A Heuristic Method for Balanced Graph Partitioning: An Application for the Demarcation of Preventive Police Patrol Areas

Thiago Assunção and Vasco Furtado

University of Fortaleza (UNIFOR) – Graduate Program in Applied Informatics (MIA)
Av. Washington Soares, 1321 - Bloco M Sala 11 - 60.811-905 - Fortaleza – Brazil
thiagoaa_unifor@yahoo.com.br, vasco@unifor.br

Abstract. In this paper we describe a heuristic method, based on graph-partitioning algorithms, with the purpose of improving the demarcation of areas for police patrolling. This demarcation seeks to homogenize the number of crimes among the patrol regions. If the map of a particular region is taken as a graph, we can say that the problem faced by police forces (typically preventive police) is similar to the problem of finding balanced connected q partitions of graphs (BCPq). Since this is a problem belonging to the NP-Hard class, approximate algorithms are the most suitable for solving such a problem. The method described in this article obtains results nearest those considered optimal for the more general case of BCPq, for $q \geq 2$.

Keywords: approximate algorithm, graphs, heuristic method, balanced connected partition.

1 Introduction

In order to contain criminal activity, it is necessary to provide the organizations that act repressively or preventively with tools for the efficient performance of their actions. Preventive Police Force has the task of carrying out ostensible law enforcement. It is assumed, therefore, that this group's presence acts as a dissuasive force, thus preventing the occurrence of crimes.

The division of patrol areas is one of the basic activities of planning for the police patrol. One example of a project in which the issue of division of patrol areas causes a major impact is being developed in Brazil: the "Block Patrol" project.

This project intends to divide the city of Fortaleza (more than 2 million inhabitants) into patrol areas that would be covered by teams of police officers. This territorial division, in principle, is being done by means of an *ad hoc* criterion: fixed areas measuring 3 km² each. The search for efficient methods to carry out this division, based on criteria such as the crime rate of the areas, is the object of this work. It is easy to understand why the technicians of the Block Patrol chose such a simple method. The division of the city into variable areas considering different optimization criteria is not a trivial task. The difficulty of such an activity is evident in this scientific paper. For this work, we decided to model the problem of the optimized division

H. Geffner et al. (Eds.): IBERAMIA 2008, LNAI 5290, pp. 62–72, 2008.

of patrol areas as a problem of balanced connected partition of graphs in q partitions (BCPq). It can be perceived, by doing so, that the problem in question is presented as NP-Hard [4].

The optimization in demarcating the regions intends to achieve the maximum homogenization possible in the distribution of crimes among the areas. Doing so, the distribution of police patrol teams can be associated to a demarcated area (i.e. each area is associated to one team).

The first contribution of this article is, therefore, to supply a formal setting for the treatment of the problem and, for this reason, is inspired by works already developed by the scientific community for seeking an approximated and viable solution to the problem. In addition to the formal modeling of the problem, this paper proposes a heuristic method for efficient resolution (in comparative terms with the state of the art) for the problem of balanced graph partition. Finally, the application of such a method to partition the city of Fortaleza into patrol areas is the practical contribution provided herein.

The remainder of the article is structured in the following way: Initially, we present a short analysis of the problem of balanced connected partition of graphs and of the main algorithms related to this problem. Then, we show how modeling the problem of dividing the patrol areas was done to a graph partition problem, and we describe this article's objective heuristic method. In the fourth section, we evaluate the heuristic method and compare it with the algorithm proposed by Chlebíková [3], which is considered the algorithm that produces the best results for the specific case of BCPq, for q = 2. Finally, the last section brings final considerations and perspectives for future works.

2 Balanced Connected Partitions of Graphs

BCPq (Balanced Connected Partition of graphs for q partitions, where $q \geq 2$) can be defined in the following way:

Given an integer number q and a connected graph G = (V, E), where its vertices possess weights, find a q-partition of G such that each subgraph associated to each partition is connected and the weight of the lightest one is the highest possible, i.e., the distribution of the weights among the subgraphs should be the most homogeneous possible.

This problem belongs to the class of the problems of graph partitioning and is classified as an NP-Hard problem, even for q-connected graphs [4]. Therefore, the most appropriate algorithms for solving this problem are approximation algorithms, which tend to find solutions nearest those considered optimal.

