Chapter 6
Hybrid Metaheuristics for the Graph Partitioning Problem

Una Benlic and Jin-Kao Hao

Abstract. The Graph Partitioning Problem (GPP) is one of the most studied NP-complete problems notable for its broad spectrum of applicability such as in VLSI design, data mining, image segmentation, etc. Due to its high computational complexity, a large number of approximate approaches have been reported in the literature. Hybrid algorithms that are based on adaptations of popular metaheuristic techniques have shown to provide outstanding performance in terms of partition quality. In particular, it is the hybrids between well-known metaheuristics and multilevel strategies that report partitions of the minimal cut-size value. However, metaheuristic hybrids generally require more computing time than those based on greedy heuristics which can generate partitions of acceptable quality in a matter of seconds even for very large graphs. This chapter is dedicated to a review on some representative hybrid metaheuristic approaches including genetic local search, basic multilevel search and recent development on hybrid multilevel search.

6.1 Introduction

The Graph Partitioning Problem (GPP) is one of the fundamental combinatorial optimization problems which is notable for its applicability to a wide range of domains, such as VLSI design [1, 43], data mining [49], image segmentation [42], etc. Since the general GPP is NP-complete, approximate methods constitute a natural and useful approach to address this problem. In the past several decades, many efforts have been made in devising a number of heuristic approaches such as graph growing and greedy algorithms, spectral methods, multilevel approaches, as well as algorithms based on well-known metaheuristics like tabu search, ant colony, simulated annealing, genetic and memetic algorithms. However, the application of these partitioning algorithms depends on several factors. An important factor is the trade-off between

Una Benlic · Jin-Kao Hao
LERIA, University of Angers, 2 Bd Lavoisier, 49045 Angers Cedex 01, France
e-mail: `{benlic,hao}@info.univ-angers.fr`

computation time and solution quality. Some algorithms run fast but deliver solutions of medium quality while others require significantly longer time but produce excellent quality partitions and even other that can be tuned between both extremes. This preference of time vs. quality is problem dependent. For instance, in the context of network layout or VLSI design, even a slight improvement of partition quality can be of significant importance. For these applications, it is worthwhile to employ a partition algorithm able to obtain excellent quality solutions even if the algorithm is computationally intensive. On the other hand, in other cases like sparse matrix-vector multiplication, a very fast algorithm is indispensable since the computing time required for the partitioning task has to be less than the time needed by a fast vector multiplication algorithm. Another factor that has to be considered when designing an appropriate partitioning algorithm is the partition balance. While some applications require partitions of perfect balance, others tolerate imbalance up to a certain degree in order to obtain a partition of better cut-size. All of these imply that there is no single best algorithm for all the cases, and that each one of them has its applications.

Hybrid metaheuristic approaches for GPP have shown to provide excellent performance in terms of solution quality. In particular, it is the hybrids between classical metaheuristics and multilevel methods that report partitions of the minimal cut-size value. However, these hybrids are generally more time consuming than those based on greedy iterative methods, which can produce partitions even for very large graphs in a matter of seconds.

This chapter is devoted to an overview of the most popular and effective hybrid metaheuristic approaches proposed in the literature for the k-way graph partitioning problem. We first provide a general definition of the graph partitioning problem and an overview of the benchmark instances, followed by a brief review on the most common heuristic approaches for GPP. In Section 6.4, we first describe the Kernighan-Lin (KL) heuristic and its improvement by Fiduccia and Mattheyses, and then review some of the most popular hybrids between KL-like algorithms and population-based methods. Before concluding, the multillevel paradigm for GP is detailed in Section 6.5 and some of the best multilevel metaheuristic hybrids are presented. Moreover, we try to provide an answer, based on a landscape analysis performed on a number of GP instances, to what makes these algorithms so effective.

6.2 Problem Definition and Bechmark Instances

6.2.1 Problem Description and Notations

The nature of a partitioning problem can greatly vary depending on the intended application. In this chapter we only focus on the partitioning approaches devised to tackle the general multi-way graph partitioning problem (also called k-way partitioning).

Consider an undirected graph $G = (V,E)$ consisting of a set of vertices (i.e. nodes) $V = \{v_1, v_2, ..., v_n\}$ and a set of edges E, such that each vertex and edge is associated with a non-negative weight that we denote by $w(v)$ and $w(e)$ respectively. Then, a k-partition of G can then be defined as a mapping (partition function) $\pi : V \rightarrow \{1,2,...,k\}$ that distributes the vertices of V among k disjoint subsets $S_1 \cup S_2 \cup ... \cup S_k = V$. The particular partitioning case when k is set to two is known as the Graph Bisection Problem (GBP).

Let $\{S_1, S_2, ..., S_k\}$ be a partition of V obtained by π, E^c the set of all the cutting edges of G induced by π, i.e., $E^c = \{\{x,y\} \in E \mid x \in S_i \text{ and } y \in S_j \text{ and } i \neq j\}$, and let φ be the set of all the partition functions of G. The k-way graph partitioning problem consists in determining $\pi^* \in \varphi$ such that the partition $\{S_1, S_2, ..., S_k\}$ given by π^* minimizes the sum of weight of edges in E^c, while ensuring that each S_i, $i \in \{1,2,...,k\}$ is of roughly equal weight. Here, the weight of a subset S_i is equal to the sum of weights of the vertices in S_i, $W(S_i) = \sum_{v \in S_i} w(v)$.

For some applications, perfect partition balance is required. On the other hand, some applications tolerate partition imbalance up to a certain limit, since allowing more imbalance may lead to partitions of better quality in terms of the total weight of edges in the cut. This notion of partition balance is defined as follows. Let $W_{opt} = \lceil |V|/k \rceil$ be the optimal subset weight, where $\lceil x \rceil$ represents the first integer $\geq x$, then the quantity $\varepsilon = max_{i \in \{1..k\}} W(S_i)/W_{opt}$ defines the degree of imbalance among the k subsets of a partition $\{S_1, S_2, ..., S_k\}$. $\varepsilon = 1$ means that the partition is perfectly balanced while $\varepsilon > 1$ indicates an imbalanced partition with larger ε corresponding to greater imbalance.

6.2.2 Benchmark Instances

In the context of various existing research studies on the graph partitioning problem defined in Section 6.2.1, a large number of benchmark instances have been used.

A well known source of graph instances that has been frequently used to compare and evaluate algorithm performance is proposed by Johnson et al. [25] and by Bui and Moon [14]. These benchmark graphs are classified into five following types where the first two classes are from [25]:

1. *Gn.d*: A random graph with n vertices, where an edge e is placed between any two vertices with probability p, such that p is chosen so that the expected vertex degree, $p(n-1)$, is d.
2. *Un.d*: A random geometric graph with n vertices uniformly distributed in a unit square. An edge is placed between two vertices if their Euclidean distance is less than or equal to $\sqrt{d/(n\pi)}$, where d is the expected vertex degree.
3. *breg.n.b*: A random regular graph with n vertices of degree 3, whose optimal bisection size is b with probability $1 - o(1)$.

4. *cat.n*: A caterpillar graph with n vertices. Starting with a spine in which every vertex is of degree 2 (except the two ending vertices), each vertex of the spine is connected to six new vertices. Given an even number of vertices in the spine, the optimal bisection size is 1. Another group of caterpillar graphs are denoted with *rcat.n* where each vertex on the spine is connected to $\sqrt{n}$ new vertices.
5. *grid.n.b*: A grid graph with n vertices whose optimal bisection size is known to be b. The same grid graph but with the boundaries wrapped around is denoted by *w-grid.n.b*.

The graph instances proposed in [25] and [14] are of rather small size (up to 1000 and 5252 vertices respectively), and as such do not represent a real challenge for recent algorithms that are designed to tackle large partitioning problems steaming from real-life applications. Another important source of GP instances is provided by the Walshaw's Graph Partitioning Archive (`http://staffweb.cms.gre.ac.uk/~c.walshaw/partition/`). These benchmark graphs represent samples of small to medium scale problems arising in different applications. Compared to the graphs provided in [25] and [14], these are of significantly larger dimensions with the biggest graph *auto* comprising 448695 vertices and 3314611 edges.

Since circuit partitioning is one of the most important applications of the GPP, the ISPD circuit benchmark suites are also used to evaluate the performance of partitioning algorithms. The circuit benchmark suite is regularly being updated with new circuits that are directly derived from real industrial designs, and that represent today's mixed-size physical design constraints in terms of size and complexity. The most recent circuit instances are presented at the ISPD-2011 placement contest [46]. These instances are large with the total number of nodes ranging from 483452 to 1293433 and the total number of nets from 468918 to 1293436. Their format can easily be converted into the standard format used by the current state-of-art k-way graph partitioning packages.

6.3 Classical Approaches for the Graph Partitioning Problem

Many different approaches have been proposed in the literature for the GPP. Some of these algorithms only take a local view of the graph and try to ameliorate the given partition, while others consider the problem globally. Some are purely deterministic always producing the same partition, while others rely on random decisions. Some operate on the graph itself, while others use some mathematical representations of it. Some are very time consuming, while others can find a partition even of very large graphs in a matter of seconds. In this section, we provide a brief review of the most common heuristic approaches applied to GPP.

Greedy graph partitioning methods are quite simple. The basic idea of these deterministic approaches is to accumulate in some way vertices into subsets, one subset at a time or alternating between subsets. Battiti and Bertossi propose two popular greedy procedures for graph bisection, the Min-Max-Greedy [7] and the Diff-Greedy [6], that assign one randomly chosen seed vertex to each bisection subset and subsequently add vertices to them alternatively. While the Min-Max-Greedy [7] method adds each time a vertex that will produce the smallest increase in the cut size of the partition, the vertex selection criterion is slightly modified in the Diff-Greedy [6] to choose vertices that have the minimum difference between the number of new external and internal edges. Another kind of greedy partitioning algorithm is proposed by Ciarlet and Lamour [15]. All of these algorithms are very fast, and have an additional advantage of being able to directly divide a graph into the desired number of subsets. This avoids applying recursive bisection that can give arbitrarily worse results than direct k-way partitioning [41]. On the other hand, the quality of partitions in terms of cut-size is not always great. Therefore, greedy partitioning algorithms are often used to generate an initial partition which is then refined with a local improvement algorithm.

