Graph Partitioning

Edited by
Charles-Edmond Bichot
Patrick Siarry

ISTE
WILEY

Table of Contents

Part 3: Other Approaches to Graph Partitioning

Graph Partitioning

Edited by
Charles-Edmond Bichot
Patrick Siarry

ISTE
WILEY

First published 2011 in Great Britain and the United States by ISTE Ltd and John Wiley & Sons, Inc

ISTE Ltd
27-37 St George's Road
London SW19 4EU
UK
www.iste.co.uk

John Wiley & Sons, Inc.
111 River Street
Hoboken, NJ 07030
USA
www.wiley.com

Library of Congress Cataloging-in-Publication Data

Graph partitioning / edited by Charles-Edmond Bichot, Patrick Siarry.

p. cm.

Includes bibliographical references and index.

ISBN 978-1-84821-233-6

1. Partitions (Mathematics) 2. Graph theory. I. Bichot, Charles-Edmond. II. Siarry, Patrick.

QA76.165.G73 2011

512.7'3--dc23

2011028388

British Library Cataloguing-in-Publication Data

A CIP record for this book is available from the British Library

ISBN 978-1-84821-233-6

Introduction

Since the early 1970s, graph partitioning has been a widely researched topic. After 40 years development, it is now time to evaluate the situation of the work done in this area.

Graph partitioning is a problem transverse to several fields in engineering, as well as research. Th diversity of backgrounds of people working on this subject may explain why no real internation community exists, despite the fact that some workshops have been organized on this subject ov different periods of time. However, the number of people who work or have worked on this topic quite significant, as shown by the abundant literature cited throughout the chapters of this boo Confronted with this profusion and diversity, it is interesting to gather the knowledge accumulate over so many years in order to synthesize it and draw the common theoretical fundamentals of th field.

This book intends to present to the neophyte reader, as well as the expert in applied mathematics the computer science researcher, tools and methods to solve graph partitioning optimization problem For this purpose, we have collected several methodological chapters detailing different grap partitioning optimization approaches such as the multilevel method, metaheuristics, partitionin parallelization, and hypergraph partitioning. In order to complete this theoretical part, several grap partitioning applications have been described on subjects as diverse as mobile networks, imag segmentation, air traffic control, social networks, etc.

Despite the large number of studies in the domain of graph partitioning, it is clear that a lot of wo remains to be done to solve this problem more efficiently. Recent years have seen the sizes of graph to be partitioned soaring from a few thousand vertices to several million, or even billions of vertice We hope that by reading this book the reader will feel inspired not only to take an interest in th problem, but also try to solve it more efficiently, both in terms of quality of partitions found an computation time required.

This book has three parts: Part 1 is dedicated to the most common application of graph partitionin numerical analysis; Part 2 describes and implements several combinatorial optimization methods f graph partitioning; Part 3 presents other uses of graph partitioning.

Chapter 1 provides a general introduction to this book and is therefore independent of any part. analyzes graph partitioning in order to identify the different problems associated with it.

A large number of studies on graph partitioning have been undertaken within the domain numerical analysis. Part 1 of this book is dedicated to their presentation. Thus, Chapter 2 describes th methods and algorithms commonly used in numerical analysis to solve graph partitioning problem the multilevel method, refinement algorithms like the Kernighan-Lin or Fiduccia-Mattheys algorithms, and the spectral method. Chapter 3 introduces the particular case of hypergrap partitioning, which often occurs in numerical analysis. Chapter 4 presents several parallel algorithm to partition a graph. Finally, Chapter 5 presents the problem of static mapping which occurs in parall computing.

Graph partitioning is often studied through a combinatorial optimization point of view. This is th theme of Part 2 of this book, which is dedicated to the study of combinatorial optimization method and more particularly metaheuristics, for graph partitioning. This part consists of two theoretic chapters followed by two application chapters. Chapter 6 focuses on the use of several loc metaheuristics, like simulated annealing or iterated local search. Chapter 7 provides details on the u

of population-based metaheuristics to optimize partitioning. This chapter explains the work done o
Ant Colony algorithms and describes several adaptations of Genetic Algorithms to graph partitionin
It also introduces a recent method for graph partitioning optimization, fusion-fission, which works
a meta-method that overlooks a multilevel algorithm. The last 2 chapters of this part provi
application examples for this part. Chapter 8 applies a Genetic Algorithm to the problem of mobi
network partitioning in tariff zones. Chapter 9 describes an Air Traffic Control problem and solves
using the fusion-fission method.

Part 3 of this book develops other approaches of graph partitioning. In this part, we have chosen
focus on graph partitioning optimization, and thus we limited the scope of this part, even if mar
other works on graph partitioning could have been added, like constraint programming for grap
partitioning or graph decomposition. Chapter 10 outlines the image segmentation problem and offe
several methods to solve this problem, based on graph partitioning. Chapter 11 compares sever
distances between vertices, in order to build communities, using classification methods. Chapter
proposes to partition networks into unconnected or overlapping communities. Finally, Chapter
concludes by describing how to partition very large networks into communities.

Further information that could not be included in this book is available at the book's Web address
In particular, as this book is being printed in black and white, some figures will have lost their clari
when compared to their original color version. This is especially true for images in Chapters 9 and 1
respectively, about airspace partitioning and image segmentation. To overcome this problem, th
original color versions of the figures concerned are also available at the book's Web address. You wi
also find at this Web address different graphs used in this book, as well as links to several softwar
for graph partitioning.

