Genetic Approaches for Graph Partitioning: A Survey

Jin Kim
School of Computer Science and Engineering
Seoul National University
Seoul 151-744, Korea
kimjin@soar.snu.ac.kr

Inwook Hwang
Mechatronics and Manufacturing Technology Center
Samsung Electronics
Gyeonggi-do 443-742, Korea
iwook.hwang@samsung.com

Yong-Hyuk Kim
Department of Computer Science and Engineering
Kwangwoon University
Seoul 139-701, Korea
yhdfly@kw.ac.kr

Byung-Ro Moon
School of Computer Science and Engineering
Seoul National University
Seoul 151-744, Korea
moon@snu.ac.kr

ABSTRACT

The graph partitioning problem occurs in numerous applications such as circuit placement, matrix factorization, load balancing, and community detection. For this problem, genetic algorithm is a representative approach with competitive performance with many related papers being published. Although there are a number of surveys on graph partitioning, none of them deals with genetic algorithms in much detail. In this survey, a number of problem-specific issues in applying genetic algorithms to the graph partitioning problem are discussed; the issues include encoding, crossover, normalization, and balancing.

Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: Optimization—*Global optimization*; G.2.2 [**Discrete Mathematics**]: Graph Theory—*Graph algorithms*

General Terms

Algorithms

Keywords

genetic algorithm, graph partitioning, graph bisection, combinatorial optimization, survey

1. INTRODUCTION

Given an undirected graph $G = (V, E)$, where V is the set of vertices and E is the set of edges, a balanced *k-way graph partitioning* is defined as k disjoint subsets $V_1, V_2, \ldots, V_k$ of V such that $||V_i| - |V_j|| \leq 1\ \forall i, j$ and $\sum_i |V_i| = |V|$, where $|S|$ means the cardinality of a set S. Each subset is called a *partition*. The *cut size* of a partitioning is defined to be the number of edges whose adjacent vertices are located in different partitions, and the edges are called *cut edges*. The *k-way graph partitioning problem* (k-way GPP) is the problem that finds the k-way graph partitioning with the minimum cut size. When $k = 2$, the problem is called *graph bisection* or *bipartitioning*. The above definition is easily generalized to weighted vertices and/or edges. Since most studies focused on unweighted graphs, graphs hereafter are assumed to be unweighted unless otherwise stated.

GPP arises in various practical applications such as VLSI circuit placement [25], sparse matrix factorization [11], parallel computing [50], and community detection in social networks [66]. There have been studies on the problem using coordinate-based methods [6], spectral ones [24], and multilevel ones [41, 70]. The problem is NP-hard for general graphs [32], as well as planar graphs and regular graphs [10].

It is also known that there is no approximation algorithm with a constant ratio factor for general graphs [10]. Leighton and Rao [53] proposed an approximation algorithm allowing for some imbalance in the range of $1/3$ to $2/3$ that guarantees $O(\log |V|)$ times the optimum. Since it is hardly expected to be able to produce quality solutions with approximation algorithms of theoretical bounds, studies have mostly concentrated on finding quality solutions without performance guarantee in a reasonable amount of time. Representative methods are metaheuristics such as simulated annealing [37], tabu search [68], ant colony optimization [16], and genetic algorithms(GAs) [15].

Among the metaheuristics, GAs have been the most successful and many related papers have been published on the subject. To our best knowledge, however, there are no survey focusing on GAs for GPP. Although a few surveys focused on GPP, they dealt with GA too briefly [1, 27, 30] or even did not mention it at all [65].

In this paper, some of the issues of GAs for GPP such as representation, normalization, balancing, and local optimization are discussed. By overviewing the previous work of GAs for GPP, researchers may be able to reduce the amount of trial and errors and acquire intuition on the reasonable combination of genetic elements. This intuition will also be useful for other combinatorial optimization problems.

GECCO'11, July 12–16, 2011, Dublin, Ireland.

The remainder of this paper is organized as follows: In Section 2 the effectiveness of GAs for GPP by investigating the problem space of GPP is discussed. Section 3 describes the core features of GA, e.g., encoding, crossover, and normalization. Section 4 is devoted to how to satisfy the balancing constraint of GPP. Local search algorithms are examined in Section 5 along with other techniques in Section 6. Finally, this paper is closed with a summary and directions for future work.

2. PROBLEM SPACES

Local search algorithms are popular for GPP. A typical local search algorithm finds solutions by searching the *neighborhood* of a solution; it repeatedly selects the best neighbor until no remaining neighbors are better than the current solution. The neighborhood denotes the set of solutions that are located within a specific distance from a solution in the fitness landscape. More formally, neighborhood $N(s, d)$ of a solution s is defined as $\{x \mid distance(s, x) \leq d\}$, where d is the specific distance called the radius. Local search algorithms obtain reasonable solutions in an appropriately short amount of time; however, they often get stuck in low-quality local optima.

To overcome this drawback, local search algorithms are combined with GAs which are good at searching the problem space globally. The combination is called *hybrid GA* or *memetic GA*. Since GAs are not so good at fine-tuning around local optima, local search algorithms can help improve GA's performance. To the best of our knowledge, Laszewski and Mühlenbein [52] suggested the first hybrid GA for GPP.

Investigating the local optimum space of GPP, Inayoshi and Manderick [36] noticed that good local solutions tend to be close one another. They thought that this was related to the effectiveness of crossover and did some experiments to examine the structure of the problem space. They showed that the distances between the local optima and their neighbors are smaller compared to those of the ordinary solutions. They insisted that it was evidence of showing the smoothness of the problem space.

