Solving Large Knowledge Base Partitioning Problems Using an Intelligent Genetic Algorithm

Shinn-Ying Ho*, Hung-Ming Chen and Li-Sun Shu
Department of Information Engineering, Feng-Chia University, Taichung, Taiwan, Republic of China
* E-mail: syho@fcu.edu.tw
TEL: 886-4-4517250 Ext. 3702 FAX: 886-4-4516101

Abstract

Developing large knowledge bases that are complex enough to be useful in real-world applications may result in large complicated systems that partitioning the system into smaller subsystems is an absolute requirement. The problem of allocation of production rules among several partitions of limited size such that the sum of inter-partition connections is minimized is termed as the knowledge base partitioning problem. In this paper, a novel intelligent genetic algorithm (IGA) is proposed to solve the *large* knowledge base partitioning problem which is a well-known NP-complete problem. IGA uses a new intelligent crossover based on the ability of orthogonal arrays that the chromosomes of the children are formed from the best combinations of the better genes representing variables of a function from the parents rather than the random combinations of parents' genes. It is shown empirically that the proposed general-purpose IGA needing no heuristic outperforms the existing methods, heuristic clustering, simple genetic algorithm, and heuristic evolutionary algorithm, available in the literature in solving knowledge base partitioning problems using the same benchmark, especially in solving very large partitioning problems.

1. INTRODUCTION

Allocation of production rules among several partitions in knowledge base can shorten the compilation and execution of expert systems applications and facilitate their verification, validation and maintenance. Developing large knowledge bases that are complex enough to be useful in real-world applications may result in large complicated systems that partitioning the system into smaller subsystems is an absolute requirement. The problem of allocation of production rules among several partitions of limited size such that the sum of inter-partition connections is minimized is termed as the knowledge base partitioning problem. This problem is formulated as a 0-1 integer programming problem with a quadratic objective function which is known to be NP-complete (Raz and Botten 1992). This partitioning problem theoretically have been shown to be equivalently hard to the existing partitioning problems in theoretical computer science, such as network partitioning problem, circuit partitioning problem, and VLSI network partitioning problem (B. KrishnaMurthy 1984). Therefor, how to minimize the sum of inter-partition connections for knowledge base partitioning problems (KBPPs) in a reasonable amount of computing time is a very important problem in real-world applications.

The important literatures related to KBPPs can be briefly described as follows. A clustering algorithm based on the nearest neighbor heuristic is first proposed to solve KBPPs with fast execution speed (Raz and Botten 1992). In 1995, Dev et al. proposed a genetic algorithm (GA) in the form of a probabilistic heuristic that the results of GA have better quality solutions than those obtained by the clustering algorithm (Dev et al. 1995). A critical review of the existing GA can be found in (Dutta et al. 1997). Dutta et al. have proposed a heuristic evolutionary algorithm (HEA) which outperforms the existing algorithms for solving knowledge base partitioning problems using the same benchmark (Dev et al. 1995). The authors highlight three things that the standard error of the experimental results is down to a very small level and the future research may be directed towards faster convergence, along with the consideration of bringing the average result close to the best result.

In this paper, we solve KBPPs using an intelligent genetic algorithm (IGA) which is an efficient general-purpose algorithm capable of solving large parameter optimization problems. Theoretical analysis and experimental studies of IGA can be found in our recent work (Ho et al. 1999). We try to achieve the following goals:

(1) Show empirically that the proposed IGA outperforms the existing methods: heuristic clustering (Raz and Botten 1992), simple genetic algorithm (Dev et al. 1995), and heuristic evolutionary algorithm (Dutta et al. 1997), for solving KBPPs using the same benchmark. The comparisons of performance evaluation takes the following into account: (1) the best solution; (2) quality of average solutions; (3) very small variance, i.e., robust; (4) fast convergence speed.
(2) In real-world applications, the knowledge base may be larger than the used benchmark. To compare the performance for solving large KBPPs with the competitive GA-based algorithm, we enlarge the size of the benchmark and apply it to test all participant algorithms. From the simulation results, we hope to

show that IGA is more superior to other algorithms for very large partitioning problems.

The knowledge base partitioning problem is given in Section 2. An IGA is also introduced in Section 3. Performance comparisons among IGA and the competitive methods are given in Section 4. Finally, conclusions are made in Section 5.