Charles-Edmond BICHOT and Patrick SIARR
August 201

---

1 The book's Website, "Graph partitioning: optimization and applications", is available a
perso.ec-lyon.fr/charles-edmond.bichot/.

Chapter 1

General Introduction to Graph Partitioning [1]

1.1. Partitioning

The word “partitioning” represents the action of creating a partition by segmenting a set into sever parts. In mathematics, a partition of a set X is a family of pairwise disjoint parts of X whose union the set X . Therefore, each element of X belongs to one, and only one of the parts of the partition.

Partitioning is used to solve a large number of engineering problems. There are many examples partitioning applications: data mining, VLSI design, load balancing, parallel computing, matr computation, fluid dynamics, image analysis, air traffic control, etc.

Creating a partition consists of distributing a set of objects into several subsets. In order distribute the objects in these different subsets, it may be useful to compare them. Thus, each obje will be associated with a few other objects. When each object is associated with all other objects, th the grouping is known as clustering, and not partitioning. The resulting associations would quantified. If the set of all these objects is finite, then all the conditions would be met to create graph from these objects, in other words to materialize the associations between objects. The next st will be to build a partition of this graph. Partitioning a set of objects generally aims to distribute the objects into parts having strong internal links and weak links between parts. That is known as t objective of partitioning or objective function. This objective function varies according to the specif problem which is to be solved. We emphasize the fact that in this book we mainly focus on th optimization problem of minimizing an objective function. Thus, we will generally seek to minimi the links between parts rather than maximize them.

The problem of graph partitioning optimization is common to several disciplines:

– it is part of graph theory problems and therefore discrete mathematics. Discrete mathemati relates to discrete structures, i.e. finite or countable sets;

– it is also part of the combinatorial optimization problems. A combinatorial optimization proble tries to find out the best solution within a set of discrete solutions;

– it is solved using computing.

Graph partitioning is, therefore, a discipline situated between computer science and appli mathematics.

Because of the diversity of its applications, the graph partitioning optimization problem evolves in multitude of problems. However, we can group them into two large categories. The nature of a grap partitioning problem is very different according to whether we seek to obtain parts of very simil sizes or without any size constraint. This observation leads us to distinguish between two differe graph partitioning problems:

– *constrained partitioning*, when the parts of the partition are of similar sizes;

– *unconstrained partitioning*, when the parts can be of (very) different sizes.

Before giving a precise definition of these two problems, this introductory chapter presents a fe

reminders on mathematics in section 1.2 and graph theory in section 1.3. These will be followed by formal description of the graph partitioning optimization problem in section 1.4 and its differe objectives in section 1.5. Then the two main categories of graph partitioning problems will b presented: first the constrained partitioning problem in section 1.6, and then the unconstraine partitioning problem in section 1.7. Finally, section 1.10 will study the NP-hardness of several grap partitioning problems.

1.2. Mathematical notions

DEFINITION 1.1. (CARDINALITY OF A SET)– *Let X be a finite set of n elements. The cardinality of the set is the number n of elements of X , and is denoted by |X |. Therefore, we have |X | = n.*

In mathematics, the partition of a set V of objects is defined as follows:

DEFINITION 1.2. (PARTITION)– *Let V be an ordinary set. A set P of subsets of V is called a partition of if:*

– no element of P is empty;

– the elements of P are pairwise disjoint;

– the union of the elements of P is equal to V.

The elements of P are called the parts of the partition P. The cardinality of the partition P i therefore the number of parts of P. In graph partitioning, the number of parts of P is often denoted k.

It is a bad habit to use the word partition instead of part, which creates ambiguity. Thus, instead saying "the partition P_1 of P", it is better to say "the part P_1 of the partition P".

As previously mentioned, the graph partitioning problem is a combinatorial optimization proble which is a branch of discrete mathematics. Discrete mathematics represent the study of mathematic structures, where the notion of continuity doesn't exist. Thus, sets studied in discrete mathematics a countable, and therefore countable sets are sometimes called discrete sets.

We can now give a formal definition of a combinatorial optimization problem:

DEFINITION 1.3. (COMBINATORIAL OPTIMIZATION PROBLEM) – *A combinatorial optimization problem defined from a triplet (S, p, f) such that*:

– S is a discrete set called research space (also called solution space);

– p is a predicate on S, i.e. a function of S in {true, false};

– $f : S \rightarrow R$ associates with each element $x \in S$ a cost $f(x)$.

The function f is called the objective function or the cost function of the problem. The predicate creates a set $S_a = \{x \in S$ such that $P(x)$ is true$\}$ which is called the set of admissible solutions of t problem. Each element of S is called a solution of the problem, and each element of S_a an admissib solution of the problem.

A combinatorial optimization problem aims to find the element $\tilde{x} \in S_a$ (or the subset of elements Sa) that minimizes f:

[1.1] $$f(\tilde{x}) = \min_{x \in S_a} f(x)$$

Note: The combinatorial optimization problem which looks for a maximum element instead of minimum element, is of the same nature because:

[1.2] $$\max_{x \in S_a} f(x) = -\min_{x \in S_a} (-f(x))$$

Combinatorial optimization problems are generally very expensive to solve optimally. This particularly the case of graph partitioning problems, which are often NP-hard, as we will see in section 1.10.

