Congress on Numerical Methods in Engineering 2011
Coimbra, 14 to 17 June, 2011

PARTITIONING GENERIC GRAPHS INTO K BALANCED SUBGRAPHS

David Portugal and Rui P. Rocha

Institute of Systems and Robotics
Department of Electrical and Computers Engineering
University of Coimbra
3030-290 Coimbra, Portugal
e-mail: {davidbsp, rprocha}@isr.uc.pt

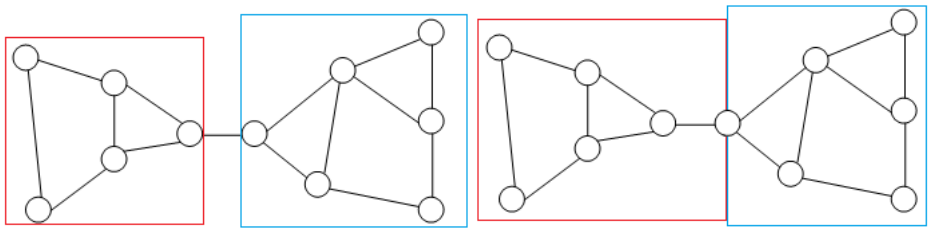
Keywords: Graph Theory, Balanced Partition, Bisection, Subgraphs.

Abstract *Graph partitioning is a classical graph theory problem that has proven to be NP-hard.*

Most of the research in literature has focused its attention on a particular case of the problem called the graph bisection problem, where $k = 2$, such that the parts have approximately equal weight and minimizing the size of the edge cut.

In this article, we describe how to obtain balanced partitioning on a given undirected, connected and weighted graph into an arbitrary number k of regions (subgraphs), by hierarchically employing a multilevel bisection algorithm not only in the general graph, but also in the originated subgraphs.

Due to the application chosen for this study, the partition consists of k subgraphs, which are subsets of vertices in the same region, not intended to be totally disjoint, sharing at least one vertex with another subgraph near their borders.



(a) Classical graph-cut. (b) Shared-vertex partitioning.

Figure 1: Simple bisectioning of a generic graph in two subgraphs.

1 INTRODUCTION

Graph theory related problems can be useful in many research fields. In particular, graph partitioning algorithms find their application in areas like computer program segmentation, sparse matrix reordering, floor planning, circuit placement, cluster ranging from computer vision, data analysis and other important disciplines. However, the Graph Partitioning Problem is known to be NP-Complete [1]. In the last decades, many authors have proposed sub-optimal algorithms and approximated heuristics for solving this problem. In this paper, the problem addressed is slightly different from the classical Partitioning Problem, in the sense that it is not required that the generated subgraphs become totally disjoint, i.e. regions are not connected by edges which have end points in different regions, like in Figure 1a; instead subgraphs may share a vertex or some vertices near their borders, as seen on Figure 1b.

Although the problem has a subtle difference to the classical partitioning problem, it can be efficiently solved using similar fast heuristics. Our approach is based on generalizing a graph bisectioning algorithm, by also employing intentionally unbalanced bisections on subgraphs in order to obtain arbitrarily high k partition sets.

This choice of design happened due to the application chosen, for which the method was developed. In [2] we have presented an algorithm which, among others, partitions a given graph into a maximum of eight regions. This graph was used as a topological environment for mobile robots navigation in 2D environments and the originated subgraphs correspond to areas, where each different robot travels. In order not to leave out any edge, shared-vertex partitioning was employed. In this paper, we describe how we have generalized the algorithm to partition a graph into an arbitrary number of k regions and show how the method usually produces balanced partition sets.

Topological maps translate into generic graphs with the following properties:

- Undirected Graphs: edges have no orientation and both ends of an undirected edge are equivalent, having a symmetric relation between them; e.g., if an edge connects vertex A to vertex B, the same edge connects vertex B to vertex A.
- Weighted Graphs: a label (weight) is associated with every edge in the graph. In the case of topological maps, the weight of an edge generally translates into the cost of

traversing the edge or the length associated to it.

