such an algorithm.

4 A factor 6 approximation algorithm

In this section, we describe and justify the augmentation algorithm for $D = 2R-1$. Then, we provide the changes for the case $D = 2R$.

4.1 Odd diameter

Assume without loss of generality that the input forest $F = (V, E)$ contains at least one edge, otherwise we simply run the algorithm on the forest obtained from F by adding an arbitrary edge. Let L be the set of leaves of F and let $L_0 \subset L$ be the set of leaves constituting one-vertex trees of F. For the rest of this paper, we denote by $d(u,v)$ *the distance in the forest* F between two vertices $u, v \in V$. Suppose that every tree of the forest F containing at least two vertices is rooted at some leaf. Let S be the set of all such roots. The algorithm picks an arbitrary root $r \in S$ and the neighbor r' of r. At the next stage, a minimum vertex $(R-1)$-dominating set C of the forest F containing r' and the set L of leaves is computed. The algorithm proceeds each rooted tree level-by-level starting from its root, and for current vertex $c \in (C-L) \cup S$ it computes the list D_c of its direct descendants in C sorted in increasing order with respect to the distances to c. For each vertex $c' \in D_c$, the algorithm selects 0,1, or 2 vertices (this number depends of the distance $d(c', c)$, see Fig. 5) on the path $P(c, c')$ between c and c'. The set consisting of C and all selected vertices is grouped into two classes A and A' such that every vertex of the forest F can be connected by two vertex-disjoint paths of length $\leq R-2$, one going to a vertex of A and another to a vertex of A' (the one-vertex components of F are included in both A and A', while S is included only in A). The algorithm returns the augmentation consisting of all edges of the form ra for $a \in A$ and $r'a'$ for $a' \in A'$.

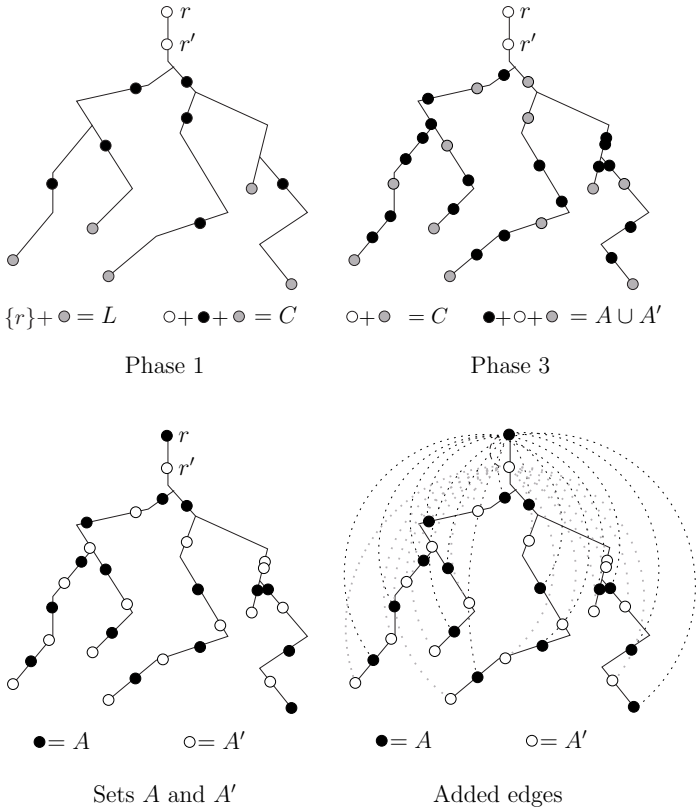


Figure 4: Illustration of Algorithm A2VDBP.

Algorithm A2VDBP(F,2R-1)

Input: A forest $F = (V, E)$ with $E \neq \emptyset$ and an odd integer $D = 2R - 1 \geq 9$.

Output: A set E' of new edges, such that any $u, v \in V$ can be connected in $H = (V, E \cup E')$ by two vertex-disjoint paths of length $\leq D$ and $|E'| \leq 6 \cdot OPT_{A2VDBP}(F)$.

Phase 0: Root every tree with at least two vertices at some leaf and denote by S the set of such roots. Set $A := L_0 \cup S$ and $A' := L_0$. Pick $r \in S$ and the neighbor r' of r.

Phase 1: Compute a minimum vertex $(R-1)$-dominating set C with $L \cup \{r'\} \subseteq C$.

Phase 2: For each $c \in (C - L) \cup S$ compute the list D_c of direct descendants of c in C and sort D_c in increasing order of the distances to c.

Phase 3: Proceed every rooted tree level-by-level. For current vertex $c \in (C - L) \cup S$ traverse the list D_c and to each vertex $c' \in D_c$ apply one of the following rules:

case $\mathbf{d(c, c') \leq R - 2}$: if $c \in A$, then insert c' in A', else insert c' in A.

case $\mathbf{R - 1 \leq d(c, c') \leq 2R - 4}$: pick the vertex $c_1 \in P(c, c')$ at distance $R - 2$ to c'. If $c \in A$, then insert c_1 in A' and c' in A, else insert c_1 in A and c' in A'.

case $\mathbf{d(c, c') \geq 2R - 3}$: pick the vertices $c_1, c_2 \in P(c, c')$, c_1 at distance $R - 2$ and c_2 at distance $2R - 4$ to c'. Find a nearest to c_2 vertex c'' of C which is not a descendant of c_2. If $c'' \in A$, then insert c_2, c' in A', and c_1 in A, else insert c_2, c' in A, and c_1 in A'.

Phase 4: Return $E' = \{ra : a \in A - \{r\}\} \cup \{r'a' : a' \in A' - \{r'\}\} \cup \{rr'\}$.

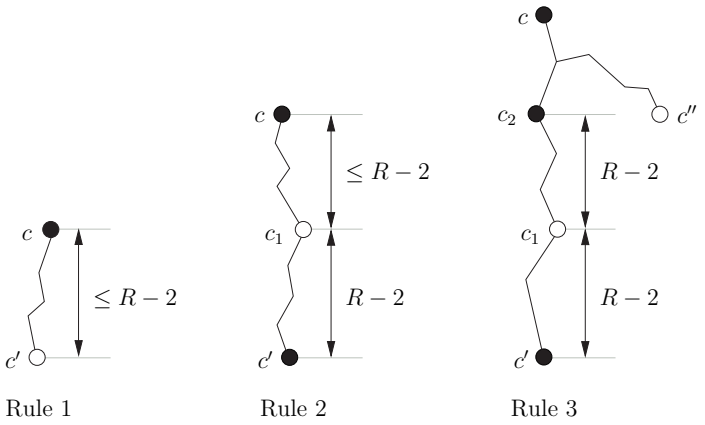


Figure 5: The three rules of Phase 3 of algorithm **A2VDBP**.

Next we establish that any pair u, v of vertices can be connected in the augmented graph $H = (V, E \cup E')$ by two vertex-disjoint paths of length $\leq D$.