The research space S can be described as the set of solutions of the combinatorial optimizatio problem, which we seek to solve without taking into account the problem's constraints. For exampl let's consider the combinatorial optimization problem aiming to find a partition of the vertices of graph $G = (V, E)$ into k parts of equal sizes (we choose k divisor of $|V|$). This problem will have research space S the set of all possible partitions of V, i.e. partitions of parts of any sizes who number of parts ranges from one to $|V|$. However, the set of admissible solutions to problem Sa mu take the constraints into account. Thus, in our example, the set Sa will be made of partitions of V in k parts of equal sizes.

DEFINITION 1.4. (GLOBAL OPTIMUM, LOCAL OPTIMUM) – *Let (S, p, f) be acombinatorial optimizatio problem and Sa the set of admissible solutions of the problem resulting from the predicate p. Let $\bar{x}$ Sa.*

– If we can prove that $\forall x \in S_a$, $f(\bar{x}) \leq f(x)$, then $\bar{x}$ is called a global optimum (minimum) of t problem;

– If there is a set of admissible solutions $X \subset S_a$, $|X| \geq 2$, and $\bar{x} \in X$ such that $\forall x \in X$, $f(\bar{x}) \leq f(x$ then $\bar{x}$ is called a local optimum (minimum) of the problem.

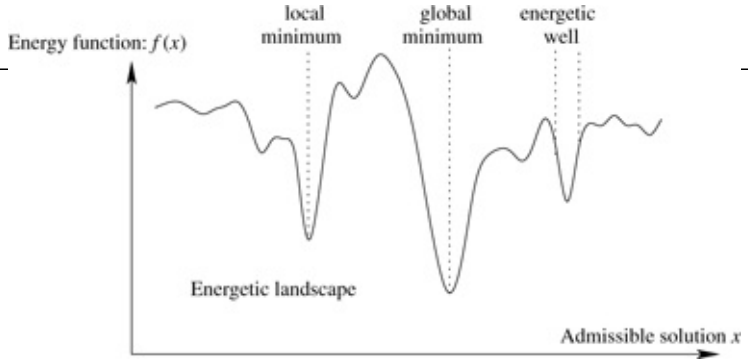
The research space S has a *topology*. It is very useful to know the characteristics of the topology of in order to understand the purpose of the metaheuristics mechanisms and to improve them (Chapters and 7 present how to adapt metaheuristics to graph partitioning). This topology results from the notio of proximity between two solutions, also called configurations. The distance between tw configurations represents the minimum number of elementary modifications necessary to go from on to another. Besides, because a value $f(x)$ is associated with each configuration x, the solution space characterized by a multi-dimensional chart called *energetic landscape*[1]. In this energetic landscap the local or global optima form energetic wells around them; these *energetic wells* are also name basins of attraction. Before saying that an admissible solution $\bar{x} \in X$ (see definition 1.4) of t problem is a local minimum, we must generally check that the size of the set X is large enough relation to the size of S_a. Figure 1.1 represents the continuous equivalent of the energetic landscape a cost function in a single-dimension solution space.

1.3. Graphs

This section is a reminder of some definitions of the graph theory.

DEFINITION 1.5. (GRAPH)– *Let V be a non-empty set of n_V elements and E a set of nE pairs of elemen of V. A graph G is the pair (V, E). The elements of V are called*

Figure 1.1. *Continuous equivalent of the energetic landscape of a cost function f (x) in a single-dimension research space*



vertices of the graph and those of E the arcs or edges of the graph, depending on whether they a directed or not.

An edge represents a connection between exactly two vertices. When there is a connection betwe at least three vertices, these vertices are connected by what is called a hyperedge. By extension, we can longer use the term graph, but hypergraph.

DEFINITION 1.6. (HYPERGRAPH)– *Let V be a non-empty set of elements and E a set of non-empty subse of elements of V . A hypergraph G is the pair (V, E). The elements of E are called hyperedges.*

Note: The problem of hypergraph partitioning is presented in Chapter 3.

When the associations between vertices are not reciprocal, for example, if the vertex v_1 is connect with the vertex v_2, without the vertex v_2 being connected with v_1, then the edges are said to be direct and are called arcs.

DEFINITION 1.7. (DIRECTED AND UNDIRECTED GRAPHS)– *Let G = (V, E) be a graph.*

If for each edge $(v_1, v_2) \in E$, *we have* $(v_2, v_1) \in E$, *then the graph is undirected and the elements E are called edges of the graph. In that case, an edge is equally denoted by:* $(v_1, v_2) = (v_2, v_1) = e \in$ *with* v_1 *and* v_2 *in V .*

In the opposite case, the graph is said to be directed and the elements of E are called arcs of th graph.

DEFINITION 1.8. (ADJACENT VERTICES AND EDGES, INCIDENT EDGES)– *Let G = (V, E) be an undirecte graph.*

Let $e = (v_1, v_2) \in E$ *be an edge. The vertices* v_1 *and* v_2 *are said to be the adjacent vertices of the edg e. Similarly, e is said to be incident to the vertices* v_1 *and* v_2.

