1 authors

- Olivier Baudon, Assistant Professor, Labri, UMR 5800, Université Bordeaux1, Talence, France

  baudon@labri.u-bordeaux.fr

  Olivier Baudon, born in 1962, graduated from the Institute of Applied Mathematics of the University of Angers (France) and obtained his PhD at the University Joseph Fourier (Grenoble, France) in 1989. Since 1990, he is assistant professor at the University Bordeaux I. Baudon's main topic concerns graphs, interconnection networks and computer tools for graph theory. He is working with the company Mondeca since 1999 on graph clustering algorithms applied to graphs issue from Topic Maps.

- Pascal Auillans, engeneer R&D, Mondeca, Montrouge, France.

  pascal.auillans@mondeca.com

  Pascal Auillans, born in 1975, graduated from the University of Bordeaux I (France) in computer sciences. He started a PhD at the University of Bordeaux I Computer science departement (LaBRI) in 2000 with Olivier Baudon and Andre Raspaud in the field of graph theory. At the same time he is working for Mondeca, a firm involved in Topic Maps software developement. His PhD project deals with graph theory concepts (such as clustering) applied to semantic graphs.

2 Abstract

In our presentation we will show that topic maps can be considered, by a mathematical point of view, as a hypergraph. We will make an overview of the different existing clustering techniques and give some results on graphs representing topic maps. In particular, we will insist on techniques issued from the Interconnection Network topic. In this case, researchers are mainly interested in making a clustering with a minimum number of clusters, so that each cluster has either a bounded diameter or a bounded radius. Using the radius has another interest in that it naturally exhibits a particular node (the center) which can be used to be the representative element of its cluster.

3 Graph Theory and Topic maps

In this part we will introduce all the different terms and results of graph theory, which we will use in the rest of the document. For more information on concepts and standard algorithms of graph theory, we propose the following books [Ber83], [GM79], [CGH96].

3.1 Elements of graph theory

3.1.1 Generality

A **graph** G is a couple (V, E), where V is the set of **vertices** and E the set of **edges**.

An edge e is a pair of vertices $\{x, y\}$. We usually say that x and y are **incident with** e, or that they are the **endpoints** of e. x and y are said to be **adjacent** or **neighbors**.

In this paper we will consider that all the graphs are simple, meaning there exists neither multiple edges nor self-loops, ie each pair $\{x, y\}$ in E is unique and with $x \neq y$.

The **order** of a graph, that we will note n, is its number of vertices, and its **size**, that we usually note m, is its number of edges.

The **degree** of a vertex is the number of edges that are incident with it.

Example

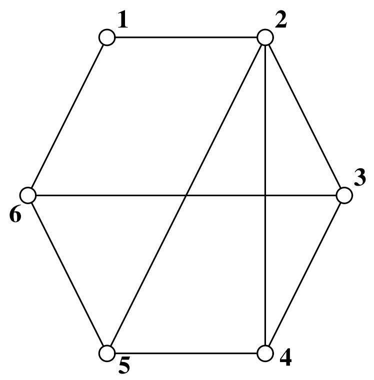

Figure 1: Graph simple

The graph in figure 1 is of order 6, and size 8. The degree of the vertex labeled 1 is 2, its neighbors are the vertices 2 and 6 and its incident edges are $\{1, 2\}$ and $\{1, 6\}$.

3.1.2 Subgraph

A **partial** graph of $G = (V, E)$ is a graph $G[F] = (V, F)$ where F is a subset of E.

An **induced subgraph** of $G = (V, E)$ is a graph $G(X) = (X, F)$ where X is a subset of V and F is the subset of E such that for every vertex couple of X, there exists an edge between them in $G(X)$, if and only if there exists one between them in G.

Let $G = (V, E)$ be a graph. We will call a **path** of G a sequence $P = x_1e_1x_2 \ldots x_{k-1}e_{k-1}x_k$ with $\forall i\ x_i \in V$ and $e_i \in E$ and $e_i = \{x_i, x_{i+1}\}$. A path is said to be **simple** if each vertex appears only once.

A **cycle** $C = x_1 \ldots e_{k-1}x_k$ is a path where endpoints are the same vertex ($x_1 = x_k$). A circuit is said to be **simple** if each vertices x_i, $i \in [1 \ldots k-1]$, appears only once.

A **connected component** of a graph G is a set in which each pair of vertices is connected by a path. A graph G is said to be **connected** if there is just one connected component.