• Simple Graphs: there are no loops and no more than one edge between any two different vertices.

• Fixed/Static Graphs: the topological properties of the graph like the relations between vertices and the edge weights never change.

• Connected Graphs: there exists at least one path between all pairs of vertices.

• Non-Complete Graphs: there are no unique edges that connect every pair of distinct vertices.

The next section represents a brief survey of the work previously done in the area and section 3 reviews the bisectioning approach used in this paper. The following section presents the proposed algorithm to partition the graph in k balanced regions. Later on, it will be shown that the results prove its efficiency. Afterwards, the article ends then with final conclusions.

2 RELATED WORK

During the last four decades, a growing interest in solving the graph partitioning problem has been noticeable. As referred before, since finding an optimal partitioning is NP hard, one is forced to settle for approximation algorithms. Most of the investigation in this area focuses on methods for bisectioning the graph into two regions, which is also a NP-Hard Problem.

One of the earliest attempts and perhaps the most well-known heuristic algorithm for partitioning graphs was described in [3], which takes two separate sets as an initial solution of the problem, and exchanges pairs of vertices between them in order to obtain the best possible solution.

Throughout the years, other heuristics based on different mathematical tools have been presented like [4], which used simulated annealing for solving the graph partitioning and the traveling salesman problems. Following that work, [5] showed that simulated annealing finds the optimal bisection in a random graph with very high probability. In addition, [6] described a partitioning algorithm that combines characteristics of the simulated annealing algorithm and the Hopfield neural network.

Branch and bound approaches to solve the partitioning problem in the case of $k = 2$ and for general weighted graphs have also been described in [7] as well as [8]. Yan and Hsiao have presented a fuzzy clustering algorithm to solve the graph bisection problem and apply it to circuit partitioning [9]. Other authors have presented approaches based on genetic algorithms [10], divide-and-conquer approximation algorithms [11] and even ant colony optimization [12].

Linear programming methods became popular after being shown that they were able to find better cuts than Kerninghan-Lin. For instance, [13] used linear programs related to multicommodity flow to approximate the minimum cut arrangement problem and [14] built upon expander flows to present a new approach for finding graph separators that

use single commodity flows. Additionally, Lisser and Rendl used linear and semidefinite programming to solve graph partitioning in the context of the telecommunication industry, presenting data from France Telecom networks with up to 900 nodes [15]. On the other hand, spectral methods also became vastly used, since they were faster and produced good results. These are based on the computation of eigenvectors of the adjacency matrix, also known as the spectrum of the graph. Several works have used such techniques like [16], [17] and [18].

As an alternative, multilevel algorithms for partitioning graphs were first described by [19] and [20]. Typically such multilevel schemes match pairs of adjacent vertices to define new coarsened graphs and recursively iterate this procedure until the graph size falls below some threshold. The coarsest graph is then cut and the partition is refined on all the graphs back to the original one. Such methods benefit from being very fast and producing cuts that are almost as good as those obtained by linear programming based methods, as shown in the works of Karypis and Kumar [21], which presented a multilevel heuristic embodied in the METIS package and Walshaw and Cross [22], which described JOSTLE, a software based on parallel multilevel graph-partitioning.

In addition to heuristics and approximate algorithms for solving the graph-partitioning problem, many authors have analyzed the lower bounds of known algorithms and in special case of graphs (e.g. [23] and [24]), which is beyond the scope of this article.

While most of the work referred herein attempt to partition graphs into two (or more) regions while minimizing the size of the graph-cut (the sum of the weights of the edges, which have endpoints in different regions), since we use shared-vertex partitioning, we are essentially concerned on dividing a graph G into k disjoint and balanced parts, such that the parts have approximately equal weight, which is also a fundamental combinatorial problem. To solve the problem, our analysis builds upon the multilevel partitioning algorithm presented in [21].