Several algorithms exist for the more general problem of graph partitioning. Among those, we can cite the following: the Cartesian Nested Dissection method [6], the Kernighan-Lin (KL) algorithm [1], the Fiduccia-Mattheyses (FM) algorithm [2], the multilevel methods [8] and [10], recursive coordinate bisection [11]. It is worthy of mention that the KL algorithm is one of the most widely referenced for solving this problem, since it produces a good local partitioning regarding the number of cut edges. For further information, we recommend reading [7].

There are few algorithms geared specifically toward the BCPq problem. For the particular case where q = 2, the algorithm proposed by Chlebíková [3] is the one that produces a partition nearest to what is considered optimal, i.e., the partitions are connected and possess weights whose values are approximations of the ideal result (each partition possesses half the weight of the graph). For the case where $q \geq 3$, BCPq has not been widely investigated, in as much as—up to until 2007—no algorithms existed that resolved any problem for $q \geq 3$ within the set of integer numbers [5]. Very recently, Pinheiro in [5] constructed several heuristics, based on spanning trees, which present good results for the general case of the problem. For further information regarding the BCPq problem and algorithms of approximation geared toward this problem, we recommend reading [9].

3 Description of the Solution

For solving the problem of optimization of patrol area demarcation, we use the graph data structure for the computer representation of geographical maps [12]. In this paper, we used the map that represents the district of *Aldeota*, located in the city of Fortaleza, Brazil. In order to explain this modeling, let's take this map as shown in Fig. 1.

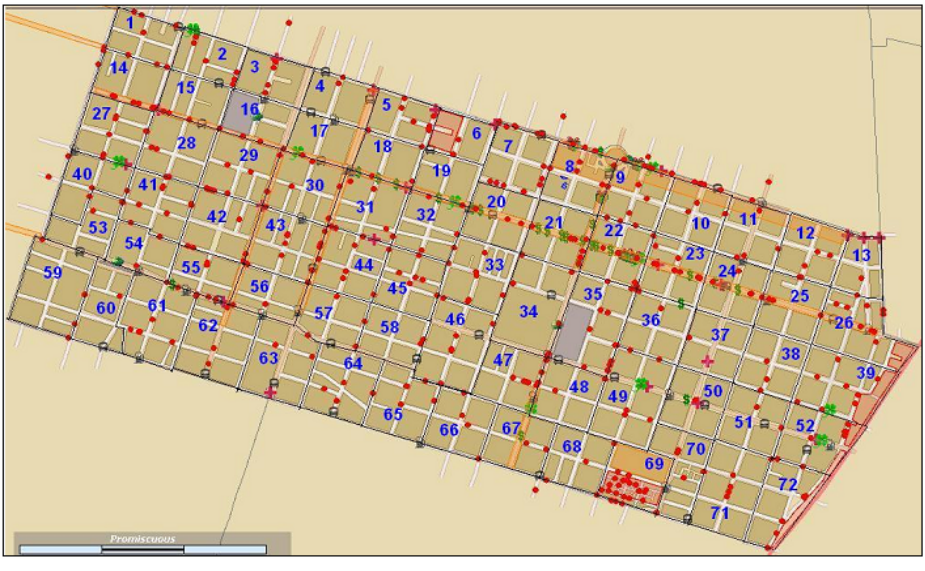

Fig. 1. Map of the Aldeota district

The map was divided into 72 parts called "blocks". One block is a sub-region of a district whose area includes about four city squares [area of buildings/lots surrounded by four streets], i.e., 0.04 km². Each block was numbered with a value from 1 to 72. The points in black indicate the occurrence of a crime at a particular location in the district.

The choice of small areas to delimit the blocks was arbitrary. However, this demarcation could be of another granularity; but—since specialists on the subject believe that this granularity is reasonable, because they would not constitute *per se* just one useful area of policing—we adopted this demarcation. Moreover, if we chose larger areas, the heuristic method would have to divide these blocks into sub-blocks in order to guarantee a greater homogeneity of crimes among the patrol areas, thus diminishing the performance of such a method.

Each block of this map was represented by a vertex on the graph. Additionally, each vertex possessed a weight, denoting the number of crimes within the region of the block represented by the vertex. The edges of the graph represent the idea of neighborhood between the blocks, i.e., two neighboring blocks on the map will be represented by an edge (u, v), where u and v are graph vertices associated to the blocks. For example, blocks 1 and 2 are neighbors on the map; therefore they will form an edge (v1, v2) on the graph. Observe that diagonal blocks are not considered neighbors. For example, blocks 1 and 15 are not neighbors. Fig. 2 illustrates the graph that results from the map in Fig. 1. The numbers in bold represent the weights of the vertices, i.e., the number of crimes that occurred within the limits of a particular block associated to the vertex.

