On the m-Way Graph Partitioning Problem

IMTIAZ AHMAD AND MUHAMMAD K. DHODHI

Department of Electrical and Computer Engineering, Kuwait University, PO Box 5969, Safat 13060, Kuwait
Email: imtiaz@eng.kuniv.edu.kw

The m-way graph partitioning problem (GPP) is an intractable combinatorial optimization problem with many important applications in the design automation of VLSI circuits and in the mapping problem for distributed computing systems. In this paper, we introduce a technique based on a problem-space genetic algorithm (PSGA) for the GPP to reduce the weighted cut-size while keeping the size of each subset balanced. The proposed PSGA based approach integrates a problem-specific simple and fast heuristic with a genetic algorithm to search a large solution space efficiently and effectively to find the best possible solution in an acceptable CPU time. Experimental study shows that our technique produces better results with respect to both the quality of the solution and the computational time over the previous work. The PSGA is a simple, versatile and a generic optimization technique which can also be applied to other combinatorial optimization problems.

Received July 21, 1994; revised December 27, 1994

1. INTRODUCTION

This paper deals with the m-way graph partitioning problem (GPP). Given a graph with weights on both the edges and the nodes, the objective of the GPP is to find a partition of the nodes of a graph into m disjoint, equal-size subsets such that the sum of the weights of those edges which are not in the same partition is minimized. GPP has a lot of useful applications in VLSI cell placement (Sherwani, 1993), high level synthesis (Gajski *et al.*, 1992) and task assignment problem in distributed computing systems (Bokhari, 1987). GPP is an NP-complete problem (Garey and Johnson, 1979) and hence heuristic techniques are applied to get suboptimal results in a reasonable amount of CPU time. In this paper, we introduce a technique based on a problem-space genetic algorithm (PSGA) to solve the general GPP. The PSGA integrates a problem-specific heuristic with a genetic algorithm to search a large solution space effectively and efficiently.

Different approaches have been reported in the literature to solve the GPP, such as the Kernighan and Lin (KL) algorithm (Kernighan and Lin, 1970) and its variants (Fiduccia and Mattheyses, 1982; Dunlop and Kernighan, 1985; Lee *et al.*, 1989), simulated annealing (Johnson *et al.*, 1989; Tao and Zhao, 1993), tabu search (Tao and Zhao, 1993; Glover, 1989), mean-field annealing (Van den Bout and Miller, 1990) and genetic algorithms (GAs) (Jones and Beltramo, 1991; Laszewski, 1991). Most of these techniques start with some initial solution, usually chosen at random, and search the neighborhood of the current solution for a new solution. This process is repeated until no further improvement in the objective function is possible. KL proposed an efficient graph bisectioning algorithm which starts with a random initial partition and then uses pairwise swapping of nodes between partitions until no further improvement is possible. The time complexity of the KL algorithm is $O(N^2 \log_2 N)$, where N is the number of nodes in a graph. The performance of the KL algorithm degrades for non-unity weights on the nodes (Lee *et al.*, 1989). Several algorithms have been developed to improve the basic KL algorithm. Fiduccia and Mattheyses (1982) (FM) used a clever implementation to achieve linear complexity by using an efficient data structure. FM moves single vertex across partition in a single move and it also permits the handling of unbalanced partitions. Dunlop and Kernighan (1985) have compared the KL algorithm with the FM and found that the results of the FM are inferior to those of the KL but that the CPU time is substantially shorter.

Lee *et al.* (1989) presented an iterative algorithm based on the KL algorithm that exploits the problem equivalence property by transforming the given problem into a max-cut problem using a graph transformation technique. It deals with nodes of different weights and outperforms the KL for m-way partitioning. Simulated annealing (SA) (Johnson *et al.*, 1989; Tao and Zhao, 1993) gives superior results, but it is time consuming. Tabu search (Tao and Zhao, 1993; Glover, 1989) is a meta-heuristic which starts from a given solution and iteratively improves it by utilizing a short-term memory in the form of a tabu list to escape from any local minimum and uses the aspiration-level to diversify the search for good solutions. Simulated annealing and tabu search approaches iteratively improve one solution at a time whereas approaches based on GAs simultaneously perform a multi-dimensional search by maintaining a population of potential solutions. Moreover, the tabu search uses memory structure whereas GAs are memoryless which rely on randomization and evolution to conquer intractability. Mean Field Annealing (Van den Bout and Miller, 1990) combines the characteristics of the simulated annealing algorithm and the Hopfield neural network, but does not handle weighted graphs.

An efficient algorithm for a general GPP is desirable to get the best possible solution within acceptable CPU times. In this paper, we introduce an approach based on the problem-space genetic algorithm (PSGA). The PSGA-based approach uses the inherent parallelism provided by the GA and exploits the problem-specific knowledge by using a simple and fast heuristic to search a large solution space efficiently and effectively to get the best possible solution within acceptable CPU times. GAs are probabilistic combinatorial optimization techniques in which the genetic operators such as selection, crossover and mutation, which are derived from the selection processes in nature, guide a population of potential solutions to a problem towards better solutions (Holland, 1975). The recent literature reports a number of successful applications of GAs to a wide range of problems in diverse fields, such as the GPP (Jones and Beltramo, 1991, Laszewski, 1991), standard cell placement (Shahookar and Mazumder, 1990), searching, machine learning and machine identification (Goldberg, 1989).

