# Un algorithme distribué d'énumération des noeuds d'un réseau et application au calcul des distances entre 2 noeuds quelconques et du diamètre d'un réseau

Yves Métivier, John Michael Robson, Akka Zemmari

# *Un algorithme distribué d'énumération des noeuds d'un réseau et application au calcul des distances entre 2 noeuds quelconques et du diamètre d'un réseau*

Y. Métivier, J.M. Robson, et A. Zemmari

*Université de Bordeaux - Bordeaux INP, LaBRI, UMR CNRS 5800*
*351 cours de la Libération, 33405 Talence, France*
{*metivier, robson, zemmari*}*@labri.fr*

On considère un réseau où les noeuds (processus) communiquent par passage de messages. On suppose qu'il existe un noeud distingué. Dans ce travail, nous proposons et analysons un algorithme distribué d'énumération des noeuds d'un réseau utilisant des messages de taille $O(1)$ et tel que si deux noeuds $u$ et $v$ ont des numéros consécutifs alors ils sont à distance au plus 3. Nous montrons que l'algorithme se termine en $O(n)$ rondes, où n est le nombre de noeuds du réseau ; la taille des messages étant $O(1)$, la complexité en bits (nombre maximum de bits transitant par un canal de communication) est $O(n)$. Comme application de cette énumération, on présente un algorithme permettant de calculer les plus petites distances entre 2 noeuds quelconques d'un réseau et un algorithme permettant de calculer le diamètre d'un réseau en un temps $O(n)$ avec des messages de taille $O(1)$ ce qui donne une complexité en bits linéaire en le nombre de noeuds du réseau. Ces résultats améliorent les résultats obtenus par Peleg et al. (ICALP 2012) et Wattenhofer et al. (SODA 2012) qui présentent des algorithmes pour le calcul des plus petites distances entre deux noeuds quelconques ou le diamètre en temps $O(n)$ avec des messages de taille $O(\log n)$. Enfin nous montrons qu'il n'existe pas d'énumération des noeuds telle que deux noeuds ayant des numéros consécutifs sont à distance au plus 2. Une présentation complète de ce travail peut être consultée dans [MRZ16].

## 1 The Model and the Problem

We consider the standard message passing model for distributed computing. The communication model consists of a point-to-point communication network described by a connected graph $G = (V(G), E(G))$ $(= (V,E)$ for short) where the vertices $V$ represent network processes and the edges $E$ represent bidirectional communication channels. Processes communicate by message passing: a process sends a message to another by depositing the message in the corresponding channel. In the sequel, we consider only connected graphs. We assume the system is fully synchronous, namely, all processes start at the same time and time proceeds in synchronised rounds.

**Time Complexity.** A round (cycle) of each process is composed of the following three steps: 1. Send messages to (some of) the neighbours, 2. Receive messages from (some of) the neighbours, 3. Perform some local computation. As usual the time complexity is the number of rounds needed until every vertex has completed its computation.

**Bit Complexity.** By definition, in a bit round each vertex can send/receive at most 1 bit from each of its neighbours. The bit complexity of algorithm $\mathcal{A}$ is the number of bit rounds to complete algorithm $\mathcal{A}$.

**Network and Processes Knowledge.** The network $G = (V,E)$ is anonymous: unique identities are not available to distinguish the processes. We only assume that there is an elected (a distinguished) vertex denoted *Leader*. We do not assume any global knowledge of the network, not even its size or an upper bound on its size. The processes do not require any position or distance information. Each process knows from which channel it receives or to which channel it sends a message, thus one supposes that the network is represented by a connected graph with a port numbering function.

**All Pairs Shortest Paths, Diameter, Girth, Cut-Edge and Cut-Vertex.** A walk in a graph $G = (V,E)$ is a finite alternating sequence of vertices and edges, beginning and ending with a vertex and where each edge is incident with the vertices immediately preceding and following it. A trail is a walk in which no edge occurs more than once. A path is a trail in which all vertices are different, except that the initial and final vertices may be the same. A walk with at least 3 edges in which the first and last vertices are the same but all other vertices are distinct is called a cycle.

**Definition 1** *Let $G = (V,E)$ be a connected graph, let $u, v \in V$.*
*The distance $dist_G(u,v)$ between u and v in G, is the length of a shortest path between u and v in G.*
*The eccentricity of v is the greatest distance from v to any other vertex.*
*The diameter of G, denoted $D(G)$, is the maximum distance between any two vertices of G.*
*The girth of G is the length of a shortest cycle of G.*
*A cut-vertex is a vertex whose removal increases the number of connected components.*
*A cut-edge is an edge whose removal increases the number of connected components.*

The all pairs shortest paths (APSP for short) problem in $G$ is to compute the length of shortest paths between any pair of vertices in $G$.

A tree is a connected acyclic graph. A rooted tree is a tree in which one of the vertices is distinguished from the others (called *Leader* in this work). A spanning-tree of a connected graph $G = (V,E)$ is a tree $T = (V,E')$ such that $E' \subseteq E$.