Let e_1 *and* e_2 *be two edges.* e_1 *and* e_2 *are said to be adjacent if they are incident to the same verte i.e.* $e_1 = (v, v_1)$ *and* $e_2 = (v, v_2)$.

DEFINITION 1.9. (DEGREE OF A VERTEX)– *In an undirected graph G = (V, E), the degree of a vertex v ∈ is the number of edges incident to the vertex :*

[1.3] $deg(v) = |\{(v, v') \in E, v' \in V\}|$

DEFINITION 1.10. (LOOP AND MULTIPLE EDGES)– A*n edge is called a loop if both of its extremities a identical.*

If two or more edges are incident to the same two vertices, then the edge is said to be multiple a these two edges are parallel. In that case, the multiplicity of an edge is the number of its parall edges, including itself.

DEFINITION 1.11. (SIMPLE GRAPH)– *An undirected graph is said to be simple if it has neither loop n multiple edges.*

DEFINITION 1.12. (REGULAR GRAPH)– *An undirected graph is said to be regular when all its vertic*

have the same number of neighbors.

When the associations between vertices are quantified, the edges of the graph have a weight, and th graph is said to be weighted.

DEFINITION 1.13. (WEIGHTED GRAPH)– *Let $G = (V, E)$ be a graph. The graph is said to be weighted if real value, $w(e)$, is associated with each edge $e \in E$. The value $w(e)$ is called the weight of e. I extension, a pair of vertices $(v_1, v_2) \in V^2$ such that $(v_1, v_2) \notin E$ is considered to have a zero weigh $w(v_1, v_2) = 0$.*

The weight of a subset X of edges of E is the sum of the weights of the elements of X : $w(X) = \sum_{e \in}$ $w(e)$.

Likewise, we often associate with the vertices $v \in V$ of the graph to a strictly positive integer calle the weight of v and denoted by $w(v)$.

Note: In the case of graph partitioning, the weight of an edge or vertex is most often a strict positive integer.

We must be careful not to confuse the cardinality of a set of vertices (respectively edges) and th weight of a set of vertices (respectively edges). They are equal if the vertices (respectively edges) a not weighted.

In the case of weighted graphs, we redefine the degree of a vertex $v \in V$ as the sum of the weights the edges adjacent to this vertex. Thus:

$$deg(v) = \sum_{(v,v') \in E} w(v, v') \quad [1.4]$$

For an unweighted graph, we can associate with each edge, and each vertex a unit weight.

For any graph $G = (V, E)$, we can define a bijection u: $V \rightarrow \{1, \ldots, n_V\}$ which associates an integ with every element of V . The function u can count the elements of V . For convenience, with eac element of the set of vertices V of a graph G, we will associate a rank in V , and we will merge th vertex v_i of V with its rank i in V . For example, for each $(v_i, v_j) \in E$, $w(v_i, v_j)$ will be denoted by $w($ $j)$, and $w(v_i)$ by $w(i)$.

For any graph, we can define a unique function, *weight:* $E \rightarrow \mathbb{N}$, which associates each edge with i weight. This *weight* function can be represented as a matrix. The adjacency matrix of a graph G can b defined as the matrix that associates each edge of G with its weight.

DEFINITION 1.14. (ADJACENCY MATRIX) – *Let $G = (V, E)$ be a simple graph. The matrix M_{Adj} ha dimension $n_V \times n_V$, such that $\forall (i, j) \in \{1, \ldots, n_V\}2$:*

$$(M_{Adj})_{ij} = \begin{cases} 0 & if\ i = j \\ w(i,j) & else \end{cases} \quad [[1.5]$$

is called the adjacency matrix *of the graph G.*

DEFINITION 1.15. (DEGREE MATRIX)– *Let $G = (V, E)$ be a simple graph. The matrix M_{Deg} has dimensio $n_V \times n_V$, such that $\forall (i, j) \in \{1, \ldots, n_V\}^2$:*

$$(M_{Adj})_{ij} = \begin{cases} 0 & if\ i = j \\ w(i,j) & else \end{cases} \quad [[1.6]$$

is called the degree matrix of the graph G.

DEFINITION 1.16. (LAPLACIAN MATRIX)– *Let $G = (V, E)$ be a simple graph. The matrix*

$$(M_{Deg})_{ij} = \begin{cases} deg(i) = \sum_{k=1}^{nv} w(i,k) & if\ i = j \\ 0 & else \end{cases} \quad [[1.7]$$

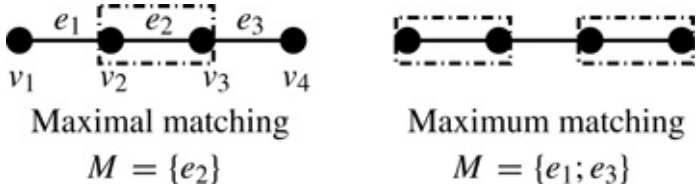
is called the Laplacian matrix of G.

DEFINITION 1.17. (PATH IN A GRAPH)– *Let G = (V, E) be a graph and* (v_1, v_2) *be two distinct vertices. If succession of edges* (*or correctly directed arcs in the case of a directed graph*) *are allowed to reach from* v_1*, then we can say that there is a* path *from* v_1 *to* v_2.