The Kernighan-Lin (KL) algorithm [29] is one of the earliest and most popular local improvement heuristic for graph partitioning. It improves iteratively the quality of an existing partition obtained by other partitioning approaches. Originally, the KL procedure was intended to be applied several times starting from a different random partition. While this produces reasonable results on small graphs, it is quite ineffective on larger problem instances. Nowadays, the KL heuristic is used to complement algorithms that have a more global view of the problem but are likely to ignore local characteristics. Numerous improvements and adaptations of the basic KL procedure have been proposed in the literature. The most important improvement of the KL algorithm for graph bisection is the one proposed by Fiduccia and Mattheyses [21], which reduces the time per KL pass to linear. Both KL and Fiduccia-Mattheyses (FM) algorithms are devised to tackle only the graph bisection problem. Several adaptations of the KL and FM procedures have been proposed in the literature for the k-way partitioning. Among these is an extended FM algorithm by Sanchis [38, 39], and an adaptation of FM proposed by Hendrickson and Leland [23]. A description of the KL procedure and its modification by Fiduccia and Mattheyses is provided in Section 6.4.1.

More sophisticated vertex move based approaches for graph partitioning rely on well-known metaheuristics such as tabu search, simulated annealing, ant colony algorithms and evolutionary approaches. Moreover, Chardaire et al. [15] apply to the partitioning problem the Population Reinforced Optimization Based Exploration (PROBE) heuristic, which has been presented as a new metaheuristic [3] inspired by genetic algorithms. Although adaptations of metaheuristic algorithms are generally more time consuming than greedy iterative methods, they often yield an improvement on solution quality. In particular, the best performing approaches for the graph partitioning problem are often hybrids between a classical metaheuristic and a multilevel method. Indeed, a great number of the current best balanced partitions for the set of benchmark graphs from Walshaw's Graph Partitioning Archive are obtained with two multilevel hybrid approaches proposed by Benlic and Hao, based on an iterative tabu search algorithm [9] and a memetic algorithm [10, 11] respectively. Section 6.4 and 6.5 review respectively some of the most effective hybrid evolutionary approaches and multilevel metaheuristic hybrids.

Spectral heuristic approaches have also been extensively used. An advantage of these approaches is the possibility of defining lower bounds for the objective function of the partitioning problem. Spectral methods are based on computing eigenvalues and eigenvectors of the Laplacian matrix associated with the graph in order to construct various geometric representations of the graph. A fundamental spectral algorithm is the spectral bipartitioning (SB) based on the linear ordering representation. The SB uses the second eigenvector of the Laplacian (called the Fiedler vector), which contains important directional information about the graph G. The components of the Fiedler vector are weights associated to vertices of G, such that the differences between the components provide information about the distances between the vertices. A bisection with SB is obtained by sorting the vertices according to the sizes of the components of the Fiedler vector, and then distributing half of the vertices to each subset of the bisection. Spectral algorithms for k-way graph partitioning can be classified according to two approaches: recursive spectral bisection algorithm and direct spectral k-way partitioning algorithm. The former consists in finding the Fiedler eigenvector of a Laplacian matrix of graph G, and recursively partitioning G until the desired number of partitions is obtained. The latter uses $p \geq k$ eigenvectors and directly partitions G into k subsets with some heuristic. For a recent survey on algorithms based on these two approaches see [34]. Since spectral partitioning is a computationally intensive process, the first papers on multilevel methods for graph partitioning [4] propose multilevel implementations of spectral algorithms to simplify the calculation for a spectral method. Spectral methods are global approaches for partitioning graphs. Therefore, it is useful to improve the obtained partition with a local optimization algorithm. These are often called partition refinement algorithms. The algorithms for refinement of partitions found by spectral methods are most often of Kernighan-Lin type.

To handle very large graphs, multilevel algorithms prove to be quite useful. Various adaptations of the general multilevel technique have been tried on a number of combinatorial optimization problems including the traveling salesman, graph coloring, and the vehicle routing problem [47]. For graph partitioning, the multilevel approach has been very successful. The multilevel method was initially proposed to accelerate the performance of existing partitioning approaches. However, it was shortly recognized to be extremely effective and to have a more global vision of a graph than standard refinement procedures. The multilevel approach thus imposed itself as a global strategy using local partitioning algorithms. The basic idea of a multilevel graph partitioning approach is to successively create a sequence of progressively smaller graphs by grouping vertices into clusters. A partition of the coarsest graph is generated and then successively projected back towards the original graph followed by partition refinement. Hybrid algorithms that combine a multilevel method with a metaheuristic approach shortly became very popular for solving the partitioning problem. Section 6.5 is dedicated to the multilevel schemes for partitioning as well as to the best metaheuristic algorithms for multilevel partition refinement.

Although the current partitioning approaches are able to produce high quality partitions in reasonable time, performing partitioning in parallel is very important and has received a lot of attention. A great deal of work has been focused on parallelizing geometric graph partitioning and spectral bisection algorithms. Parallel formulations of multilevel graph partitioning schemes have also been proposed in the literature, although their development is quite challenging. Moreover, parallel versions of the three well known partition packages Jostle [48], Metis [40] and Scotch [36], based on the multilevel paradigm, have also been developed. Perhaps the fastest available parallel code is the parallel version of Metis (parMatis). It produces partitions which are worse that those obtained with the sequential version of Metis (kMetis). In general, parallelization of graph partitioning algorithms induces some penalty in terms of solution quality. However, Holtgrewe et al. [24] demonstrate in their recent work that high quality graph partitioning can be obtained in parallel in a scalable way. Moreover, their parallelization approach even seems to ameliorate partition quality, and in some cases improves the best-known partitions reported in the literature.

6.4 Evolutionary Hybrids for Graph Partitioning

Many hybrid evolutionary algorithms have been proposed in the literature for the graph partitioning problem. The success of these approaches lies in combining advantages of both recombination operator that discovers unexplored promising regions of the search space, and local search that finds good solutions by concentrating the search around these regions. Most of the popular population-based graph partitioning algorithms use the well-known Fiduccia-Mattheyses (FM) improvement of the Kernighan-Lin (KL) algorithm (or some slight modification of it) for fast iterative local improvement of partitions created in the recombination process. Before reviewing the current state-of-art hybrid population-based approaches, we thus describe the KL heuristic and its FM modification.

6.4.1 Kernighan-Lin Bisection Algorithm, Improvement and Adaptation

The Kenighan-Lin (KL) heuristic [29] improves upon a given initial bisection by exchanging two equal-size vertex subsets of the bisection. Let (A,B) be a graph bisection, i.e., $A \cup B = V$ and $A \cap B = \emptyset$. We denote by $g(a,b)$ the reduction in the cut size when two vertices $a \in A$ and $b \in B$ exchange their subsets, and by $g(v)$ the reduction when vertex v is moved to the opposite subset. The gain $g(a,b)$ can then be computed as

$$g(a,b) = g_a + g_b - 2\delta(a,b),$$

where

$$\delta(a,b) = \begin{cases} 1 & \text{if } (a,b) \in E \\ 0 & \text{otherwise.} \end{cases}$$

The KL algorithm selects a pair of vertices (a,b) which maximizes gain $g(a,b)$. Once a and b are selected, they are not considered any more for further exchange. This process is repeated to form a sequence of pairs $(a_1,b_1),...,(a_{n/2-1},b_{n/2-1})$. The algorithm then exchanges vertices in $X = \{a_1,...,a_k\}$ from one bisection subset with vertices in $Y = \{b_1,...,b_k\}$ from another bisection subset, such that $\sum_{i=1}^{k} g(a_i,b_i)$ is maximized. The above constitutes one KL pass of complexity $O(n^3)$. This process is repeated until no improvement on the bisection is possible.

To reduce the total running time per pass, Kernighan and Lin [29] suggest considering only several highest gain vertices in each subset and then selecting the pair with the maximum gain among all the combinations. This reduces the running time per pass to $O(n^2)$ and introduces only a slight degradation in solution quality.

Fiduccia and Mattheyses [21] modify the KL bisection heuristic by suggesting to move one vertex at a time instead of exchanging two vertices. Moreover, the authors propose an effective bucket data structure that reduces the time per pass to linear $O(|E|)$ by avoiding unnecessary search for the highest gain vertex and by minimizing the time needed for updating the gains of vertices affected by each move. The idea of the bucket data structure consists in placing all vertices with the same gain g in a bucket that is ranked g. Finding a vertex with the maximum gain simply consists in finding the non-empty bucket with the highest rank, and selecting a vertex from the bucket. After a vertex v has been moved to another subset, the bucket structure is updated by recomputing gains of vertex v and its neighbours, and transferring these vertices to appropriate buckets.

The bucket data structure consists of two arrays of buckets, one for each subset of a bisection, where each bucket of an array is represented by a doubly linked list. The arrays are indexed by the possible gain values for a move, ranging from g_{max} to g_{min}. A special pointer *maxgain* points to the highest index in the array whose bucket is not empty. The structure also keeps an additional array of vertices where each element (vertex) points to its corresponding vertex in the doubly linked lists. This enables direct access to the vertices in buckets and their transfer from one bucket to another in constant time. An example of the bucket data structure is illustrated in Fig 6.1.

Both KL and FM heuristics are devised only for the Graph Bisection Problem (GBP). Several adaptations of the FM algorithm have been proposed for the k-way partitioning. In [38], Sanchis proposes maintaining $k(k-1)$ previously described bucket structures, one for each of the $k(k-1)$ possible directions to move a cell (i.e., vertex) between partition subsets. The author also adopts the notion of level gain [31] that enables the algorithm to better distinguish between cells whose first level gains (regular gains) are the same. However, Sanchis suggests a more space efficient way to maintain level gains than that proposed in [31]. Moreover, making k^2 comparisons to determine the next legal cell move (i.e., move which preserves partition balance) is avoided by keeping a sorted list of the *maxgain* pointers corresponding to legal move directions. This is done by using a binary heap whose entries are *maxgain*

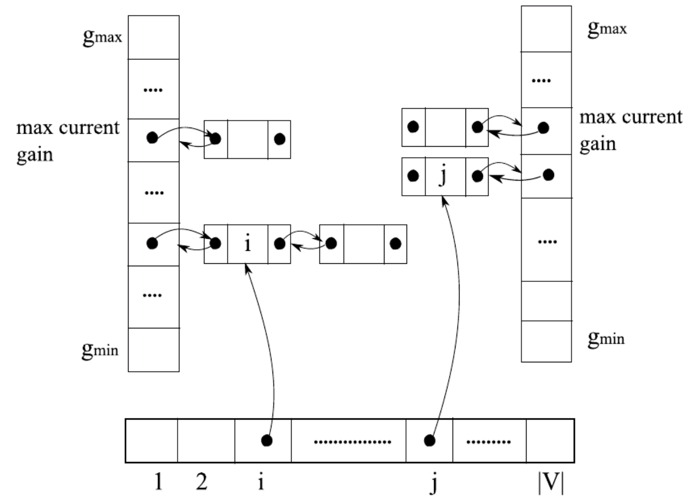

Fig. 6.1 The bucket sorting data structure [21] for graph bisection

pointers for the currently legal move directions. The pointer with the highest value is located at the root of the tree, and an array indexed by move directions is maintained holding pointers to the elements in the heap. The complexity involved in maintaining the heap is $O(lk|E|logk)$, where l is the number of gain levels.