3 MULTILEVEL BISECTIONING ALGORITHM

3.1 Fast Multilevel Approach for Bisectioning Generic Graphs

In [21], a high quality multilevel approach for partitioning irregular graphs was presented. The heuristic proceeds by collapsing random edges until the resulting graph is quite small. It finds a good partition in this collapsed graph, and successively induces it up to the original graph, using local search. There are three main phases in this method: coarsening, partitioning and uncoarsening.

Basically, in the coarsening phase, a sequence of smaller and less complex graphs, each with fewer vertices, is obtained by collapsing vertices and edges into single vertices of the next level, which are called multinodes. An example of a coarsened graph with multinodes is presented in figure 2. Then, in the partitioning phase, a simple bisection is found for the coarsened graph and lastly, in the uncoarsening phase, the partitioning is refined while the

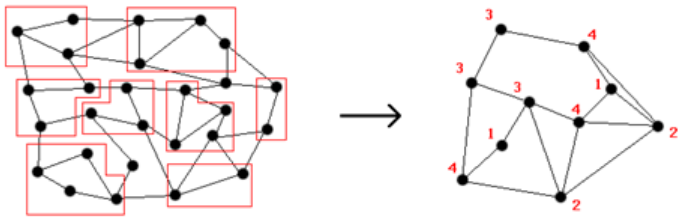


Figure 2: Example of a coarsened graph.

original graph is being restored. Diverse methods for all the three phases of the scheme are described in the original paper and comparisons were made between these methods. Analyzing the results and their properties, some methods were selected for this work with some minor changes in a few cases.
For the coarsening method, the Heavy-Edge Matching was used. Vertices of the graph are visited in random order. To form multinodes, a vertex is matched with a neighbor vertex, such that the weight of the edge between them is maximum over all valid incident edges (heavier edge). Careful observation of the final multinodes generated shows that vertices which are highly related tend to be included in the same multinode. This method was chosen mainly due to its good results and simplicity.
In the partitioning phase, the approach used was based on the Kerninghan-Lin Algorithm [3], which starts with an initial bipartition of the graph and, in each iteration, it searches for a subset of vertices, from each part of the graph such that swapping them leads to a partition with smaller edge-cut. In our case, the edges are not cut. Every edge belongs to a unique region and the algorithm is adapted so that the portioning is done in frontier vertices. The frontier vertices define the boundaries of each subgraph. The main difference in the approach used in this work is that the initial bipartition starts by separating the largest multinode from the rest, evaluating an equilibrium condition (the sum of edge weights in both regions) and swapping multinodes from one side to the other only if the equilibrium condition improves.
As for the uncoarsening system, the KL(1) Refinement algorithm is used. The projected partition of the coarsened graph is assumed as a good initial partition for the upper level graph and vertices are swapped during the first uncoarsening phase, to improve the global equilibrium condition.

3.2 Bisectioning graphs into subgraphs with different weights

As it was previously seen, the employed multilevel approach creates a bisection of the graph, which results in two smaller subgraphs. To be able to create more than two subgraphs (k $>$ 2), unbalanced partitions were also produced by changing the parameters of the equilibrium condition in the partitioning phase of the multilevel bisection algorithm. This is a necessary condition in our method to create arbitrarily high k subgraphs from a generic input graph.

For instance, a three region balanced partitioning is done by firstly dividing the main graph into two regions with an unbalanced partition condition of 33.33% and 66.66% of its dimensions. Then, the largest subgraph obtained is divided into two regions with a 50% - 50% balanced condition. Three regions with 33.33% of the graphs dimension is the aim for the final result, as seen in figure 3 for the case of $k = 3$.