Lemma 4.1 *For any vertex $x \in V$ there exists two vertices $a \in A$ and $a' \in A'$ such that $d(x,a) \leq R-2$, $d(x,a') \leq R-2$, $P(a,a') \cap C \subseteq \{a, a'\}$, and x belongs to the (possibly degenerated) path $P(a,a')$.*

Proof. The result trivially holds if $x \in L_0$, because $L_0 \subseteq A \cap A'$ and we can set $a := x =: a'$. If $x \in S$, then $x \in A$ and we can set $a := x$. Let c' be the closest to x vertex of C. According to the algorithm, we can set $a' := c'$ if $d(x,c') \leq R-2$ and $a' := c_1$ if $R-1 \leq d(x,c') \leq 2R-4$. In the remaining case $d(c,c') \geq 2R-3$, the role of a' is played by the vertex c_2 described in the algorithm. If $x \in C - S$, then x is a direct descendant of some vertex $c \in C$, and, according to the algorithm, we can define $\{a, a'\} := \{x, c\}$ if $d(x,c) \leq R-2$ and $\{a, a'\} := \{x, c_1\}$ otherwise.

So, assume that $x \notin C \cup L_0$. Let c be the nearest ancestor of x in C and let c' be the nearest descendant of x in C. Clearly c' is a direct descendant of c. If $d(c,c') \leq R-2$, then we can set $\{a, a'\} := \{c, c'\}$ because c and c' belong to distinct sets A, A'. On the other hand, if $R-1 \leq d(c,c') \leq 2R-4$, then set $\{a, a'\} := \{c, c_1\}$ if $x \in P(c, c_1)$ and set $\{a, a'\} := \{c_1, c'\}$ if $x \in P(c_1, c')$. From the algorithm we infer that the vertices c, c' belong to one set A, A' and c_1 belongs to another set, establishing the assertion.

Finally suppose that $d(c,c') \geq 2R-3$, i.e., we are in the third case of Phase 3. Recall that the algorithm picks two vertices $c_1, c_2 \in P(c,c')$, c_1 at distance $R-2$ and c_2 at distance $2R-4$ to c' and considers a closest in C non-descendant c'' of c_2. The vertices c' and c_2 belong to one set A, A', while c_1 and c_2 (as well as c' and c_1) belong to distinct sets. Therefore we can assume that $x \in P(c_2, c)$, otherwise the proof is immediate by setting $\{a, a'\} = \{c', c_1\}$ if $x \in P(c', c_1)$ and $\{a, a'\} = \{c_1, c_2\}$ if $x \in P(c_1, c_2)$. Let z be the vertex of $P(x,c')$ at distance R to c' ($z \in P(c_1, c_2)$ because $R \geq 5$, thus $z \neq x$). Denote by $\hat{c} \in C$ any closest to z center of an $(R-1)$-ball covering z. Since $d(z, \hat{c}) \leq R-1 < R = d(z,c')$, the choice of c' implies that $\hat{c}$ cannot be a descendant of z. For the same reason, $\hat{c}$ cannot be a descendant of x either.

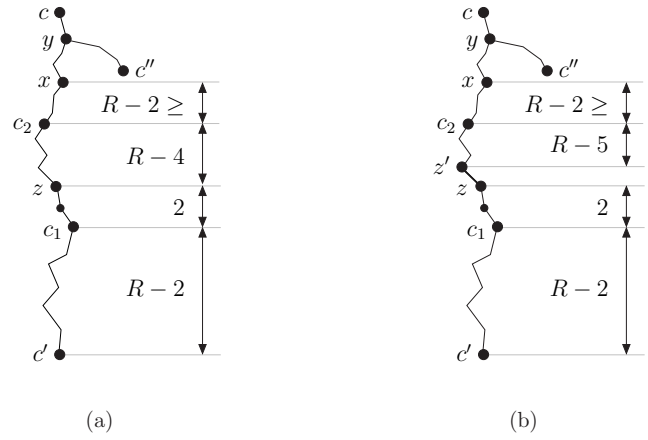


Figure 6: To the proof of Lemmas 4.1 and 4.5.

Hence $d(z,\hat{c}) = d(z,c_2) + d(c_2,\hat{c})$, $d(z,c'') = d(z,c_2) + d(c_2,c'')$, and $d(c_2,c'') \le d(c_2,\hat{c})$ by the choice of c'', yielding $d(z,c'') \le d(z,\hat{c}) \le R-1$. The choice of $\hat{c}$ implies $d(z,c'') = d(z,\hat{c})$, therefore c'' can play the role of $\hat{c}$. In particular, this implies that c'' is not a descendant of x as illustrated by Fig. 6a. Thus $R-1 \ge d(z,c'') = d"(z,c_2) + d(c_2,x) + d(x,c'') \ge R-4+d(c_2,x)$ because $d(z,c_2) = R-4$, yielding $d(c_2,x) \le 3 \le R-2$. Notice also that $d(x,c'') < d(z,c'') \le R-1$ because $x \in P(z,c'') - \{z\}$, i.e., $d(x,c'') \le R-2$. Finally, we assert that $d(c,c'') < d(c,c')$. This is obviously true if $c'' \in P(c,x) \subset P(c,c')$ (in fact, $c'' = c$.) Otherwise, let y be the nearest common ancestor of c'' and x. Then $d(y,c'') \le R-3$ while $d(y,c') = d(y,x) + d(x,c_2) + d(c_2,c') \ge 1 + 2R - 4 = 2R-3 > R-3$. Thus $d(c,c'') < d(c,c')$, and at the moment when the algorithm analyzes the pair c,c', the vertex c'' is already affected to A or A'. Since c_2 and c'' belong to distinct sets A, A', we can set $\{a,a'\} := \{c'',c_2\}$. Indeed $d(x,c_2) \le R-2, d(x,c'') \le R-2$, and $x \in P(c_2,c'')$, thus completing the proof. □

Lemma 4.2 *Any pair u,v of distinct vertices can be connected in the augmented graph $H = (V, E \cup E')$ by two vertex-disjoint paths P_1 and P_2 of length at most $D = 2R-1$.*

Proof. By Lemma 4.1, there exist four vertices $a,b \in A$ and $a',b' \in A'$ such that $\max\{d(a,u), d(a',u), d(b,v), d(b',v)\} \le R-2$ and $u \in P(a,a'), v \in P(b,b')$. Moreover, $P(a,a') \cap C \subseteq \{a,a'\}$ and $P(b,b') \cap C \subseteq \{b,b'\}$. Then the vertices r and r' may occur in the paths $P(a,a')$ and $P(b,b')$ only as their end-vertices, namely, $r' \in P(a,a')$ implies that $r' = a'$, while $r \in P(a,a')$ implies that $r = a$ and $u \in \{r,r'\}$.