Boese *et al.* [7] further investigated the local optimum space of GPP by computing the distances between many local optima and the global optimum. The result showed that the quality of the local optima overall decreases as the distance from the global optimum increases. They conjectured that the local optimum space is globally convex around the global optimum. If the crossover operator is regarded as a convex search, the observation implies that a hybrid GA is a good search method because it takes advantage of the convexity of the problem space.

Kim and Moon [45] supported these results with a series of experiments on the local optimum space of GPP. They reported that solutions near the center of the local optimum space usually had small cut sizes. It suggests that the central region is quite attractive. Whether explicit or not, crossover operators tend to drive solutions toward the central area of the problem space. This makes crossover an attractive search operator for the problem.

Merz and Freisleben [58] observed that if graphs are different from each other, the corresponding problem spaces are also different. They designed experiments with two local heuristics and observed from geometric graphs that the problem space became smoother as the average degree increased; this observation goes against common intuition.

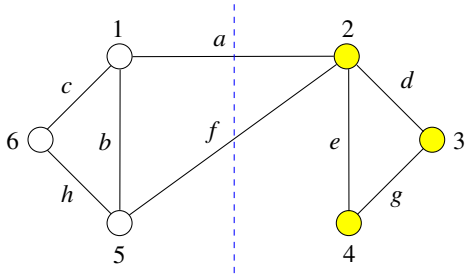

Figure 1: An example of bisection. The vertices are labeled by numbers and the edges are by alphabets. The cut edges are a and f; the cut size is 2.

3. REPRESENTATION AND OPERATORS

3.1 Encoding

In GAs for the k-way GPP, a chromosome corresponds to a partitioning of the vertices. Among the schemes representing a chromosome, the *group-numbers encoding*, also called the *vertex-to-cluster encoding*, is the most natural and prevalent one. It assigns each gene a specified number 0 to $k-1$ depending on which partition the gene belongs to. It is widely used because it is simple and intuitive, however, it has *redundancy*. There are $k!$ encodings for an identical partition, e.g., a bipartition in Figure 1 may be encoded as 011100 or 100011. It is known that redundancy undermines the performance of GA [8, 69]. To cope with this situation, normalization is used, which is covered in Section 3.3.

In addition, there is another encoding that can be considered as a kind of group-numbers encoding. In the encoding by Steenbeek *et al.* [75], a gene corresponds to a cluster, not a vertex. The clusters are identified in the early stage of the GA.

There are a considerable amount of work [21, 34] using order-based encoding which was originally used in combinatorial problems for optimization of permutations including the traveling salesman problem (TSP). In this encoding, a chromosome is represented by a permutation of $|V|$ integers representing the vertices of graph. To reconstruct or evaluate a partitioning of vertices represented by this encoding, a decoding process is required. The process assigns each vertex to the partition that maximizes the sum of the edge weights between the vertices in the same partition; the resulting partitioning becomes the phenotype of the chromosome. Although this encoding does not suffer from redundancy, it cannot represent all feasible solutions and its decoding process takes a long time.

Armbruster *et al.* [3] and Boulif [9] independently proposed the edge encoding, which maps a chromosome to an edge set, not a vertex set. In the representation, each gene of a chromosome is assigned to 1 if its corresponding edge is on the cut and 0 otherwise. For example, the partitioning of the graph on Figure 1 is represented as 10000100 in alphabetical order. This representation is well adapted to their formulations. However, it requires a rather long chromosome of length $|E|$ to represent the partitions, while the group-numbers encoding requires only $|V|$. Moreover, one-

point crossover, which Armbruster *et al.* used, creates infeasible offsprings more frequently in the representation.

Gene reordering is an encoding scheme using the linkage information of GPP. In the GA for GPP, Bui and Moon [13] first proposed the gene reordering method. The method rearranges gene positions in a BFS (breadth-first search) visiting order. They insisted that gene reordering helps to preserve the building blocks in multi-point crossover because gene reordering tends to shorten the defining lengths of good schemata. They also proposed DFS-based (depth-first search) gene reordering and compared it with BFS gene reordering [12]. Through further study on BFS gene reordering, they developed a hybrid GA [15] that became one of the most cited articles in the field of GPP. Martin's spectral hybrid GA also adopted this method [57]. Hwang *et al.* [35] improved BFS gene reordering by starting the BFS from multiple points.

Representing a solution into a one-dimensional string inevitably causes the loss of information contained in the original graph with a multi-dimensional nature. This led to multi-dimensional encodings. Cohoon and Paris [22] first used a two-dimensional chromosome for a VLSI placement problem. For GPP, Bui and Moon [14] introduced multi-dimensional encoding and developed crossovers for it. The details are explained in the next subsection.

3.2 Crossover

Crossover is closely related to encoding. Crossovers that can be applied to ordinary one-dimensional group-numbers encoding for GPP include one-point, multi-point, and uniform crossovers. The standard one-point crossover is simple but effective enough for some GAs to use it as their main operation [2, 3, 77]. Among multi-point crossovers, five-point is the most popular [12, 13, 14, 15, 35, 40, 44, 46, 56, 57]. Such a high perturbation is allowable particularly in hybrid GAs because of the strong search power of the local optimization heuristics.

Uniform crossover is prevalently used [36, 38, 48, 55, 58, 75] because it can produce the most various kinds of combinations. Maini *et al.* [55] invented two variants of this, *Knowledge-based Non-Uniform Crossover* (KNUX) and *Dynamic KNUX* (DKNUX). KNUX uses the adjacency information of the vertices and makes variant masks for the crossover, while DKNUX changes the masks adaptively. Inayoshi and Manderick [36] proposed a variant that inherits the common gene shared by both parents and fills the others randomly. Merz and Freisleben [58]'s *greedy crossover* works similar to this except that it fills the others with the min-max greedy heuristic of Battiti and Bertossi [5].