DEFINITION 1.18. (CONNECTED GRAPH)– *Let G = (V, E) be a graph. This graph is said to be* connected *for any pair of vertices* v_1 *and* v_2 *a path from* v_1 *to* v_2 *exists.*

The matching of a graph is a subset of edges of this graph which have no incident vertex in commo

Figure 1.2. *Difference between a maximal matching and a maximum matching*



DEFINITION 1.19. (MATCHING)– *Let G = (V, E) be a graph. The matching M of the graph G is a set pairwise non-adjacent edges.*

A matching M is maximal if every edge of the graph is incident with at least one edge of M.

A matching is maximum when it contains the highest possible number of edges.

Note: Every *maximum* matching is also a *maximal* matching. Let M be a *maximum* matching ar assume that there is a set of edges M' of E, not in M , such that $M \cup M'$ is a *maximal* matching. The M is not a *maximum* matching, as there is a matching, $M \cup M'$ that contains more edges. That absurd, therefore M is a *maximal* matching. The reciprocal is not necessarily true: in a linear grap with four vertices, the matching made up of the edge formed by the two central vertices is *maxim* but not *maximum* (see Figure 1.2). A graph can have several *maximum* matchings (and a *fortio* several *maximal* matchings).

1.4. Formal description of the graph partitioning problem

Before presenting the general problem of graph partitioning, we have to define the partition of graph. As mentioned in the previous section, a graph is a pair formed by a set of vertices and a set edges. It is therefore possible to partition, in a mathematical sense, the set of vertices as well as the s of edges. However, although some problems seek to partition the edges of a graph [HOL 81], grap partitioning is mostly understood as the partition of the vertices of the graph.

DEFINITION 1.20. (PARTITION OF THE VERTICES OF A GRAPH)– *Let G = (V, E) be a graph and* $P_k = \{V_1, \ldots V_k\}$ *a set of k subsets of V . P_k is said to be a partition of G if:*

– no element of P_k *is empty:*

$\forall i \in \{1, \ldots, k\}, S_i \neq \emptyset;$

$\forall i \in \{1, \ldots, k\}, S_i \neq \emptyset;$

– the elements of P_k *are pairwise disjoint:*

$\forall (i, j) \in \{1, \ldots, k\}^2, i \neq j, V_i \cap V_j = \emptyset;$

– the union of all the elements of P_k *is equal to V :*

$\bigcup_{i=1}^{k} V_i = V$

The elements V_i of P_k are called the parts of the partition. The number k is called the cardinality o the partition, or the number of parts of the partition.

The definition of a partition of the vertices of a graph does not restrict itself to the size of the part To remedy this and measure the size of the parts of a partition, the concept of a partition *balance* introduced. The term balance refers to the *measure of the weight*.

DEFINITION 1.21. (BALANCE OF A PARTITION)– *Let $G = (V, E)$ be a graph and $P_k = \{V_1, \ldots, V_k\}$ partition of this graph into k parts. The average weight of a part V_i of P_k is*

$$weight_{avg} = \left\lceil \frac{w(V)}{k} \right\rceil$$

where $\lceil x \rceil \in \mathbb{N}$ *is the first integer which is greater than or equal to x.*

The balance bal(P_k) of the partition P_k is the division of P_k 's maximal weighted element by th average weight of a part:

$$bal(P_k) = \frac{\max_i w(V_i)}{weight_{avg}} \quad [1.8]$$

Thus, a partition whose partitioning balance is unitary has parts of identical weights (rounded b one). For example, the bisection of a graph of size $(2n + 1)$ will have a balance of 1.00, if and only the largest part is of size $n + 1$ and the other part is of size n.

As we have previously seen, the graph partitioning problem is a combinatorial optimizatio problem. To present it simply, the general graph partitioning problem consists in finding a partition o a graph which satisfies a predicate p and minimizes an objective function f . Here a more form definition is provided:

DEFINITION 1.22. (GENERAL GRAPH *k*-PARTITIONING OPTIMIZATION PROBLEM)– *Let G = (V, E) be a grap and let $k \geq 2$ be an integer. We can define the triplet (S, p, f) that characterizes the general graph k partitioning optimization problem such that:*

– the research space S is defined as the set of all possible partitions of V (these partitions have cardinality that ranges from 1 to $|V|$;

– let p' be a predicate defined by $P \in V$, $p'(P)$ = true if and only if $|P| = k$. The predicate p of th general problem is defined in S such that $p(P)$ = true $\Rightarrow p'(P)$ = true, thus it can be more restrictiv than p';

– let $f : S \rightarrow \mathbb{R}$ be an objective function.

The set of admissible solutions of the problem is defined by:

$$S_a = \{P \in S \text{ such that } p(P) = true\}$$

The general graph partitioning problem consists in finding the partition $\tilde{P}_k \in S_a$ that minimizes f:

$$f(\tilde{P}_k) = \min_{P_k \in S_a} f(P_k) \quad [1.9]$$

It can be noticed that $S_a \subseteq \{P| \in S$ *such that* $|P| = k\}$. This means that an admissible solution of graph partitioning optimization problem can be more restrictive than merely satisfying the predica p' (i.e. $|P| = k$). That is why we have introduced the predicate p and not just the predicate p'.