If $\{u,v\} = \{r,r'\}$, then P_1 and P_2 are the two parallel edges between r and r'. On the other hand, if $u \in \{r,r'\}$, $v \notin \{r,r'\}$, then as P_1 and P_2 we take the two (u,v)-paths in the simple cycle of length at most $2R-4+3 = 2R-1$ formed by the path $P(b,b')$ and the new edges br, rr', and $r'b'$ (if $r' \ne b'$). So, assume that $r,r' \notin \{u,v\}$. If the paths $P(a,a')$ and $P(b,b')$ are disjoint, then as P_1 we take the path formed by $P(u,a), P(b,v)$, and the new edges ar, rb, while as P_2 we take the path formed by $P(u,a'), P(b',v)$, and the new edges $a'r'$ (if $r' \ne a'$), $r'b'$ (if $r' \ne b'$). These paths have length at most $2R-4+2 = 2R-2 < D$. They are disjoint

because they may intersect only in r and r' and it is easy to see that $r \notin P_2$ and $r' \notin P_1$. Now, consider the case when $P(a,a') \cap P(b,b') \neq \emptyset$. Then the length of the path $P(u,v)$ is at most D because $d(u,t) + d(t,v) \leq R-2+R-2 < D$ for any vertex $t \in P(a,a') \cap P(b,b')$. Set $P_1 := P(u,v)$. It remains to specify the second path P_2. First suppose that $r' \in P(u,v)$. Then $a' = r' = b'$ because r' may appear only as an end-vertex on the paths $P(a,a')$ and $P(b,b')$. Therefore $(P(u,a) \cup P(v,b)) \cap P(u,v) = \{u,v\}$. Since $r \notin P_1$, we take as P_2 the path of H consisting of $P(u,a)$, the new edges ar and rb, and the path $P(b,v)$. Its length is at most $R-2+2+R-2 = 2R-2 < D$. Hence, let $r' \notin P(u,v)$. If $P(u,v) \cap P(u,a) = \{u\}$, then set $Q_1 := P(u,a), \alpha_1 := a, \beta_1 := r$. Otherwise, we have $P(u,v) \cap P(u,a') = \{u\}$, and set $Q_1 := P(u,a'), \alpha_1 := a', \beta_1 := r'$. In both cases, $Q_1 \cap P(u,v) = \{u\}$ and the length of Q_1 is at most $R-2$. Analogously, if $P(u,v) \cap P(v,b) = \{v\}$, then set $Q_2 := P(v,b), \alpha_2 := b, \beta_2 := r$, otherwise set $Q_2 := P(v,b'), \alpha_2 := b', \beta_2 := r'$. Again, in both cases $Q_2 \cap P(u,v) = \{v\}$ and the length of Q_2 is at most $R-2$. The paths Q_1 and Q_2 are disjoint, because all vertices of $Q_1 \cap Q_2$ would lie on the unique path of F connecting u and v and we know that $P(u,v)$ intersects $Q_1 \cup Q_2$ only in the vertices u, v. We take as P_2 the path of H consisting of Q_1, the edges $\alpha_1\beta_1, \beta_1\beta_2$ (if $\beta_1 \neq \beta_2$), $\alpha_2\beta_2$, and the path Q_2. Its length is at most $2(R-2)+3 = 2R-1 = D$ and $P_1 \cap P_2 = \{u,v\}$. This establishes that indeed H is a feasible solution to problem A2VDBP. □

Lemma 4.3 $|A| + |A'| \leq 3|C| - 4$.

Proof. In Phase 3, for vertex $c \in (C-L) \cup S$ and each its direct descendant c' in C, the algorithm insert in $A \cup A'$ at most two new vertices (if $c = r$ and $c' = r'$, then no new vertex is added). Any vertex $c' \in C - S$ either belongs to L_0 or is a direct descendant of a unique vertex of $(C-L) \cup S$. Hence the number of new vertices is at most $2(|C| - |S| - |L_0|) - 2$. Since the vertices of L_0 are included in both sets A and A' and the remaining vertices of C in only one such set, we conclude that $|A| + |A'| \leq 2(|C| - |S| - |L_0|) - 2 + 2|L_0| + (|C| - |L_0|) = 3|C| - 2|S| - |L_0| - 2 \leq 3|C| - 4$. □

Hence the number of edges added by the algorithm is $|A| + |A'| + 1 \leq 3|C| - 3$. From Proposition 3.1 and Lemma 4.2 we obtain the following result:

Theorem 4.4 *Algorithm A2VDBP is a factor 6 approximation algorithm for the problems A2VDBP, A2EDBP, and ADCE on forests $F = (V,E)$ for any odd $D = 2R - 1 \geq 9$.*

4.2 Even diameter

Suppose now that $D = 2R$. The algorithm is the same as in the case of odd D except that the covering of the vertices of F by balls of radius R is replaced by the covering of the edges with balls of the same radius. The proof of Lemma 4.5 is different from that of Lemma 4.1. In particular, in the proof of Lemma 4.5 we must require that $D \geq 12$. All other proofs are similar.

Lemma 4.5 *For any $x \in V$ there exists two vertices $a \in A$ and $a' \in A'$ such that $d(x,a) \leq R-2$, $d(x,a') \leq R-2$, $P(a,a') \cap C \subseteq \{a,a'\}$, and x belongs to the (possibly degenerated) path $P(a,a')$.*