4 ALGORITHM FOR GENERALIZING THE PARTITION IN K REGIONS

4.1 Description

At this point, we have a fast bisection algorithm which produces both balanced and unbalanced subgraphs as desired. In this section, we describe a simple algorithm to generalize the partition of a graph in k pieces, based on the multilevel bisection algorithm. Figure 3 illustrates our approach. The idea is to hierarchically apply the bisectioning algorithm in the produced subgraphs to efficiently obtain the final k subgraphs as intended. As shown in the pseudo-code of algorithm 1 (section 4.2), the method accepts as input a graph G and an intended number of regions or subgraphs (k). During the procedure, the algorithm keeps track of the fraction of the graph that each subgraph represents in a table of integers called $Numerator$, which is similar to a label. The original graph has a $Numerator$ value of k (index 1). From then on, the algorithm attempts to successively divide the graph and subgraphs in half or around half of its dimensions, using the $floor$ and *ceil* functions, until the current subgraph $Numerator$ value falls to 1, which means that that portion should no longer be divided.
Note that in the algorithm, the function *bisection* accepts as parameters the graph G; the index of the subgraph to partition from (*prevregion*), which will be inherited to one of the originated subgraphs; the index of the other originated subgraph (*newregion*) and the fractions to include in the equilibrium condition of the bisectioning algorithm, which will determine if it is a balanced (50%-50%) or unbalanced condition. In addition the function returns a boolean variable, which is true in case the bisectioning was successful or false otherwise. One of the possible reasons for returning a false value is when the subgraph can no longer be partioned. When this happens, the routine is aborted.
The method stops when all subgraphs have been originated, having a resulting $Numerator$ value of 1 and the number of *partlevel* (partition levels) exceeds $\log_2 k$.

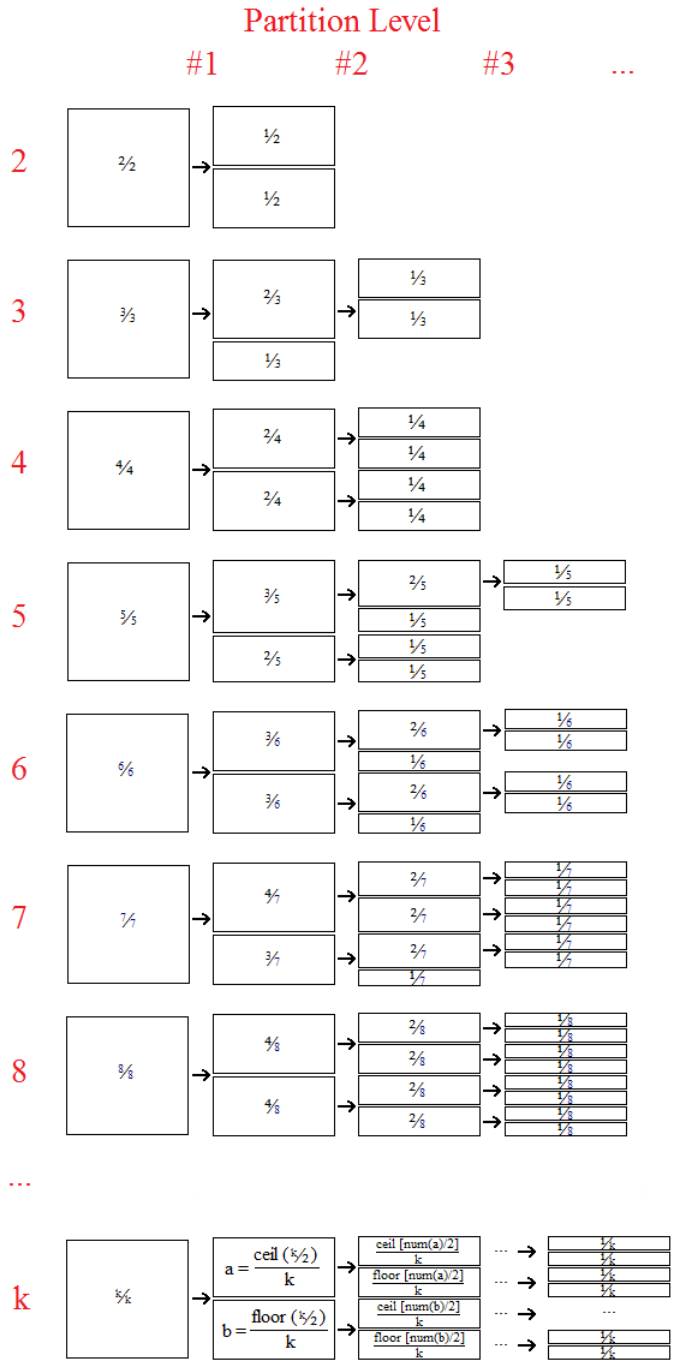


Figure 3: Generalizing the partitioning in k parts.