By definition, the set of vertices of the graph is finite. The number of partitions of a set of elements is called the Bell number relative to n, and is denoted by B_n. It represents the number distinct equivalence relations on a set of n elements. $B_0 = 1$. The Bell number satisfies the followin recursive equation:

$$B_{n+1} = \sum_{k=0}^{n} C_n^k B_k \quad [1.10]$$

where $C_n^k = \frac{n!}{k!(n-k)!}$ is the binomial coefficient of n and k, i.e. the number of parts of k elements from a s of n elements.

The number of partitions into k parts of a set of n elements is called the Stirling number of t second kind. This number is denoted by $S_{n,k}$ and its value is:

$$S_{n,k} = \sum_{i=0}^{k} (-1)^i \frac{(k-i)^n}{i!\,(k-i)!} \quad [1.11]$$

There is a relation between the Bell number and the Stirling number of the second kind:

$$B_n = \sum_{i=1}^{n} S_{n,i} \quad [1.12]$$

Thus, because the set of vertices is finite, the set of solutions S is a finite set. It is therefore discrete set, according to the definition of a combinatorial optimization problem (see section 1.2).

The general graph partitioning optimization problem is defined by using a pair formed by predicate and an objective function: (p, f). Thus, graph partitioning can be divided into numero problems, depending on the nature of p and f . However, these problems can be classified into tw main different categories, depending on the nature of the predicate p: the constrained grap partitioning problem introduced in section 1.6 and the unconstrained graph partitioning proble described in section 1.7.

1.5. Objective functions for graph partitioning

The different objective functions for graph partitioning are based on two concepts: inter-cla disparities and intra-class similarities. The cut between parts often represents inter-class disparit and the weight of parts is used to characterize intra-class similarities.

DEFINITION 1.23. (CUT FUNCTION)– *Let $G = (V, E)$ be a graph. Let V_a and V_b be two subsets of V , a we can define the cut between these two subsets as:*

$$cut(V_a, V_b) = \sum_{u \in V_a} \sum_{v \in V_b} w(u, v) \quad [1.13]$$

For the rest of this section, let $P_k = \{V_1, \ldots, V_k\}$ be a partition of V into k parts. The objecti functions presented will be defined with regard to this partition.

The simplest of the objective functions used in graph partitioning is called the cut of a partition. aims to minimize the sum of the weight of the edges between parts of the partition P_k . It is alread used by B. Kernighan and S. Lin [KER 70]:

DEFINITION 1.24. (CUT OBJECTIVE FUNCTION)

$$cut(P_k) = \sum_{i<j} cut(V_i, V_j) = \frac{1}{2} \sum_{i=1}^{k} cut(V_i, V - V_i) \quad [1.14]$$

The objective function that aims to minimize, for each part, the ratio between its cut and its weigh is called ratio cut. It was introduced by Y.-C. Wei and C.-K. Cheng [WEI 89]:

DEFINITION 1.25. (RATIO CUT OBJECTIVE FUNCTION)

[1.15] $$ratio(P_k) = \sum_{i=1}^{k} \frac{cut(V_i, V - V_i)}{w(V_i)}$$

However, when the term ratio cut is used, we have to be careful that other objective functions a also called by the same name in the literature [WAN 03].

The third and last function widely used in graph partitioning is called normalized cut. This functio which is the most recent, was presented by J. Chi and J. Malik [SHI 00]. It aims to minimize, for ea part, the ratio between its cut and the sum of the weight of the edges adjacent to at least one of i vertices. In other words, it aims to minimize, for each part, the ratio between the sum of the weight the edges adjacent to exactly one of its vertices and the sum of the weight of the edges adjacent to least one of its vertices:

DEFINITION 1.26. (NORMALIZED CUT OBJECTIVE FUNCTION)

[1.16] $$norm(P_k) = \sum_{i=1}^{k} \frac{cut(V_i, V - V_i)}{cut(V_i, V)} = \sum_{i=1}^{k} 1 - \frac{cut(V_i, V_i)}{cut(V_i, V)}$$

The normalized cut function can isolate regions of the graph where the vertices are strongly inte connected. Its authors created it to replace the ratio cut in their image segmentation problem.

Note: In the case of graphs weighted by strictly positive weights, which is almost always the case graph partitioning problems, all the objective functions presented before are positive.

All these objective functions can define the *energy* of a part of a partition. The term energy is us in reference to the concept of energetic landscape presented in section 1.2. This energy is the valu brought by the part of the partition to the objective function. In the case of the cut objective functio the energy of a part V_i will be:

[1.17] $$energy_{cc}(V_i) = cut(V_i, V - V_i)$$

In the case of the ratio cut, the energy of a part V_i will be:

[1.18] $$energy_{rc}(V_i) = \frac{cut(V_i, V - V_i)}{w(V_i)}$$

In the case of the normalized cut, the energy of a part V_i will be:

[1.19] $$energy_{cn}(V_i) = \frac{cut(V_i, V - V_i)}{cut(V_i, V)}$$

1.6. Constrained graph partitioning

Classic literature on graph partitioning focuses on the constrained graph partitioning optimizatio problem. Most graph partitioning tools solve this problem, such as METIS, JOSTLE, CHACO, SCOTCH PARTY, as we will see in Chapter 2 and in the Appendix.