Proof. The result trivially holds if $x \in L_0$, because $L_0 \subseteq A \cap A'$ and we can set $a := x =: a'$. If $x \in S$, then $x \in A$ and we can set $a := x$. Otherwise, let c' be the nearest to x vertex of C. According to the algorithm, we can set $a' := c'$ if $d(x,c') \leq R-2$ and $a' := c_1$ if $R-1 \leq d(x,c') \leq 2R-4$. Finally, if $d(x,c') \geq 2R-3$, we can set $a' := c_2$. Indeed, let uv be the edge of $P(x,c')$ such that $d(u,x) = R$ and $d(v,x) = R+1$. Since both end-vertices of this edge must be covered, there exists another ball of radius R centered at a vertex of C located at distance at most R from u. Hence, by the choice of c', $d(x,c') \leq R+R \leq 2R$ and $d(x,c_2) \leq 2R-(2R-4) \leq 4 \leq R-2$ because $R \geq 6$. Now, suppose that $x \in C-S$. In this case, the vertex x is a direct descendant of a vertex $c \in C$ and $x \in D_c$. Therefore, according to the algorithm, we set $\{a,a'\} := \{x,c\}$ if $d(x,c) \leq R-2$ and $\{a,a'\} := \{x,c_1\}$ otherwise. Next, suppose that $x \notin C \cup L_0$. Let c be a nearest ancestor of x in C and let c' be a nearest descendant of x in C. Then the vertex c' is clearly a direct descendant of c, i.e. $c' \in D_c$. If $d(c,c') \leq R-2$ or $R-1 \leq d(c,c') \leq 2R-4$, then a et a' can be chosen as in the proof of Lemma 4.1. Thus, suppose that $d(c,c') \geq 2R-3$, i.e, the third rule has been applied. We can assume that $x \in P(c_2,c)-\{c_2\}$, otherwise the lemma follows by setting $\{a,a'\} := \{c',c_1\}$ if $x \in P(c',c_1)$ and $\{a,a'\} := \{c_1,c_2\}$ if $x \in P(c_1,c_2)$. Let z and z' be the vertices of $P(x,c')$ located at distance R and $R+1$ from c', respectively ($\{z,z'\} \subseteq P(c_1,c_2) - \{c_2\}$ since $R \geq 6$, thus $x \notin \{z,z'\}$). Denote by $\hat{c} \in C$ the nearest to z' center of a ball of radius R which covers the edge z,z'. The vertex $\hat{c}$ is not a descendant of z, otherwise we would have $d(z',\hat{c}) \leq R$ and $d(z,\hat{c}) \leq R-1 < R$, contrary to the choice of c'. Now, suppose that $\hat{c}$ is a descendant of x. Let t be the nearest common ancestor of z' and $\hat{c}$. Since $t \in P(z',x)$, $d(t,c') > R$, and $d(t,\hat{c}) \leq R$, we obtain a new contradiction with the choice of c'. Thus $\hat{c}$ is not a descendant of x either (see Fig. 6b for an illustration). We conclude that $d(z',\hat{c}) = d(z',c_2)+d(c_2,\hat{c})$, $d(z',c'') = d(z',c_2)+d(c_2,c'')$ and $d(z',c'') = d(z',\hat{c})$, therefore the vertex c'' can play the role of $\hat{c}$. This implies that c'' cannot be a descendant of x. Thus $R \geq d(z,c'') = d(z,c_2)+d(c_2,x)+d(x,c'') \geq R-4+d(c_2,x)$ because $d(z,c_2) = R-4$, from which we infer that $d(c_2,x) \leq 4 \leq R-2$. Note that $d(x,c'') < d(z',c'') \leq R-1$ because $x \in P(z',c'')-\{z'\}$, i.e., $d(x,c'') \leq R-2$. Finally, we assert that $d(c,c'') < d(c,c')$. This is obviously true if $c'' \in P(c,x) \subset P(c,c')$ (in fact, $c'' = c$.) Otherwise, let y be the nearest common ancestor of c'' and x. Then $d(y,c'') \leq R-3$ while $d(y,c') = d(y,x)+d(x,c_2)+d(c_2,c') \geq 1+2R-4 = 2R-3 > R-3$. Thus $d(c,c'') < d(c,c')$, and at the moment when the algorithm analyzes the pair $\{c,c'\}$, the vertex c'' is already assigned to A or A'. Since c_2 and c'' belong to different sets A, A', we can set $\{a,a'\} := \{c'',c_2\}$. Indeed $d(x,c_2) \leq R-2, d(x,c'') \leq R-2$, and $x \in P(c_2,c'')$, thus completing the proof. □

From Proposition 3.1 and Lemmas 4.2, 4.3, and 4.5 we derive the following theorem

Theorem 4.6 *Algorithm A2VDBP is a factor 6 approximation algorithm for problems A2VDBP, A2EDBP, and ADCE on forests $F = (V,E)$ for any even $D = 2R \geq 12$.*

5 A factor 4 approximation algorithm for ADCE

In this section, we modify (and simplify) the algorithm A2VDBP in order to return feasible solutions of smaller size for the problem ADCE. As in previous section, we have different algorithms depending of the parity of D.

5.1 Odd diameter

We use the same notations and conventions as in the algorithm A2VDBP for odd D, except that we do not need to consider the vertex r' and instead of sets A, A' we will use one multiset A (A contains two copies of each vertex from $L_0 \cup S - \{r\}$). Namely, given a minimum vertex $(R-1)$-dominating set C of F containing the set of leaves L, we proceed each rooted tree of F and complete C to a set A with the property that for every vertex x there exist two vertices $a, a' \in A$ such that $x \in P(a, a')$ and $d(x, a) \leq R-2, d(x, a') \leq R-1$. The algorithm returns the augmentation $E' = \{ar : a \in A\}$. Every element of A gives rise to an added edge, therefore r will be connected with every vertex of $L_0 \cup (S - \{r\})$ by two parallel edges.

Algorithm ADCE(F,2R-1)
Input: A forest $F = (V, E)$ and an odd integer $D = 2R - 1 \geq 5$.
Output: A set E' of new edges, such that $\mathrm{diam}(H - e) \leq D$ for any $e \in E \cup E'$ where $H = (V, E \cup E')$ and $|E'| \leq 4 \cdot OPT_{ADCE}(F)$.
Phase 0: Root every tree containing at least two vertices at some leaf. Denote by S the list of such roots and pick $r \in S$.
Phase 1: Compute a minimum vertex $(R-1)$-dominating set C with $L \subseteq C$. Set $A := (C - \{r\}) \cup (L_0 \cup (S - \{r\}))$.
Phase 2: For each vertex $c \in C - (L_0 \cup S)$, find the closest in C ancestor c' of c. If $d(c, c') \geq R$, then find $c_1 \in P(c, c')$ at distance $R-1$ to c and insert c_1 in A.
Phase 3: Return $E' = \{ra : a \in A\}$.

We present several auxiliary results establishing the feasibility of the augmentation E'.

Lemma 5.1 *For any vertex $x \in V - \{r\}$, there exists two different elements a, a' of the multiset A such that $x \in P(a, a')$ and $d(x, a) \leq d(x, a') \leq R-1$.*

Proof. First, let $x \in V - S$. The result straightforwardly follows from the algorithm if x is a vertex of C or if x has been inserted in A. So, let $x \notin A$, and let c be a nearest to x vertex of C (clearly, $d(c, x) \leq R-1$). If x is an ancestor of c, then c and one of the vertices c' or c_1 described in Phase 2 form the required pair $\{a, a'\}$. Now, assume that c is not a descendant of x. Suppose by way of contradiction that all descendants of x in A are located at distance $\geq R$ from x. Consider an arbitrary direct descendant a of x in A. Since $d(x, a) \geq R$, from the algorithm we infer that $a \in A - C$. Indeed, if $a \in C$, then, since all ancestors of a in C are also ancestors of x, in Phase 2 of the algorithm a vertex $c_1 \in P(a, x)$ would be inserted in A, contrary to the choice of a. Thus $a \in A - C$ and a has been added to the multiset A

in Phase 2 of the algorithm. This means that the vertex a is at distance $R-1$ from some descendant $b \in C$, i.e. $d(x,b) \geq R + (R-1) = 2R-1$. From this we can deduce that x is at distance at least $2R-1$ from all its descendants from C. Pick a descendant x' of x satisfying $d(x,x') = R-1$. Since any descendant of x' in C is at distance at least $(2R-1)-(R-1) = R$ from x' and $d(x,x') = R-1$, the center $c'' \in C$ of any $(R-1)$-ball covering x' is a descendant of x but not a descendant of x'. Let x'' be the nearest common ancestor of x' and c''. Then $d(c'',x'') \leq R-2$ and $d(x,x'') \leq R-2$, yielding $d(x,c'') \leq 2R-4 < 2R-1$, contrary to the fact that x is at distance $\geq 2R-1$ from any of its descendants from C. This contradiction completes the proof of the case $x \in V - S$. If $x \in S - \{r\} \cup L_0$, we can take as a and a' the two occurrences of x in A. □