4.2 Pseudo-Code

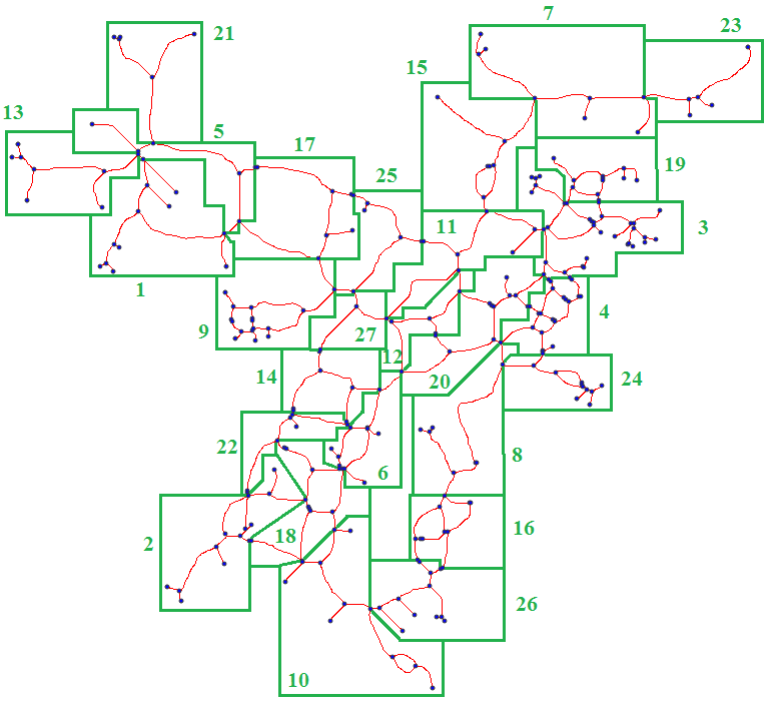
Algorithm 1: Partitioning a Generic Graph into k Regions.

```
/* Inputs                                                                    */
G: Generic Connected Graph
k : Number of Total Regions

if k < 2 then return;

zeros(Numerator [k + 1]); // Table [0, ..., k] initialized with zeros.
Numerator [1] ← k;
partlevel ← 1;
keep_going ← true;

while keep_going do
    if k > 2^(partlevel−1) then
        count ← 2^(partlevel−1) ;
        inc ← 1;

        while inc ≤ count do
            prevregion ← 0;

            for i ← inc to count do
                if Numerator [i] > 1 then
                    prevregion ← i;
                    break;

            if prevregion ≠ 0 then
                newregion ← count +inc;
                Numerator [newregion ] ← floor( Numerator[prevregion] / 2 );
                Numerator [prevregion ] ← ceil( Numerator[prevregion] / 2 );
                keep_going ← bisection(G, prevregion, newregion, Numerator[prevregion] / k,
                Numerator[newregion] / k );
            else
                break;

            inc ++ ;

    if partlevel ≥ log_2 k ∧ keep_going then
        return;

    partlevel ++ ;
```