GAs are blind search techniques and they require problem-specific genetic operators (crossover, mutation) to get good solutions. Storer *et al.* (1992) have proposed a new search method, which integrates a fast, problem-specific heuristic with the local search. The key concept in this method is to base the definition of search neighborhood on a heuristic/problem pair (h, p) where h is a known fast heuristic and p represents the problem data. Since a heuristic h is mapping from a problem to a solution, the pair (h, p) is an encoding of a specific solution. By perturbing the problem p, a neighborhood of solutions is generated. This neighborhood forms the basis for a local search. The problem space is generated by perturbing the problem data. Let P be a set of m problems obtained by perturbing the original problem data. That is, $P = \{p_j = p_0 + \delta, j = 1, \ldots, m\}$, where p_0 is the data for the original problem and δ is the randomly generated perturbation vector. The perturbation range depends on the specific problem. In order to keep the generated 'dummy' problem values in the proximity of the original problem values, upper and lower limits on the perturbation can be introduced. The solution subset S corresponding to the problem set P can be created by the application of an heuristic, h, $S = \{h(p_j), j = 1, \ldots, m\}$.

PSGA is different from the hybrid GAs (Jones and Beltramo, 1991; Laszewski, 1991) for the GPP. Jones and Beltramo (1991) solve the partitioning problems using GAs by encoding partitions by two different methods; one method uses group-number encoding in which partitions are encoded as strings of group numbers and the other method uses permutation encoding in which partitions are encoded as permutations of N-objects and $K - 1$ group separators. They use a special crossover operator for each of the encoding methods, otherwise crossover may result in illegal solution. Laszewski (1991) uses a parallel GA to solve the partitioning problem. The selection of mates is limited to a local neighborhood and an intelligent structural crossover operator was used which copies a whole partition from one solution into another in order to avoid loosing information from one generation to another. In PSGA (Storer *et al.*, 1992; Dhodhi, 1992) the chromosome is based on the problem data and all the genetic operators are applied in the problem space, so there is no need to modify genetic operators for each application. The solution is obtained by applying a simple and fast known heuristic to map from problem space to solution space, where each chromosome guides the heuristic to generate a different solution.

The PSGA-based technique (Storer *et al.*, 1992; Dhodhi, 1992) offers several advantages over the simulated annealing-based approach as well as traditional GAs. By operating in problem-space, standard crossover operators are trivially constructed and applied on the perturbed problem data. PSGA uses a fast heuristic to map from problem-space to solution space, therefore it avoids disadvantages of probabilistic approaches such as local fine tuning in the last stage of traditional GAs and, moreover, PSGA has a fast convergence rate as compared with standard GA. The PSGA-based technique is objective independent, and thus it can be applied to combinatorial optimization problems with any objective functions. The rest of the paper is organized as follows. Section 2 discusses the proposed heuristic technique for the GPP. Experimental results are provided in Section 3. Section 4 concludes the paper.

2. THE PARTITIONING STRATEGY

In this section, first we formalize the GPP and then give a summary of the proposed graph partitioning strategy followed by its detailed discussion.

2.1 Problem formulation

Consider a finite vertex-weighted, edge-weighted, undirected multigraph $G = (V, E, \Omega_1, \Omega_2)$, where $V = \{v_1\ v_2, \ldots, v_N\}$ is the set of nodes, $E \subseteq V \times V$ is the set of edges, $\Omega_1: E \rightarrow Z^+$ defines the weights on edges and $\Omega_2: V \rightarrow Z^+$ defines the weights on nodes, where Z^+ is the set of positive integers. For example, a graph with nine nodes adopted from Sarje and Sagar (1991) is shown in Figure 1. The m-way graph partitioning problem is to find a mapping function:

$$\Pi : V \rightarrow \{\pi_1, \pi_2, \pi_3, \ldots, \pi_m\} \text{ where } \pi_i = \{v \in V | \Pi(v) = \pi_i\} \forall i | 1 \leqslant i \leqslant m \quad (1)$$

such that

$$\pi_i \cap \pi_j = \phi \text{ for } i \neq j \text{ and } \cup \pi_i = V \forall i | 1 \leqslant i \leqslant m \quad (2)$$

Let $S(\pi_i)$ denote the size of subset π_i and be defined as $\Sigma\Omega_2(v) \forall v \in \pi_i$ and let $C_{i,j}$ denote the weighted

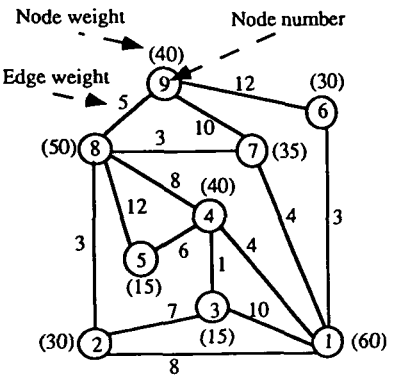

FIGURE 1. A graph with nine nodes.