Crossovers for one-dimensional encoding can also be applied to multi-dimensional encoding; however, they ignore higher dimensional relationship between genes. Bui and Moon [14] proposed *Z3 crossover* which is a generalized multi-point crossover suitable for multi-dimensional encoding. Comparing it with one-dimensional crossover, they obtained better results for GPP benchmark graphs. Since it lacks diversity, Kahng and Moon [39] proposed *geographic crossover* which uses various cutting strategies to create diverse forms of schemata. It outperformed Z3.

Since these operators are not attractive to order-based encodings, permutation crossover operators are designed for the encodings. Höhn and Reeves [34] first applied PMX (partially matched crossover) to the GPP inspired by [38]. Cincotti *et al.* [21] takes into consideration the *leaders*, which are the starting points for a greedy heuristic, because the other vertices have little to do with the result. The crossover of the leaders is a variant of the uniform crossover. Moraglio *et al.* [59, 60] extended cycle crossover to preserve feasibility. Its details are given in Section 4.

Soper *et al.* [74]'s crossover is unique, in that the structure of a graph is not mapped directly to a chromosome, but is used as a base to weight the edges for a multi-level algorithm. Vertices and edges near the cut are weighted higher and the others are weighted lower. In a multi-level algorithm, the higher weighted edges tend to be selected as cut edges. More details are given in 5.2.

Lin and Cheng's crossover [54] is based on *autogamy*, not *allogamy*, which is commonly used in other GAs. Two offsprings are generated from each selected parent by a *pizza-cutting* operation, which divides the border area[1] into two partitions and then the combination of this border area cut and the original cut create two new cuts, i.e., two offsprings. They reported that it effectively worked on partitioning internet-like graphs.

3.3 Normalization

There are three types of relationships between *genotype* and *phenotype*, i.e., encoding and the real solution: one-to-one, one-to-many, and many-to-one. There is no problem in one-to-one encoding. One-to-many encoding, which was suggested in the context of some order-based encodings [21, 34] does not require normalization. Many-to-one encoding causes somewhat serious problems to the crossover's consistency for searches. In this encoding, more than one genotype represents the same partition. The good features of parent solutions may be lost as a result of crossover by being confused with diverse genotypes.

The commonly used group-numbers encoding has redundancy, which raises some problems. If genotypes corresponding to the same phenotype are similar (synonymously redundant), even redundant encoding does not create a serious problem [69]. Unfortunately for GPP, the prevalent group-numbers encoding is nonsynonymously redundant and genotypes are fairly different so that it undermines the performance of GAs. Choi *et al.* [18] showed that both fitness-distance correlation and epistasis variance representing the problem space become zero, which implies that redundant encoding obstructs the search of GA.

Normalization is a method that overcomes this situation. Commonly used normalization locks the genotype of one parent chromosome and transforms the other parent of the closest chromosome to the locked parent. Figure 2 shows an example. In the example, the offspring obtained by normalization better preserves the characteristics of the parents. In [15], crossover produces two offsprings by complementing a parent chromosome and selecting the better parent. This method shares the same motivation as normalization.

The first normalization of GPP appeared in Laszewski [51]. Although they did not call it normalization, their adaptive crossover operator plays the role of normalization. The operator chooses a partition of one parent and copies it to the other. During the process, the number of chosen partition is replaced with that of the most overlapped partition. Kang and Moon [40] extended this method to multi-way partitioning by considering the whole partition. They used the term

[1] the vertices near the cut

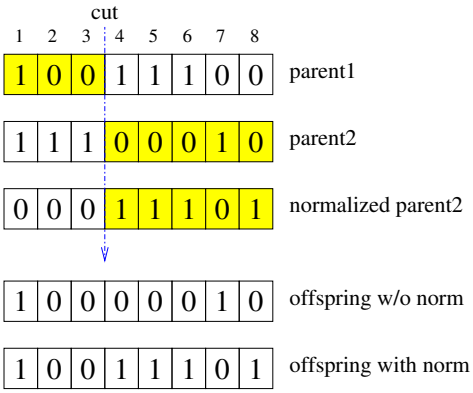

Figure 2: An example of normalization for group-numbers encoding.

normalization for the first time. Their method is a kind of greedy heuristic. Choi and Moon [19, 20] did experiments on the problem space of GPP and reported that the problem space varies according to the crossover operators. They also showed that normalization improves GA's performance.

Normalization is modeled by the optimal assignment problem finding a numbering of partitions that minimizes the distance between two chromosomes. The greedy algorithm used in previous methods works well but does not guarantee that optimal numbering is obtained in the sense of genotypic consistency. Moraglio *et al.* [59] proposed a method to find the optimal numbering using the Hungarian method [49].

All known normalization methods are based on the Hamming distance. As an extension of the Hamming distance, Moraglio *et al.* [59] devised *labeling-independent distance* defined as the minimum Hamming distance between all pairs of one solution. However, this is also based on the Hamming distance. No method for normalization based on other metrics has been studied.

4. BALANCING

In the GPP, balancing constraint denotes that the difference of cardinalities between the largest partition and the smallest one should be at most one. It is a critical constraint that makes the problem difficult to solve. If the constraint is ignored, the GPP can be seen as the same problem as MIN-k-CUT. Especially, if k is two, it can be solved in polynomial time by max-flow algorithms [31]. When allowed an imbalance of one-third, an $O(\log |V|)$-approximation algorithm can be obtained [53]. On the other hand, if the constraint must be satisfied, no OPT + $O(|V|^2)$-approximation algorithm exists unless P=NP [10].