Now that we already have the graph representing the geographical map of the Aldeota district, all we have to do is apply the method of balanced connected partitioning in order to produce the regions that will be patrolled by the police teams.

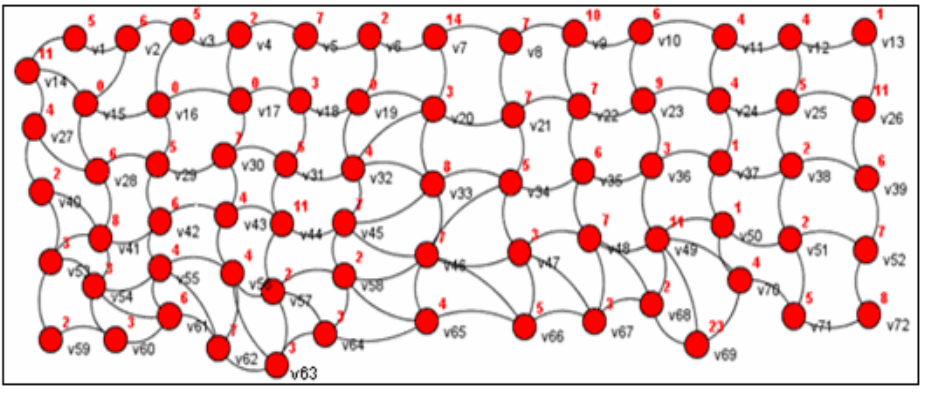

Fig. 2. Graph representing the map in Fig. 1

3.1 Description of the Heuristic Method

The method is comprised of two phases: initial partitioning and partition refinement. In the initial partitioning phase, we can use any method that partition a graph in two connected sub-graphs. Particularly, for the application in the demarcation of patrol areas we have chosen to use the Cartesian Parallel Nested Dissection algorithm [6]. The rationale behind this choice is two-fold. Such an algorithm assumes that the vertices possess geographical coordinates which it is the case in the patrol area domain where coordinate values are associated to each vertex according to the position of the

block that this vertex represents on the map. Furthermore, the Cartesian Nested Dissection algorithm is simple to implement, efficient having low complexity.

In the partition refinement phase, we use a heuristic algorithm that we constructed based on the strategies of the KL algorithm [1]. This heuristic algorithm swaps the vertices between the partitions with the intent of maximizing the homogeneity of the weights between them. Such a refinement algorithm is this article's main contribution toward the BCPq problem.

3.2 Algorithm for Partition Refinement

The partition refinement algorithm is based on concepts adopted by the Kernighan-Lin method [1]. The KL algorithm executes swaps among partitions with the aim of minimizing the number of edges that will be cut. Similarly, our algorithm executes such swaps with the intent of better distributing the weights among the partitions. Both receive a graph and the partitions thereof as input and return the refined partitions according to their refinement criteria. Before detailing the algorithm, it is important to define some of the concepts:

- Cut vertex or articulation point is a vertex of a graph such that its removal causes an increase in the number of connected components. If the graph was connected before the removal of the vertex, it will be disconnected afterwards;
- A vertex of a partition is said to be admissible if there is an edge linking it to a vertex in another partition. Moreover, this vertex can't be a cut vertex;
- Weight of a partition, represented by the notation *w(P)*, denotes the total value of the weights of the vertices belonging to partition *P*;
- Two partitions are considered adjacent if there is an edge that goes from a vertex in one these partitions to a vertex in the other one.

The algorithm possesses the following characteristics: it carries out the swap, where only one vertex is swapped at a time, between two adjacent partitions at a time. The vertices are swapped or moved from one partition whose weight is greater to another partition whose weight is less. Vertices that are swept must be admissible, which guarantees the connectivity of the partitions after the swap, and they possess the lowest values of preference among the possible vertices to be swapped. These preference values must be lower than a local homogeneity index for the vertices to be transferred.