cut-size between two subsets π_i and π_j of Π and be defined as $\Sigma\Omega_1(u,v)\forall(u,v) \in E|u \in \pi_i$ and $v \in \pi_j$. Now the objective of partitioning is to minimize the weighted cut-size of Π:

$$\text{Cut } (\Pi) = \Sigma C_{i,j} \ \forall i,j|1 \leqslant i < j \leqslant m \tag{3}$$

such that the imbalance $W(\Pi)$ between the size of each π_i is minimum.

$$W(\Pi) = \Sigma|S(\pi_i) - S(\pi_j)| \ \forall i,j|1 \leqslant i < j \leqslant m \tag{4}$$

2.2. Outline of the algorithm

The outline of the proposed scheme is depicted in Figure 2. First we read the graph and build a database which includes the adjacency list for each node in the graph. Then we get the user defined parameters such as the population size N_p, the number of generations N_g, the crossover rate P_c, the mutation rate P_m, the number of disjoint subsets into which we want to divide the nodes of the given graph m, scaling factor α which controls the perturbation range in the proximity of the problem data and weight parameters ω_1, ω_2. The first chromosome in the initial population is built based on problem data and the rest of the chromosomes are generated by perturbing the first chromosome. Then we apply the partitioning heuristic to generate a solution for each chromosome in the initial population, and evaluate the cost and fitness for each solution. We save the current best solution in a database. Then we select chromosomes from the current population based on their fitness, and apply crossover and mutation to generate a new population. The whole process of applying partitioning heuristic and cost and fitness calculation is repeated until the termination criterion (the number of generation N_g) is met. Finally, we report the best solution.

2.4. Initial population

An initial population of size 6 for the graph in Figure 1 is shown in Figure 3. Each chromosome in the initial population for the GPP at hand consists of an array of real numbers representing the priority for each node of the graph. The priority $(P_{ro})^i$ of each node i for the first

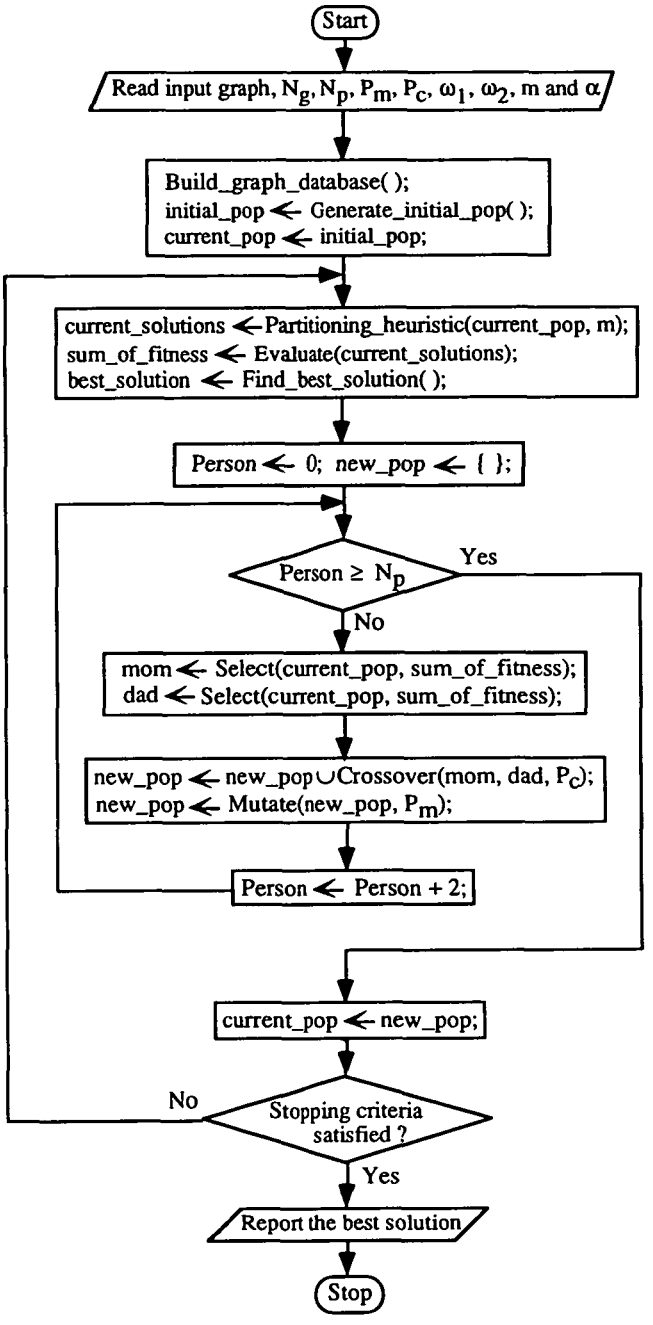

FIGURE 2. Outline of the proposed scheme.