2. PROBLEM STATEMENT

For the sake of completeness and to make the present endeavor complete, the following problem statement is abstracted from (Dev et al. 1995). The total knowledge base consists of production rules, and the related rules (rules pertaining to a particular topic) can be grouped together to form areas. The knowledge base partitioning problem involves an optimal assignment of these rules/areas to different chunks or partitions so as to minimize the switching among partitions. The assignment of the rules to various partitions has to satisfy the following two constraints:

(1)The assignment must be exhaustive mutually exclusive, i.e., each and every area must be assigned to exactly one partition.

(2) The size of partition should not exceed a specified maximum size.

The objective function to be optimized reflects the extent to which the partitions are independent of each other as measured by the number of connections across partitions. A connection is defined as an instance of a rule with a premise element in one partition and a conclusion element in a different partition. Mathematically the problem is posed as follows. The following notation is used:

N	= total number of areas to be assigned among M partitions.
x_{ik}	= 1 if area i is assigned to partition k, = 0 otherwise.
a_{ij}	= weight of the connection between area i and area j.
A	= adjacency matrix containing the weights of the connections between areas (a_{ij}).
b_i	= size of area i, in terms of number of rules or some relevant measure.
B	= maximum partition size.

The smallest number of partitions that permit a feasible solution is the smallest integer not smaller than $(\sum_{i=1}^{N} b_i)/B$. For any partition k, the quantity $x_{ik}(1\text{-}x_{jk})$ is equal to 0 if two areas i and j belong to partition k, and is equal to 1 if area i belongs to partition k and area j belongs to a different partition. Consequently, the objective function may be formulated as

$$MIN \sum_{k=1}^{M} \sum_{i=2}^{N} \sum_{j=1}^{i-1} x_{ik}(1-x_{jk})a_{ij} \,. \tag{1}$$

There are three sets of constraints:

(1) The decision variables x_{ik} must be 0 or 1:

$x_{ik} \in \{0,1\}$ for all areas $i = 1, \ldots, N$ and all partitions $k = 1, \ldots, M$. (2)

(2) Each area must belongs to exactly one partition:

$$\sum_{k=1}^{M} x_{ik} = 1 \text{ for all areas } i = 1, \ldots, N. \tag{3}$$

(3) The partition size should not exceed the limit B:

$$\sum_{i=1}^{N} x_{ik} b_i \le B \text{ for all partition } k = 1, \ldots, M. \tag{4}$$

3. AN INTELLIGENT GENETIC ALGORITHM

The IGA uses a novel intelligent crossover (IC) based on the ability of orthogonal arrays (OAs). The principle of the IC approach relies on OAs which are descried in Section 3.1. Section 3.2 presents the use of OAs to achieve intelligent crossover. The IGA is provided in Section 3.3. To find the optimal solution economically with limited small population sizes, a modified IGA with a two-stage IC is proposed in Section 3.4.

3.1 ORTHOGONAL ARRAY AND FACTOR ANALYSIS

Orthogonal Arrays (OAs) and factor analysis, which are representative methods of quality control (Taguchi and Konishi 1987), also work to improve the crossover operator more efficiently. We provide a definition of the OA as follows. Let there be N factors of two levels. The number of total combinations is 2^N. Columns of two factors are orthogonal when 4 pairs, (1, 1), (1, 2), (2, 1), and (2, 2), occur equally in all experiments. When any two factors in an experimental set are orthogonal, the set is called an OA. To establish an OA of N factors of two levels, we obtain an integer $n=2^{\lceil log(N+1)\rceil}$, build an orthogonal array $L_n(2^{n-1})$ with n rows and (n-1) columns, and select N columns.

Factor analysis can evaluate the effects of factors on the evaluation function, rank the most effective factors, and determine the best level for each factor such that the evaluation function is optimized. Orthogonal experiment design can reduce the number of experiments for the factor analysis. The number of OAs for single factor analysis is only n. For instance, Table 1 shows an orthogonal array $L_8(2^7)$.