6.4.2 Selected Evolutionary Approaches for Graph Partitioning

6.4.2.1 Hybrid Genetic Algorithm by Bui and Moon

A classical approach for GBP is the Breadth First Search Genetic Bisection Algorithm (BFS-GBA), proposed by Bui and Moon [14], that can easily be adapted for the k-way partitioning problem .

The approach uses a standard solution encoding that represents a partition as a bit string (or integer string in case of k-way partitioning), where the i^{th} element in the string indicates the partition subset of vertex i. Although this solution representation is widely used for the GPP, it is not the most suitable because of the high redundancy which grows exponentially with the number of partition subsets k. That is, each solution can be represented in $k!$ possible ways which deteriorates the performance of traditional crossovers by introducing severe inconsistencies in an offspring partition.

In each iteration, BFS-GBA picks from the population two parents for recombination, such that the probability of selecting the best individual for recombination is four times as high as the probability of selecting the worst parent. Two offspring partitions are then created by recombining the selected parents using the standard multi-point (MP) crossover and a variation of MP (call it VMP) respectively. The VMP crossover is the same as MP except that it copies to the child the complement values from the second parent. Therefore, it can only be applied in the case of bisection. BFS-GBA selects the better of the two offspring solutions and passes it to a local optimization procedure. The motivation for using the two crossovers is the following. If two partitions are exactly (or almost exactly) the complement of each other, the MP crossover will cause much stronger perturbation in an offspring partition than the VMP, probably resulting a partition of poor quality. The algorithm reinforces the diversification by applying a mutation to offspring I^0 that selects at random m positions in the chromosome and changes their values. This generally leads to an infeasible solution which is repaired by using a simple scheme of flipping bits.

For fast improvement of offspring, Bui and Moon propose a variation of Fiduccia–Mattheyses linear time KL implementation (see Section 6.4.1). Only one pass of the local optimizer is allowed, and the size of the sets to be swapped is restricted. This decreases even more the computation time of the local serach phase by about an order of 10. In case of the k-way partitioning, the KL extension [29] for the k-way partitioning is used.

Finally, BFS-GBA applies a replacement strategy that first tries to replace I^0 with the more similar parent based on the Hamming distance measure, and if it fails, it tries to replace the other parent (replacement is carried out only when I^0 is better than one of the parents). Although this scheme preserves longer a diversified population, it is very time consuming since with large diversity the algorithm takes more time to converge. For this reason, the BFS-GBA scheme replaces offspring I^0 with the worst individual from the population in case when I^0 is worse than both of the parents.

The algorithm stops when 80% of the population is occupied by solutions of the same quality, which are not necessarily identical solutions.

An important component of this hybrid genetic algorithm is the preprocessing phase that can dramatically improve the performance on some types of graphs (geometric and caterpillar graphs) at very little cost in time. The rational behind this preprocessing scheme is to reorder vertices on the chromosome in an attempt to ensure that clusters of highly connected vertices are included in short schemas that have more chance to survive in a crossover. It consists in performing a breadth first search (BFS) on the input graph starting at a random vertex. The order in which vertices are visited by the BFS is used to reorder vertices on the chromosome. That is, the i^{th} vertex in the BFS ordering takes the position i in the chromosome. This preprocessing phase is carried out only once before the start of the hybrid genetic algorithm.

The performance of BFS-GBA was tested on the set of graphs from [25], as well as on a number of instances specially designed for this evaluation [14]. In this study, BFS-GBA competes very favourably with the multi-start KL algorithm and the simulated annealing proposed by Johnson et al [25]. Moreover, it is considerably faster then the simulated annealing approach [25]. Unfortunately, the performance of BFS-GBA has not been demonstrated on larger instances ($|V| > 10000$).

6.4.2.2 A Memetic Algorithm by Merz and Freisleben

Merz and Freisleben [32] propose a memetic algorithm for GBP, which is based on the observations made from an exhaustive landscape analysis on a set of local optima sampled with the KL and greedy heuristics respectively for the instances proposed in [14, 25]. The results of this analysis showed that the landscape of the GBP is highly dependent on the graph structure and that some landscapes have slightly more correlated local optima than others. However, all these analysed graphs generally have a rather structured landscape (fitness-distance correlation coefficient $\rho_{fdc} > 0.15$), and local optima that are concentrated within a limited region of the search space (see Section 6.5.4.1). Therefore, the authors propose a memetic algorithm (MA) that attempts to exploit the landscape structure of the problem.

The algorithm uses the same solution encoding as BFS-GBA. For the local search phase, MA employs the standard KL algorithm which runs in $O(|E|)$ instead of $O(n^2)$ time by means of the bucket data structure proposed by Fidducia and Mattheyses.

Instead of generating the initial population randomly, the algorithm uses the randomized Diff-Greedy heuristic by Battiti and Bertossi [6] since it is one of the best constructive heuristics for the GBP and is able to generate a wide range of high quality solutions. The idea of the Diff-Greedy algorithm consists in generating a partition by adding vertices alternatively to partition subsets in a greedy way. Let S^0 and S^1 be two subsets of the bisection. At each stage, the vertex selected to enter a subset, say S^0, is the vertex for which the number of neighbour vertices in S^0 minus the number of neighbour vertices in S^1 is maximized. The rationale behind this selection criterion is that a bisection that minimizes the cut size maximizes at the same time the number of internal edges.

The Diff-Greedy heuristic thus exploits the structure of the search space that has shown to be very effective for instances of the GPP. The authors therefore propose a new crossover called greedy crossover (GC) which is based on the same idea as the Diff-Greedy heuristic. In the first phase of the GC, all vertices that are contained in the same partition subset in both parents are placed in the same subset in the offspring. Then, the rest of vertices is assigned to both subsets according to the selection strategy used in the Diff-Greedy algorithm. If $|S_0| < |S_1|$ a vertex is added to subset S_0, else to S_1.

Selection for recombination in MA is done uniformly at random without bias to fitter individuals, while selection for survival is performed by choosing the best individuals from the union of parents and children by taking care that there is no duplicate in the population.

Due to the computing time required by the local search phase, the population size is kept very small (up to 40) compared to genetic algorithms. This leads to premature convergence, especially in the absence of mutation. To overcome this problem, MA triggers a restart mechanism that has shown to be very effective for combinatorial optimization problems, including the QAP and the TSP. Upon convergence (the average Hamming distance has dropped below a threshold (d = 10) or there was no changes in the population for more than 30 generations), the whole population except the best individual is mutated by exchanging subsets of randomly chosen pairs of vertices (v_1, v_2) such that $v_1 \in S$ and $v_2 \notin S$. After mutation, each individual is improved with the KL local search and MA proceeds with performing the crossover as usual.

This memetic algorithm shows to be effective, scalable and very robust on different types of graphs, and is able to produce better average cut size than any previous heuristic search method including tabu search, simulated annealing, and hybrid genetic algorithms.

6.4.2.3 A PROBE Based Heuristic by Chardaire et al.

In [15], the authors propose an adaptation of a new population-based metaheuristic technique named PROBE (Population Reinforced Optimization Based Exploration) for the GBP. The PROBE method is conceptually much simpler than genetic algorithms as it does not include selection, replacement, and mutation procedures. The basic idea of PROBE is to find optimized solutions by exploring promising search subspaces, starting from solutions in which common characteristics found in both parents are preserved. These optimized solutions then constitute a new population in the next generation.

As in [14], the PROBE bisection algorithm (PROBE-BA) uses the bit string solution encoding where the i^{th} bit indicates the subset of vertex i. Although this scheme is quite intuitive, the authors note that a less redundant encoding might improve the partition quality.

PROBE-BA uses standard graph partitioning approaches for exploration and exploitation of the search space, namely the Diff-Greedy heuristic by Battiti and Bertossi [6] (see Section 6.4.2.2) and the variation of the KL algorithm by Bui and Moon [14] (see Section 6.4.2.1) . Given a population $POP = \{I_q^1, I_q^2, ..., I_q^{|POP|}\}$ of feasible solutions at generation q, the next generation of solutions is obtained as follows. For each $i = 1, ..., |POP|$, a partial bisection I_{q+1}^i is computed from the pairs (I_q^i, I_q^{i+1}) (where the superscript is taken modulo $POP + 1$) by fixing the vertices corresponding to the bits shared by the two parent solutions I_q^i and I_q^{i+1}. This partial solution is then used as input to the Diff-Greedy algorithm to obtain a complete bisection of the given graph. Note that this recombination process is exactly the same as with the previously described greedy crossover devised by Merz and Freisleben [32], which tries to exploit the landscape structure. Once solution I_{q+1}^i has been constructed, its quality is improved with the fast KL local optimizer designed by Bui and Moon [14].

The performance of PROBE-BA has extensively been evaluated on a large number of graphs of different sizes (the largest graph *auto* has $|V| = 448695$ and $|E| = 331461$). The results show that PROBE-BA can compete with other population-based algorithms, reactive tabu search, or more specialized multilevel partitioning approaches. Moreover, it was able to improve, in reasonable time, the previous best cut values for a number of real world instances.

6.5 Multilevel Graph Partitioning

As illustrated in [47], the multilevel paradigm is a useful approach to solving combinatorial optimization problems that even appears to impart a 'global' quality to local search heuristics. Basically, the approach allows one to approximate the initial problem by approximating successively smaller (and easier) problems. Moreover, the coarsening helps filter the solution space by placing restrictions on which solutions the refinement algorithm can visit. We dedicate this section to some of the best performing and most popular multilevel hybrids that combine a refinement algorithm based on a metaheuristic approach. After a formal definition of the general multilevel procedure, we provide a review on two basic types of multilevel schemes and on the most effective adaptations of metaheuristic techniques that have been proposed for partition refinement of coarsened graphs.