chromosome is computed as below:

$$(P_{ro})^i = \Omega_2(v_i) * \text{Random}(0,1) * N; \tag{5}$$

where N is number of nodes in a graph. The objective is to keep some knowledge about the problem data in determining the priority of the nodes. The rest of the chromosomes in the initial population are generated by a random perturbation in the priority as given below:

$$(P_r)^i = (P_{ro})^i + \text{Uniform } (-\eta, \mu) \tag{6}$$

Where $(P_{ro})^i$ is the priority of node i in the first chromosome based on the original problem data, Uniform $(-\eta, \mu)$ is a random number generated uniformly between $-\eta$ and μ. We are taking the value

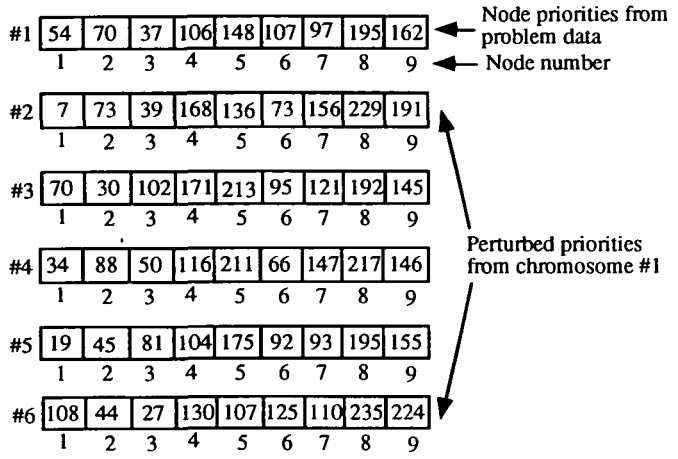

FIGURE 3. An initial population of six chromosomes.

of $\eta = \mu = \text{Max}_i\{(P_{ro})^i\}/\alpha$, but η and μ do not need to have same values. $(P_r)^i$ is the priority for node i of the chromosome calculated by perturbing the original problem data. α is a user defined parameter to keep the lower bound (η) and upper bound (μ) on the perturbed dummy values in proximity to the original problem. As one can see from Figure 3, each chromosome has a different priority value for each node in different chromosomes. So each chromosome guides the heuristic to generate a different solution. We show a small population size for demonstration purposes only. Once the initial population (G^0) is constructed, the PSGA-based technique generates subsequent populations G^1, $G^2, \ldots, G^{i-1}$, G^i by applying the genetic operators (selection, crossover and mutation).

2.5. Partitioning heuristic

Pseudocode for the partitioning heuristic is given in Figure 4. The inputs to this heuristic are the chromosome and m, the number of disjoint subsets into which we want to divide the nodes of the given graph. The partitioning heuristic was applied to generate a solution from a given chromosome with the objective of balancing the size $S(\pi_i)$ among given m-subsets. In this heuristic the nodes in each chromosome of the population are sorted into descending order according to their priority. Then the partitioning heuristic selects a node with the highest priority and assigns it to one of the subsets π_i which has the lowest size $S(\pi_i)$ currently among the given m-subsets to keep the size of each subset nearly equal. The size of each subset is the sum of the weight of all the nodes mapped to that subset.

```
Partitioning_heuristic (chromosome, m):
        Build priority_list of nodes based on their priority value in the chromosome.
        while (priority_list <> null) do begin
                Select πi with minimum S(πi) currently.
                Select the node j from priority_list with the highest priority.
                Assign node j to πi.
                Update S(πi) by adding to it Ω2(vj).
                Delete node j from priority_list.
        end while
End Partitioning_heuristic.
```

FIGURE 4. Pseudocode for the partitioning heuristic.

The partitioning heuristic gives the size $S(\pi_i)$ of each subset, the cut-size Cut(Π) and the imbalance $W(\Pi)$ of each solution generated by this heuristic is computed by a separate procedure using (3) and (4). When we apply the above heuristic the solutions for each chromosome in the initial population are shown in Table 1 with the value of m as 3.

2.6. Cost and fitness function

The cost function is the key issue as it reflects the goal of the optimization. We take the sum of both the weighted cut-size Cut(Π) (3) and the imbalance $W(\Pi)$ (4) as a cost function as shown below:

$$\text{Cost(i)} = \omega_1 * \text{Cut}(\Pi) + \omega_2 * W(\Pi) \tag{7}$$

Where ω_1, ω_2 are the weight parameters which control the objective function, whether we want the minimum cut or more balanced partitions. The following cost-to-fitness mapping function was used to calculate the fitness of each chromosome (Storer *et al.*, 1992):

$$f(i) = \frac{(\text{MaxCost} - \text{Cost}(i))^\tau}{\sum_{j=1}^{j=N_p}(\text{MaxCost} - \text{Cost}(j))^\tau} \tag{8}$$

Where $f(i)$ is fitness of chromosome i, MaxCost is the maximum cost of a chromosome in the population, Cost(i) is the cost of chromosome i, N_p is the population size, and τ is a parameter which is used for fitness scaling to balance convergence and diversity. Generally τ is in the range of 1 to 5. The fitness of each chromosome in the initial population is given in Table 1 with the value of τ as 3 and $\omega_1 = \omega_2 = 1$.

