Parallel Genetic Algorithms on Line Topology of Heterogeneous Computing Resources

Yiyuan Gong
Faculty of Information Engineering, University of the Ryukyus
Okinawa 903-0213 Japan
gongyy@ads.ie.u-ryukyu.ac.jp

Morikazu Nakamura
Faculty of Information Engineering, University of the Ryukyus
Okinawa 903-0213 Japan
morikazu@ie.u-ryukyu.ac.jp

Shiro Tamaki
Faculty of Information Engineering, University of the Ryukyus
Okinawa 903-0213 Japan
shiro@ie.u-ryukyu.ac.jp

ABSTRACT

This paper evaluates a parallel genetic algorithm (GA) on the line topology of heterogeneous computing resources. Evolution process of parallel GAs is investigated on two types of arrangements of heterogeneous computing resources: the ascending and descending order arrangement of computing capability. Their differences in chromosome variety, migration frequency and solution quality are investigated. The results in this paper can help to design parallel GAs in grid computing environments.

Categories and Subject Descriptors

F.2 [**Analysis of algorithms and problem complexity**]: Miscellaneous

General Terms

Algorithms, Design

Keywords

parallel genetic algorithm, migration interval, heterogeneous computing capability, grid

1. INTRODUCTION

In recent years, a grid has been becoming a promising computing environment for large scale computation [4]. Since a grid can provide many computational resources, researchers are interested in solving their large scale computing problems on a computational grid. A grid can include heterogeneous computing resources and various network media. Therefore, the capability of computing nodes is different and the network bandwidth between nodes is not uniform.

Since Genetic Algorithms (GAs) require usually lots of computation time, there are many reports on parallel GAs. However, when we solve very large scale optimization problems, we need tremendous computing resources. Therefore, it is straightforward to implement parallel GAs on a grid. Recently, some researchers have studied GAs on a grid [6] [7]. In [6] they present an efficient model of master/slave computing in grid environments. In [7] they propose a framework for researchers to easily develop GAs in a grid.

The purpose of our research is to develop efficient schemes of parallel GAs in grid computing environments to solve very large scale optimization problems. Gong [11] developed a parallel GA with a tree-based migration, which is an island model, but the migration is performed through logical tree-topology network.

A logical tree-topology network is established on a set of computing nodes in which we can set a logical link if two nodes can communicate each other. Note that we can establish any logical topology, even completely-connected topology, on (a part of) the Internet but it is important to consider the physical network underlying the logical topology since the communication delay of the logical link depends on the physical network. The reason we employ tree topologies is that those can be easily implemented and extended in real network environments with considering the network latency. Note that tree-topologies are various: not only a balanced binary tree but also a line and a star are a kind of tree-topologies.

Previously, Gong [11] investigated the effects of topologies and delay for parallel GA execution in distributed environments. In that paper, four types of tree-topologies: *star*, *line*, *balanced binary tree* and *sided binary tree* were examined. From the experimental results, we found that the cooperative evolution among nodes contributes to solution quality and the *line* topology performs the best among the four topologies from the viewpoint of solution quality.

However, in [11], we assumed an ideal situation that all the computing resources have the same computing capability, but in the real system, it cannot be the same. Therefore, in this paper, we investigate the case of heterogeneous computing resources. In this case, it is natural to think that the arrangement of heterogeneous computing resources should affect the performance of parallel GAs. Here, we consider the line topology since it is the simplest and showed the good performance in our previous evaluation [11].

In this paper, we use a continuous multimodal landscape generator based on Gaussian functions [1] to generate test problems. Traditional function optimization problems such as Schwefel functions, Griewank functions and Rastrigin functions [3] are additionally used. A real-coded GA with UNDX [10] crossover and MGG [9] alternation model is implemented for optimization of Gaussian functions and a binary-coded GA for the traditional functions.

This paper is organized as follows: In Sect.2, we present our parallel GA. We show the experimental results and discuss on them in Sect.3, and conclude this paper in Sect.4.

2. PARALLEL GA WITH TREE-BASED MIGRATION

We present a parallel GA, called PDGA in this section. The PDGA is a distributed parallel GA, that is, an island model with migration in which the migration is performed through a logical tree-topology.

A tree-topology is established before execution. When we use a part of the Internet as a grid computing environment, any logical topology even the completely connected topology can be set. However, because of the communication latency, usually we need to consider the physical network topology, namely, we should set a logical link between two nodes in which they can communicate with small communication overhead. of communication delay. With considering communication overhead of the migration topology, we can say tree topologies are easily implemented and extensible on real network environments. Moreover, hierarchical structure of the tree topologies may be useful for cooperative evolution of parallel GAs. It is why we choose tree-topologies for the migration.

When we are given a network environment, for example a part of the Internet, we set a spanning tree with a single root node, that is, all the computing nodes are included in the topology in which every node except the root knows its parent and every node except the leaves knows its children.