The local homogeneity index α indicates how far away the weights of both partitions are from a weight that is considered ideal. This ideal weight is the average among the weights of the partitions. The index is calculated according to the equation defined in (1), where:

$$\alpha = \left| w\left(P_1\right) - w\left(P_2\right) \right| \tag{1}$$

- $w(P_1)$ *denotes the weight of partition* P_1*;*
- $w(P_2)$ *denotes the weight of partition* P_2*;*

The preference criterion for a vertex $\gamma(v)$ represents the value of α after swapping this vertex. Such value is calculated according to the equation defined in (2), where:

$$\gamma(v) = \left|\left(w_{\max}(P_1) - w_{vertex}(v)\right)\right| + \left|\left(w_{\min}(P_2) + w_{vertex}(v)\right)\right| \quad (2)$$

- $w_{\max}(P_1)$ *denotes the weight of the partition with highest weight, i.e., that to which vertex* v *belongs;*
- $w_{vertex}(v)$ *denotes the weight of vertex* v*;*
- $w_{\min}(P_2)$ *denotes the weight of the partition adjacent to* P_1 *with the lowest weight;*

To optimize the swapping process, the algorithm prefers swaps between the partition with the highest weight and the partition adjacent thereto with the lowest weight in a given iteration of the algorithm. Following this preference criterion, the weights of the most valued partitions are first diminished and, consequently, the weights of their respective smaller valued adjacent partitions are increased, improving the homogeneity among the partitions. The process finishes when there is no vertex that can be swapped among adjacent partitions, i.e., the solution cannot be more refined according to the algorithm's criteria.

The algorithm of the GLOBALREFINEMENT procedure, which carries out the process described previously. See it below.

```
GLOBALREFINEMENT (Partitions[])

1    For each partition do
2      Attribute an optimal weight value to the partition;
3    INITIALIZEADJACENCYPARTITIONS (Partitions);
4    Place partitions in decreasing order according to their weights;
5    indexCurrentPartition ← 0;
6    numberIterations ← 0;
7    currentPartition ← null;
8    While (indexCurrentPartition = Partitions.length) do
9      currentPartition ← Partitions[indexCurrentPartition];
10     Place partitions adjacent to currentPartition in increasing order according to
11     their weights;
12     For each partition adjacent to currentPartition do
13       numberIterations ← LOCALREFINEMENT (adjacentPartition, currentPartition);
14       If (numberIterations = 0) then
15         stop;
16     If (numberIterations = 0) then
17       indexCurrentPartition ← indexCurrentPartition + 1;
18     Else
19       indexCurrentPartition ← 0;
20       numberIterations ← 0;
21       UPDATEADJACENCYPARTITIONS (Partitions);
22       Place partitions in decreasing order according to their weights;
```

The procedure begins by attributing an optimal weight value to each partition (lines 1 and 2). This optimal value is the result of the division between the weight of the graph and the number of partitions. Then, the procedure initializes the relations of adjacency among the partitions invoking the INITIALIZEADJACENCYPARTITIONS procedure (line 3). Immediately thereafter, the partitions are placed in decreasing order according to their weights (line 4), and the variables: indexCurrentPartition, numberIterations and

currentPartition are initialized (lines 5, 6 and 7). The outermost loop repeats until all of the partitions have been tested for the swapping process (line 8). Within that loop, first the partition with the greatest weight is attributed to the currentPartition variable (line 9), and the partitions adjacent thereto are placed in increasing order according to their weights (line 10). Then, for each adjacent partition, the swapping process is executed by calling the LOCALREFINEMENT function (lines 12 and 13). If this method executes any swap, the innermost loop ends (lines 14 and 15). Otherwise, it continues until reaching the end of the list of adjacent partitions or until some swap is carried out. Right after this innermost loop, if no swap was made between the current partition with the highest weight and one of its adjacent partitions, the indexCurrentPartition variable is increased (line 17), and the outermost loop goes on to its next iteration. Otherwise, the indexCurrentPartition and numberIterations variables are again initialized (lines 19 and 20), the relations of adjacency among the partitions are updated (UPDATEADJACENCYPARTITIONS procedure) (line 21), the partitions are again placed in decreasing order according to their weights (line 22), and the outermost loop goes on to its next iteration.

4 Evaluation and Comparison with the Chlebíková Algorithm

In order to utilize an approximative method, we will use the approximation ratio or performance ratio (represented by the letter p) as a metric for evaluating the method described in the previous section. This measure informs how close the solution obtained by an algorithm is to the optimal solution.

Using equation (3), we can calculate the approximation ratio of an algorithm, where A(I) represents the value of a solution obtained by executing algorithm A for an instance I of the problem, and opt(I) represents the value of an optimal solution for instance I of the problem.

$$A(I) \leq p * opt(I) \tag{3}$$

Since—in our case—the partitioning method may produce partitions with weights above those considered optimal, the value of A(I) is calculated in the following manner: optimal weight for a partition (weight of the graph divided by the number of partitions) – the greatest difference in module between the weight of a partition and the optimal weight for a partition.