One of the first chapters to present this problem was from B. Kernighan and S. Lin [KER 70]. Th constrained graph partitioning problem is often used to illustrate new graph partitioning method Indeed, in the chapter introducing the spectral method [POT 90], A. Pothen, H. Simon and K.-P. Lio applied this new method to this problem. Moreover, the chapters of S. Barnard and H. Simo introducing the multilevel method [BAR 93, BAR 94] again applied this new method to a constraine graph partitioning problem.

Constrained graph partitioning is mainly used to solve different engineering problems like hig performance computing, linear systems resolution, meshing, and VLSI circuits conception. Howeve criticism has been raised regarding the use of this approach to model these problems, mo particularly the case for [HEN 98].

In literature, constrained graph partitioning is often called *multi-way graph partitioning* [YAR 0 or *k-way graph partitioning* [KAR 98].

The problem of constrained graph partitioning consists of finding a partition into k parts, whic should imperatively respect a strict balance while minimizing an objective function f.

The objective function, minimized by constrained partitioning, is most often the *cut* objecti function (see definition 1.24).

Let's consider the other objective functions. If a partition P_k has parts of identical weights, then i ratio cut (section 1.5) is equal to its cut, up to a multiplicative constant. In fact, in that case:

$$\begin{aligned} ratio(P_k) &= \sum_{i=1}^{k} \frac{cut(V_i, V - V_i)}{w(V_i)} \\ &= \frac{k}{w(V)} \sum_{i=1}^{k} cut(V_i, V - V_i) \\ &= \frac{k}{w(V)} cut(P_k) \end{aligned} \quad [1.20]$$

Thus, for the problem of constrained graph partitioning, minimizing the ratio cut is equivalent minimizing the cut.

Besides, when considering a partition P_k whose parts have identical weight, the use of th normalized cut loses its meaning. Indeed, this cost function has been created to isolate from oth regions those whose vertices are connected, which is often contradictory to the search for equal part Therefore, to the best of our knowledge, this cost function is never used in problems of constraine partitioning.

DEFINITION 1.27. (CONSTRAINED GRAPH PARTITIONING)– *Let $G = (V, E)$ be a graph, k a number of part and bal_max a maximal partitioning balance. Let S be the solution space, defined by the general grap partitioning problem. The set S_a of admissible solutions of the problem is defined by:*

$$S_a = \{P_i \in S \text{ such that } |P_i| = k \text{ and } bal(P_i) \leq bal\_max\}$$

The constrained graph partitioning *problem consists of finding $\bar{x} \in S_a$ that minimizes the co function f:*

$$f(\bar{x}) = \min_{x \in S_a} f(x) \quad [1.21]$$

Notes:

– the maximal partitioning balance of constrained graph partitioning problems is often very sma *bal_max* ∈ [1, 0; 1, 05];

– within constrained graph partitioning problems, a small relaxation of the partitioning balance often accepted. Indeed, with a value of the balance slightly greater than the unit, partitions of low cut can be found, and such solutions are often technically not detrimental to the resolution of th problem [SIM 97].

1.7. Unconstrained graph partitioning

The unconstrained graph partitioning optimization problem is slightly more exotic than th constrained graph partitioning optimization problem. We used the term exotic because this problem divided into very similar problems and its objective functions have been regularly improved, or least modified. Finally the main aim of constrained graph partitioning consists of finding a partitio respecting a restrictive balance, the main aim of unconstrained graph partitioning is to minimize a objective function. Moreover, if there is no other aim for unconstrained graph partitioning, there is second one for constrained graph partitioning, which is to minimize the cost function.

Unconstrained graph partitioning is quite similar, in its formulation, to data partitioning *clustering*. The data partitioning problem consists of merging similar elements into sets that are distinct as possible. To compare two elements, the notion of distance is used. That brings the da partitioning problem down to the creation of subsets of elements as distinct from each other possible. And yet, we can easily create a graph whose vertices are these elements and whose edg have for weight the inverse of the distance between two elements. In the case where the number parts is fixed, a data partitioning problem is similar to an unconstrained graph partitioning proble As a result, many studies have sought to solve data partitioning problems using graph partitionin tools. However, graphs modeled in this way are often complete, because there is a distance for eac pair of elements, and so, all the vertices of the graph will be connected.

Unconstrained graph partitioning is widely used not only in image segmentation[SHI 98b, SHI 98 PER 98, MEI 00, SHI 00, GDA 01, WAN 01, MAR 04, BEN 05], but also in texts classification [DH 01, ZHA 01, DIN 01, ZHA 04, BIC 09]. In the case of image segmentation, numerous chapte compare different objective functions and suggest new functions including new criteria [SHI 98b, SH 00, PER 98, GDA 01, WAN 01, FEL 04]. Several chapters solve VLSI circuit partitioning problems b using an unconstrained graph partitioning approach [HAG 92, RIE 94, ALP 95]. However, in that cas VLSI circuits are more often considered as hypergraphs than graphs [KER 70]. Other more unusu problems, such as airspace partitioning (see Chapter 9), are also unconstrained graph partitionin problems.