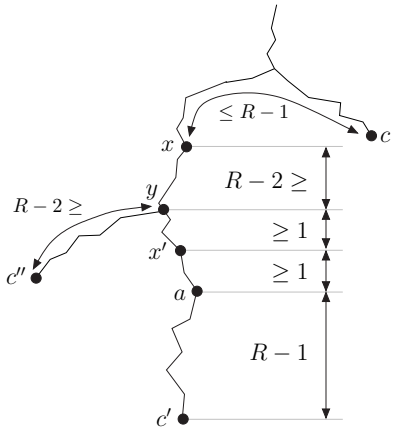


Figure 7: To the proof of Lemma 5.1.

Lemma 5.2 *For any vertex $x \in V - \{r\}$, there exists $a \in A$ such that $d(x,a) \leq R-2$.*

Proof. Let $x \notin L_0 \cup S$, otherwise we can set $a = x$ by definition of A. Suppose by way of contradiction that all vertices of A are at distance $\geq R-1$ from x and pick a vertex $c' \in C \subseteq A$ at distance $R-1$ to x. If c' is a descendant of x, then x will be inserted in A when c' will be analyzed in Phase 2 of the algorithm, a contradiction. So, all descendants of x from C are at distance $\geq R$ to x. Let x' be the son of x and let c'' be any closest to x' vertex of C. Since $d(x,c'') \geq R-1$, c'' must be a descendant of x'. Thus $d(x',c'') = R-1$ by what has been shown about x. As $P(c'',x) \cap C = \{c''\}$, the vertex x' will be inserted in A when c'' will be analyzed. Since $R \geq 3$, we get a contradiction with the choice of x. □

Lemma 5.3 *For any edge $e \in E \cup E'$, we have $diam(H-e) \leq D$.*

Proof. Pick two arbitrary vertices $u, v \in V$. Combining Lemmas 5.1 and 5.2 we can conclude that there exist four elements $a, a', b, b' \in A$ such that $a \neq a', b \neq b'$, $u \in P(a,a'), v \in P(b,b')$, $d(u,a) \leq R-2, d(u,a') \leq R-1$, and $d(v,b) \leq R-2, d(v,b') \leq R-1$. If $r \in \{u,v\}$ then obviously we have two disjoint (u,v)-paths of length at most R in H. Now, suppose that

$r \notin \{u, v\}$. For any edge e, the distances in $H - e$ between u and r and between v and r are at most R. If one of these two distances is at most $R - 1$, we are done. Otherwise, we must have either $a = b$ and $e = ar \in E'$ or e belong to $P(u, a) \cap P(v, b)$. In the first case, e does not belong to the path $P(u, v)$ and the length of this path is at most D, because $d(u, v) \leq d(u, a) + d(v, b) \leq 2R - 4 \leq D$. Now, suppose that e belong to $P(u, a) \cap P(v, b)$. Then u and v belong to the same tree component T of F. If e does not belong to $P(u, v)$, then $d(u, v) \leq d(u, a) + d(v, b) \leq 2R - 4$. Otherwise, if $e \in P(u, v) \cap P(u, a) \cap P(v, b)$ then the pairs u, b and v, a lie in different connected components of $T - e$. Therefore $d(v, a) + d(u, b) \leq d(u, a) + d(v, b) \leq 2R - 4$ and the smallest of the distances $d(v, a)$ and $d(u, b)$, say the first, is at most $R - 2$. Thus $d_{H-e}(u, v) \leq d_{H-e}(u, r) + d_{H-e}(r, a) + d_{H-e}(a, v) \leq R + 1 + R - 2 = 2R - 1$, yielding $\mathrm{diam}(H - e) \leq 2R - 1 = D$ for every edge $e \in E \cup E'$. □

The next lemma follows immediately from the analysis of the algorithm:

Lemma 5.4 $|A| \leq 2|C|$.

From Proposition 3.1 and previous lemmata, we obtain the following result:

Theorem 5.5 *Algorithm ADCE is a factor 4 approximation algorithm for problem ADCE on forests $F = (V, E)$ for any odd $D = 2R - 1 \geq 5$.*

5.2 Even diameter

Now suppose that $D = 2R$. As in the case of the factor 6 algorithm, we use a covering of edges of F with a minimum number of balls of radius R. Then the algorithm is similar to the factor 4 algorithm in the odd case, however its correctness proof must be modified.

Algorithm ADCE(F,2R)
Input: A forest $F = (V, E)$ and an even integer $D = 2R - 1 \geq 6$.
Output: A set E' of new edges, such that $\mathrm{diam}(H - e) \leq D$ for any $e \in E \cup E'$ where $H = (V, E \cup E')$ and $|E'| \leq 4 \cdot OPT_{ADCE}(F)$.
Phase 0: Root every tree containing at least two vertices at some leaf. Denote by S the list of such roots and pick $r \in S$.
Phase 1: Compute a minimum edge R-dominating set C with $L \subseteq C$. Set $A := C \cup (L_0 \cup S)$.
Phase 2: For each vertex $c \in C - (L_0 \cup S)$, find the closest in C ancestor c' of c. If $d(c, c') \geq R + 1$, then find $c_1 \in P(c, c')$ at distance R to c and insert c_1 in A.
Phase 3: Return $E' = \{ra : a \in A,\ a \neq r\}$.

Lemma 5.6 *For each $x \in V - A$, there exist two different elements a and a' of the multiset A such that $x \in P(a, a')$ and $d(x, a) \leq d(x, a') \leq R - 1$.*

Proof. Let c and c' be the nearest descendant and the nearest ancestor of x in C. First suppose that $d(c, x) \leq R$. If $d(c, c') \leq R$, since $x \notin \{c, c'\}$, it suffices to set $\{a, a'\} := \{c, c'\}$. Thus we

can suppose that $d(c, c') > R$. In this case, the algorithm will insert in A a vertex c_1 which is located at distance R from c. Since $x \neq c_1$ and $d(c, x) \leq R$, we infer that $d(c, x) \leq R - 1$ and $d(c_1, x) \leq R - 1$ and in this case we can set $\{a, a'\} := \{c, c_1\}$.