Examples

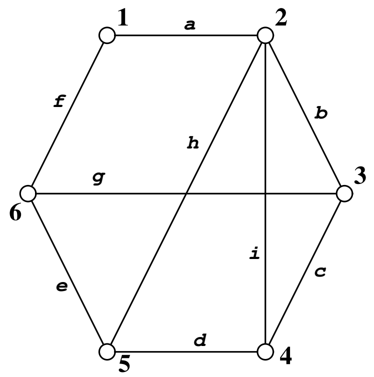

Figure 2: Graph G

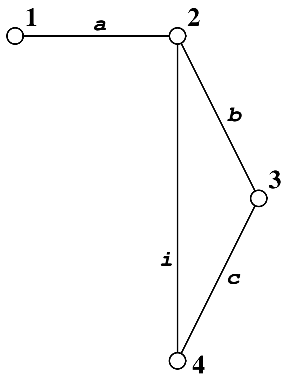

Figure 3: Induced subgraph $G(\{1,2,3,4\})$

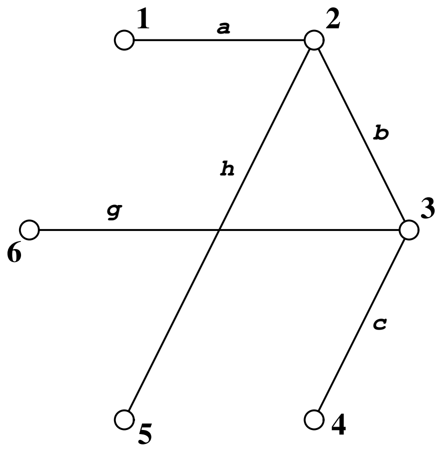

Figure 4: Partial subgraph of G

The sequence 1,a,2,b,3,c,4 is a path of length 3 of the graph in figure 2. The subgraph induced by the vertices $\{1,3,4\}$ is not connected. The graph in figure 4 is a partial graph of the graph in figure 2.

3.1.3 Tree

A **Tree** is a connected graph without cycle. The size of an order n tree is then $n-1$. The vertices of degree 1 are called the **leaves**.

A **spaning tree** of a graph G is a partial graph of G which is a tree.

Examples

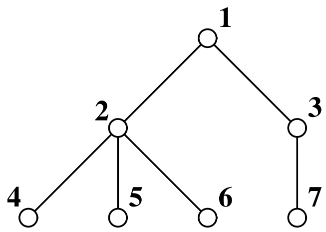

Figure 5: Tree

The tree in figure 4 is a spanning tree of the graph in figure 2.

3.1.4 Distances

The **length** of a path or a cycle is the number of edges it contains. The **distance** between two vertices x, y in a graph G, noted $d(x, y)$, is the minimum length of all the path connecting them. The **eccentricity** of a vertex is the maximum of all the distances of that vertex : $ecc(v) = max_{w \in V} d(v, w)$. If the graph is not connected then the eccentricity of any vertex is infinite.

The **diameter** of a graph is the maximum of all the vertices eccentricity : $D(G) = \max_{v,w \in V} d(v, w)$.

Let $G = (V, E)$ be a simple undirected graph. A **k-dominating** set of G is a subset S of V, such that each vertex of V is at most at distance k from a vertex of S. For $k = 1$, we will use the term **dominating** set.

Examples

The graph in figure 2 is of diameter 2, every vertices having an eccentricity of 2. The distance 1 neighborhood of the vertex 1 are the vertices 2 and 6 and its distance 2 neighborhood are the vertices 3, 4, 5.

The set $\{1, 2\}$ is a dominating set of the graph in figure 2. As this graph is of diameter 2 every set of vertices of size at least one is a 2-dominating set.

3.1.5 Graph clustering

A **clustering** of a graph $G = (V, E)$ is a set $\mathcal{S} = \{S_1 \ldots S_k\}$ of **clusters**, which are subsets of V, such that their union is equal to V. If every pair of sets of S are disjoint then S is called a **partition** and we will use the term **graph partitioning**.

3.1.6 Hypergraphs

Hypergraphs are a generalization of graph concept, where an edge is incident with an unspecified number of vertices. In that case we will use the term **hyperedges**. The representing graph of a hypergraph, $H = (V_H, G_H)$ is a graph G where the set of vertices is the union of the set of vertices and the edges of the hypergraph, and there exists an edge between two vertices if and only if one of the vertices is an hyperedge incident with the other vertex : $G = (V_H \bigcup E_H, \{\{v, e\}, e \in E_H, v \in e\})$.