Let y_t be the positive function evaluation value of experiment no. t. Define the main effect of factor j with level k S_{jk},

$$S_{jk} = \sum_{t=1}^{n} Y_t^2 \times [\textit{the level of Exp no t of factor j is k}], \tag{5}$$

where

$$[condition] = \begin{cases} 1 & \textit{if the condition is true} \\ 0 & \textit{oterwise,} \end{cases} \tag{6}$$

and

$$Y_t = \begin{cases} y_t & \textit{if the function is to be maximized} \\ 1/y_t & \textit{if the function is to be minimized.} \end{cases} \tag{7}$$

Note that the main effect reveals the individual effect of a factor. The most effective factor j has the largest main effect difference (MED) $|S_{j1}\text{-}S_{j2}|$. If main effect $S_{j1}>S_{j2}$, the level 1 of factor j is better than the level 2 on the contribution for the optimization function. Otherwise, level 2 is better.

3.2 INTELLIGENT CROSSOVER

Genetic algorithm (GA) uses binary variables of a function to represent a chromosome. For instance, a function with N variables of l bits is encoded in binary codes of Nl length in a chromosome, or Nl binary variables. The representation of chromosomes for IC approach is the same as that of traditional GA. Two parents breed two children using IC at a time. How to use the OA to achieve the IC is described as the following steps.

Step 1: Select the first N columns of OA $L_n(2^{n-1})$ where $n = 2^{\lceil \log(N+1) \rceil}$. Note that one variable of a function is regarded as a factor in OA.

Step 2: Let level 1 and level 2 of factor j represent the jth variable of a function coming from the parent 1 and parent 2, respectively.

Step 3: Evaluate the function values y_t for experiment no. t where $t = 1, 2, \ldots, n$.

Step 4: Compute the main effect S_{jk} where j = 1, 2, …, N and $k = 1, 2$.

Step 5: Determine the best level for each variable. Select level 1 for the jth variable if $S_{j1} > S_{j2}$. Otherwise, select level 2.

Step 6: The chromosome of the first child is formed from the best combinations of the better variables from the derived corresponding parents.

Step 7: Rank the most effective factors from rank 1 to rank N. The factor with large MED has higher rank.

Step 8: The chromosome of the second child is formed similarly as the first child except that the variable with the lowest rank adopts the other level.

3.3 INTELLIGENT GENETIC ALGORITHM

Our IGA can be written as follows:

Step 1: Initialization: Randomly generate an initial population of N_{pop} individuals, I_i, i=1,2, …, N_{pop}.

Step 2: Elitist strategy: Repeat the following steps for i =1 to N_{pop}-1:

- 2a: Select I_i and I_i as the parents and produce the two children I_{c1} and I_{c2} using IC.
- 2b: Replace I_i and I_{i+1} with the second and the best individuals using fitness performance among I_i, I_{i+1}, I_{c1} and I_{c2}, respectively.

Step 3: Evaluation: Evaluate the function values for all individuals.

Step 4: Selection: Use the rank selection that replace the worst $N_{pop} \times P_s$ individuals by the best $N_{pop} \times P_s$ individuals to form the new population. According to the selection probability P_c, select $N_{pop} \times P_c$ parents for intelligent crossover operations.

Step 5: Crossover: Apply IC to the selected pairs of parents. The two children are replaced by two individuals with the better fitness function values among the parents and children for the elitist strategy.

Step 6: Mutation: Apply the mutation operator to the generated new population using mutation probability P_m. To prevent the fitness value from deteriorating, mutation is not applied to the best individual.

Step 7: Termination test: If a prespecified stopping condition is satisfied, end the algorithm. Otherwise, return to step 3.

It is note that the individuals generated by OA's experiments may be an infeasible solution which is not preferred by IC. The penalty-based approach is a good method for efficient use of IC. In solving knowledge base partitioning problem, we modify the objective function as follows:

$$MIN \sum_{k=1}^{M} \sum_{i=2}^{N} \sum_{j=1}^{i-1} x_{ik}(1 - x_{jk}) a_{ij} + M \cdot P(s) \qquad (8)$$

where $P(s)$ is the total size of overflow for infeasible solution s. If s is a feasible solution, $P(s)$ is equal to zero.