In GAs for GPP, crossover often does not produce feasible partitions. There are two ways of resolving this. One is allowing unbalanced chromosomes and adding an imbalance penalty to the fitness function. The imbalance penalty is commonly defined as the multiplication of a constant α and the squared or absolute sum of the differences between the sizes of each partition. The constant α is used to control the degree of penalty; if α is higher, the imbalance penalty increases faster. Imbalance penalty has been adopted by many studies [23, 28, 55, 75].[2] If a balanced solution is not obtained by the end of a GA process, the final solution should be adjusted by moving the vertices of the large partitions to the small ones.

The other is to repair an unbalanced partition to a balanced one immediately after crossover or mutation. Modification of the fitness function is not necessary and feasible solutions are always available all through the GA process. Moving the vertices of a larger partition to a smaller one is the most common repair method. Hwang *et al.* [35] showed that contiguous repair improves the performance by synergy with gene reordering. It implies that repairing algorithms influence the performance of the GAs for GPP.

Allowing penalty has a merit that it is easily applied to GAs that use primitive operations only. Its demerit is its performance. Pirkul and Rolland [64] compared the method of repair to the method that allows the imbalance and reflects the penalty in the fitness function. They reported that the performance of repair method was better; however, the details of the experimentation were not presented.

When repair frequently occurs, a risk arises that the offspring may be fairly different from their parents. Höhn and Reeves [34] explained this by the concept of *strict transmission* [67], which is a property that all alleles of the child chromosome come from one of the corresponding parent values. They observed that crossover does not work well when strict transmission rarely occurs and thus, the repair rate grows. However, this is not necessarily true for hybrid GAs which allow for a rather high rate of mutation.

Another approach is to modify a crossover to prevent unbalanced solutions from being produced. The previously mentioned greedy crossover of Merz and Freisleben [58] is an example. Since the min-max greedy heuristic included in the crossover repairs unbalanced partitioning, no other balancing method is necessary. Moraglio *et al.* [59, 60] extended cycle crossover for permutation-with-repetitions encoding and applied it to multi-way partitioning. It always produces feasible solutions.

5. LOCAL OPTIMIZATION

Since a GA is weak at fine-tuning around local optima, it is extremely hard to get a competitive result without local optimization for larger than toy-size NP-hard problems. In this section, representative local optimization algorithms are described and explaining how they are combined with GAs.

5.1 KL Heuristic and Its Variants

Among local optimization algorithms for graph partitioning, Kernighan-Lin (KL) algorithm [43] is often considered the first competitive heuristic for GPP.

The KL algorithm starts with a balanced partitioning of vertices. It proceeds in a series of *passes*. During each pass, the algorithm improves the initial solution by swapping pairs of vertices to create a new solution. This process is repeated for the new solution until no more improvements can be obtained.

Let (A, B) be an initial bipartition of vertices V. Define the *gain* g_v of a vertex v to be the cut size reduction by moving v to the opposite set. The gain $g(a, b)$ as a result of swapping vertices $a \in A$ and $b \in B$ is $g_a + g_b - 2\delta(a, b)$, where $\delta(a, b)$ is 1 if $(a, b) \in E$ and 0 otherwise.

The pair (a, b) that maximizes $g(a, b)$ is selected. Once a and b are selected, they are assumed to be exchanged and not considered for further exchange. A sequence of pairs

[2] A study of simulated annealing [37] influenced these. The study also found that reasonable values of α may be small enough.

$(a_1, b_1), (a_2, b_2), \ldots, (a_{|V|/2-1}, b_{|V|/2-1})$ are selected in this way. The algorithm chooses l that maximizes $\sum_{i=1}^{l} g(a_i, b_i)$ and exchanges $\{a_1, a_2, \ldots, a_l\}$ and $\{b_1, b_2, \ldots, b_l\}$. This is a pass of KL. KL repeats the above pass until there is no more improvement.

The KL algorithm has been frequently used because it finds quite competitive solutions in a reasonable amount of time. A number of variants have been proposed [26, 46, 47]. Among them, Fiduccia-Mattheyses (FM) algorithm [29] is widely used. The main difference between KL and FM lies in the unit of vertex-moving during a pass. While KL exchanges a pair of vertices, FM moves one vertex at a time. This makes FM faster than KL with little loss of partitioning quality. Bui and Moon devised a fast variation of KL [15]; its time complexity is mostly $O(|E|)$ while that of the traditional KL algorithm is mostly $O(|V|^3)$.

There are three main schemes using the above algorithms for multi-way partitioning which were originally developed for bipartitioning: recursive method, pairwise one, and direct one [44]. The recursive method repeats bipartitioning until the given graph is partitioned into k subgraphs. The pairwise method repeatedly chooses two partitions randomly and runs the KL algorithm [43] to them. The direct method uses a multi-way generalization of the FM algorithm [29]. Sanchis [72] showed that the direct method performed better than the recursive one.

5.2 Multi-level Heuristics

The multi-level heuristic is one of the most popular methods that are combined with GAs. It works as follows:

1. *Coarsening*: find and collapse appropriate vertices to make a coarser graph.
2. *Partition*: if the graph is small enough, partition it.
3. *Uncoarsening*: restore the collapsed vertices.

Since the balance of the graph may be violated during uncoarsening, a balance-adjustment is required. Appropriate heuristics such as KL may help improve the performance.

In the coarsening phase, maximal matching is primarily used. Küçükpetek *et al.* [48] proposed a method that finds a maximal matching by a GA. They compared the method with a famous multi-level package, METIS [42], and reported better results in a number of benchmark graphs, though it took a longer time.