An important property of representative graph of hypergraph is that the vertices can be decomposed into two sets such that there exists no edges between any couple of vertices belonging to the same set. This type of graphs are therefore named **bipartite** graphs.

Example

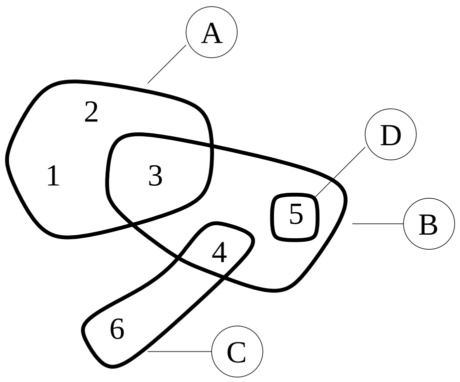

Figure 6: Hypergraph H

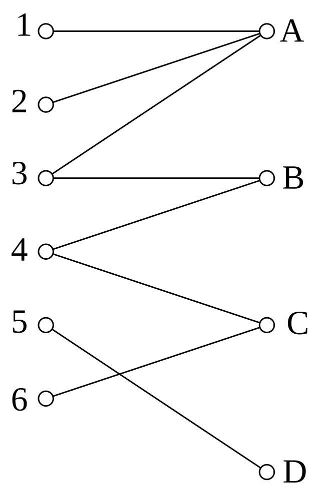

Figure 7: Representing graph of H

3.2 Notions of complexity

3.2.1 Definitions

A **decision problem** is a problem for which the answer is "yes-no". A lot of optimization problems can be reduced to a sequence of decision problems. For example, looking for the biggest stable of a graph (set of vertices pairwise not adjacent) can be reduced to a sequence of at most $n-1$ problems "Does a stable of cardinality k in G exist ?", for k between 1 and n.

A **NP problem** (non-deterministic polynomial) is a decision problem for which the yes answer can be verified in polynomial time. That means the number of elementary instructions (comparisons, additions, products ...) is a polynome according to the size of input data. For a graph, the usual size of input data is the sum of the number of edges and vertices : $m+n$.

For example, proving there exist a stable of size k in a graph can be done easily by checking that the possible solution is of size k and vertices are not pairwise adjacent.

A **NP-complete problem** is a NP problem for which the existence of a polynomial algorithm implies that there exists a polynomial algorithm for every other NP problem. The existence (or non existence) of such an algorithm is unknown for the moment.

3.2.2 Main results on graph

Most of the main graph problems are NP-complete : the existence of a stable or a clique (set of vertices all pairwise adjacent) of a given size. The problem of the existence of a k-dominating set (k constant) of size at most k' is NP-complete.

3.3 Graph Coding

According to the needs and the importance given to the edges, we can represent the graph either by the adjacency, incidence relation or both of them. Therefore there exist two matrical coding of a graph.

3.3.1 Adjacency Matrix

this matrix is a $n \times n$ integer matrix, where the integer of the cell at the intersection of line i and row j is equal to the number of edges from the vertex i to the vertex j.

3.3.2 Incidence Matrix

For a undirected graph, This matrix is a boolean ($\{0, 1\}$) matrix of size $n \times m$, where the value of the cell at the intersection of line i and row j equal 1 if the edge j is incident with the vertex i, 0 otherwise.

The matrix

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

is the adjacency matrix of the graph in figure 2.

The matrix

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

is the incidence matrix of the same graph. The rows are sorted following the lexicographic order of the edges.

3.3.3 Lists

The inconvenience of the matrical coding of the graph is the difficulty of going through all of the neighbors or edges incident with a vertex. That is why we frequently represent a graph under the form of a n list of neighborhood arrays. Finally, if it is necessary to be able to answer quickly to the double question "What are the neighbors of vertex i?" and "Are the vertices i and j neighbors?" We should use both representations simultaneously (or a good compromise such as the hash tables).

3.4 From Topic Maps to graphs

In this part we will show that a topic map, restricted to the part defining the index management, can be considered as a hypergraph and therefore, due to the definitions already given, can be considered as its representative graph. We won't make a precise presentation of the topic map norm, but instead try to give an overview of its concepts.