3.4 IGA WITH A TWO-STAGE IC

The crossover is the main operator for global search. Although single-point crossover was inspired by biological process and is used in traditional GA frequently, it has one major drawback in that certain combinations of schema cannot be combined in some situation (Z. Michalewicz 1994). Therefore, it is an inferior crossover of traditional GA in solving large parameter optimization problems. The multipoint crossover can be used to improve the performance of generating offspring in the aspect of diversification. Nevertheless, the preference of which crossover techniques to use is problem-dependent (Spears and DeJong 1991). However, traditional single-point and multipoint crossover operators function well for small string lengths, but not efficient for large string lengths due to the small variance between the parents and the children compared to the huge search space. Another useful approach is the bit mask crossover which generates offspring from the parents based on a randomly generated crossover mask (Z. Michalewicz 1994). The resultant offspring contains a mixture of genes from each parent. Therefore, the bit mask crossover performs well than the single-point and multipoint crossover operators for a large string length based on the high variance of each subsrting representing each parameter. The responsibility of the crossover is to maintain a good balance between exploiting the currently good regions and exploring new regions where better solutions may be found. Since the good substrings are hardly survived due to the violation of the bit mask crossover, it is better to consider the performance of individual substrings rather than wholes strings and choose the better individual substrings from each parent to form the chromosomes of the children using their combinations.

While very small size population is used for economical purpose, the performance of generating offspring in the aspect of diversification for IC in solving large optimization problems is relatively not superior. For making use of the advantages of both the bit mask for diversification and IC for large string length, a modified IGA (IGA2) with an efficient two-stage IC (IC2) is proposed to cope with this problem. At stage 1 of IC2, the bit masks are first used to modify the parents' chromosomes for achieving wide exploitation and at stage 2 of IC2, the IC described earlier is then used. In this way the ability of improvement between the parents and the children in one generation can be increased greatly.

4. PERFORMANCE EVALUATION

In order to demonstrate the superiority of our algorithm, we compare its performance with those of heuristic clustering (HC), simple genetic algorithm (SGA), and heuristic evolutionary algorithm (HEA) proposed in the literatures (Raz and Botten 1992), (Dev et al. 1995), and (Dutta et al. 1997), respectively. The KBPP problem involved the design of knowledge base with 178 rules divided 15 topical areas. Full details of this problem are available in (Raz and Botten 1992). The entries a_{ij} of adjacency matrix A, i.e., benchmark available in (Dev et al. 1995) and shown in Table 2, represent the number of rules in area i that either have premise element in area i and conclusion element in area j or vice versa.

4.1 PERFORMANCE EVALUATION USING THE BENCHMARK

We have executed our algorithms, IGA and IGA2, using the same number of function evaluation evaluations used by Dev et al. In each case we run our algorithms using parameters: P_s=0.2, P_c=0.5, P_m=0.05 (for IGA) and P_m=0.4 (for IGA2). Twenty independent runs were conducted for each case and the simulation results are averaged with different random seeds. The comparative results of IGA and IGA2 are added to the Table 3 which the other partial results are abstracted from (Dutta et al. 1997). From the experimental result, it can be seen that:

(1) From Table 3, we can see that the quality of solutions obtained by IGA and IGA2 is superior to the participate algorithms in performance of the best value and average value.
(2) IGA and IGA2 outperform the simple genetic algorithm in convergence speed and accuracy, as show in Fig. 1.
(3) The comparisons of variance of solutions are shown in Fig. 2. It can be seen that the variance of IGA2 is superior to IGA which is superior to other algorithms.

4.2 ENLARGED BENCHMARK TEST

To compare the efficiency of the participated algorithms in solving the large knowledge base partitioning problems, we need a large knowledge base. In order to permit replication, we replicate the benchmark $A = [a_{ij}]$ to form the new enlarged benchmark A_k.

$$A_k = \begin{bmatrix} A_{k-1} & A_{k-1} \\ A_{k-1} & A_{k-1} \end{bmatrix} \qquad (9)$$

where A_0=A and k is a positive integer. Furthermore, the size of area i, b_i, is replaced by $b_i \times 2^{k-1}$. The parameters of IGA are P_s=0.2, P_c=0.5, P_m=0.05, N_{pop}=100, and 200 generations. Since the population size is not small (N_{pop}=100), IGA2 is not applied. The same parameters are used for SGA except that the same stopping condition that function evaluation calls are equal to the function evaluation used by IGA. Fig. 3 summarizes the comparative results. From this figure, we can see that IGA is more superior to SGA and HEA while the size of the used benchmark is larger.