Suppose now that $d(c, x) \geq R + 1$. Let x' be the neighbor of x in the path $P(c, x)$ (it exists because $x \notin L$). As c is located at distance at least $R + 1$ from x, the edge xx' is not covered by the ball of radius R centered at c. Thus, by the choice of c, there exists a vertex c'' at distance at most R from x' which is not a descendant of x (because the edge xx' must be covered). Then we can set $a := c''$ because $d(c'', x) \leq R - 1$. Now, suppose by way of contradiction that all descendants of x in A are located at distance at least R from x. Then, as $d(c, x) \geq R + 1$ and $x \neq c'$, the algorithm will add to A the vertex c_1 of the path $P(c, x)$ at distance R from c, yielding $d(c, x) \geq d(c, c_1) + d(c_1, x) \geq R + R \geq 2R$. Hence, by the choice of c, all descendants of x in C have distance at least $2R$ to x. Let y and y' be two adjacent descendants of x located at distance $R - 1$ and R from x. Since $x \notin C$, the edge yy' can be covered only by a ball of radius R centered at a descendant of x which is located at distance at most $R - 1 + R = 2R - 1$ from x. This contradicts the assumption that all descendants of x in C have distance at least $2R$ to x. Hence at least one descendant a' of x in A is located at distance at most $R - 1$ from x. □

Lemma 5.7 *For each $x \in V - A$ there exists $a \in A$ such that $d(x, a) \leq R - 2$.*

Proof. Suppose by way of contradiction that all vertices of the set A are located at distance at least $R - 1$ from x. Let c' be a nearest descendant of x in C. Then $d(x, c') \geq R - 1$ by our assumption. Let x' be the neighbor of x on the path $P(x, c')$. If $d(x, c') \leq R + 1$, then according to Phase 2 of the algorithm, either the vertex x (if $d(x, c') = R$), or its father (if $d(x, c') = R - 1$), or the vertex x' (if $d(x, c') = R + 1$) belongs to the multiset A, which leads to a contradiction because $R \geq 3$. Therefore all descendants of x in the set C have distance at least $R + 2$ to x. Let x'' be the son of x'. From what have been shown above, the edge $x'x''$ cannot be covered by a ball of radius R centered at a descendant of x. Hence, there exists a vertex $c'' \in C$ at distance at most R from x'' which is not a descendant of x. Since $d(x, x'') = 2$, we conclude that $d(x, c'') \leq R - 2$, a contradiction. □

Lemma 5.8 *For each vertex $c \in A - \{r\}$, there exists a vertex c' different from c (or a copy of c) in A such that $d(c, c') \leq R$.*

Proof. If $c \in A - C$, then by construction of A, there exists a vertex $c' \in C$ at distance R from c. On the other hand, if $c \in (S - \{r\}) \cup L_0$, the vertex c occurs at least twice in the multiset A. Finally, let $c \in C - (S \cup L_0)$. Then according to the algorithm there exists a vertex $c' \in A$ located at distance R from c. □

Lemma 5.9 *For any edge $e \in E \cup E'$, we have $diam(H - e) \leq D$.*

Proof. Pick two arbitrary vertices $u, v \in V$. We distinguish several cases in function of the position of u and v with respect to A. First, let $\{u, v\} \cap A = \emptyset$. In this case we can simply use the proof of Lemma 5.3 because Lemmas 5.6 and 5.7, applied to a vertex of $V - A$, are equivalent to Lemmas 5.1 and 5.2.

Suppose that $u \in A - \{v\}$ and $v = r$. Then uv is an edge of E'. Let c be a nearest to u vertex in $A - \{u\}$. From Lemma 5.8, we know that $d(c, u) \leq R$. Then the edge uv and the path consisting of the path of F between u and c followed by the edge cv represent two edge-disjoint (u, v)-paths of H of length at most $2R$ (see Fig. 8a). Thus, independently of the choice of the removed edge e, we have $d_{H-e}(u, v) \leq 2R = D$.

Now, let $\{u, v\} \subseteq A - \{r\}$. By Lemma 5.8 there exists two vertices c and c' such that $d(u, c) \leq R$ and $d(v, c') \leq R$. Again, we can provide two edge-disjoint (u, v)-paths of length at most $2R$. If $P(u, c) \cap P(v, c') \neq \emptyset$, then we obtain the situation depicted in Fig.8b and $P(u, v)$ has length at most $2R$. In this case, the second path (of length 2), consists of two added edges ur and vr. If $P(u, c) \cap P(v, c') = \emptyset$, then the situation is depicted in Fig.8c. In

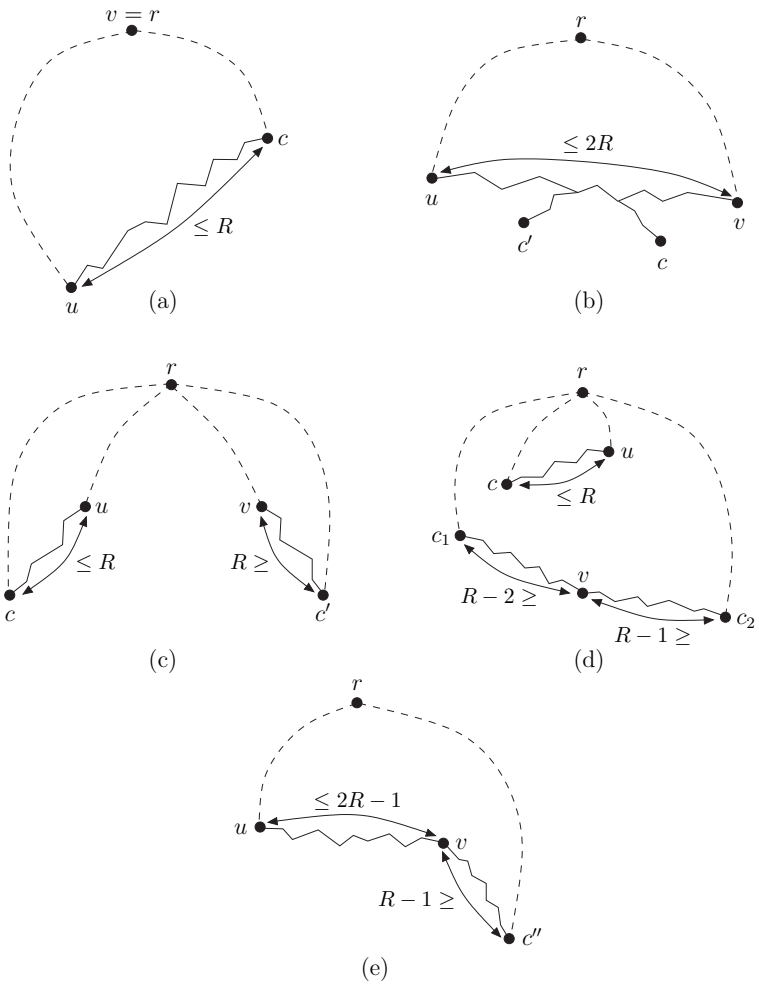


Figure 8: To the proof of Lemma 5.9.

this case, the first path consists of $P(u,c)$ and two added edges $\{c,r\}$ and $\{v,r\}$. The second path consists of $P(v,c')$ and two added edges $\{c',r\}$ and $\{u,r\}$. The length of each of these two (u,v)-paths is at most $R+2 \leq 2R$.