Soper *et al.* [74] took advantage of multi-level heuristic as a local optimization method for GAs. The heuristic works like crossover. Recombination and mutation operators are used for computing weight biases of vertices and edges, and the results give hints for multi-level heuristics. That is, good characteristics of previously promising solutions are reflected in multi-level heuristics. They reported better results than graph partitioning packages such as Chaco [33] and METIS, but their method took a much longer time. They insisted that the method is useful when the quality of partitioning is important.

5.3 Local Search Algorithm in Hybrid GAs

In hybrid GAs for GPP, there have been some studies using fast but weak local optimization algorithms. In [51, 61], 2-opt was applied only to the border vertices which are connected by the cross edges between the partitions. Bui and Moon [15] obtained better results by allowing only one pass of KL instead of full passes, together with restricting the size of the sets to be swapped. They believed that the improved results were due to avoiding premature convergence by a slightly weaker local optimization technique.

On the other hand, there have been a number of studies reporting notable performance improvement by enhanced local optimization algorithms. For bisection, Kim and Moon [46] suggested a new KL-based local search algorithm, which introduces a new type of gain, called *lock gain*, of a vertex in a manner that takes into account only the edges connected to the vertices that have already been moved. Combining it with a GA, they obtained an impressive result for most benchmark graphs. Hybrid GAs with local optimization algorithms specialized for multi-way partitioning showed good results [40, 44].

Steenbeek *et al.* [75] proposed a cluster enhancement heuristic (CEH) and a CEH-combined hybrid GA. Their hybrid GA partitions vertices using a node-swap heuristic and identifies clusters. It only then moves clusters to another partition in the GA process. They reported successful results.

6. OTHER APPROACHES

6.1 Spectral Methods

Martin [56, 57] introduced singular value decomposition (SVD), which is a spectral technique that can obtain relationships between genes, to enhance GAs. In his hybrid GA, the initial population was created using eigenvectors of the incidence matrix of a graph. Then in each generation, it applies a process named *genetic engineering*, which constructs a graph using the gene relationships that are found by rank-2 SVD on several promising solutions. The SVD hybrid GA has found minimum bisections for some benchmark graphs. A weak point of SVD is its running time; it runs in $O(|E|^2|V| + |E||V|^2 + |V|^3)$ for an $|E| \times |V|$ matrix. Martin expected that his method might be useful when the fitness function is expensive to compute.

6.2 Branch-and-cut

Armbruster *et al.* [2, 3] formulated integer programming for GPP, and solved it using branch-and-cut and linear programming (LP), a very effective method for integer programming. The lower and upper bound of branch-and-cut framework are obtained respectively by the LP primal method and by a GA whose initial population is provided by LP rounding with various random pivots. They used common operators except for four kinds of mutations: an ordinary swap; moving one vertex into the opposite cluster; a variation of constructing initial population; and a local search. The gap of branch-and-cut was small for about 300 vertices of graphs, but increased for larger graphs. It is notable that the role of GA is secondary in the whole framework of the method.

6.3 Non-traditional GAs

Non-traditional GAs are not comprised of traditional operators such as selection, recombination, and mutation. Saab and Rao [71] devised a stochastic evolution method which uses a complicated mutation instead of crossover. Barake *et al.* [4] proposed a new metaheuristic named PROBE, which can be viewed as a GA without selection. When applied to the GPP [17], it outperformed other metaheuristics.

Table 1: List of GA papers for graph partitioning

author(s)	brief description	year	section	reference
Laszewski and Mühlenbein	the first hybrid GA for graph partitioning	1991	2, 6.4	[52]
Laszewski	the first normalization	1991	3.3, 5	[51]
Talbi and Bessière	parallel GA	1991	6.4	[77]
Bui and Moon	BFS gene reordering	1993	3.1, 3.2, 3.3, 5	[12, 13, 15]
Inayoshi and Manderick	investigating problem space	1994	2	[36]
Maini *et al.*	variants of uniform crossover	1994	3.2	[55]
Höhn and Reeves	order based encoding	1996	3.1, 3.2, 4	[34]
Steenbeek *et al.*	cluster based encoding and local search	1998	3.1, 5	[75]
Schwarz and Očenášek	BMDA and BOA	1999	6.3	[73]
Kang and Moon	normalization and multi-way partitioning	2000	3.3, 5	[40]
Merz and Freisleben	investigating problem space	2000	2	[58]
Kim and Moon	multi-way partitioning	2001	5	[44]
Cincotti *et al.*	order based encoding	2002	3.1, 3.2	[21]
Mühlenbein and Mahnig	EDA	2002	6.3	[62]
Choi and Moon	normalization	2003	3.3	[19, 20]
Kim and Moon	lock gain based local search heuristic	2004	5	[46]
Kim and Moon	investigating problem space	2004	2	[45]
Soper *et al.*	multi-level hybrid	2004	3.2, 5.2	[74]
Küçükpetek *et al.*	multi-level hybrid	2005	5.2	[48]
Armbruster *et al.*	LP-based branch-and-cut hybrid	2005–6	3.1, 6.2	[2, 3]
Martin	spectral method	2005–6	6.1	[56, 57]
Boulif and Atif	edge encoding	2006	3.1	[9]
Hwang *et al.*	multi-attractor BFS gene reordering	2006	3.1, 4	[35]
Chardaire *et al.*	PROBE metaheuristic	2007	6.3	[4, 17]
Moraglio *et al.*	geometric crossover and multi-way partitioning	2007, 2011	3.2	[59, 60]
Lin and Cheng	*pizza-cutting* crossover and initialization	2008	3.2	[54]
Farshbaf and Feizi-Derakhshi	multi-objective optimization	2009	4	[28]
Boulif	survey on representation	2010	3.1	[8]