6.5.1 Formal Definition of the Multilevel Paradigm

Let $G_0 = (V_0, E_0)$ be the initial graph, and let k denote the number of partition subsets. The multilevel paradigm can be summarized by the following steps.

1. Coarsening phase: The initial graph G_0 is transformed into a sequence of smaller graphs $G_1, G_2, ..., G_m$ such that $|V_0| > |V_1| > |V_2| > ... > |V_m|$. Each coarse graph represents the original problem, but with fewer degrees of freedom. Coarsening stops when $|V_m|$ reaches a fixed threshold (coarsening threshold).
2. Initial partitioning phase: A k-partition P_m of the coarsest graph $G_m = (V_m, E_m)$ is generated. It allows to get the first approximation of the problem.
3. Uncoarsening phase: Partition P_m is progressively projected back to each intermediate G_i $(i = m-1, m-2, ..., 0)$. Before each projection, the partition is first refined (improved) by a refinement algorithm.

This process leads thus to a sequence of partitions $P_m, P_{m-1}, P_{m-2}, ...P_0$. The last one, i.e., P_0 is returned as the final partition of the original graph G_0.

In Figure 6.2, we illustrate this multilevel procedure for the 4-way partitioning problem.

6.5.2 Multilevel Schemes

Two main classes of multilevel schemes have been proposed in the graph partitioning literature. In general, any coarsening can be defined as a process of aggregation of graph vertices to form the vertices of the next coarser graph. In the following, we describe the strict aggregation scheme, as well as the more effective weighted aggregation which has recently been proposed for the partition problem. While the former scheme makes hardened local decisions at each graph level, the latter

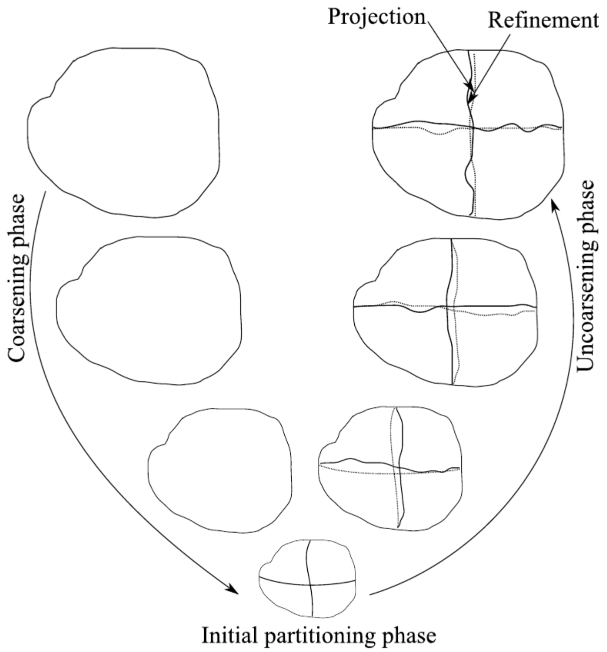

Fig. 6.2 The coarsening, initial partitioning and the uncoarsening (projection/refinement) phases for the 4-way partitioning problem. During the uncoarsening phase, the dashed lines represent projected partitions and the dark ones indicate partitions refined after projection.

introduces more freedom in solving the coarser levels and avoids making local decision before gathering the pertinent global information.

6.5.2.1 Strict Aggregation Scheme (SAG)

Strict aggregation (SAG) [11, 23, 28, 44], also called edge contraction or matching of vertices, is employed by most multilevel partitioning algorithms. The idea of the SAG is to form a new vertex $v \in V_{i+1}$ of a coarser graph G_{i+1} by merging a subset of vertices $V_i^c \subset V_i$ of G_i that (usually) have a strong local coupling (i.e., connectivity).

The weights of the resulting vertices and edges of the coarsened graph $G_{i+1} = (V_{i+1}, E_{i+1})$ are set accordingly. The weight of the new vertex $v \in G_{i+1}$ becomes equal to the sum of weights of the vertices that are aggregated to form v. Similarly, let $v_a, v_b \in V_{i+1}$ be two vertices formed by collapsing $\{v_1, v_2, v_3\} \in V_i^a$ and $\{v_4, v_5, v_6\} \in V_i^b$. All the edges incident to $\{v_1, v_2, v_3\}$ and $\{v_4, v_5, v_6\}$ are merged to form a new edge $\{v_a, v_b\} \in E_{i+1}$ with a weight that is set equal to the sum of weights of the edges incident to $\{v_1, v_2, v_3\}$ and $\{v_4, v_5, v_6\}$. Updating of vertex and edge weights is illustrated in Fig 6.3. For simplicity, the cardinality of the vertex subset that is merged to form a new vertex of a coarser graph is set to two.

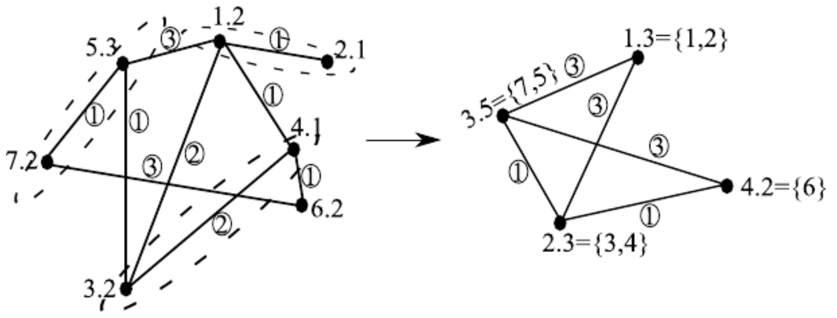

Fig. 6.3 An example of updating weights of new vertices and edges formed after vertex aggregation. For each vertex v, we indicate its corresponding weight w separated by a dot (e.g., $v.w$). For each newly formed vertex v_n, we also indicate the subset of vertices that are aggregated to form v_n.

Two main SAG schemes have been proposed for coarsening graphs. The first scheme is based on finding an independent subset of edges (matching) $\Gamma \subset E_i$, and then collapsing the two vertices of each edge in Γ to form a new vertex of a coarsened graph $G = (V_{i+1}, E_{i+1})$ [11, 27, 28, 44]. Any vertex that is not part of Γ is simply copied over to G_{i+1}. The second scheme is based on finding c-cliques for $c > 2$, and then collapsing the vertices of the cliques to form new vertices of the coarsened graph [19, 22]. We concentrate on the matching scheme, since it is the most commonly used coarsening method.

The key issue for the matching scheme is the selection of the independent subset of graph edges Γ to be collapsed at each step of the coarsening phase. This can be achieved by finding a maximal matching of the graph [35]. That is, the objective is to find the maximal number of edges no two which are incident on the same vertex. There exist polynomial time algorithms for tackling this problem, with running time of at least $O(|V|^{2.5})$. Unfortunately, this is too slow to be applicable to the partitioning problem. Therefore, several fast heuristic approaches have been proposed to compute an approximate maximal matching such as the heavy-edge matching heuristic (HEM), which has $O(|E|)$ time complexity [27]. The HEM considers vertices in random order, matching each unmatched vertex v with its unmatched neighbor u, if any, such that the weight of edge $\{u, v\}$ is maximal among all the edges incident to v. Other similar heuristics for computing an approximate maximal matching of a graph include Random Matching (RM), Light Edge Matching (LEM), and Heavy Clique Matching (HCM). If time is not a issue, metaheuristic algorithms might be used to approximate maximal matchings of a graph at the expense of running time. In [30], the authors propose a genetic algorithm to determine an approximate maximal matching that has shown to provide a significant improvement on solution quality.

The projection in case of the SAG is a trivial process. If a vertex $v \in V_i$ is in subset S_m, then the matched pair of vertices $v_1, v_2 \in V_{i-1}$ which represents vertex $v \in V_i$ will also be in subset S_m.

6.5.2.2 Weighted Aggregation Scheme (WAG)

Another class of multilevel schemes that has been applied to several combinatorial optimization problems [37] (including the graph partitioning [17]) is based on the algebraic multigrid (AMG) method [12, 13]. The essential difference between this and the previously described SAG scheme is in the coarsening phase. While the SAG is based on grouping of vertices into small disjoint subsets, the AMG coarsening is a weighted aggregation where each vertex is divided in fractions, and different fractions belong to different subsets of vertices (i.e., aggregates). That is, the vertices that belong to more than one vertex subset will be divided among the corresponding aggregates. Like with the SAG scheme, all the vertices from a subset are merged to form a vertex of the coarser level, where they will be blocked into larger aggregates, forming vertices of a still coarser graph, and so on. Weighted instead of strict aggregations is important to express a likelihood of vertices to belong to the same subset. These likelihoods are accumulated at the coarser levels, indicating tendencies of vertices to be associated together. In that way, weighted coarsening avoids making hardened local decisions, such as edge contractions, before gathering important global information.

The WAG coarsening scheme for graph partitioning first starts by selecting a set $C \subset V_i$ of seed vertices of the finer graph G_i, that will constitute the vertices of the coarser graph G_{i+1}. This process is guided by the principal that each vertex from $F = V \backslash C$ should have a strong dominant connection to C. Then, starting from $C = \emptyset$ and $F = V$, vertices are being transferred from F to C until all the remaining vertices from F satisfy the following condition:

$$\sum_{j \in C} w_{ij} / \sum_{j \in V} w_{ij} \geq Q,$$

where Q is a parameter (usually $Q \approx 0.5$).

Each vertex from C becomes a seed of one aggregate that will form one vertex of the coarser graph. Define for each $i \in F$ a coarse neighborhood $N_i = \{j \in C, w_{ij} \geq \alpha_i\}$, where α_i is a parameter that limits the nighborhood size in order to control complexity. The AMG interpolation matrix is then defined by

$$P_{ij} = \begin{cases} w_{ij} / \sum_{k \in N_i} w_{ik} & \text{for } i \in F, j \in N_i \\ 1 & \text{for } i \in C, j = i \\ 0 & \text{otherwise,} \end{cases}$$

where each entry P_{ij} of the matrix represents the likelihood of vertex i to belong to the j^{th} aggregate. Let $I(k)$ be the order number in the coarser graph G_{i+1} of a vertex that constitutes an aggregate around a seed vertex whose order number in the finer graph G_i is k. The weight of an edge that connects two aggregates $p = I(i)$ and $q = I(j)$ in G_{i+1} becomes equal to $w_{pq} = \sum_{k \neq l} P_{ki} w_{kl} P_{lj}$. As in the case of the SAG scheme, the total sum of vertex weights is conserved throughout each graph level.