5 RESULTS AND DISCUSSION

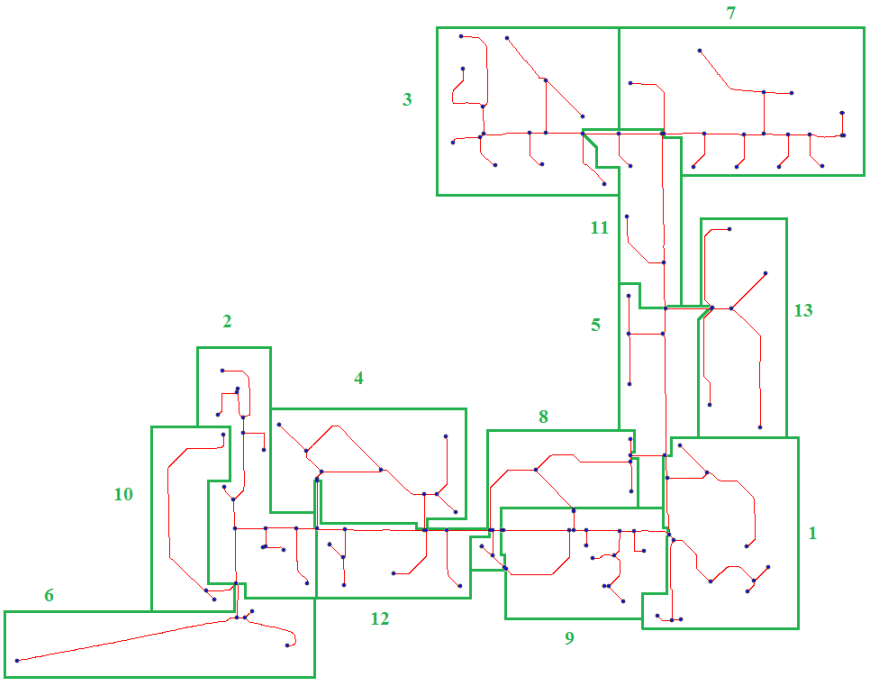
In this section we present some results of the partitioning algorithm applied to diverse graphs:

- Graph A, presented in figure 4, has 268 vertices and is partitioned in 27 regions.
- Graph B, presented in figure 5, has 135 vertices and is partitioned in 13 regions.
- Graph C, presented in figure 6, has 70 vertices and is partitioned in 15 regions.
- Graph D, presented in figure 7, has 78 vertices and is partitioned in 12 regions.



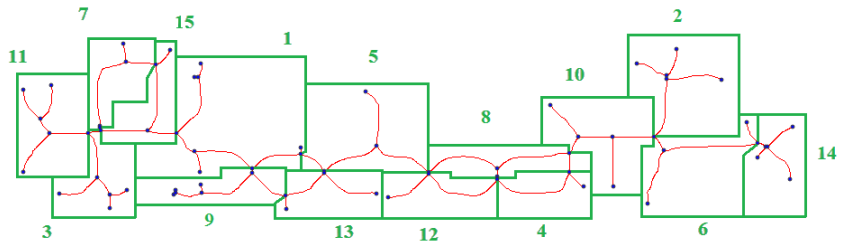
Region	Proportion (%)
1	4.92
2	5.07
3	6.10
4	3.82
5	3.63
6	3.35
7	4.14
8	4.01
9	4.63
10	5.00
11	4.22
12	3.14
13	3.40
14	3.09
15	3.66
16	3.69
17	4.58
18	4.45
19	3.35
20	3.92
21	2.88
22	2.77
23	2.46
24	2.72
25	2.77
26	2.82
27	1.42

Figure 4: Graph A: 27-way partition.



Region	Proportion (%)
1	8.94
2	10.96
3	10.73
4	6.85
5	8.77
6	6.34
7	10.44
8	6.07
9	7.88
10	4.39
11	6.65
12	6.80
13	5.18