5. CONCLUSIONS

Large knowledge base partitioning problem (LKBPP) is an important problem in real-world applications. This partitioning problem theoretically have been shown to be equivalently hard to the existing partitioning problems in theoretical computer science, such as network partitioning problem, circuit partitioning problem, and VLSI network partitioning problem. Due to the huge search space, traditional GA-based algorithms suffer from both the slow convergence speed and accuracy. In this paper, a novel IGA is used to solve LKBPPs. High performance of IGA and IGA2 is demonstrated from the experimental results that the proposed IGA outperforms the existing methods, heuristic clustering, simple genetic algorithm, and heuristic evolutionary algorithm, for solving KBPPs, especially for solving vary large partitioning problems.

References

Botten, Nancy Adrew Kusiak and Tazi Raz (1989). Knowledge base: Integration, verification, and partitioning. *European Journal of Operational Research 42, 1989.*

De Jong, K. A. (1975). *An analysis of the behavior of a class of genetic adaptive systems.* Doctoral dissertation, U. of Michigan.

Dev, Keshav C. Siva Ram Murthy (1995). A genetic algorithm for the knowledge base partitioning problem. *Pattern Recognition Letters 16, 873-879.*

Dutta, Paramartha (1997). An evolutionary heuristic for knowledge base partitioning problem. *1997 IEEE International Conference on Evolution Computation.*

Goldberg, D. E. (1989). *Genetic Algorithms in search, Optimization and Machine Learning.* Addison – Wesley Publishing Company.

Ho, Shinn-Ying Li-Sun Shu and Hung-Ming Chen (1999). Intelligent Genetic Algorithm with a New Intelligent Crossover Using Orthogonal Arrays. *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference, July 13-17, 1999, Orlando, Florida USA .*

Michalewicz, Z. (1994). *Genetic Algorithm + Data Structure = Evolution Program*, 2nd Ed. Berlin: Springer-Verlag.

Murthy, B. Krishna (1984). An improved min-cut algorithm for partitioning VLSI networks, *IEEE Trans. on Computer,* vol. C-33, no. 5.

Raz, Tzvi and Nancy A. Botten (1992). The Knowledge base partitioning problem: Mathematical formulation and heuristic clustering. *Data and Knowledge Engineering 8, 329-337.*

Spears, W. M. and K. DeJong (1991). An analysis of multi-point crossover. *In Foundations of Genetic Algorithms*, G.J.E. Rawlines, 301-315.

Taguchi, G. and S. Konishi (1987). *Orthogonal Arrays and Linear Graphs.* Dearbon, MI: American Supplier Institute.

Table 1. Orthogonal array $L_8(2^7)$

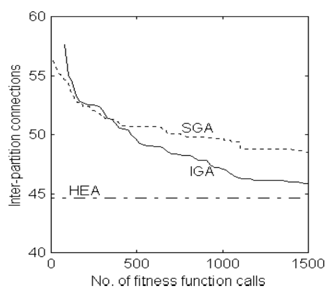
Exp. no.	Factors 1	2	3	4	5	6	7	Function Evaluation value
1	1	1	1	1	1	1	1	y_1
2	1	1	1	2	2	2	2	y_2
3	1	2	2	1	1	2	2	y_3
4	1	2	2	2	2	1	1	y_4
5	2	1	2	1	2	1	2	y_5
6	2	1	2	2	1	2	1	y_6
7	2	2	1	1	2	2	1	y_7
8	2	2	1	2	1	1	2	y_8

Table 2.   Problem benchmark

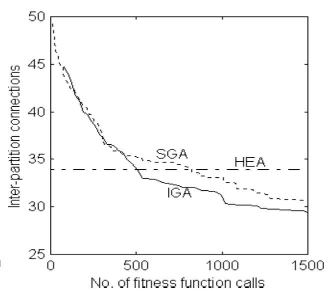
area	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	1	1	1	0	1	2	1	4	1	0	3	3	0	0
2	1	0	1	3	0	0	2	0	4	3	0	0	0	0	0
3	1	1	0	5	0	1	1	1	1	4	2	0	0	0	0
4	1	3	5	0	0	0	2	0	4	3	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0
7	2	2	1	2	0	0	0	2	4	3	0	0	0	0	0
8	1	0	1	0	0	0	2	0	0	0	0	0	1	1	0
9	4	4	1	4	0	0	4	0	0	3	0	0	1	0	0
10	1	3	4	3	0	0	3	0	3	0	0	0	0	0	0
11	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0
12	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	3	0	0	0	0	1	0	1	1	0	0	0	0	0	0
14	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
size	11	10	5	8	9	4	11	16	5	18	17	8	15	22	19

Table 3. Comparative results.