For example: suppose that the partitioning algorithm for a graph G with weight 30 produced three partitions with the following weights: 11, 11 and 8. The opt(I) value will be 10 (1/3 of the weight of the graph), and the A(I) value will be 8, i.e., (10 - |8 – 10|). Therefore, the approximation ratio (p) will be 0,8 (80%), i.e., the weights of the three partitions approximated the optimal weight by at least 80% for a partition of this graph.

4.1 Application and Evaluation of the Algorithm in the Graph of the Aldeota District

In order to verify the efficacy of the heuristic method that is the subject of this article, we applied it to the graph that represents the Aldeota district (Fig. 2) for a number q of partitions varying from 2 to 15. The results are presented in Table 1.

Observe that, for the most part, the results were good, with an approximation percentage greater than or equal to 80%, except for the case where the number of partitions is equal to 4, whose approximation percentage was 69.56%. This occurred due to the fact that the Initial Partitioning phase did not produce—for that case—partitions that better aided the Partition Refinement phase regarding adjacency among the partitions, in order to produce a solution with a higher approximation percentage. However, the greatest contribution of this article, i.e., the refinement algorithm that we constructed, applied in the Partition Refinement phase, showed that given an initial connected partitioning, one can produce good solutions for a number of partitions greater than or equal to two. Furthermore, for these tests, the partitioning method takes no more than 0.5s to produce the solutions.

It should be pointed out that we applied the method on a planar graph with coordinate values associated with its vertices. We were not able to affirm that the method described in this article produces solutions with the same percentage of approximation for all types of graphs. However, we believe that, given an initial connected partitioning, the heuristic algorithm (Partition Refinement phase) manages to produce connected partitions with good percentages of approximation.

Table 1. Results of the tests on the Aldeota graph

Number of Partitions	Percentage of Approximation
2	99.45%
3	99.18%
4	69.56%
5	97.26%
6	93.44%
7	86.53%
8	93.47%
9	87.50%
10	88.88%
11	90.90%
12	90%
13	82.14%
14	80.76%
15	87.50%

4.2 Comparison with the Chlebíková Algorithm

As already mentioned previously, the Chlebíková algorithm [3] is the one that produces a balanced connected partitioning closest to what is considered optimal for the BCP2 problem, where the input graph is 2-connected.

However, the algorithm proposed by Chlebíková, in certain situations, fails to produce better solutions because it always chooses the admissible vertex with the lowest weight among all admissible vertices.

Unlike Chlebíková's algorithm, the refinement algorithm that we constructed chooses admissible vertices that best balance the weights among the partitions in any given iteration. This difference makes the algorithm that we created produce solutions that are equal to or better than the solutions obtained by applying the algorithm proposed by Chlebíková.

We compared the two methods through several tests conducted by varying the number of vertices and the density (number of edges) of biconnected graphs. We have used the Jung electronic library [13], version 1.7.6, to randomly generate the graphs. The graph's vertex weighs are also randomly set up with values varying between one and the half of the graph size (total number of vertices).

We mean by initial partition the phase the returns two partitions in which one of them has the heaviest graph vertex and another has the rest of the vertices. This is the same strategy used in the initial phase of Chlebíková's algorithm. We have adopt it for comparison purposes since we keep the same conditions between our approach and Chlebíková`s one. Otherwise the results could be biased by the results obtained during the initial phase what would compromise the evaluation of the refinement phase which is our main goal. Moreover, the use of the Cartesian Dissection would require the association of coordinates for the vertices what would constraint the choice of graphs in the evaluation process. The test results are presented in Table 2. The first column and second columns show the graph edges and vertices number, respectively. For each test, ten graphs were used and we run the algorithm for each one of them. The third and fourth columns represent the average approximation ratio of our approach and Chlebíková `s, respectively. Finally, the fifth and sixth columns show, also for our approach and Chlebíková `s one, the average iteration number for finding a solution. Note thus that the average iteration number accounts the average of the number of times one vertex was transferred from one partition to another during the algorithm execution. According to these tests, the algorithm that we describe in