The projection process of the WAG scheme is more complex than in the case of the SAG. One of the simpler ways to project a partition consists in computing the probability that a fine vertex belongs to a particular subset of the partition. In the case of a graph bisection, only the probability of being part of subset 0 (or 1) matters. Then, the probability that vertex i belongs to subset 0 can be determined with the following relation:

$$P_0(i) = \sum_{k \in N, I(k) \in S_0} P_{iI(k)}.$$

With this strategy, vertex i is assigned to subset 0 if the probability $P_0(i) \geq 0.5$, and to subset 1 otherwise.

Although the SAG scheme has been used by most multilevel partitioning approaches, experimental evaluations have shown that the usage of WAG can significantly improve the quality of a partition. For a comparison between the two types of schemes for the graph partition problem, the reader is referred to a recent paper by Chevalier and Safro [17].

6.5.3 Effective Refinement Strategies Based on Metaheuristics

Partition refinement approaches that are used in conjunction with the multilevel paradigm are most often based on the KL linear-time complexity improvement by Fiduccia and Mattheyses. The KL heuristic has shown to be efficient in finding locally optimal partitions when it starts with a fine initial partition. Since the projected partition is already of fairly good quality, the KL considerably decreases the cut-size within a small number of iterations [27]. For that reason, four out of five public-domain graph partitioning packages (Chaco, Jostle, Metis, and Scotch), whose aim is to find reasonably good partitions in very short computing time, use a multilevel KL hybrid as the default setting. However, it has been shown that, given a longer computing time, other multilevel refinement algorithms that are based on well-known metaheuristic techniques, such as tabu search or evolutionary approaches, are able to largely improve on the solution quality in terms of cut-size.

We next review some of the best performing metaheuristic refinement approaches for the k-way partitioning. Note that each refinement procedure of a multilevel strategy can solely be applied to solve the partitioning problem. The multilevel paradigm is integrated since it can often either accelerate the convergence of the local serach or even improve the asymptotic convergence in solution quality [47].

6.5.3.1 Perturbation-Based Tabu Search by Benlic and Hao

A recently proposed perturbation-based iterated tabu search (ITS) procedure [9], combined with the multilevel matching scheme (see Section 6.5.2.1), has shown to be extremely effective in finding balanced ($\varepsilon = 1.0$) k-way partitions. Perfect partition balance for this algorithm is thus imposed as a constraint and is progressively established during the search, while the objective is to minimize the total sum of cutting edge weights.

The ITS employs two neighborhood relations (call them N_1 and N_2) which are explored in a token-ring way. That is, one neighborhood search is repeatedly applied to the best local optimum produced by the other neighborhood. Given a subset S_i of a k-partition $p = \{S_1, S_2, ..., S_k\}$, the basic idea of the neighborhood relations N_1 and N_2 is to move a vertex v from its current subset to subset S_i, under the constraint that v must be a border vertex relative to S_i, i.e., $v \notin S_i$ has at least one adjacent vertex in S_i. Note that in this way, the size of the neighborhoods is limited, since the set of border vertices relative to S_i is generally of small size. In addition, such a neighborhood allows the search to concentrate around these critical vertices.

Let $I = \{S_1, S_2, ..., S_k\}$ be a k-partition, $V(S_i)$ the set of border vertices relative to subset S_i, and $S_{max} = \{S_i | max_{i \in \{1..k\}} \{W(S_i)\}\}$ the subset with the maximum vertex weight. The neighborhood relations N_1 and N_2 can be explained by the two move operators given below.

Move 1: Move one highest gain vertex v_m. Choose randomly a subset $S_m \in \{S_1, S_2, ..., S_k\} - \{S_{max}\}$. Then, select the *highest gain* vertex $v_m \in V(S_m)$ whose current subset is S_c, such that $S_c \in \{S \in I | W(S) > W(S_m)\}$. Move the selected vertex v_m to subset S_m.

Move 2: Move two highest gain vertices v_m and v_n. Choose vertex v_m and its new subset S_m as with the first move operator. Choose randomly a new subset $S_n \in \{S_1, S_2, ..., S_k\} - \{S_{max}, S_m\}$. Then, select vertex $v_n \in V(S_n)$ whose current subset is S_c, such that $S_c \in \{S \in I | S \neq S_n\}$. Move v_m to S_m, and v_n to S_n.

As defined in Section 6.4.1, the gain for moving vertex v to subset S_m is the reduction in the cut size. The selection of the vertex with the highest gain, as well as the updates needed after each move, are achieved efficiently by using a new adaptation of Fiduccia-Mattheyses bucket sorting [21] for the k-way partitioning that maintains k arrays of buckets, one for each partition subset.

It is important to note that these move operators progressively lead the search toward a balanced partition since they basically constraint (partially with Move 2) vertex migration from heavy weight subsets to light weight subsets.

Let $V_{cand} \subset V(S_m)$ be the set of the highest gain vertices which are considered for migration to subset S_m. The selection of vertex v, which is moved to S_m, is based on several pieces of history information. This selection strategy is first conditioned by the tabu status. It also employs two additional criteria which are based on *vertex move frequency* and *vertex weight*. The move frequency is a long term memory that records, for each vertex v, the number of times v has been moved to a different subset. It gives priority to moves that have been applied less often. If there is more than one vertex with the same move frequency in the set V_{cand}, the second criterion is used to distinguish them and prefer a vertex v which, when moved to subset S_m, minimizes the weight difference between the target subset S_m and the original subset S_c.

Each time a vertex v is moved from a subset S_c to another subset S_m, it is forbidden to move v back to its original subset S_c for the next tt iterations (tabu tenure), where tt is tunded adaptively.

The described TS procedure applies a very aggressive search procedure since it focuses only around border (critical) vertices. Therefore, to avoid getting trapped in a local optimum, the algorithm periodically triggers a simple perturbation which consists in moving a fixed number of vertices γ, including non-border ones, such that the partition balance is not degraded.

This multilevel ITS algorithm (MITS) is designed to produce excellent quality partitions with the possibility to generate solutions of various qualities depending on the amount of computing time allowed. Indeed, experimental studies on a set of graphs from Walshaw's graph partitioning archive have shown that partitions generated with MITS within short computation time (from 1 second up to several minutes for a graph with $|V| = 143437$ and $|E| = 409593$) are generally far better than those produced by the current public-domain partitioning packages. When the running time is prolonged up to one hour, the described algorithm often outperforms the existing state-of-art graph partitioning algorithms in terms of solution quality.

6.5.3.2 An Evolutionary Approach by Soper et al.

A popular approach for generating high quality graph partitions, proposed by Soper et al. [44], is a combination of an evolutionary search algorithm and a multilevel partitioner.

The employed multilevel partitioner, known as JOSTLE, is based on the matching scheme with the HEM heuristic (see Section 6.5.2.1) and the linear-time KL improvement by Fiduccia and Mattheyses that the authors extend for use with non-integer gains by integer scaling. The fitness function used by the evolutionary approach is defined to be $-f\lambda$, where f is the number of edges in the cut and λ the degree of imbalance. The partition imbalance is thus not considered as a constraint, but induces a heavy penalty in case of greater imbalance. In this way, partitions within the balance constraint eventually dominate the population as the search progresses.

The basic idea of the approach is to assign a bias (≥ 0) to each vertex, and a weight to each edge that is equal to one plus the sum of the biases of its incident vertices. When applying JOSTLE to a graph with biased vertex and edge weights, vertices with a small bias are more likely to appear as boundary vertices than those with a larger one, and edges of a lower weight have higher probability to be cut. In this way, JOSTLE concentrates its search to a rather limited region of the search space just like the ITS from Section 6.5.3.1.

Each new offspring is obtained with a crossover or a mutation operator by determining a set of biased values from one or more parents from the population and than applying JOSTLE to generate a partition. The crossover creates a new set of biases from a given number of partitions in the following way. For each vertex v in the graph, check whether v appears as a border vertex (ends a cut edge) in two or more of the parent partitions. If so, assign to v a bias value selected uniformly at random from the range [0, 0.1]. Otherwise, assign to vertex v a bias value of 0.1 plus a random number chosen in the same range.

The mutation operator generates a new set of biases by considering information from only one parent in the following way. For each vertex v in the graph, check whether v is a border vertex, the neighbour of a border vertex, or the neighbour of a neighbour of a border vertex. If so, assign to v a bias value selected uniformly at random from the range [0, 0.1]. Otherwise, assign to vertex v a bias value of 2.0 plus a random number chosen in the same range.

After an offspring partition has been created, the associated biased values are removed.

The population size $|POP|$ is kept quite small (around 50) due to size of graphs and the time required to execute JOSTLE. Each new generation is produced as follows. $|POP|$ new offspring are created by either crossover or mutation at a given ratio. Mating groups of individuals for crossover and candidates for mutation are randomly selected from the current generation, such that each individual participates in at least one trial. The union of parent and offspring individuals are ranked by fitness, and the best $|POP|$ individuals are then selected to form the new generation. The proposed algorithm is thus a simplified version of the CHC Adaptive Search Algorithm [18] that lacks incest prevention and restarts.

Each run of this evolutionary approach consist of 50,000 calls to JOSTLE, and therefore requires very long execution time of hours and even days for large graphs. As expected, it thus provides higher quality partitions than any of the existing package that usually take less than a minute (and often less than a second) to generate a partition.

6.5.3.3 The Memetic Algorithm by Benlic and Hao

In [11], Benlic and Hao extend their MITS algorithm for balanced k-way partitioning from Section 6.5.3.1 to a multilevel memetic approach (MMA) by integrating a dedicated multi-parent crossover operator based on the notion of backbone and a distance-preserving pool updating strategy that maintains a healthily diversified population. To avoid high solution redundancy introduced by the standard string solution encoding, an individual $I = \{S_1, ..., S_k\}$ for MMA corresponds to a partition of V into k disjoint groups or subsets, such that each subset S_j, $j \in \{1, ..., k\}$ is composed of vertices that are assigned to the j^{th} subset.

The success of the MMA partly lies in the dedicated backbone-based multi-parent crossover operator (BBC) that exploits an existing structure of a problem by preserving the elements which hopefully belong to the optimal partition, while permitting limited perturbations within offspring solutions. It thus provides high quality partitions for instances with exploitable global structure and search landscapes with highly correlated local optima.