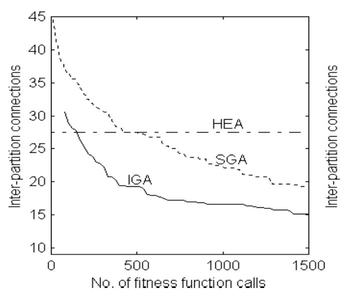
B	N_{pop}	HC	SGA	HEA		IGA		IGA2	
				Best	Avg.	Best	Avg.	Best	Avg.
25	5	48	57	52	53.9	40	47.35	40	46.96
	10		53	53	53.26	40	44.1	40	42.18
	15		53	52	52.52	40	44.75	40	41.95
50	5	28	33	30	40.82	25	29.95	25	30.22
	10		29	29	38.26	25	28	25	27.46
	15		33	29	37.52	25	28.75	25	26.08
75	5	12	14	15	29.34	12	15.6	12	15.10
	10		14	14	23.98	12	13.3	12	13.06
	15		23	15	20.37	12	12.85	12	12.52
100	5	9	20	9	19.16	5	7	5	6.6
	10		7	7	16.821	5	5.1	5	5.00
	15		9	7	12.52	5	5.2	5	5.00



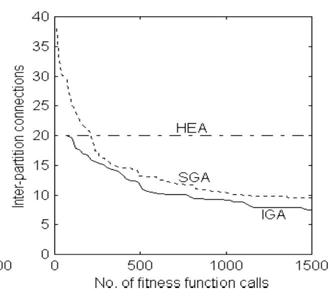
(1) N_{pop}=5, B=25



(2) N_{pop}=5, B=50



(3) N_{pop}=5, B=75



(4) N_{pop}=5, B=100

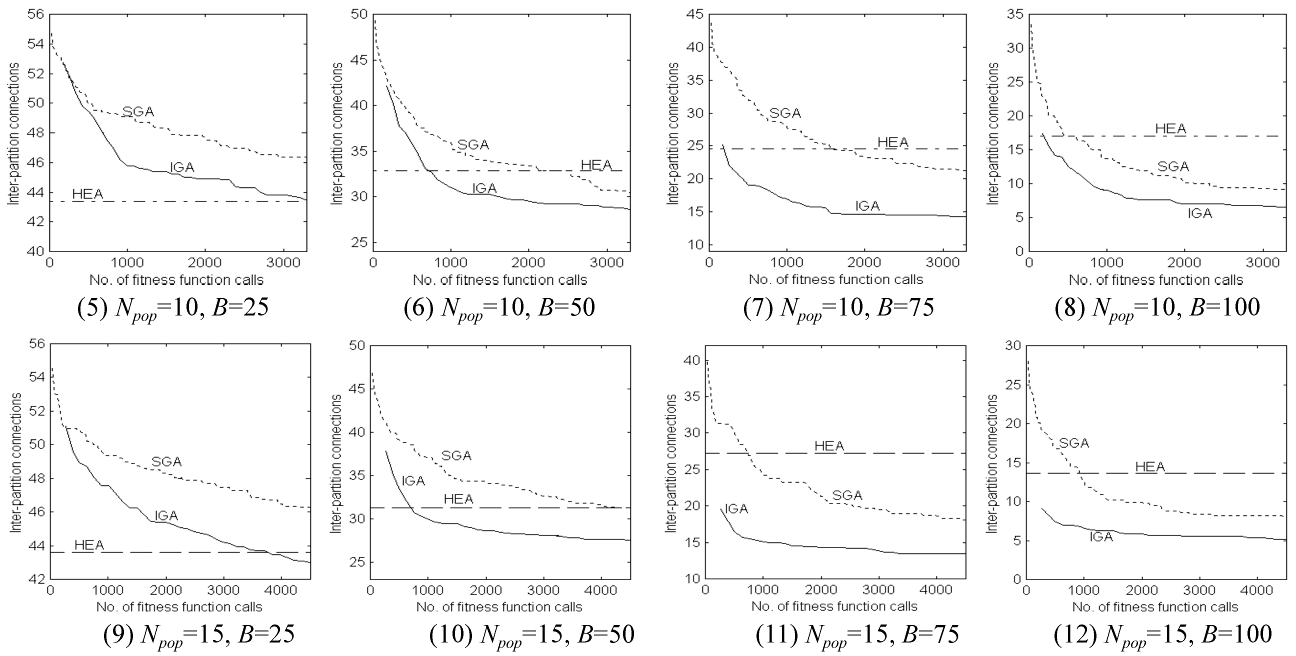


Fig. 1. Comparisons of convergence speed and accuracy.