Recently, estimation of distribution algorithm (EDA), a generalization of GAs using a probabilistic model, became popular. Pelikan *et al.* [63] applied Bayesian optimization algorithm (BOA), a variant of EDA, on two simple structured graphs and obtained good partitioning. However, the instance graphs were too small to show that EDA can deal with general instances well. Schwarz and Očenášek [73] reported that a bivariate marginal distribution algorithm (BMDA) was superior to a simple GA with respect to performance and time cost, but inferior to a *SGAH*, a simple GA hybridized with a local search heuristic. They also observed that the BOA consistently performed good and fast for large graphs, however, it could not completely beat the SGAH. Mühlenbein and Mahnig [62] did experiments on graphs with hundreds of vertices using a univariate marginal distribution algorithm (UMDA) and a learning factorized distribution algorithm (LFDA) that was combined with one pass of KL, and obtained better results than the previous work.

6.4 Parallel GAs

Since GAs mimic natural evolution which is an inherently parallel process, it is not surprising that papers in the early 90's had much interest in parallel GA (PGA). There are two main issues for PGA: topology and population. In most papers, the grid topology is used, and communication occurs between close processors that are nearer than the specific distance, usually two or three. Laszewski and Mühlenbein's PGA [52] is a representative case in which one solution corresponds to one processor and communication occurs between the adjacent processors in the ring topology. Also in Talbi [77]'s torus topology, one processor corresponds to one solution.

Collins and Jefferson [23] compared local mating and panmictic mating and reported that local mating was superior to traditional GAs as well as panmictic mating. Unlike the other researchers who have major interest in the GPP itself, they considered the GPP as just a test problem.

7. SUMMARY

The graph partitioning problem is a representative combinatorial optimization problem belonging to the class of NP-hard. As shown in Table 1, a great amount of research has been done on the problem. The facility of representation and the difficulty of finding optimal solutions have made many researchers challenge the problem. In addition, many designers choose the graph partitioning problem as the target problem to verify the superiority of their methods.

In this paper, we summarized the various approaches for the problem and, in particular, went over the issues of GAs related to the problem. These issues include the main topics of GAs such as representation, crossover, normalization, constraint handling, local optimization, and hybridization. We hope that this survey helps to lead research in a new direction for further developing effective and efficient methods.

Recently, as large problems such as the problem of partitioning *complex network* [76] including social networks have

drawn a great deal of attention from researchers, the requirement for algorithms that manage huge graphs is increasing. Since huge graphs require enormous memory, traditional GAs cannot manage it effectively. Designing novel genetic operators tailored for huge graphs will be a good direction for further study.

8. ACKNOWLEDGMENTS

The ICT at Seoul National University provides research facilities for this study. This work was supported by the Engineering Research Center of Excellence Program (2011-0000966), Basic Science Research Program (2011-0004215), and Mid-career Researcher Program (2010-0014218) of Korea Ministry of Education, Science and Technology (MEST) / National Research Foundation of Korea (NRF).

9. REFERENCES

[1] C. J. Alpert and A. B. Kahng. Recent directions in netlist partitioning: A survey. *Integration, the VLSI Journal*, 19(1-2):1–81, 1995.

[2] M. Armbruster, M. Fugenschuh, C. Helmberg, N. Jetchev, and A. Martin. LP-based genetic algorithm for the minimum graph bisection problem. In *Operations Research Proceedings*, pages 315–320, 2005.

[3] M. Armbruster, M. Fugenschuh, C. Helmberg, N. Jetchev, and A. Martin. Hybrid genetic algorithm within branch-and-cut for the minimum graph bisection problem. In *Evolutionary Computation in Combinatorial Optimization*, pages 1–12, 2006.

[4] M. Barake, P. Chardaire, and G. P. McKeown. The PROBE metaheuristic and its application to the multiconstraint knapsack problem. In *Metaheuristics: computer decision-making*, pages 19–36. Kluwer Academic Publishers, 2004.

[5] R. Battiti and A. Bertossi. Greedy, prohibition, and reactive heuristics for graph partitioning. *IEEE Transactions on Computers*, 48(4):361–385, 1999.

[6] M. J. Berger and S. H. Bokhari. A partitioning strategy for nonuniform problems on multiprocessors. *IEEE Transactions on Computers*, 36(5):570–580, 1987.

[7] K. D. Boese, A. B. Kahng, and S. Muddu. A new adaptive multi-start technique for combinatorial global optimizations. *Operations Research Letters*, 15:101–113, 1994.

[8] M. Boulif. Genetic algorithm encoding representations for graph partitioning problems. In *International Conference on Machine and Web Intelligence*, pages 288 –291, 2010.

[9] M. Boulif and K. Atif. A new branch-&-bound-enhanced genetic algorithm for the manufacturing cell formation problem. *Computers & Operations Research*, 33(8):2219 – 2245, 2006.

[10] T. N. Bui and C. Jones. Finding good approximate vertex and edge partitions is NP-hard. *Information Processing Letters*, 42:153–159, 1992.

[11] T. N. Bui and C. Jones. A heuristic for reducing fill-in in sparse matrix factorization. In *Sixth SIAM Conference on Parallel Processing for Scientific Computing*, pages 445–452, 1993.

[12] T. N. Bui and B.-R. Moon. Hyperplane synthesis for genetic algorithms. In *Fifth International Conference on Genetic Algorithms*, pages 102–109, 1993.