Given the set $P = \{I^1, ..., I^p\}$ of p parent individuals chosen with the well-known tournament selection strategy, the BBC constructs the offspring $I^0 = \{S_1^0, ..., S_k^0\}$ in k passes (one for each subset of the partition). In each pass μ it performs the following steps:

1. Select a subset S_j^i of I^i such that the weight $W(S_j^i)$ is maximal across the subsets $j \in \{1..k\}$ of each individual $I^i \in P$, i.e. $max_{i \in \{1..p\}, j \in \{1..k\}} \{W(S_j^i)\}$, with the constraint that at most $\lceil k/p \rceil$ subsets can be chosen from each individual $I^i \in P$.
2. Given I^i and S_j^i determined in Step 1, for *each* individual $I^t \in P$ $(t \neq i)$, let $\prod_t$ contain the largest number of vertices that are shared by the subset S_j^i of I^i and a subset S_η^t of I^t, i.e. $\prod_t = \{S_j^i \cap S_\eta^t | max_{\eta \in \{1..k\}} |S_j^i \cap S_\eta^t|\}$. Then, $\prod = \{\prod_1, .., \prod_{p-1}\}$ forms a set of these vertex subsets.
3. Set $S_\mu^0 = \prod_1 \cap \prod_2 \cap ... \cap \prod_{p-1}$. S_μ^0 is the largest subset of vertices that are shared by all the parent individuals. For each vertex $v \in S_j^i$ and $v \neq S_\mu^0$, v is assigned to subset S_μ^0 of I^0 if $c(v)/p - 1$ is greater than or equal to some random real number in the range $[0, 1]$, where $c(v)$ is the number of subsets of $\prod$ in which v occurs.
4. When a vertex v is assigned to subset S_μ^0 of I^0 in the μ^{th} pass, v is removed from all the parent individual subsets in which it occurs, and the weights of these subsets are adjusted accordingly.

After the previous four steps, the last step handles the unassigned vertices. Any vertex v missing from I^0 is placed at random to a subset S_r of I^0 such that $W(S_r \cup \{v\}) \leq W_{opt}$, where W_{opt} is defined in Section 6.2.1. This step introduces a degree of diversification in the crossover process.

Notice that the proposed BBC operator never degrades the balance with respect to the set of parent individuals P, since given a subset S_j^i of individual I^i which is chosen in the μ^{th} pass, at most $|S_j^i|$ vertices can be transmitted to the subset S_μ^0 of offspring I^0. In addition, an unassigned vertex v in I^0 is assigned to a subset S_r^0 only if adding v to S_r^0 does not exceed the expected optimal subset weight W_{opt}. An example of this crossover with three parent individuals $(p = 3)$ for $k = 3$ is provided in Figure 6.4.

After offspring I^0 has been generated with the BBC operator, it is improved with the ITS from Section 6.5.3.1. The MMA then decides whether I^0 should be inserted into the population by considering both the solution quality and the set-theoretic partition distance [20] (call it d) between individuals from the population. Offspring I^0 is inserted into POP if it is of the best quality relative to the population, or if the minimum distance between I^0 and any other individual in the population is greater than the minimum distance between any two individuals in the population. To determine the individual that is to be replaced by I^0, the authors adopt a strategy proposed in [33] that uses the following quality-and-distance scoring function H to rank the individuals of the population:

$$H_{i,POP} = f(I^i) + \beta / D_{i,POP}$$

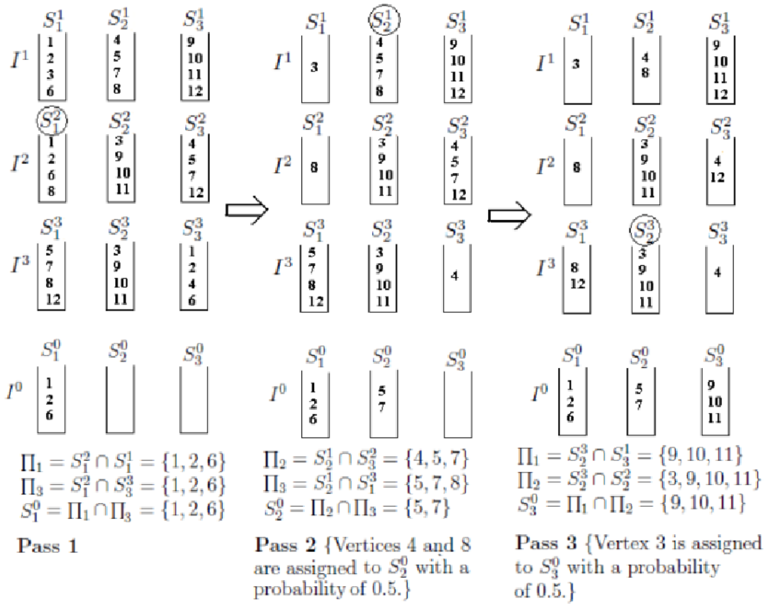
$\Pi_1 = S_1^2 \cap S_1^1 = \{1,2,6\}$
$\Pi_3 = S_1^2 \cap S_3^3 = \{1,2,6\}$
$S_1^0 = \Pi_1 \cap \Pi_3 = \{1,2,6\}$
Pass 1
$\Pi_2 = S_2^1 \cap S_3^2 = \{4,5,7\}$
$\Pi_3 = S_2^1 \cap S_1^3 = \{5,7,8\}$
$S_2^0 = \Pi_2 \cap \Pi_3 = \{5,7\}$
Pass 2 {Vertices 4 and 8 are assigned to S_2^0 with a probability of 0.5.}
$\Pi_1 = S_2^3 \cap S_3^1 = \{9,10,11\}$
$\Pi_2 = S_2^3 \cap S_2^2 = \{3,9,10,11\}$
$S_3^0 = \Pi_1 \cap \Pi_2 = \{9,10,11\}$
Pass 3 {Vertex 3 is assigned to S_3^0 with a probability of 0.5.}

Fig. 6.4 An illustration of the BBC crossover with three parents taken from [11]. A circled subset of a parent corresponds to the subset chosen in the μ^{th} pass, i.e. the subset of maximal weight across all the parent individuals with the constraint that at most $\lceil k/p \rceil$ subsets can be chosen from each individual.

where f is the objective function (i.e., the sum of cutting edge weights), β a parameter, and $D_{i,POP}$ the minimum distance between individual I^i and any other individual from the population.

An extensive experimental evaluation of MMA has been performed on a set of benchmark instances from Walshaw's archive. It has been shown that the MMA can provide even better partitions in terms of solution quality than the ITS from Section 6.5.3.1. Moreover, the authors compare two version of MMAs integrating respectively the BBC and a standard uniform crossover where the diversification is further reinforced by a random mutation operator. The results show that there is no significant statistical difference between the solution sets generated by the two MMA versions for lower values of k, i.e., $k \in \{4,8\}$. However, as k increases, the BBC operator visibly outperforms the uniform crossover in almost each case for $k \in \{16,32\}$. One explanation is that intuitively, given the semantics of the BBC crossover, it favours the preservation of backbone information for larger k whereas the number of parts has a weak influence for the uniform crossover operator as to backbone preservation.

6.5.3.4 Other Partition Refinement Approaches

Beside the aforementioned multilevel refinement partitioning procedures, some other approaches are also worth mentioning.

In [8], Battiti et al. propose a multilevel algorithm for the balanced bipartitioning which integrates the Diff-Greedy algorithm [6], used in the initial partitioning phase, and a tabu search algorithm [7] employed as a partition refinement procedure. The TS algorithm is related to the KL heuristic [29]. However, the main differences are that each selected move is applied immediately to the current solution, and that worsening moves are also accepted. The authors consider two alternative choices to adjust the prohibition parameter T (i.e., the tabu tenure) of the TS algorithm. The first choice is to maintain T fixed during the search with a value that is selected by a preliminary off-line tuning phase for different types of graphs (FIXED-TS or FTS). The other choice is to determine the right value of T in a dynamic and on-line way depending on the past search history (Reactive Randomized Tabu Search RRTS). In this way, the tedious task of tuning by the user is avoided and the T value can automatically change during the search depending on the properties of a specific task.

A refinement procedure based on a mixture of simulated annealing and tabu search algorithm (RLrMSATS) is presented by Baños et al [5]. The idea of this hybrid approach is to employ the simulated annealing procedure to escape from local optima, while preventing the occurrence of cycles by means of a tabu search mechanism. A move with the proposed approach consist in moving a vertex v from its current to another partition subset. To jump from a local optimum, RMSATS accepts worsening moves as in simulated annealing. Once a move increasing the evaluation function cost is accepted, it is forbidden to apply the reverse move during a certain number of iterations as in tabu search in order to avoid cycling. A similar hybrid refinement approach inspired by RMSATS is proposed in [45].

6.5.4 The Key to Effectiveness of Partition Refinement Procedures

The success of a multilevel algorithm is greatly dependent on its two main components: the coarsening scheme and the solution refinement procedure. Since vertex aggregation filters the solution space by putting restrictions on which solutions the algorithm can visit, it is obvious that the way coarsening decisions are made is of an extreme importance for the quality of resulting solutions. As pointed out previously, most of the graph partitioning approaches are based on the same or very similar coarsening schemes. However, given the same amount of computing time, some of these algorithms perform better than others highlighting the importance of a refinement procedure.

In the previous sections, we described the three best performing partition refinement procedures in terms of partition quality. A common characteristic of these refinement algorithms is that they are based on stronger intensification and concentrate the search only around a limited region of the search space. In ITS [9], this is done by performing (most of the time) moves with only border (critical) vertices according to a selection strategy. A similar idea is also used in [44], where a smaller bias is assigned to vertices that appear as border vertices in one or more parent partitions, which results higher probability that these vertices will remain border in successive generations. In [11], besides the ITS, the BBC crossover also limits the region of explored search space by preserving vertex groupings that are common to a number of population individuals. Moreover, the intensification plays a major role in other popular graph partitioning approaches.

We next provide an explanation, based on observation made from a landscape analysis [11, 32], to why a more pronounced intensification mechanism constitutes a highly effective search in case of the graph partitioning problem.