2.7. Selection, crossover and mutation

Genetic operators such as selection, crossover and mutation are the key elements of GAs. The selection operator selects chromosomes for reproduction from the current population based on their relative fitness. Chromosomes with higher fitness will have a higher probability of contributing one or more offspring in the next generation. The selection method was implemented using a biased roulette wheel where each chromosome in the population has a slot sized in proportion to its fitness (Holland, 1975; Goldberg, 1989). Each time we require an offspring, a simple spin of the weighted roulette wheel gives a parent chromosome. The chromosomes selected for reproduction from an initial population depending upon their fitness are given in Figure 5.

The crossover operator takes two parent chromosomes selected by the selection operator from the current population and generates two children by incorporating features from both parents. The premise here is that through this process desirable features are enhanced while most undesirable features are suppressed. We have applied a one-point crossover operator to the priority of the chromosome. In the one-point crossover operator, a

TABLE 1. Cost and fitness calculation for population G^0

Solution	π_1	π_2	π_3	$W(\Pi)$	$Cut(\Pi)$	*Cost*	*Fitness*
1	{1,2,8}	{3,4,9}	{5,6,7}	120	82	204	0.000000
2	{1,2,8}	{3,7,9}	{4,5,6}	110	69	179	0.013368
3	{5,6,9}	{1,3,8}	{2,4,7}	80	74	154	0.106943
4	{1,2,8}	{5,6,9}	{3,4,7}	110	72	182	0.009110
5	{6,8}	{2,3,4,5}	{1,7,9}	110	68	178	0.015037
6	{1,8}	{3,5,6,9}	{2,4,7}	20	84	104	0.855542

{1,2,8} means node v_1, v_2, v_8

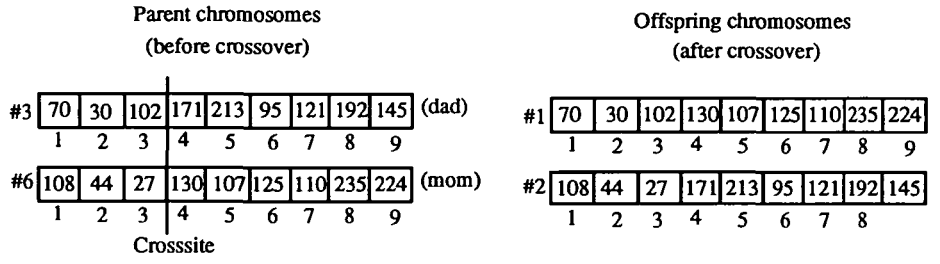

FIGURE 5. Crossover to generate a new population.

cross site is selected randomly and the value of the priority to the right of the cross site is swapped among the two mating chromosomes. The crossover is applied with a certain crossover rate (P_c), which is the ratio of the number of offspring produced by crossover in each generation to the population size. It controls the amount of crossover being applied.

In nature, mutation refers to spontaneous and random changes in genes. In a GA-based approach, mutation introduces new features into the current population by altering a randomly picked gene value. Mutation was implemented by selecting a gene at random with a mutation rate P_m and perturbing its value. The mutation rate (P_m) is the percentage of the total number of genes in the population which are mutated in each generation. The default values for the crossover rate and the mutation rate were selected based on simple GAs studies, i.e. $P_c = 0.6$ and $P_m = 0.001$. In Figure 5, chromosomes selected for crossover, the crossover site for the priority is shown to form a new population. We do not show the mutation operator in Figure 5 for the sake of simplicity.

2.8. Parameter tuning

We experimented with various population sizes N_p, number of generations N_g, crossover rates P_c and mutation rates P_m to determine parameter values which give good results at a reasonable computation cost. A population size between 150 and 200, number of generations between 80 and 100, $P_c = 0.6$ and $P_m = 0.001$ are sufficient to arrive at good solutions for the GPP.