The objective functions that an unconstrained graph partitioning problem seeks to minimize a various. Section 1.5 presents the ratio cut and the normalized cut. These two objective functions, wi several similar variants, are those that occur the most within unconstrained graph partitionin problems. In some cases, and to some extent, we can consider, for unconstrained graph partitionin that the constraint on the partitioning balance is integrated with the objective function. Thus, unlik the constrained case, most problems omit the constraint on the partitioning balance and only enuncia the objective function to minimize. The constraint on the partition balance will therefore not b included in the definition of unconstrained graph partitioning, but may be added afterward:

DEFINITION 1.28. (UNCONSTRAINED GRAPH PARTITIONING)– *Let $G = (V, E)$ be a graph and k a number parts. Let S be the solution space, defined by the general graph partitioning problem. The set S_a admissible solutions of the problem is defined by:*

$$S_a = \{P_i \in S \text{ such that } |P_i| = k\}$$

The problem of constrained graph partitioning consists of finding a partition $\tilde{x} \in S_a$ that minimizes th cost function f (as the normalized cut or the ratio cut):

[1.22] $$f(\tilde{x}) = \min_{x \in S_a} f(x)$$

As the set of admissible solutions in the constrained case is much smaller than in the unconstraine case, finding a partition that satisfies the latter is easier. In constraint case, an algorithm that deviat from the required balance will struggle to find a partition that verifies it afterward. It is therefo

difficult to start a search outside the admissible solution space in order to get back to it. Thus, it ma seem easier to find a good solution with a smooth balance than a strong one. However, this is not th case. Indeed, if it is easier to find an admissible solution in the unconstrained case, this is because th admissible solution space is bigger. This does not mean that the admissible solution space is easier t explore, and that it is easier to seek a good solution. Besides, a local minimum in the constrained ca is not necessarily close to a local minimum in the unconstrained case.

To a certain extent and when a maximal balance is imposed, unconstrained graph partitioning can b considered as a multiobjective optimization problem that consists of minimizing one of the objectiv functions of section 1.5 and minimizing the balance of the partition [RUM 02, GIL 06, SEL 03, SE 06]. To overcome the shortcomings of the classic objective functions and to integrate other partiall conflicting objectives, other chapters have considered the graph partitioning problem as multiobjective problem [SCH 99].

1.8. Differences between constrained and unconstrained partitioning

In the previous sections, we have presented the two main graph partitioning optimization proble categories: the constrained graph partitioning problem and the unconstrained graph partitionin problem. However, although these two problems have a similar formulation, algorithms used to solv them are generally quite different, if not in their principles then in their implementations and purpos In fact, at the risk of oversimplifying, these two problems primarily differ in their priorities:

– the first purpose of the **constrained graph partitioning** problem is to find a partition with parts identical weights. Its second purpose is to reduce the cut of the partition;

– the **unconstrained graph partitioning** problem aims to maximize the cut inside parts whi minimizing the cut between parts. The principle adopted is to avoid extremely different parts size except in the case of an extremely low cut between parts. Thus, the first purpose of unconstraine graph partitioning is to minimize the cut, and a second, but not obligatory purpose, is to balance th parts.

If we take an efficient algorithm designed for one of these two problems and then we use it to solv the other problem, then we are very likely to get poor results. Indeed, reversing the order of prioriti between these two problems leads to poor results [BIC 06, BIC 07].

To the best of our knowledge, we deplore the absence of chapters in the literature, which address and exhaustively compares these two problems. However, unconstrained graph partitioning problem are numerous and efforts have been made by Inderjit S. Dhillon to synthesize them [DHI 04, DHI 07]

1.9. From bisection to *k*-partitioning: he recursive bisection method

Many graph partitioning algorithms only focus on the case of graph bisection [RON 05, MAR 0 MAR 06, CHA 07]. In this section we will present two ways of creating *k*-partitions from a grap bisection algorithm. The method used will depend on the characteristics of the bisection algorithm.

When it is possible to precisely adjust the partitioning balance, the recursive bisection technique ca adapt a bisection algorithm to k-partitioning. However, when it is not possible to adjust th partitioning balance, the recursive bisection technique can only create 2^l-partitions (with l an integer where the partitions with the number of parts are a power of 2.

We draw the reader's attention to the fact that, despite all the interest in being able to adap bisection methods to *k*-partitioning, it was shown that the recursive bisection method could n explore, and therefore find, some solutions to the *k*-partitioning problem [SIM 97]. These solution can only be found using direct *k*-partitioning methods, hence their interest. However, in practic recursive bisection methods are generally at least as efficient as the direct *k*-partitioning methods.

Initially we will study the case of a recursive bisection algorithm that can create 2^l-partitions fro an ordinary bisection algorithm. Secondly, we will study the case of a bisection algorithm that ca create a bisection that is strongly unbalanced, which means that both parts have significantly differe weights. We will see that this bisection algorithm can be used recursively to create a *k*-partition.

1.9.1. *Creating a partition with a number of parts a power of 2, from a graph bisection algorithm*

In this section, we present a simple and efficient algorithm that allows us to create a 2^l-partitio (with l an integer), from an ordinary bisection algorithm. In practice, only strongly balanced bisectio algorithms are used within this algorithm, those allowing a configurable partition balance rather u the algorithm presented in the next section.

Algorithm 1.1 describes the method of adapting a *bisection* algorithm named bisection to 2 partitioning. It uses the recursive function Iterate that takes as parameters the part V' of the graph partition, the number of parts k' into which we will partition V', and the rank num of the first part the partition of V' in the final partition P_k .