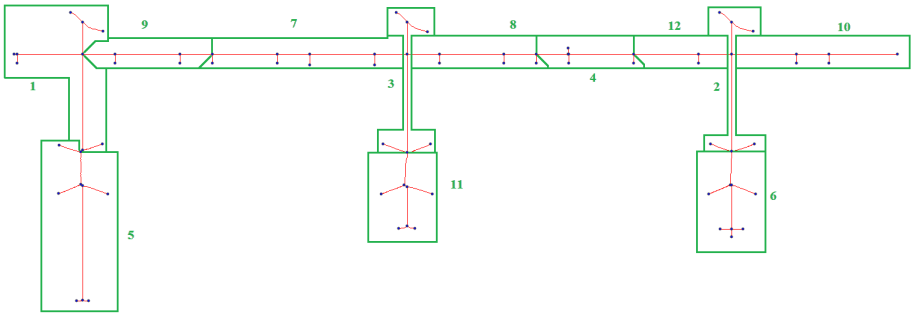
Figure 5: Graph B: 13-way partition.



Region	Proportion (%)
1	10.30
2	7.75
3	6.54
4	4.28
5	8.44
6	8.16
7	5.74
8	7.51
9	5.74
10	7.84
11	6.99
12	5.09
13	4.85
14	3.68
15	7.11

Figure 6: Graph C: 15-way partition.

It is shown that the method obtains balanced partitions using arbitrarily high k regions, and different topologies. The algorithm is extremely fast, being programmed in C++ and obtaining the final partitions typically in a few hundredths of seconds using a single-core AMD Athlon64 Processor 3500+, 2.21GHz, with 1GB RAM, running Ubuntu Linux 10.10. The method also presents a low mean deviation on the proportion of the weights of the regions ($\overline{\sigma_{prop}} \simeq$ 1-2%), which confirms how balanced the final partitions usually are. These results may vary, depending essentially on the dimension, in terms of overall weight; the number of vertices of the graph and the intended number k of partitions, as shown in Table 1.



Region	Proportion (%)
1	11.97
2	9.59
3	9.73
4	5.65
5	10.77
6	7.22
7	10.91
8	7.27
9	6.88
10	8.56
11	6.46
12	4.98

Figure 7: Graph D: 12-way partition.

Graph	Vertices	k	$\overline{\sigma_{prop}}$ (%)
A	268	27	0.77
B	135	13	1.78
C	70	15	1.43
D	78	12	1.92

Table 1: Summary of the Results

6 CONCLUSIONS

In this article, a new balanced graph partitioning method based on the generalization of a hierarchical multilevel bisectioning approach with shared vertices has been proposed. The method uses an approximation heuristic, which does not guarantee finding the optimal solution. Nevertheless, it benefits from its simplicity, efficiency and computation speed, which were proven in the presented results, being able to partition undirected weighted graphs into any arbitrary number k of regions in a balanced way, up to the point where the graph can no longer be partitioned.
The method also has the potential to be used in a wide range of applications, since it is viable for running in real-time, as shown in [2], where it was applied in multi-robot patrolling missions. Additionally, it is not limited only to the multilevel bisection algorithm described and shared-vertex partitioning, the generalization method can also be applied with a different *bisection* function and classical graph-cut, as long as the equilibrum condition takes into account the minimization of the graph-cut.

REFERENCES

[1] M. R. Garey, D. S. Johnson and L. J. Stockmeyer, *Some Simplified NP-Complete Problems*. In Proceedings of the sixth annual ACM symposium on Theory of computing (STOC'74), 47-63, 1974.

[2] D. Portugal and R. Rocha, *MSP Algorithm: Multi-Robot Patrolling based on Territory Allocation using Balanced Graph Partitioning*. In Proceedings of 25th ACM Symposium on Applied Computing (SAC 2010), Special Track on Intelligent Robotic Systems, 1271-1276, Sierre, Switzerland, March 22-26, 2010.

[3] B. W. Kernighan, and S. Lin, *An efficient heuristic procedure for partitioning graphs*. Bell Systems Technical Journal 49: 291-307, 1970.

[4] S. Kirkpatrick, *Optimization by Simulated Annealing: Quantitative Studies*. Journal of Statistical Physics, Vol 34, No. 5/6, 1984.

[5] M. Jerrum and G. B. Sorkin, *Simulated Annealing for Graph Bisection*. In Proceedings of the 34th Annual Symposium on Foundations of Computer Science, 94-103, Palo Alto, California, USA, 3-5 November, 1993.