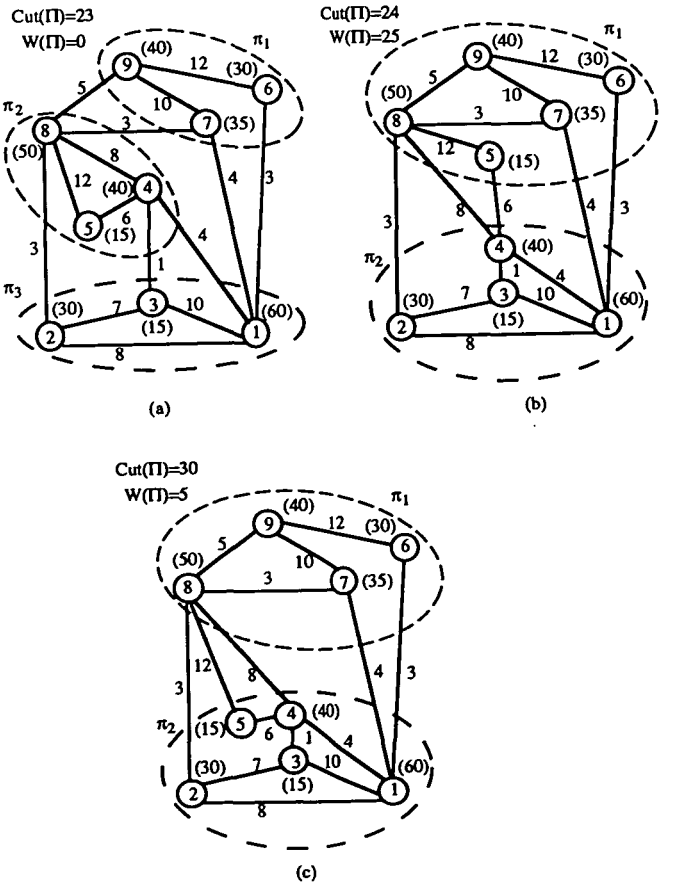

FIGURE 6. Partitioning of the graph into (a) three subsets, (b) two subsets with minimum cut-size and (c) with minimum imbalance.

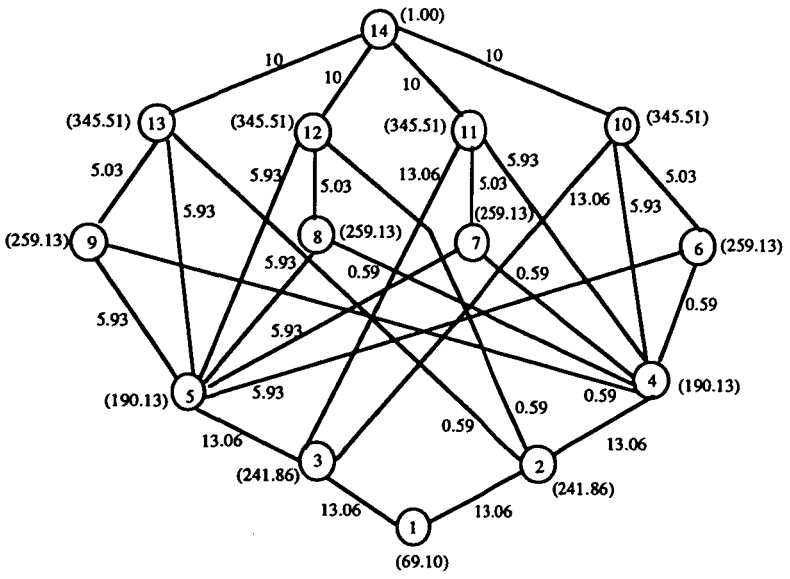

FIGURE 7. A 14 node FFT graph.

The solution for the graph of Figure 1 for $m = 3$ is shown in Figure 6(a). The solution in Figure 6(b) shows the minimum cut-size, while Figure 6(c) shows minimum imbalance for $m = 2$. One can generate different solutions by controlling the values of weight parameters ω_1, ω_2.

3. EXPERIMENTAL RESULTS

The described PSGA for the GPP has been implemented in C on a SUN SPARCstation 10 and has been tested on many test examples. We report results for previously published examples (Lee *et al.*, 1989; Tao and Zhao, 1993; Sarje and Sagar, 1991) and on random graphs (Johnson *et al.*, 1989; Tao and Zhao, 1993). The first example is a 14 node FFT graph adopted from Sarje and Sagar (1991) as shown in Figure 7. Sarje and Sagar's (1991) objective is to partition this graph to minimize the weighted cut-size with minimum load imbalance among partitions. Comparison of result with Sarje and Sagar is shown in Table 2 for $m = 2, 3$ and 4, respectively. The proposed technique produces better results both in terms of the weighted cut-size and the imbalance among partitions. The CPU time is less than 1 s on the SPARCstation 10.

Our second example is a 16 node graph adopted from (Lee *et al.*, 1989; Tao and Zhao, 1993) as shown in Figure 8. We are comparing results for this example with Tao and Zhao (1993) as shown in Table 3. The value of the weighted cut-size reported Lee *et al.* (1989) is 72 for $m = 3$, which is an inferior solution to that already reported in Tao and Zhao (1993). The proposed technique produced the same best possible results reported in Tao and Zhao (1993) for $m = 3$. We also show different results for $m = 3$, which will result in smaller cut-size with a small imbalance. The CPU time is less than 1 s on the SPARCstation 10.