# 2 Our Contribution

We present a distributed enumeration algorithm, denoted *DEA*, which assigns to each vertex of a graph $G$ of size $n$ having a distinguished vertex, denoted *Leader*, a unique integer of $\{1,2,\ldots,n\}$ such that the distance between any two vertices having two consecutive numbers is at most 3. This algorithm uses messages of size $O(1)$ and has a time complexity equal to $O(n)$.

The steps of Algorithm *DEA* are:

1. computation of a Breadth-First-Search (BFS) spanning-tree of $G$ whose root is *Leader*;
2. enumeration of the vertices with respect to a special traversal of the BFS spanning-tree.

In the following, we explain how the nodes of the BFS spanning-tree are numbered (we skip the BFS construction for the lack of place). This algorithm is denoted *Trav* and can be described as follows:

We add a loop on each leaf of the BFS spanning-tree and vertices are visited twice in a Depth-First-Search (DFS) traversal (a leaf is visited on arriving and by following the loop).

The traversal *Trav* is defined by:

- if it is the first visit to a vertex then go to the first child of the vertex if it has a child; if it has no child (i.e., it is a leaf) go from the leaf to itself;
- if it is the second visit to a vertex go to the next sibling of the vertex (two vertices $v$ and $w$ are siblings if they have the same parent), if it has a next sibling; if it has no next sibling go to the parent of the vertex if it has a parent else stop since it is the root of the BFS spanning-tree and the traversal is finished.

Thus, a visit to $v$ is immediately followed by a visit to a child or to a sibling (through the parent) or to the parent of $v$ or to $v$ itself (if $v$ is a leaf).

Let $v$ be a vertex, $\mathrm{v}_v^{(1)}$ (resp. $\mathrm{v}_v^{(2)}$) denotes the number of vertices visited before the first visit to $v$ (resp. before the second visit to $v$). We can prove (by induction) that $\mathrm{v}_v^{(1)}$ is even iff the level of $v$ is even and $\mathrm{v}_v^{(2)}$ is even iff the level of $v$ is odd. The number of a vertex $v$ is obtained by computing the number of visited vertices during the tree traversal before the first or the second visit to $v$. More precisely: *the vertex numbered k is the $k^{th}$ visited vertex such that an even number of vertices have been visited before it; it is denoted $v_k$*. An example of a run of *Trav* and the numbering of vertices is given in Fig. 1.

This enumeration enables the initialisation of anonymous waves, with respect to the enumeration order, i.e., the first wave is initialised by the vertex numbered 1, the second wave by the vertex numbered 2, etc. Anonymous waves reach vertices with respect to the enumeration order (i.e., the wave initialised by the

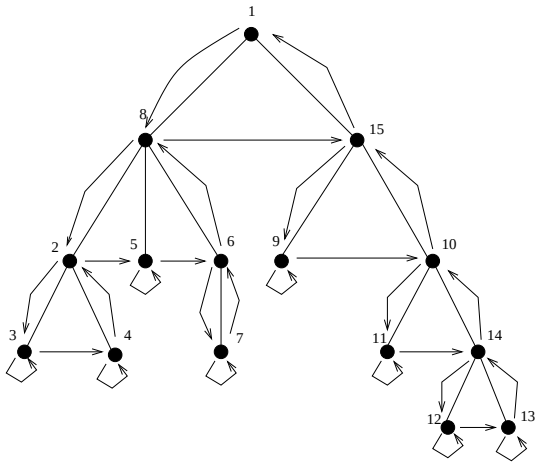


**Fig. 1:** An example of a run of *Trav* with the associated numbering of the vertices.

vertex numbered $i$ reaches any vertex after the wave initialised by the vertex numbered $i-1$ and before the wave initialised by the vertex numbered $i+1$) and thus are implicitly identified. This fact allows each vertex to compute its distance to any vertex without the computation and the use of the distance itself but by inference from the time; in this way all pairs shortest paths are obtained in time $O(n)$, each message having a constant size so that the bit complexity is $O(n)$. We deduce also a distributed algorithm for graph diameter with a bit complexity and a time complexity equal to $O(n)$. The diameter is obtained by the following steps:

1. Breadth-First-Search Tree Computation initiated by Leader;
2. Numbering of vertices;
3. Calculating distance between Leader and each vertex;
4. Waves initiation and all pairs shortest paths calculation;
5. Centralisation of the maximum distance and broadcast of the diameter.