The ISO Topic Maps standard (ISO/IEC 13250) defines the **topics** as objects allowing the gathering of groups of information because of their contribution to the same idea. One can, for example, regroup the photography and bibliography of the classical music composer, Mozart, as documents contributing to the definition of the topic "Mozart". The norm allows as well the association of a group of topics in order to define an interaction between them. Therefore, if we take the previous example of Mozart, one can associate the topic "Mozart" with one of his works, saying the "Magic Flute" but also with the topic "Mr X" who has performed it.

We can already see that the norm defines its model as a hypergraph in which the topics are the nodes and the association between two or more topics are hyperedges.

To give more meaning to topic maps, the norm allows users to define semantics on the different kind of objects, which we will present later on. To put semantics on hyperedges, the norm explicitly defined the hypergraph by its representative graph. We saw in the section presenting the graph theory that a hypergraph can be represented by a graph whose set of vertices is the union of the set of vertices and the set of hyperedges of the hypergraph, and the set of edges is defined by the relation of incidence between vertices and hyperedges of the hypergraph. In the norm these objects are called: **associations** for the vertices representing hyperedges and **members** for the edges of the bipartite graph.

At the moment our work modelizes just that part of the topic map norm, but we will continue a little further to show that graph theory can be very useful for modelisation and solving topic map problems.

As stated previously, the topic map norm allows the definition of semantics on the different objects. There exist, therefore, two features to define semantics :

The first is the definition of the objects' types (**InstanceOf** property for topics and **RoleSpec** property for edges). So one could say that the topic "Mozart" is of "human" type but also of "composer" type, the association between "Mozart" and his work is a "To Compose" type, the member linking the topic "Mozart" to the association is of "composer" type, and the member linking the association to "The Magic Flute" topic is of "musical composition" type.

The second feature, **scopes**, allows the definition of a set of types sharing the same semantics. Therefore, all the types previously defined fall under the scope of "Classical Music". It must be noted that objects (topics, associations and members) can be valid under several scopes.

This mechanism has already been modelized in graph theory by the definition of **labeled graph**. A labeled graph is a couple (G, λ) where G is a graph and λ ($\lambda : V \bigcup E \longrightarrow L$) is a function which associates to every vertex and every edge of the graph G an element of a set of labels L. In the case of topic maps the label function associates to each vertex and each edge a subset of the label set ($\lambda : V \bigcup E \longrightarrow \mathcal{P}(L)$, where $\mathcal{P}(L)$ is the set of L subsets).

4 Algorithms and results

4.1 Bibliography

This problem of graph clustering has been studied in several computer science and mathematical fields. Therefore we looked at books and articles dealing with classification [Gor99], load balancing in distributed programming environments [Pel94], interconnection networks [Pel00, EGP98], domination and k-domination in graphs [AK95, DKS97, Hed98, Hen98], and the visualization of large graphs [JE99, BMZ99]. To improve the efficiency of the algorithms we also looked at random approaches [MR95].

Most of algorithms proposed are quadratics, so they are not useful for the kind of graphs we are using. However, we get inspiration from some of them to form linear time algorithms.

4.2 Problem formalization

One can consider two approaches of the problem. The first one consists of setting up the number of clusters to be obtained, saying k, to choose a criteria, then to find a clustering of the graph's vertices in k subsets which optimize the criteria.

The most interesting example of this approach is that which consists of clustering the graph in k sets balanced in terms of the number of vertices and trying to minimize the number of intercluster edges. Such an approach has already been studied on very large graphs, but only for $k = 2$ or in a recursive way $k = 2^p$. Effective implementations have been proposed, for example in the Scotch project [Pel94].

The second approach consists of giving a criteria then clustering the graph in a way to obtain an optimal number of subsets. This is the approach that we chose, searching to cluster the graph with subsets of a given radius or diameter, trying to minimize the number of subsets, which we will call balls.

The very large graphs clustering with a given ball diameter has been studied in [JE99] This article proposes heuristics to estimate the calculation of graph's diameter. The complexity of the standard algorithm which calculates the diameter of a graph is quadratic ($\mathcal{O}(n(n+m))$), and it is therefore difficult in practice to use on large graphs. The article also shows, on example of the order of several thousand vertices graphs, that the use of an exact value for the calculation of the diameter gives much better results than that of the value approached. We are, at the moment, still developing the algorithm presented in this article, by replacing the approximate calculation of the diameter by an exact calculation using an algorithm of Mark Lesk [Les84] of the same theoretical complexity as the standard algorithm, but faster in average.