[13] T. N. Bui and B.-R. Moon. A genetic algorithm for a special class of the quadratic assignment problem. *The Quadratic Assignment and Related Problems, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 16:99–116, 1994.

[14] T. N. Bui and B.-R. Moon. On multi-dimensional encoding/crossover. In *Sixth International Conference on Genetic Algorithms*, pages 49–56, 1995.

[15] T. N. Bui and B.-R. Moon. Genetic algorithm and graph partitioning. *IEEE Transactions on Computers*, 45(7):841–855, 1996.

[16] T. N. Bui and L. C. Strite. An ant system algorithm for graph bisection. In *Genetic and Evolutionary Computation Conference*, pages 43–51, 2002.

[17] P. Chardaire, M. Barake, and G. P. McKeown. A PROBE-based heuristic for graph partitioning. *IEEE Transactions on Computers*, 56(12):1707–1720, 2007.

[18] S.-S. Choi, Y.-K. Kwon, and B.-R. Moon. Properties of symmetric fitness functions. *IEEE Transactions on Evolutionary Computation*, 11(6):743–757, 2007.

[19] S.-S. Choi and B.-R. Moon. Normalization in genetic algorithms. In *Genetic and Evolutionary Computation Conference*, pages 862–873, 2003.

[20] S.-S. Choi and B.-R. Moon. Normalization for genetic algorithms with nonsynonymously redundant encodings. *IEEE Transactions on Evolutionary Computation*, 12(5):604–616, 2008.

[21] A. Cincotti, V. Cutello, and M. Pavone. Graph partitioning using genetic algorithms with ODPX. In *IEEE Congress on Evolutionary Computation*, pages 402–406, 2002.

[22] J. P. Cohoon and W. Paris. Genetic placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 6(6):956–964, 1987.

[23] R. Collins and D. Jefferson. Selection in massively parallel genetic algorithms. In *Fourth International Conference on Genetic Algorithms*, pages 249–256, 1991.

[24] W. Donath and A. Hoffman. Lower bounds for the partitioning of graphs. *IBM Journal of Research and Development*, 17:420–425, 1973.

[25] A. Dunlop and B. Kernighan. A procedure for placement of standard-cell VLSI circuits. *IEEE Transactions on Computer-Aided Design*, CAD-4(1):92–98, 1985.

[26] S. Dutt and W. Deng. A probability-based approach to VLSI circuit partitioning. In *Design Automation Conference*, pages 100–105, 1996.

[27] J. Elsner. Graph partitioning - a survey. *Technische Universität Chemnitz*, 1997.

[28] M. Farshbaf and M.-R. Feizi-Derakhshi. Multi-objective optimization of graph partitioning using genetic algorithms. In *Proceedings of the Third International Conference on Advanced Engineering Computing and Applications in Sciences*, pages 1–6, 2009.

[29] C. Fiduccia and R. Mattheyses. A linear time heuristics for improving network partitions. In *19th ACM/IEEE Design Automation Conference*, pages 175–181, 1982.

[30] P. O. Fjällström. Algorithms for graph partitioning: A survey. *Linköping Electronic Atricles in Computer and Information Science*, 3(10), 1998.

[31] L. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.

[32] M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete problems. In *Sixth Annual ACM Symposium on Theory of Computing*, pages 47–63, 1974.

[33] B. Hendrickson and R. W. Leland. The Chaco user's guide, version 2.0. Technical Report SAND95-2344, Sandia National Laboratories, Albuquerque, 1995. Open-source software distributed at `http://www.cs.sandia.gov/~bahendr/chaco.html`.

[34] C. Höhn and C. Reeves. Graph partitioning using genetic algorithms. In *Second International Conference on Massively Parallel Computing Systems*, pages 27–43, 1996.

[35] I. Hwang, Y.-H. Kim, and B.-R. Moon. Multi-attractor gene reordering for graph bisection. In *Genetic and Evolutionary Computation Conference*, pages 1209–1216, 2006.

[36] H. Inayoshi and B. Manderick. The weighted graph bi-partitioning problem: A look at GA performance. In *The Third Conference on Parallel Problem Solving from Nature*, pages 617–625, 1994.

[37] D. S. Johnson, C. Aragon, L. McGeoch, and C. Schevon.

Optimization by simulated annealing: An experimental evaluation; Part 1, graph partitioning. *Operations Research*, 37:865–892, 1989.

[38] D. R. Jones and M. A. Beltramo. Solving partitioning problems with genetic algorithms. In *Fourth International Conference on Genetic Algorithms*, pages 442–449, 1991.

[39] A. B. Kahng and B.-R. Moon. Toward more powerful recombinations. In *Sixth International Conference on Genetic Algorithms*, pages 96–103, 1995.

[40] S.-J. Kang and B.-R. Moon. A hybrid genetic algorithm for multiway graph partitioning. In *Genetic and Evolutionary Computation Conference*, pages 159–166, 2000.

[41] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

[42] G. Karypis and V. Kumar. Multilevel k-way partitioning scheme for irregular graphs. *Journal of Parallel and Distributed Computing*, 48(1):96–129, 1998.

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

[44] J.-P. Kim and B.-R. Moon. A hybrid genetic search for multi-way graph partitioning based on direct partitioning. In *Genetic and Evolutionary Computation Conference*, pages 408–415, 2001.

[45] Y.-H. Kim and B.-R. Moon. Investigation of the fitness landscapes in graph bipartitioning: An empirical study. *Journal of Heuristics*, 10(2):111–133, 2004.

[46] Y.-H. Kim and B.-R. Moon. Lock gain based graph partitioning. *Journal of Heuristics*, 10(1):37–57, 2004.