We can note that only the maximum distance between any two nodes needs to be known, which is independent of node names. Furthermore, Frischknecht et al. proved ([FHW12], Theorem 5.1) that: "For any $n \geq 10$ and $B \geq 1$ and sufficiently small $\varepsilon$ any distributed randomized $\varepsilon$-error algorithm A that computes the exact diameter of a graph requires at least $\Omega(n/B)$ time for some $n$-node graph even when the diameter is at most 5," where $B$ is the size of messages. From this result we deduce that the bit complexity of our algorithm is optimal and the time complexity is also optimal for messages of size $O(1)$

| | Time | Message size (number of bits) | bit complexity |
|---|---|---|---|
| Almeida et al. [ABC11] | $O(D)$ | $O(n\log n)$ | $O(Dn\log n)$ |
| Holzer and Wattenhofer [HW12] | $O(n)$ | $O(\log n)$ | $O(n\log n)$ |
| Peleg et al. [PRT12] | $O(n)$ | $O(\log n)$ | $O(n\log n)$ |
| This paper | $O(n)$ | $O(1)$ | $O(n)$ |

This table summarises the comparison between the complexities of various diameter algorithms and the complexities of the diameter algorithm presented in this work.

The waves initiated by vertices with respect to the numbering of vertices can be used also for computing girth and for the determination of cut-edges and cut-vertices.

# 3 On the Enumeration

We may wonder whether it is possible to obtain an enumeration of vertices such that the distance between any two vertices having two consecutive numbers is at most 2. We explain now why the answer is negative. It indicates that in some certain sense our enumeration is optimal.

The enumeration of vertices of a connected graph such that two consecutive vertices of the enumeration are at distance at most 3 is also presented in [Sek71]; this paper presents a sequential algorithm for computing such an enumeration.

Let $G$ be a graph. A Hamiltonian path in $G$ is a path that includes all the vertices of $G$.

We recall that the cube of a graph $G$, denoted $G^3$, is the graph with the set of vertices of $G$ in which there is an edge between two vertices $u$ and $v$ if the distance between $u$ and $v$ in $G$ is at most 3. It was noticed by C. Gavoille [Gav14] that the existence of such an enumeration is equivalent to the fact that the cube of a connected graph $G$ contains a Hamiltonian path.

A cycle containing all vertices of $G$ is called a Hamiltonian cycle of $G$, and $G$ is called a Hamiltonian graph. From our enumeration result we deduce a well known result [CK69]:

**Theorem 1** *If $G$ is a connected graph then $G^3$ is a Hamiltonian graph.*

As for the cube of a graph, the square of a graph $G$, denoted $G^2$, is the graph with the set of vertices of $G$ in which there is an edge between two vertices $u$ and $v$ if the distance between $u$ and $v$ in $G$ is at most 2. The previous theorem is no longer true for the square of a tree as indicated by the next theorem. Let $K_{1,3}$ be the tree with one internal vertex and three leaves. Let $S(K_{1,3})$ be the subdivision of $K_{1,3}$ formed by inserting a vertex of degree two on each edge of $K_{1,3}$. A graph $G' = (V', E')$ is called a subgraph of a graph $G = (V, E)$ if $V' \subseteq V$, $E' \subseteq E$ and $V'$ contains all the endpoints of the edges in $E'$. Regarding the characterisation of trees with Hamiltonian square, Harary and Schwenk [HS71] proved that:

**Theorem 2** *Let $T$ be a tree with at least 3 vertices. $T^2$ is a Hamiltonian graph if and only if $T$ does not contain $S(K_{1,3})$ as a subgraph.*

In fact for our work, a priori, we only need a Hamiltonian path. In [RR11], it is proved that the square of a tree $T$ has a Hamiltonian path if and only if $T$ is a horsetail. The definition of a horsetail is rather technical thus we do not provide it; in our context the important fact is that the family of trees which are not horsetails is infinite. Finally, it is not possible to obtain an enumeration of vertices such that the distance between any two vertices having two consecutive numbers is at most 2.

# References


[ABC11] P. S. Almeida, C. Baquero, and A. Cunha. Fast distributed computation of distances in networks. *CoRR*, abs/1111.6087, 2011.

[CK69] G. Chartrand and S.F. Kapoor. The cube of every connected graph is 1-hamiltonian. *J. Res. Nat. Bur. Standards Sect B.*, 73B:47–48, 1969.

[FHW12] S. Frischknecht, S. Holzer, and R. Wattenhofer. Networks cannot compute their diameter in sublinear time. In *SODA*, pages 1150–1162, 2012.

[Gav14] C. Gavoille. Private communication. 2014.

[HS71] F. Harary and A. Schwenk. Trees with hamiltonian square. *Mathematika*, 18:138–140, 1971.

[HW12] S. Holzer and R. Wattenhofer. Optimal distributed all pairs shortest paths and applications. In *PODC*, pages 355–364, 2012.

[MRZ16] Y. Métivier, J. M. Robson, and A. Zemmari. A distributed enumeration algorithm and applications to all pairs shortest paths, diameter.. *Inf. Comput.*, 247:141–151, 2016.

[PRT12] D. Peleg, L. Roditty, and E. Tal. Distributed algorithms for network diameter and girth. In *ICALP (2)*, pages 660–672, 2012.

[RR11] J. Radoszewski and W. Rytter. Hamiltonian paths in the square of a tree. In *ISAAC*, pages 90–99, 2011.

[Sek71] M. Sekanina. On an algorithm for ordering of graphs. *Canad. Math. Bull.*, 14(2):221–224, 1971.