Finally, suppose that $u \in A$ and $v \notin A$. Let c_1 and c_2 be two closest to v vertices of A such that v lies on the path $P(c_1,c_2)$. Assume without loss of generality that $d(v,c_1) \leq d(v,c_2)$. From Lemmas 5.6 and 5.7 we deduce that $d(v,c_1) \leq R-2$ and $d(v,c_2) \leq R-1$. If $P(c_1,c_2) \cap P(u,c) = \emptyset$, then we can derive two edge-disjoint (u,v)-paths of length at most $2R$ as illustrated in Fig. 8d. Namely, the first path consists of $P(u,c)$, two edges cr and rc_1, followed by the path $P(c_1,v)$. Its length is at most $R+2+R-2 = 2R = D$. The second (u,v)-path consists of two added edges ur and rc_2, followed by the path $P(c_2,v)$. Its length is at most $2+R-1 = R+1 < D$. Now, let $P(c_1,c_2) \cap P(u,c) \neq \emptyset$. In this case, the length of the path $P(u,v)$ is at most $R+R-1 = 2R-1 < D$ and it can be used as the first path. Since the vertex v belongs to the path $P(c_1,c_2)$ and F is a forest, either $P(u,v) \cap P(v,c_1) = \{v\}$ or $P(u,v) \cap P(v,c_2) = \{v\}$ holds. If $P(u,v) \cap P(v,c_1) = \{v\}$, then we can set $c'' := c_1$, otherwise we can set $c'' := c_2$. As shown in Fig. 8e, the second (u,v)-path consists of two added edges ur and rc'', followed by the path $P(c'',v)$. Its length is at most $2+R-1 = R+1 < D$. □

From these lemmas and Proposition 3.1, we obtain the following result:

Theorem 5.10 *The algorithm* ADCE *is a factor 4 approximation algorithm for problem ADCE on forests $F=(V,E)$ for any even $D = 2R \geq 6$.*

We do not know if our analysis of the approximation ratio of the algorithms presented in last two sections is tight. Actually, we conjecture that all four algorithms have approximation ratio better than 6 and 4. The worst example we know is a forest consisting of $2k$ isolated vertices and $D = k$. In this case, both algorithms output solutions consisting of $4k-2$ edges: A2VDBP outputs two parallel edges running between r and r' and two edges xr and xr' for every vertex $x \notin \{r,r'\}$, while algorithm ADCE adds two edges between r and all vertices $x \neq r$. On the other hand, the optimal solution consists of the $2k$ edges of the cycle C_{2k}.

6 Implementation issues

In this section, we describe how to implement the algorithms A2VDBP and ADCE in $O(|V|\log|V|)$ time. Since each phase of the algorithms except the first and the last deals with vertices lying in a common tree of the forest F, further we will assume that F consists of a single rooted tree T.

Phase 1 of both algorithms computes a minimum vertex $(R-1)$-dominating set C or a minimum edge R-dominating set C and takes linear time [2]. In Phase 2 of the algorithm A2VDBP we need to compute the sets D_c of direct descendants of all $c \in (C-L) \cup S$ and to sort them according to their distances to c. By definition, any vertex of C is a direct descendant of at most one vertex of $(C-L) \cup S$, thus the total size of the lists D_c is $O(|C|)$. To compute D_c for some vertex c, we perform a partial Breadth-First-Search (BFS) of the subtree T_c rooted at c and consisting of c and all its descendants in T. During this search,

each time when a vertex c' of C is encountered, c' is inserted at the end of the current list D_c and the subtree $T_{c'}$ is not traversed. Since BFS treats the vertices of T_c level-by-level, the vertices of the resulting list D_c are sorted according to their distances to c. During the computation of the sets D_c, each vertex of C is visited twice and each other vertex of T is visited once, thus Phase 2 of A2VDBP needs $O(|V|)$ time.

To implement Phase 3 of algorithm A2VDBP and Phase 2 of algorithm ADCE in $O(|V|\log|V|)$ time we use a data structure supporting the operations INSERT, MAXIMUM, EXTRACT-MAX, and DISJOINT UNION in $O(\log|V|)$ time (for example, binomial heaps described in [6]) as well as three arrays $\texttt{anc}, \texttt{anc}'$, and $\texttt{nearest}$. The arrays $\texttt{anc}$ and $\texttt{anc}'$ contain for each vertex $c \in C$ the ancestors of c located at distance $R-2$ and $2R-4$, respectively, from c. The array $\texttt{nearest}$ contains, for each vertex x of the tree T, a closest to x vertex $\texttt{nearest}[x]$ of C which is not a descendant of x.

To compute $\texttt{anc}$ (the computation of $\texttt{anc}'$ is similar) the algorithm handles the vertices of the rooted tree T in a upward way. For a vertex p of T, the algorithm maintains in a binomial heap $BH(p)$ the set of all descendants of p from the set C having distance at most $R-2$ to p in such a way that the vertex in head of $BH(p)$ is furthest from p. As a key of each vertex in the heap we use its depth in the tree T (i.e., its distance to the root of T). A current vertex p is processed in the following way. If p is a leaf, then $p \in C$ by the algorithm and we set $BH(p) := \{p\}$. Otherwise, if p has $k > 0$ sons $x_1, \ldots, x_k$, the algorithm merges the heaps $BH(x_1), \ldots, BH(x_k)$ into a single binomial heap $BH(p)$ by using $k-1$ DISJOINT UNION operations. Then, while the vertex c in head of $BH(p)$ has distance exactly $R-2$ to p, c is removed from $BH(p)$ and p is assigned to $\texttt{anc}[c]$ (i.e., $\texttt{anc}[c] := p$). Finally, if p belongs to C, then p is inserted in $BH(p)$. Notice that the distance $d(p,c)$ can be computed in constant time from the depths of the vertices p and c in the rooted tree T and the fact that p is an ancestor of c. Since each of the operations INSERT, MAXIMUM, EXTRACT-MAX, and DISJOINT UNION on binomial heaps can be executed in $O(\log|V|)$ time [6] and for a vertex p with k sons we perform the union of k disjoint binomial heaps, the arrays $\texttt{anc}$ and $\texttt{anc}'$ can be computed in total $O(|V|\log|V|)$ time.