[6] D. E. Van Den Bout and T. K. Miller, *Graph Partitioning Using Annealed Neural Networks*. IEEE Transactions on Neural Networks, Vol. 1, No. 2, June 1990.

[7] C. Roucairol and P. Hansen, *Cut cost minimization in graph partitioning*. Numerical and Applied Mathematics, pp. 585-587, 1989.

[8] S. E. Karisch, F. Rendl and J. Clausen, *Solving Graph Bisection Problems with Semidefinite Programming*. INFORMS Journal on Computing, Vol. 12, Issue 3, July 2000.

[9] J. Yan and P. Hsiao, *A fuzzy clustering algorithm for graph bisection*. Information Processing Letters, Vol. 52, Issue 5, pp. 259-263, December 1994.

[10] T. N. Bui and B. R. Moon, *Genetic Algorithm and Graph Partitioning*. IEEE Transactions On Computers, Vol. 45, No. 7, 841-855, July 1996.

[11] G. Even, J. Naor, S. Rao and B. Schieber, *Fast Approximate Graph Partitioning Algorithms*. In SIAM Journal on Computing, Vol. 28, pp. 639-648, 1997.

[12] T. N. Bui and L. C. Strite, *An Ant System Algorithm For Graph Bisection*. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '02), W. B. Langdon et al. (Eds.), pp. 43-51, Morgan Kaufman Publishers, 2002.

[13] F. Leighton and S. Rao, *Multi-commodity max-flowmin-cut theorems and their use in designing approximation algorithms*. Journal of the ACM (JACM), Vol. 46, Issue 6, 787-832, 1999.

[14] R. Khandekar, S. Rao, U. Vazirani, *Graph partitioning using single commodity flows*. Journal of the ACM (JACM), Vol. 56, Issue 4, June, 2009.

[15] A. Lisser and F. Rendl, *Graph partitioning using linear and semidefinite programming*. Mathematical Programming, Vol. 95, No. 1, 2003.

[16] B. Hendrickson and R. Leland, *An Improved Spectral Graph Partitioning Algorithm for Mapping Parallel Computations*. SIAM Journal on Scientific Computing, 16(2),452-469, 1995.

[17] F. Rendl and H. Wolkowicz, *A projection technique for partitioning the nodes of a graph*. Annals of Operations Research, Vol. 58, pp. 155-179, 1995.

[18] D. A. Spielman, S. Teng, *Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems*. In Proceedings of the 36th Annual ACM Symposium on Theory of Computing, pp. 81-90, 2004.

[19] S. T. Barnard and H. D. Simon. *A Fast Multilevel Implementation of Recursive Spectral Bisection for Partitioning Unstructured Problems*. In Proceedings of PPSC. 711-718, 1993.

[20] B. Hendrickson and R. Leland. *A Multilevel Algorithm for Partitioning Graphs*. In Proceedings of Supercomputing '95, San Diego. ACM Press, New York, 1995.

[21] G. Karypis and V. Kumar. *A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs*. Society for Industrial and Applied Mathematics Journal of Scientific Computing, 359-392, 20 (1), 1998.

[22] C. Walshaw and M. Cross. *JOSTLE: Parallel Multilevel Graph-Partitioning Software - An Overview*. In F. Magoules, editor, Mesh Partitioning Techniques and Domain Decomposition Techniques, pages 27-58. Civil-Comp Ltd., 2007.

[23] U. Feige, R. Krauthgamer and K. Nissim, *Approximating the Minimum Bisection Size (Extended Abstract)*. In Proceedings of the 32nd Annual ACM symposium on Theory of computing, May 21-23, Portland, Oregon, USA, 2000.

[24] E. M. Arkin, R. Hassin, *Graph Partitions with Minimum Degree Constraints*. Journal of Discrete Mathematics, Vol. 190, Issues 1-3, pp. 55-65, August, 1998.