We also present results from randomly generated graphs (Johnson *et al.*, 1989) with unity weights on nodes and edges. A standard random graph can be denoted by $R_{N,D}$, where N is the number of nodes and D is the expected degree for each node. Given the values of N and D, the parameter $P = D/(N-1)$ specifies the probability that any given pair of nodes in the graph constitutes an edge. We have implemented the KL algorithm for comparison purposes. For all the test examples, the population size is 200 and the number of generations is 100. The comparison of results for different randomly generated graphs with the KL algorithm for $m = 2$ (bipartitioning) is shown in Table 4. The average cut-size

TABLE 2. Comparison of results for the FFT example

	Proposed Technique			*Sarje and Sagar (1991)*		
m	*nodes partitioning*	*W*(Π)	*Cut*(Π)	*nodes partitioning*	*W*(Π)	*Cut*(Π)
2	{1, 2, 3, 4, 7, 10, 11}, {5, 6, 8, 9, 12, 13, 14}	33.56	46.97	N/A	N/A	N/A
3	{1, 2, 3, 4, 11}, {6, 9, 10, 13, 14}, {5, 7, 8, 12}	312.76	72.17	{1, 2, 3, 4, 10}, {5, 6, 7, 8}, {9, 11, 12, 13, 14}	658.28	78.47
4	{1, 2, 4, 7}, {5, 8, 12}, {3, 10, 11, 14}, {6, 9, 13}	589.98	94.71	N/A	N/A	N/A

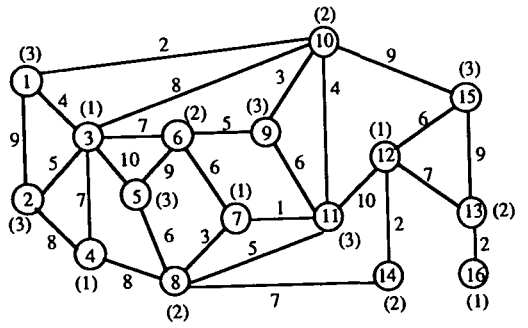

FIGURE 8. Sixteen node graph [adopted from (Lee *et al.*, 1989; Tao and Zhao, 1993)].

were given as an input to our algorithm and the results for cut-size and imbalance for a variety of partitions were obtained. In order to provide a comparative study of the results for weighted graphs obtained by our technique, we have implemented simulated annealing (SA) with the same values of parameters as reported in Johnson (1989). Comparison of results with SA are reported in Table 5 for different values of m. The proposed technique gives better or at least comparable results with less CPU time.

4. CONCLUSIONS

Graph partitioning has useful applications in many disciplines. In this paper, we have proposed a novel scheme based on the PSGA for the m-way GPP to reduce the weighted cut-size by proper balancing of the size

TABLE 3. Comparison of results for the 16 node example

	Proposed Technique			*Tao and Zhao (1993)*		
m	*nodes partitioning*	*W*(Π)	*Cut*(Π)	*nodes partitioning*	*W*(Π)	*Cut*(Π)
2	{1,2,3,4,5,6,7,8,14} {9,10,11,12,13,15,16}	3.0	23.0	N/A	N/A	N/A
	{1,2,4,8,14}, {3,5,6,7,9,16}, {10,11,12,13,15}	0.0	54.0	{1,2,4,8,14}, {3,5,6,7,9,16}, {10,11,12,13,15}	0.0	54.0
3	{1,2,9} {3,4,5,6,7,8,14}, {10,11,12,13,15,16}	6.0	49.0	N/A	N/A	N/A
	{1,2,3,4,5,6,7,8}, {9,11,14}, {10,12,13,15,16}	16.0	47.0	N/A	N/A	N/A
4	{1,2}, {4,8,14,16} {3,5,6,7}, {9,10,11,12,13,15}	25.0	58.0	N/A	N/A	N/A
	{1,2,14} {3,4,5,6,7,8}, {9,10,11}, {12,13,15,16}	9.0	66.0	N/A	N/A	N/A

is reported for the KL and the proposed technique for 10 test runs of each graph. The proposed technique produces better or at least the same average cut-size as obtained by using the KL algorithm. The proposed scheme takes a longer time as compared with the KL algorithm. The proposed technique is not intended for graph with unity weight on nodes and edges.

The KL algorithm can neither handle the graphs with weight on nodes and edges nor can it be used to partition a graph into an odd number of subsets (partitions), whereas the proposed technique can be used to partition a weighted graph into any number of partitions. We have generated random weighted graphs with different degrees and number of nodes. The weights of the nodes and the edges are selected randomly between an integer range of 1–10 for 50 node graphs and between an integer range of 1–20 for 100 and 150 node graphs. These graphs

TABLE 4. Comparison of results for bipartitioning of random graphs

Nodes (*N*)	*Degree* (*D*)	*Edges* (*E*)	*Proposed technique* *Cut*(Π)	*KL* *Cut*(Π)
20	4	42	10	10
	6	61	19	20.1
	8	78	27	28.2
40	4	78	16	18.6
	8	170	55	55.1
	12	237	81	81.2
60	4	115	26	26
	8	232	71	71.9