Table 2. Comparison between the Chlebíková algorithm and this article's method

Density	n	p Our Approach	p Chlebíková	Nb. Iteration Our Approach	Nb. Iteration Chlebíková
2n	10	0.95	0.89	2.9	4.5
	20	0.99	0.95	6.4	12
2.5n	10	0.98	0.94	3	4.7
	20	0.99	0.96	6.4	11.7
	30	0.99	0.98	9.4	18.1
	40	0.99	0.98	12.4	25.5
3n	10	0.95	0.92	2.9	4.8
	20	0.99	0.96	5.7	12.2
	30	0.99	0.97	9.3	18.6
	40	0.99	0.98	12.3	25.4
	50	0.99	0.98	15.1	33.1
	60	1	0.99	18.1	39.8
	70	1	0.99	21.7	46.3

section 3 produces solutions with an approximate ratio better than those proposed by Chlebíková. Note that these results were achieved in less iterations than Chlebíková`s.

The Chlebíková algorithm used in our tests was implemented by us in JAVA. In order to perform the tests presented in this section, we used an Intel Pentium 4 Processor with 512Mb of memory.

5 Final Considerations and Future Works

The intent of this article is to help preventive police forces. Such help consists of better delimiting the patrol areas and distributing the police teams among these areas. For such, we showed that the problem faced by the preventive police can be modeled as a problem of balanced connected partitioning of graphs (BCPq). Such a problem has not been widely investigated for its most general case, where the number of partitions is greater than 2.

We developed a heuristic method based on strategies pertaining to the foremost graph-partitioning algorithms. We applied it on a graph that represents the Aldeota district in the city of Fortaleza, Brazil. The results obtained from the tests on this graph showed that, in general, the approximation percentage was equal to or greater than 80% for a number of partitions varying from 2 to 15. Moreover, we compared this method to the approximation algorithm proposed by Chlebíková. Several tests were conducted, varying the number of vertices and of edges of the graphs. The results obtained showed that the algorithm proposed in this article possessed a degree of approximation equal to or higher than that obtained with the algorithm proposed by Chlebíková.

Based on this work, we show the feasibility, in the future, of delimiting patrol areas that obey certain limits of extension. In order for this to be possible, an area value would be associated with each vertex, and such value would represent the extension of the block associated therewith. Thus, the patrols would not be under-utilized, i.e., there would be no patrols in small regions. In the long term, it is also possible to envision the use of this algorithm to improve patrolling efforts in other districts of the city of Fortaleza, as well as in other cities.

References

1. Kernighan, B.W., Lin, S.: An efficient heuristic procedure for partitioning graphs. Bell Sys. Tech. J., 291–307 (1970)
2. Fiduccia, C.M., Matteyses, R.M.: A linear time heuristic for improving network partitions. In: 19th Design Automaton Conference, pp. 175–181 (1982)
3. Chlebíková, J.: Approximating the maximally balanced connected partition problem in graphs. Information Processing Letters 60, 225–230 (1996)
4. Chataigner, F., Salgado, L.R.B., Wakabayashi, Y.: Approximability and inaproximability results on balanced connected partitions of graphs. Discrete Mathematics and Theoretical Computer Science (DMTCS) 9, 177–192 (2007)
5. Lucindo, R.P.F.L.: Partição de Grafos em Subgrafos Conexos Balanceados. Dissertação de mestrado, Universidade de São Paulo (2007)

6. Heath, M.T., Raghavan, P.: A Cartesian Parallel Nested Dissection Algorithm. SIAM Journal on Matrix Analysis and Applications (1994)
7. Pereira, M.R.: Particionamento Automático de Restrições. Tese de Doutorado, Universidade Federal do Rio de Janeiro (COPPE) (2006)
8. Karypis, G., Kumar, V.: A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. Technical Report TR 95-035. Departament of Computer Science. University of Minnesota (1995)
9. Salgado, L.R.B.: Algoritmos de Aproximação para Partições Conexas em Grafos. Tese de doutorado, Universidade de São Paulo (2004)
10. Moretti, C.O., Bittencourt, T.N., André, J.C., Martha, L.F.: Algoritmos Automáticos de Partição de Domínio. Escola Politécnica da Universidade de São Paulo, Boletim Técnico, BT/PEF-9803 (1998) ISSN 0103-9822
11. Simon, H.D., Teng, S.H.: Partitioning of Unstructured Problems for Parallel Processing. Computing Systems in Engineering 2, 135–148 (1991)
12. Gersting, J.L.: Mathematical Structures for Computer Science, 6th edn. W.H. Freeman, New York (2006)
13. Jung. Java Universal Network/Graph Framework (2003) (Last access: February 14 , 2008), `http://jung.sourceforge.net/index.html`