[47] B. Krishnamurthy. An improved min-cut algorithm for partitioning VLSI networks. *IEEE Transactions on Computers*, C-33:438–446, 1984.

[48] S. Küçükpetek, F. Polat, and O Oğuztüzün. Multilevel graph partitioning: an evolutionary approach. *Journal of the Operational Research Society*, 56(5):549–562, 2005.

[49] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97, 1955.

[50] V. Kumar, A. Grama, A. Gupta, and G. Karypis. *Introduction to Parallel Computing*. The Benjamin/Cummings Publishing Company, Inc., 1994.

[51] G. Laszewski. Intelligent structural operators for the k-way graph partitioning problem. In *Fourth International Conference on Genetic Algorithms*, pages 45–52, 1991.

[52] G. Laszewski and H. Mühlenbein. Partitioning a graph with a parallel genetic algorithm. In *First Workshop on Parallel Problem Solving from Nature*, pages 165–169, 1991.

[53] F. T. Leighton and S. Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. In *29th Symposium on Foundations of Computer Science*, pages 422–431, 1988.

[54] S. Lin and X. Cheng. BC-GA: A graph partitioning algorithm for parallel simulation of internet applications. In *16th Euromicro Conference on Parallel, Distributed and Network-Based Processing*, pages 358–365, 2008.

[55] H. S. Maini, K. G. Mehrotra, C. K. Mohan, and S. Ranka. Genetic algorithms for graph partitioning and incremental graph partitioning. In *International Conference on Supercomputing*, pages 449–457, 1994.

[56] J. G. Martin. Subproblem optimization by gene correlation with singular value decomposition. In *Genetic and Evolutionary Computation Conference*, pages 1507–1514, 2005.

[57] J. G. Martin. Spectral techniques for graph bisection in genetic algorithms. In *Genetic and Evolutionary Computation Conference*, pages 1249–1256, 2006.

[58] P. Merz and B. Freisleben. Fitness landscapes, memetic algorithms, and greedy operators for graph bipartitioning. *Evolutionary Computation*, 8(1):61–91, 2000.

[59] A. Moraglio, Y.-H. Kim, Y. Yoon, and B.-R. Moon. Geometric crossovers for multiway graph partitioning. *Evolutionary Computation*, 15(4):445–474, 2007.

[60] A. Moraglio, Y.-H. Kim, Y. Yoon, and B.-R. Moon. Geometric crossovers for multiway graph partitioning. *Theory and Principled Methods for the Design of Metaheuristics*, 2011. (to appear).

[61] H. Mühlenbein. Parallel genetic algorithm in combinatorial optimization. In O. Balci, R. Sharda, and S. Zenios, editors, *Computer Science and Operations Research*, pages 441–456. Pergamon Press, 1992.

[62] H. Mühlenbein and T. Mahnig. Evolutionary optimization and the estimation of search distributions with applications to graph bipartitioning. *International Journal of Approximate Reasoning*, 31(3):157–192, 2002.

[63] M. Pelikan, D. E. Goldberg, and K. Sastry. Bayesian optimization algorithm, decision graphs, and Occam's razor. IlliGAL Report No. 2000020, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, 2000.

[64] H. Pirkul and E. Rolland. New heuristic solution procedures for the uniform graph partitioning problem: extensions and evaluation. *Computers and Operations Research*, 21(8):895–907, 1994.

[65] A. Pothen. Graph partitioning algorithms with applications to scientific computing. In D. E. Keyes, A. H. Sameh, and V. Venkatakrishnan, editors, *Parallel Numerical Algorithms*, pages 323–368. Kluwer Academic Press, 1997.

[66] J. M. Pujol, V. Erramilli, and P. Rodriguez. Divide and conquer: Partitioning online social networks. *CoRR*, abs/0905.4918, 2009.

[67] N. J. Radcliffe. Forma analysis and random respectful recombination. In *International Conference on Genetic Algorithms*, pages 222–229, 1991.

[68] E. Rolland, H. Pirkul, and F. Glover. Tabu search for graph partitioning. *Annals of Operations Research*, 63(2):209–232, 1996.

[69] F. Rothlauf and D. E. Goldberg. Redundant representations in evolutionary computation. *Evolutionary Computation*, 11(4):381–415, 2003.

[70] Y. G. Saab. An effective multilevel algorithm for bisecting graphs and hypergraphs. *IEEE Transactions on Computers*, 53(6):641–652, 2004.

[71] Y. G. Saab and V. Rao. Stochastic evolution: a fast effective heuristic for some genetic layout problems. In *27th ACM/IEEE Design Automation Conference*, pages 26–31, 1990.

[72] L. A. Sanchis. Multiple-way network partitioing. *IEEE Transactions on Computers*, 38(1):62–81, 1989.

[73] J. Schwarz and J. Očenášek. Experimental study: hypergraph partitioning based on the simple and advanced genetic algorithm BMDA and BOA. In *5th International Conference of Soft Computing*, pages 124–130, 1999.

[74] A. J. Soper, C. Walshaw, and M. Cross. A combined evolutionary search and multilevel optimisation approach to graph-partitioning. *Journal of Global Optimization*, 29(2):225–241, 2004.

[75] A. G. Steenbeek, E. Marchiori, and A. E. Eiben. Finding balanced graph bi-partitions using a hybrid genetic algorithm. In *IEEE International Conference on Evolutionary Computation*, pages 90–95, 1998.

[76] S. H. Strogatz. Exploring complex networks. *Nature*, 410:268–276, 2001.

[77] E.-G. Talbi and P. Bessière. A parallel genetic algorithm for the graph partitioning problem. In *Fifth International Conference on Supercomputing*, pages 312–320, 1991.