However, the approach which seems most interesting to us is that of graph clustering with balls of given radii. The use of the radius rather than the diameter has the negative effect of giving less homogenous induced subgraph (a maximum distance is guaranteed with relation to a single vertex and not between every pair of vertices), but possesses the advantage of distinguishing naturally a vertex, the center of the ball.

The problem of graph clustering in a minimum number of induced subgraphs of given radii k is, as we said in paragraph dealing with distances, known as the k-domination. It is a NP-complete problem and therefore the existence of an exact and efficient algorithm is strongly improbable. This problem is largely treated for diverse purposes, in [Pel00, Hed98, Hen98, Cha82].

If the subject was largely treated from a theoretical point of view, few heuristics were proposed. For $k = 1$, Johnson [Joh74] proposes heuristics natural enough which consist of construction at each iteration a ball centered on the vertex having the most neighbors not already forming a part of a ball already constructed. It gives as well a limit for the relationship between the produced solution and the optimal solution. We are inspired by his heuristics in adapting them to the general problem for any k.

In [EGP98] the authors propose heuristics which consist of for every vertex v to calculate $nb(v)$ the number of balls of radius k to which it belongs, then to chose the center of the balls in decreasing order with relation to nb, eliminating all vertices belonging to the ball defined by the chosen center. The first step being quadratic, we tried a more efficient version in only doing the first step on a constant number of vertices, prepared to reiterate the second step several times.

Finally, in [Hen98] we find heuristics based on the decomposition of a spaning tree, which we tested as is.

4.3 Implemented Algorithm Presentation

As explained in the previous paragraph, we implemented three algorithms of graph clustering with given ball radii, the first inspired by [Joh74], the second by [EGP98], and the third directly issued from [Hen98]. We implemented these algorithms using a data structure coding the representative graph with vertices lists of neighborhoods. We chose this coding because we knew that the vertices' degrees were limited by a constant c.

4.3.1 First algorithm

The first algorithm implemented consists of ball construction starting with the highest weight vertex, the weight being a function depending on the degree of the vertex, and on the other hand, on the sum of the distances to the centers already chosen. It is a sort of modification of Johnson's algorithm in which the weight functions are the number of neighbors not already included in a ball.

4.3.2 Second algorithm

The second algorithm consists of constructing balls starting with centers chosen randomly the first time, until every vertex appears in at least one ball, then to reiterate with choosing as centers the vertices not already included in a ball and belonging to a maximum of balls of the previous stage.

4.3.3 Third algorithm

The third algorithm extracts from the graph spanning tree, and calculates the centers using only the edges of that tree. For that, we calculate at each step a path of length equal to the diameter of the tree and we choose as the center the vertex at distance k of one of the path's endpoints. Then we delete the subtree rooted at the center and we reiterate on the remaining tree.

4.4 results

4.4.1 Test data

We compared the graph clustering algorithm on three graphs:

- a grid 1000x1000, as an example of a regular graph. The grid $n \times m$ is the graph in which the vertices are couples (i,j), $1 \leq i \leq n$, $1 \leq j \leq m$ and where two vertices are adjacent, if and only if, they share a common coordinate and the other one differs by only one unit. Every vertex is of degree 4 except the one on the first and last lines and the first and last rows whose degrees are 3 or 2 for the corners. The number of edges is therefore $2nm - n - m$ The grid 1000x300 possesses a 300 000 vertices and almost 600 000 edges (598 700).
- a random graph with 300 000 vertices and 600 000 edges, the edges having equal probability of being taken.
- a graph issued from a topic map managing classical music information. This topic map is of order 219116 (number of topics plus number of associations) and of size 1496592 (number of members).

4.4.2 Measurement

To compare the different algorithms we watched the following properties :

- The time of execution (in seconde) which is important on large graphs.
- The number of balls obtain by th algorithm.
- the minimum, maximum and average number of balls a vertex belongs to, which gives an idea of the balls' overlapping.

4.4.3 Results

- Grid graph 1000 x 300 with radius 50.

	Time	Nballs	Min	Max	Average
First algorithm	2	74	1	4	1
Second algorithm	40	240	1	4	3
Third algorithm	1046	5144	7	106	80

- Random graph RANDOM GRAPH n=300000, m=600000 with radius 5.