6.5.4.1 Landscape Analysis for the Graph Partitioning Instances

The performance of a stohastic algorithm crucially depends on the characteristics of search landscape like the average distance between local optima and the relative distance of local optima to the nearest global optimum. The fitness distance correlation (FDC) coefficient ρ_{fdc} [26] is a well-known tool for landscape analysis and can provide useful indications about the problem hardness, even if such an analysis has some known shortcomings and limits. FDC estimates how closely related are the fitness and distance to the nearest optimum. For a minimization problem, if the fitness of a solution decreases with the decrease of distance from the optimum, then it would be easy to reach the target optimum for an algorithm that concentrates around the best candidate solutions found so far, since there is a "path" to the optimum via solutions with decreasing (better) fitness. A value of $\rho_{fdc} = 1$ indicates perfect correlation between fitness and distance to the optimum. For correlation of $\rho_{fdc} = -1$, the fitness function is completely misleading. FDC can also be visualized with the FD plot, where the same data used for estimating ρ_{fdc} is displayed graphically.

A landscape analysis for the GPP has been performed in two works. In [32], Merz and Freisleben provide a thorough analysis of the landscape for the GBP on a set of instances introduced in [14, 25], and perform a fitness distance correlation analysis (FDA) [26] based on solutions samples with the KL [29] and Diff-Greedy [6] heuristics respectively. In [11], Benlic and Hao make a FDA for the k-way partitioning problem (for $k \in \{4, 8, 16, 32\}$) on a set of graphs from the Walshaw's graph partitioning archive (used for performance evaluation in [9, 11, 44]), based on a sample of local optima obtained after 1500 independent runs of the ITS [9]. While Merz and Freisleben use the Hamming distance, Benlic and Hao use the set-theoretic distance to perform the landscape analysis.

Tables 6.1 and 6.2 show the results from [32] and [11] respectively. Column 'ρ_{fdc}' of the two tables reports FDC coefficients ρ_{fdc} for the analysed graphs. For illustrative purpose, FD plots of only two graphs (*3elt* and *vibrobox*) are given in Figure 6.5 for $k \in \{4, 8, 16, 32\}$. As it can be seen from Tables 6.1 and 6.2, there is a signification fitness distance correlation in many cases. However, the FDA

Table 6.1 Analytical results for graph bisection, taken from [32], for graph partitioning instances provided in [14, 25]. Columns 'd_{lo}' and 'd_{go}' report respectively the average distance between local optima and the average distance of local optima from the best local optimum, expressed as a percentage of $|V|$. Column 'ρ_{fdc}' shows the correlation coefficients with respect to fitness and distance.

	KL heuristic			Diff-Greedy heuristic		
Graph	*avg d_{lo}*	*avg d_{go}*	ρ_{fdc}	*avg d_{lo}*	*avg d_{go}*	ρ_{fdc}
G1000.0025	22.53	21.29	0.37	22.09	20.86	0.34
G1000.005	22.79	21.97	0.22	22.36	21.68	0.18
G1000.01	22.6	21.54	0.37	22.21	21.36	0.29
G1000.02	22.47	21.24	0.47	22.2	20.86	0.41
U1000.05	22.47	21.62	0.28	14.44	11.15	0.63
U1000.10	20.65	19.16	0.36	15.39	12.81	0.58
U1000.20	15.76	12.94	0.63	14.94	13.53	0.58
U1000.40	13.02	9.45	0.82	13.6	10.47	0.66
Breg5000.16	3.95	2.08	0.99	17.9	11.98	0.99
Cat.5252	24.14	23.91	0.02	14.58	11.52	0.21
Rcat.5114	22.79	22.36	0.07	14.46	11.45	0.7
Grid5000.50	4.3	2.4	0.91	15.07	12.09	0.7
W-grid5000.100	14.43	13.56	0.66	14.43	13.55	0.7

Table 6.2 Analytical results, taken from [11], for seven graph partitioning instances from Walshaw's graph partitioning archive when $k \in \{4,8,16,32\}$. Columns 'd_{lo}' and 'd_{go}' report respectively the average distance between local optima and the average distance of local optima from the best local optimum, expressed as a percentage of $|V|$. Column 'ρ_{fdc}' shows the correlation coefficients with respect to fitness and distance.

	k=4			k=8			k=16			k=32		
Graph	*avg d_{lo}*	*avg d_{go}*	ρ_{fdc}	*avg d_{lo}*	*avg d_{go}*	ρ_{fdc}	*avg d_{lo}*	*avg d_{go}*	ρ_{fdc}	*avg d_{lo}*	*avg d_{go}*	ρ_{fdc}
data	30.5	34.8	0.57	17.8	16.0	0.68	22.5	23.7	0.08	24.9	23.1	0.6
3elt	19.1	18.7	0.7	17.2	14.6	0.53	14.0	12.1	0.75	20.5	17.1	0.53
uk	18	14.3	0.61	26.3	25.7	0.24	26.9	25.1	0.33	27.4	24.9	0.44
crack	3.5	2.2	0.89	22.5	19.6	0.51	27.7	22.9	0.74	28.1	26.3	0.58
wing-nodal	26.1	21.6	0.81	17.1	13.6	0.91	31.0	27.3	0.56	37.5	35.6	0.4
fe-4elt2	9.8	6.7	0.74	26.0	24.4	0.68	16.4	14.7	0.51	28.7	25.5	0.51
vibrobox	40.1	41.4	-0.02	22.4	19.7	0.03	41.5	45.5	0.65	49.7	46.8	0.21

analysis also reveals the existence of several cases among the selected instances for which there is virtually no correlation between fitness and distance, i.e., cases where $\rho_{fdc} < 0.15$. Indeed, from plots in Figure 6.5, it is clear that there is practically no correlation for 'vibrobox' when $k \in \{4,8\}$. On the other hand, the plots indicate the strongest correlation for the graph '3elt' when $k \in \{4,16\}$ and 'vibrobox' when $k = 16$.

The existence of a strong correlation between solution quality and its distance to the nearest global optimum, as observed from the FDC analysis in [11, 32], is often refered to as a big valley structure of the landscape. Intuitively, in this structure a global optimum is surrounded by local optima with evaluation values that deteriorate with the increase of distance to the global optimum. In case of landscapes with a big valley structure, stronger intensification leads to algorithms of better performance.

Additionally, tables 6.1 and 6.2 report the average distance between local optima (column '*avg d_{lo}*') and the average distance of local optima from the best local optimum (column '*avg d_{go}*'), expressed as a percentage of $|V|$. Given that the maximum

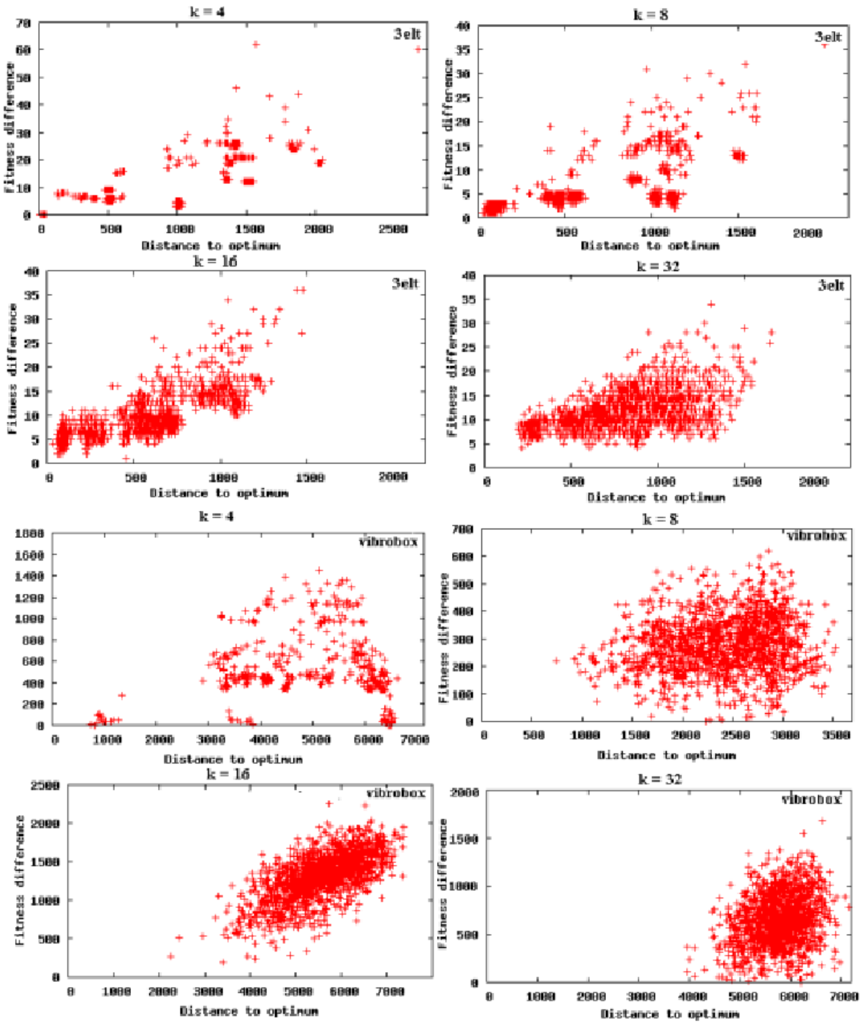

Fig. 6.5 FD correlation plots with respect to the normalized solution fitness and distance to the optimum for 3*elt* and *vibrobox* when $k \in \{4, 8, 16, 32\}$. The first four plots are related to the *elt* graph, while the last four are related to the *vibrobox*. The plots are taken from [11].

distance between any two solutions is $|V|$, these results imply that local optima are not uniformly distributed, but are rather concentrated within a limited number of regions in the search space.

These observations constitute an explanation to why algorithms that perform a stronger intesification do so well on these GP instances.

6.6 Conclusion

In this chapter, we provided a review on hybrid metaheuristics for solving the well-known *k*-way partitioning problem (GPP). GPP is an NP-complete problem with a broad spectrum of applicability. Therefore, many efforts have been made in devising a number of different heuristic approaches such as spectral methods, graph growing and greedy heuristics, multilevel approaches, as well as algorithms based on popular metaheuristics. The application of these methods depends on several factors including time vs. quality and the degree of imbalance. The most popular graph bisection heuristic is the linear time implementation of the Kernighan-Lin algorithm by Fiduccia and Mattheyses, which improves iteratively the quality of an existing partition. Different adaptations and modifications of its basic procedure have been proposed in the literature. These KL-like algorithms are often hybridized with other approaches such as multilevel and genetic algorithms. The current best performing GPP algorithms in terms of solution quality are hybrids between classical metaheuristic techniques and multilevel methods. Indeed, the three most effective algorithms reviewed in this chapter, that were able to produce state-of-art partitions, are hybrids between multilevel methods and adaptations of well-known metaheuristics. We noted that a common characteristic of these approaches is that they are based on a strong intensification mechanism which seems to work well on most GP instances whose landscapes generally display the big valley structure.