Next we explain how to compute in linear time the array $\texttt{nearest}$. First, for each vertex $p \in V$, we compute a nearest descendant $\texttt{nearest\_desc}[p]$ of p from the set C. The array $\texttt{nearest\_desc}$ can be computed by traversing the vertices of T in a upward way. Given the current vertex p, we set $\texttt{nearest\_desc}[p] := p$ if $p \in C$, otherwise $\texttt{nearest\_desc}[p]$ is selected among the nearest descendants of the sons of p. Namely, if x_1 is a son of p providing the minimum of the list of distances $D[p] = \{d(x, \texttt{nearest\_desc}[x]) : x \text{ is a son of } p\}$, then we set $\texttt{nearest\_desc}[p] := \texttt{nearest\_desc}[x_1]$. The array $\texttt{nearest}$ is computed in a downward way. Suppose that p is a current vertex for which $c' = \texttt{nearest}[p]$ has been computed and we wish to compute the entries $\texttt{nearest}[x]$ for all sons x of p. If $p \in C$, then clearly we must set $\texttt{nearest}[x] := p$. Now, let $p \notin C$. If p contains a unique son x, then obviously $\texttt{nearest}[x] := c'$. So suppose that p has at least two sons. Since all leaves of T belong to the set C, the list $D[p]$ defined above must contain at least two entries. Let x_1 and x_2 be the two sons of p providing the two smallest values of $D[p]$. Let $c_1 = \texttt{nearest\_desc}[x_1]$ and $c_2 = \texttt{nearest\_desc}[x_2]$, and suppose without loss of generality that $d(x_1, c_1) \leq d(x_2, c_2)$. If

$d(p, c') \leq d(p, c_1) = d(x_1, c_1) + 1$, then we set $\texttt{nearest}[x] := c'$ for all sons x of p. Otherwise, we set $\texttt{nearest}[x] := c_1$ for all sons x of c distinct from x_1. For x_1 we set $\texttt{nearest}[x_1] := c'$ if $d(p, c') \leq d(p, c_2)$ and $\texttt{nearest}[x_1] := c_2$ otherwise. The four occurring cases are illustrated in Fig. 9. For a given vertex p, the computation of `nearest` for its sons requires linear time in the number of sons of p, thus array `nearest` can be calculated in total $O(|V|)$ time.

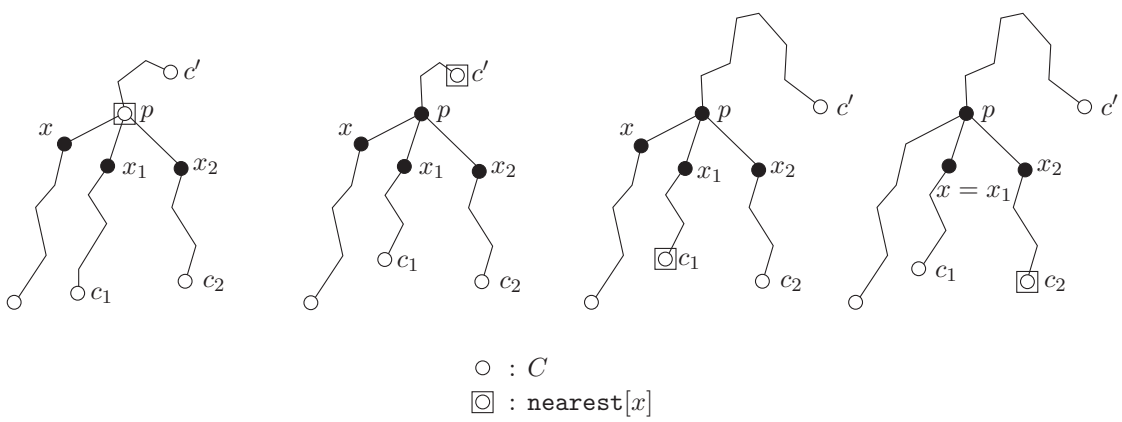


Figure 9: Four cases in the computation of $\texttt{nearest}[x]$.

Finally, we show how to implement Phase 3 of algorithm A2VDBP and Phase 2 of algorithm ADCE in constant time using the arrays $\texttt{anc}, \texttt{anc}'$, and `nearest`. We illustrate this on case $d(c, c') \geq 2R - 3$ of Phase 3 of A2VDBP. In this case, c_1 is the vertex $\texttt{anc}[c']$ and c_2 is the vertex $\texttt{anc}'[c']$. On the other hand, c'' is nothing else but the vertex $\texttt{nearest}[c_2]$. Therefore it remains to check if c'' belongs to the set A and to insert the vertices c_1, c_2, and c' in A or A' following the Rule 3. Summarizing, we obtain the following result:

Theorem 6.1 *The algorithms* A2VDBP *and* ADCE *can be implemented in* $O(|V| \log |V|)$ *time.*

References

[1] N. Alon, A. Gyarfas, M. Ruszinko, Decreasing the diameter of bounded degree graphs, *J. Graph Theory*, **35** (2000), 161–172.

[2] G.J. Chang, Labeling algorithms for domination problems in sun-free chordal graphs, *Discrete Appl. Math.*, **22** (1988/1989), 21–34.

[3] V. Chepoi, B. Estellon, K. Nouioua, Y. Vaxès, Mixed covering of trees and the augmentation problem with diameter constraints, *Algorithmica* **45** (2006), 209-226.

[4] V. Chepoi, Y. Vaxès, Augmenting trees to meet biconnectivity and diameter constraints, *Algorithmica*, **33** (2002), 243–262.

[5] F. R. K. Chung, M. R. Garey, Diameter bounds for altered graphs, *J. Graph Theory*, **8** (1984), 511–534.

[6] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*, Second Edition, The MIT Press, 2001.

[7] D. Dolev, J. Halpern, B. Simons, H.R. Strong, A new look at fault tolerant network routing, *Information and Computation*, **72** (1987), 180–196.

[8] K. P. Eswaran, R. E. Tarjan, Augmentation problems, *SIAM J. Computing*, **5** (1976), 653–665.

[9] A. M. Farley, A. Proskurowski, Self-repairing networks, *Parallel Processing Letters*, **3** (1993) 381–391.

[10] M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness,* Freeman, San Francisco, CA, 1979.

[11] T. Ishii, S. Yamamoto, H. Nagamochi, Augmenting forests to meet odd diameter requirements, *International Symposium on Algorithms and Computation* (ISAAC'03), Lecture Notes in Computer Science, **2906** (2003), pp. 434–443 and *Discrete Optimization,* **3** (2006), 154-164.

[12] G. Kant, H.L. Bodlaender, Planar graph augmentation problems, *Workshop on Algorithms and Data Structures* (WADS'91), Lecture Notes in Computer Science **519** (1991), pp. 286–298.

[13] Ch.-L. Li, S.Th. McCormick, D. Simchi–Levi, On the minimum-cardinality-bounded-diameter and the bounded-cardinality-minimum-diameter edge addition problems, *Operations Research Letters* **11** (1992), 303–308.

[14] A.A. Schoone, H.L. Bodlaender, J. van Leeuwen, Diameter increase caused by edge deletion, *J. Graph Theory* **11** (1987), 409–427.

[15] V.V. Vazirani, *Approximation Algorithms*, Springer-Verlag, Berlin, 2001.

[16] D. B. West, *Introduction to Graph Theory,* Prentice Hall, London, 2001.