	Time	Nballs	Min	Max	Average
First algorithm	1350	10961	1	51	9
Second algorithm	816	12217	1	47	8
Third algorithm	1639	4676	7	197	101

- Topic map graph TOPIC MAP n=219116, m=1496592 with radius 5.

	Time	Nballs	Min	Max	Average
First algorithm	55	373	1	137	20
Second algorithm	71	408	1	192	33
Third algorithm	1123	537	1	484	185

5 Furtherworks and conclusion

We noticed that the implemented algorithms usually generate a big ball in the center of the graph and lot of small ones in the periphery of it. We think that we can obtain a better solution by recursively clustering the subgraph induced by the big ball.

There remain, as well, certain ways to explore, such as graphic data processing. In order to do that it will be necessary to have a factorial analysis tool allowing the representation of a graph, in a simplified way, by a set of points. Once this is done, we may study several algorithms of set of points clustering in the space, issued from pictures analysis.

References

[AK95] C. J. Alpert and A. B. Kahng. Recent developments in netlist partitioning: A survey. *Integration: the VLSI Journal*, 19(1-2):1–81, 1995. `http://nexus6.cs.ucla.edu/~cheese/survey.html`.

[Ber83] C. Berge. *Graphes.* Gauthier-Villard, 1983.

[BMZ99] Vladimir Batagelj, Andrej Mrvar, and Matjaz Zaversnik. Partitioning approach to visualization of large graphs. *Lectures Notes in Computer Sciences*, 1731:90–97, 1999.

[CGH96] I. Charon, A. Germa, and O. Hudry. *Méthodes d'optimisation combinatoire*. Masson, 1996.

[Cha82] G.J. Chang. *k-domination and graph covering problems*. PhD thesis, School of OR and IE, Cornell University, 1982.

[DKS97] Jitender S. Deogun, Dieter Kratsch, and George Steiner. An approximation algorithm for clustering graphs with dominating diametral path. *Information Processing Letters*, 61:121–127, 1997.

[EGP98] Tamar Eilam, Cyril Gavoille, and David Peleg. Compact routing schemes with low stretch factor. In 17^{th} *Annual ACM Symposium on Principles of Distributed Computing (PODC)*, pages 11–20. ACM PRESS, August 1998.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and Intractability, A guide to the theory of NP-Completeness*. W.H. Freeman and Company, 1979.

[GM79] M. Gondran and M. Minoux. *Graphes et Algorithmes*. Eyrolles, 1979.

[Gor99] A.D. Gordon. *Classification*. Chapman & HALL/CRC, 1999.

[Hed98] Stephen T. Hedetniemi. Distance domination in graphs. In Teresa W. Haynes, S.T. Hedetniemi, and Peter J. Slater, editors, *Complexity results*, pages 233–269. Pure and Applied Mathematics - Dekker, 1998.

[Hen98] Michael A. Henning. Distance domination in graphs. In Teresa W. Haynes, S.T. Hedetniemi, and Peter J. Slater, editors, *Domination in graphs, advanced topics*, pages 321–349. Pure and Applied Mathematics - Dekker, 1998.

[JE99] F. Brandenburg J. Edachery, A. Sen. Graph clustering using distance-k cliques. *Lecture Notes in Computer Sciences*, 1731:98–106, 1999.

[Joh74] David S. Johnson. Approximation algorithms for combinatorial problems. *Journ. of Computer and System Sciences*, 9:256–278, 1974.

[Les84] M. Lesk. Diametre de graphes et qualite du service d'un reseau de donnees. *R.A.I.R.O.*, 18:247–261, 1984.

[MB83] David W. Matula and Leland L. Beck. Smallest-last ordering and clustering and graph coloring algorithms. *Journal of the Association for Computing Machinery*, 30:417–427, 1983.

[MR95] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[Pel94] Francois Pellegrini. Static mapping by dual recursive bipartitioning of process and architecture graphs. *IEEE*, pages 486–493, 1994.

[Pel00] D. Peleg. *Distributed Computing, a locality-sensitive approach.* SIAM Monographs on Discrete Mathematics and Applications, 2000.

[RS97] Tom Roxborough and Arunabha Sen. Graph clustering using multiway ratio cut. *Proceedings of Symposium on Graph Drawing GD'97, Berlin*, 30:291–296, 1997.