References

1. Alpert, J.C., Kahng, B.A.: Recent directions in netlist partitioning: A survey. Integration, the VLSI Journal 19(12), 1–81 (1995)
2. Alpert, J.C., Hagen, W.L., Kahng, B.A.: A hybrid multilevel/genetic approach for circuit partitioning. In: Proceedings of the IEEE Asia Pacific Conference on Circuits and Systems, pp. 298–301 (1996)
3. Barake, M., Chardaire, P., McKeown, G.P.: The PROBE metaheuristic for the multiconstraint knapsack problem. In: Resende, M.G.C., de Sousa, J.P. (eds.) Metaheuritics, pp. 19–36. Springer (2004)
4. Barnard, T.S., Simon, D.H.: A Fast Multilevel Implementation of Recursive Spectral Bisection for Partitioning Unstructured Problems. In: Proceedings of the 6th SIAM Conference on Parallel Processing for Scientific Computing, pp. 711–718 (1993)
5. Baños, R., Gil, C., Ortega, J., Montoya, F.G.: Multilevel Heuristic Algorithm for Graph Partitioning. In: Raidl, G.R., Cagnoni, S., Cardalda, J.J.R., Corne, D.W., Gottlieb, J., Guillot, A., Hart, E., Johnson, C.G., Marchiori, E., Meyer, J.-A., Middendorf, M. (eds.) EvoIASP 2003, EvoWorkshops 2003, EvoSTIM 2003, EvoROB/EvoRobot 2003, EvoCOP 2003, EvoBIO 2003, and EvoMUSART 2003. LNCS, vol. 2611, pp. 143–153. Springer, Heidelberg (2003)
6. Battiti, R., Bertossi, A.: Differential Greedy for the 0-1 Equicut Problem. In: Proceedings of the DIMACS Workshop on Network Design: Connectivity and Facilities Location, pp. 3–21 (1997)
7. Battiti, R., Bertossi, A.: Greedy, prohibition, and reactive heuristics for graph partitioning. IEEE Transactions on Computers 48(4), 361–385 (1999)

8. Battiti, R., Bertossi, A., Cappelletti, A.: Multilevel reactive tabu search for graph partitioning. Preprint UTM 554. Dip. Mat., University Trento, Italy (1999)
9. Benlic, U., Hao, J.K.: An effective multilevel tabu search approach for balanced graph partitioning. Computers and Operations Research 38(7), 1066–1075 (2010)
10. Benlic, U., Hao, J.K.: An Effective Multilevel Memetic Algorithm for Balanced Graph Partitioning. In: ICTAI, vol. (1), pp. 121–128 (2010)
11. Benlic, U., Hao, J.K.: A multilevel memetic approach for improving graph k-partitions. To appear in IEEE Transactions on Evolutionary Computation (2011)
12. Brandt, A., McCormick, S., Ruge, J.: Algebraic multigrid (AMG) for sparse matrix equations. In: Evans, D.J. (ed.) Sparsity and its Applications, pp. 257–284 (1984)
13. Brandt, A.: Algebraic multigrid theory: The symmetric case. In: Preliminary Proceedings of the International Multigrid Congerence, vol. 19, pp. 23–56 (1986)
14. Bui, T.N., Moon, B.R.: Genetic Algorithm and Graph Partitioning. IEEE Transactions on Computers 45(7), 841–855 (1996)
15. Chardaire, P., Barake, M., McKeown, G.P.: A PROBE-based heuristic for graph partitioning. IEEE Transactions on Computers 56(12), 1707–1720 (2007)
16. Ciarlet, P., Lamour, F.: On the validity of a front-oriented approach to partitioning large sparse graphs with a connectivity. Numerical Algorithms 12(1), 193–214 (1996)
17. Chevalier, C., Safro, I.: Comparison of Coarsening Schemes for Multilevel Graph Partitioning. In: Stützle, T. (ed.) LION 3. LNCS, vol. 5851, pp. 191–205. Springer, Heidelberg (2009)
18. Eshelman, L.J.: The CHC adaptive search algorithm: How to have a safe search when engaging in non-traditional genetic recombination. In: Rawlings, G.J.E. (ed.) Foundations of Genetic Algorithms, pp. 265–283 (1991)
19. Garbers, J., Prome, H.J., Steger, A.: Finding clusters in VLSI circuits. In: Proceedings of IEEE International Conference on Computer Aided Design, pp. 520–523 (1990)
20. Gusfield, D.: Partition-Distance: A Problem and Class of Perfect Graphs Arising in Clustering. Information Processing Letters 82(3), 159–164 (2002)
21. Fiduccia, C., Mattheyses, R.: A linear-time heuristics for improving network partitions. In: Proceedings of the 19th Design Automation Conference, pp. 171–185 (1982)
22. Hagen, L., Kahng, A.: A new approach to effective circuit clustering. In: Proceedings of IEEE International Conference on Computer Aided Design, pp. 422–427 (1992)
23. Hendrickson, B., Leland, R.: A multilevel algorithm for partitioning graphs. In: Proceedings of Supercomputing, CDROM (1995)
24. Holtgrewe, M., Sanders, P., Schulz, C.: Engineering a scalable high quality graph partitioner. In: Proceedings of IEEE International Parallel & and Distributed Processing Symposium, pp. 1–12 (2010)
25. Johnson, D.S., Aragon, C.R., Mcgeoch, L.A., Schevon, C.: Optimization by Simulated Annealing: An Experimental Evaluation; Part-I, Graph Partitioning. Operations Research 37, 865–892 (1989)
26. Jones, T., Forrest, S.: Fitness Distance Correlation as a Measure of Problem Difficulty for Genetic Algorithms. In: Proceedings of the 6th International Conference on Genetic Algorithms, pp. 184–192. Morgan Kaufmann (1995)
27. Karypis, G., Kumar, V.: A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. SIAM Journal on Scientific Computing 20(1), 359–392 (1998)
28. Karypis, G., Kumar, V.: Multilevel k-way Partitioning Scheme for Irregular Graphs. Journal of Parallel and Distributed Computing 48(1), 96–129 (1998)
29. Kernighan, B.W., Lin, S.: An efficient heuristic procedure for partitioning graphs. Bell System Technical Journal 49, 291–307 (1970)
30. Küçükpetek, S., Polat, F., Oğuztüzün, H.: Multilevel graph partitioning: an evolutionary approach. Journal of the Operational Research Society 56, 549–562 (2005)
31. Krishnarnurthy, B.: An Improved Min-Cut Algorithm for Partitioning VLSI Networks. IEEE Transactions on Computers 33, 438–446 (1984)
32. Merz, P., Freisleben, B.: Fitness Landscapes, Memetic Algorithms. and Greedy Operators for Graph Bipartitioning. Journal of Evolutionary Computation 8(1), 61–91 (2000)

33. Lü, Z., Hao, J.K.: A Memetic Algorithm for Graph Coloring. European Journal of Operational Research 203(1), 241–250 (2010)
34. Nascimento, M., de Carvalho, A.: Spectral methods for graph clustering: A survey. European Journal of Operational Research 211(2011), 221–231 (2010)
35. Papadimitriou, C., Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity. Prentice-Hall (1982)
36. Pellegrini, F.: Scotch home page, `http://www.labri.fr/pelegrin/scotch`
37. Safro, I., Dorit, R., Brandt, A.: Multilevel algorithms for linear ordering problems. Journal of Experimental Algorithmics 13, 1–14 (2008)
38. Sanchis, L.: Multiple-Way Network Partitioning. IEEE Transactions on Computers 38(1), 62–81 (1989)
39. Sanchis, L.: Multiple-Way Network Partitioning with Different Cost Functions. IEEE Transactions on Computers 42(12), 1500–1504 (1993)
40. Schloegel, K., Karypis, G., Kumar, V.: Graph partitioning for high performance scientific simulations. In: Dongarra, J., et al. (eds.) CRPC Parallel Computing Handbook. Morgan Kaufmann (2000)
41. Simon, H., Teng, S.H.: How good is recursive bisection. SIAM J. Sci. Comput. 18(5), 1436–1445 (1997)
42. Shi, J., Malik, J.: Normalized Cuts and Image Segmentation. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 731–737 (1997)
43. Słowik, A., Białko, M.: Partitioning of VLSI Circuits on Subcircuits with Minimal Number of Connections Using Evolutionary Algorithm. In: Rutkowski, L., Tadeusiewicz, R., Zadeh, L.A., Żurada, J.M. (eds.) ICAISC 2006. LNCS (LNAI), vol. 4029, pp. 470–478. Springer, Heidelberg (2006)
44. Soper, A.J., Walshaw, C., Cross, M.: A Combined Evolutionary Search and Multilevel Optimisation Approach to Graph-partitioning. Journal of Global Optimization 29(2), 225–241 (2004)
45. Sun, L., Leng, M.: An Effective Multi-Level Algorithm Based on Simulated Annealing for Bisecting Graph. In: Yuille, A.L., Zhu, S.-C., Cremers, D., Wang, Y. (eds.) EMMCVPR 2007. LNCS, vol. 4679, pp. 1–12. Springer, Heidelberg (2007)
46. Viswanathan, N., Alpert, C.J., Sze, C., Li, Z., Nam, G.J., Roy, J.A.: The ISPD-2011 Routability-Driven Placement Contest and Benchmark Suite. In: Proc. ACM International Symposium on Physical Design, pp. 141–146 (2011)
47. Walshaw, C.: Multilevel refinement for combinatorial optimisation problems. Annals of Operations Research 131, 325–372 (2004)
48. Walshaw, C., Cross, M.: JOSTLE: Parallel Multilevel Graph-Partitioning Software – An Overview. In: Magoules, F. (ed.) Mesh Partitioning Techniques and Domain Decomposition Techniques, pp. 27–58 (2007)
49. Zha, H., He, X., Ding, C., Simon, H., Gu, M.: Bipartite Graph Partitioning and Data Clustering. In: Proceedings of the ACM 10th International Conference on Information and Knowledge, pp. 25–31 (2001)