TABLE 5. Comparison of results for m-way partitioning of random graphs

Nodes (N)	Degree (D)	Edges (E)	m	Proposed			SA		
				W(Π)	Cut(Π)	CPU time (s)	W(Π)	Cut(Π)	CPU time (s)
			4	13	231	2.0	15	240	5.0
50	4	96	6	47	290	2.0	49	311	5.0
			8	15	337	2.0	29	362	6.0
			6	13	696	2.0	25	699	5.0
50	8	199	8	51	736	2.0	19	761	6.0
			10	95	771	2.0	45	780	8.0
			4	18	976	2.0	6	954	7.0
50	12	312	8	43	1243	2.0	15	1292	8.0
			12	99	1408	3.0	35	1456	10.0
			10	107	3225	4.0	95	3238	16.0
100	8	398	15	478	3334	5.0	192	3784	18.0
			10	113	4961	4.0	49	5145	17.0
100	12	596	15	584	5163	5.0	458	5181	20.0
100	16	820	15	502	7280	5.0	260	7173	24.0
			10	70	4861	7.0	66	4877	27.0
150	8	596	15	444	4904	8.0	172	5275	35.0
			10	108	7560	7.0	76	7684	28.0
150	12	893	15	394	7821	8.0	146	7824	38.0
			10	110	10077	8.0	24	10233	36.0
150	16	1196	15	462	10703	9.0	132	11268	40.0

of each subset. The PSGA-based scheme combines the power of GAs, a global search method, with a known fast heuristic to search a large solution space in an intelligent way in order to find a best possible solution within acceptable CPU times. It has been demonstrated through experimental results that the proposed algorithm is better in terms of both the CPU times and the quality of solution over previous work. Our algorithm can easily be adapted for hypergraph partitioning. The parallel implementation of the algorithm and for mapping to special architectures for multichip designs can be extended very easily.

ACKNOWLEDGEMENT

This research is funded by Kuwait University grant EE-060.

REFERENCES

[1] Bokhari, S. H. (1987) *Assignment Problems in Parallel and Distributed Computing*. Kluwer, Dordrecht.

[2] Dhodhi, M. K. (1992) *Data Path Synthesis Using Concurrent Scheduling and Allocation Based on Problem-Space Genetic Algorithms*, PhD Dissertation, Department of Electrical Engineering and Computer Science, Lehigh University, Bethlehem, PA.

[3] Dunlop, A. E. and Kernighan, B. W. (1985) A procedure for placement of standard-cell VLSI circuits. *IEEE Trans. Computer-Aided Design of Integrated Circuits & Systems*, **CAD-4**, 92–98.

[4] Fiduccia, C. M. and Mattheyses, R. M. (1982) A linear-time heuristic for improving network partitions. In *Proc. IEEE/ACM 17th Design Automation Conf.*, pp. 175–181.

[5] Gajski, D. D., Dutt, N. D., Wu, A. C. and Lin, S. Y. (1992) *High-Level Synthesis: Introduction to Chip and System Design*. Kluwer, Dordrecht.

[6] Garey, M. R. and Johnson, D. S. (1979) *Computers and Intractability: A Guide to the Theory of NP Completeness*. W.H. Freeman, San Francisco, CA.

[7] Glover, F. (1989) Tabu Search—Part 1. *ORSA J. Comput.*, **1**, 190–206.

[8] Goldberg, D. E. (1989) *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA.

[9] Holland, J. H. 1975) *Adaption in Natural and Artificial Systems*. MIT Press, Cambridge, MA.

[10] Johnson, D. S., Aragon, C. R., Mcgeoch, L. A. and Schevon, C. (1989) Optimization by simulated annealing: an experimental evaluation; Part I, graph partitioning. *Operations Res.*, **37**, 865–892

[11] Jones, D. R. and Beltramo, M. A. (1991) Solving partitioning problem with genetic algorithm. In *Proc. Int. Conf. on Genetic Algorithms*, pp. 442–449.

[12] Kernighan, B. W. and Lin, S (1990) An efficient heuristic procedure for partitioning graphs. *Bell Syst. Tech. J.*, **49**, 291–307.

[13] Laszewski, G. V. (1991) Intelligent structural operators for the K-way graph partitioning problem. In *Proc. Int. Conf. on Genetic Algorithms*, pp. 45–52.

[14] Lee, C. H., Park, C. I. and Kim, M. (1989) Efficient algorithm for graph-partitioning problem using a problem transformation method. *Computer-Aided Design*, **21**, 611–618.

[15] Sarje, A. K. and Sagar, G. (1991) Heuristic model for task allocation in distributed computer systems. *Proc. IEEE*, **138**, 313–318.

[16] Shahookar, K. and Mazumder, P. (1990) A genetic approch to standard cell placement using meta-genetic parameter optimization. *IEEE Trans. Computer-Aided Design of Integrated Circuits & Systems*, **CAD-9**, 500–511.

[17] Sherwani, N. (1993) *Algorithms for VLSI Physical Design Automation*. Kluwer, Dordrecht.

[18] Storer, R. H., Wu, D. S. and Vaccari, R. (1992) New search space for sequencing problems with application to job shop scheduling. *Management Sci.*, **38**, 1495–1509.

[19] Tao, L. and Zhao, Y. C. (1993) Effective heuristic algorithms for VLSI circuits partition. *IEE Proceedings-G*, **140**, 127–134.

[20] Van den Bout, D. E. and Miller, T. K. (1990) Graph partitioning using annealed neural networks. *IEEE Trans. Neural Networks*, **1**.