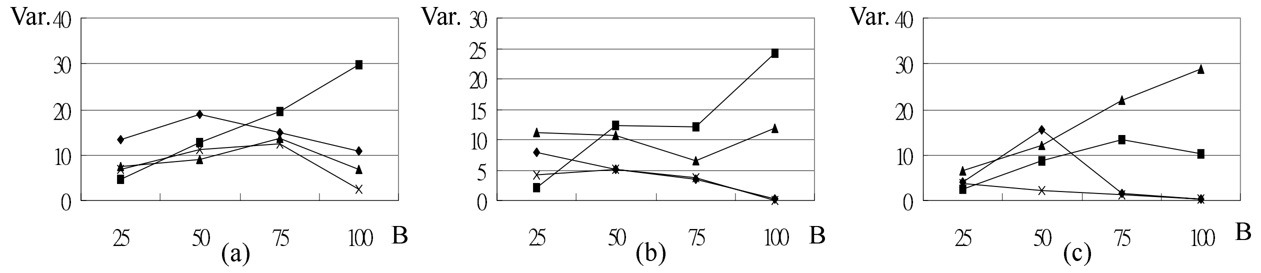


Fig. 2. Comparisons of variance of solutions. (a) N_{pop}=5 (b) N_{pop}=10 (c) N_{pop}=15

Notation: IGA ─◆─ HEA ─▲─ SGA ─■─ IGA2 ✕

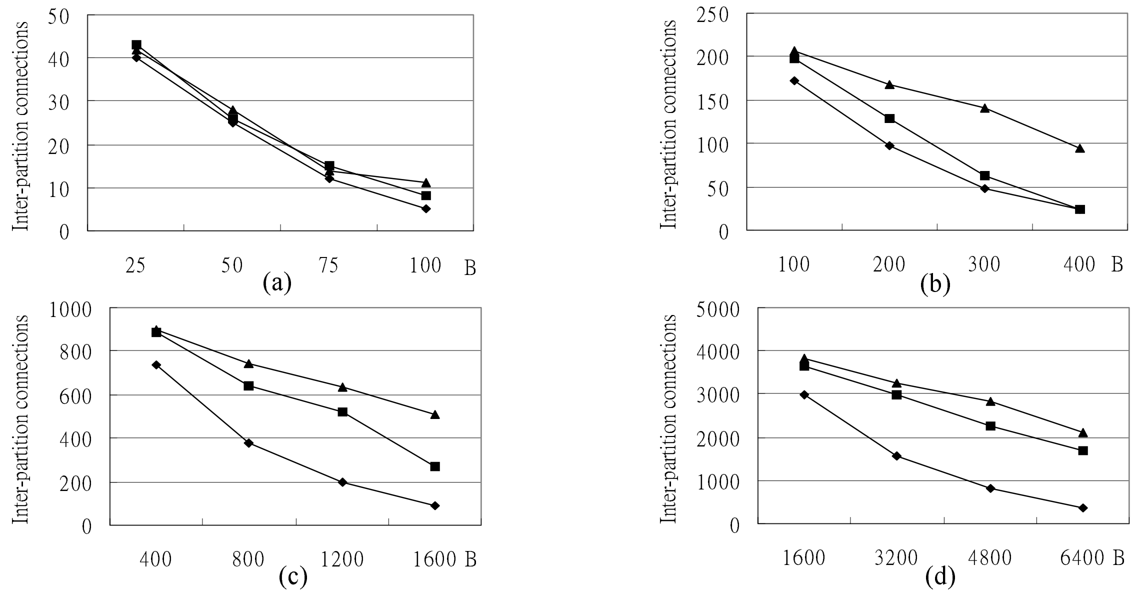


Fig. 3. Comparisons of quality for various sizes of benchmark. (a) A_0 (b) A_1 (c) A_2 (d) A_3

Notation: ─◆─ IGA ─▲─ HEA ─■─ SGA