Figure 1 is an example of a spanning tree. The arrowed edges are chosen for the spanning tree.

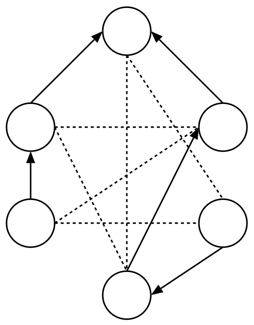

Figure 1: Example of spanning tree

In our algorithm, each computing node generates its own initial chromosome set as a subpopulation, and carries out genetic operations on its subpopulation independently as an island model. The migration can be performed from a child to its parent when the migration condition is satisfied. The migration condition is determined by two factors: the migration interval and the timing of the current best update. The migration interval is set before execution, for example, 30, 50, 100 and so on. When the migration interval is 30, each node tries the migration at every 30 generation. However, it can perform the migration only when the current best was updated since the last migration. When a node receives a chromosome sent from its child, the node replaces the worst chromosome in his population by the migrated one. The migration can be described as *push type* since a child node pushes its current best to his parent when it is updated (at the migration interval).

We show the procedure for each computing node when the target problem is a Gaussian function.

```
 0: procedure PDGA for Gaussian functions;
 1: begin
 2:   Generation_no := 0;
 3:   initialize a subpopulation;
 4:   evaluation;
 5:   while(termination condition does not hold)
 6:   begin
 7:     randomly select two parents and
         apply UNDX crossover;
 8:     evaluation;
 9:     select the best and the second chromosomes
           from the children and these two parents;
10:     replace the original parents by
           the selected chromosomes;
11:     check the communication buffer;
12:     if(received chromosomes) begin
13:         replace the worst one
               by the received one;
14:     endif;
15:     if(migration condition is satisfied)
16:       send the best to the parent;
17:     endif;
18:   end while;
19: end
```

3. EXPERIMENTAL EVALUATION

In the experiment, we evaluate effects of arrangements of heterogeneous computing resources on cooperative evolution in our parallel GA. As we described, in this paper we focus on the line topology since it is the simplest and showed good performance [11].

3.1 Simulation System

Since it is still difficult to use a real grid environment for experimental evaluation, we simulate it on a PC-cluster which is composed of Intel 2.0GHz Xeon processors connected by a fast-ethernet switch. All the processors of the PC-cluster have the same capability. In order to simulate heterogeneous computing resources, we insert dummy codes to the program on a node which simulates a slower computing resource. Let T_{GA} be the execution time of one GA generation of the fastest node in the resources, and use it to represent the capability of all the computing nodes. For example, the number in a node represents the capability in Figure 2, the root node has the weight 8, it means the node requires $8T_{GA}$ to complete one GA generation, and so on. Therefore, the root node iterates eight times of the same calculation to simulate $8T_{GA}$. We call the coefficient of T_{GA} *capability weight*. Note that smaller capability weights correspond to more powerful computing capability.

The parallel program for the PDGA is written in C language with MPICH (a MPI-based communication library) [5].

3.1.1 Test Problems and Parameters

The dimension of Gaussian functions was set to two, and the number of GFs was 200 in this experimental evaluation. The range of rotation angle was set to [-$\pi/4$, $\pi/4$], and the search space was bounded to [-300, 300] in each dimension. The range of variance value was set to [0.25, 5.25]. The problems were generated randomly.

UNDX parameters are set as Kita's recommends [8]: α is 0.5, β is 0.35 and the number of crossover time is 100.

The dimension was set to 40, one point crossover, one bit reverse mutation and the roulette selection were used for the traditional function optimization problems. The elite reservation was also incorporated.

A parallel GA execution terminates when the root node terminates. We set the total number of generations to the leaf nodes beforehand, and each node sends a terminated message to its parent when it terminates. All the nodes except the leaves can terminate after they receive terminate messages from all the children.

All results we show here are the average values of 50 trials. We show here results mainly for Gaussian functions' optimization.

3.1.2 Arrangement of Heterogeneous Computing Resources

We ran our algorithm for several arrangements of heterogeneous computing nodes.

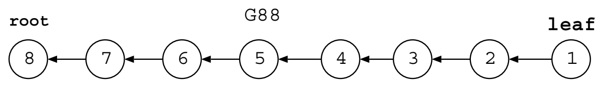

Figure 2: Descending arrangement: G88

Figure 2 shows graphically the arrangement of 'G88'. The number in each node represents the capability weight of the node. An arrow represents the parent-child relation, that is, the migration direction. The source of an arrow shows the child and the destination the parent. Table 1 lists all the arrangements used in the experiments. In the table, 'Name' represents the name of the arrangement, 'Computing capability' shows the capability weight of each node, and note that 'root' is the most left node in this column, the 'leaf' node is on the most right of this column. 'Total' represents the sum of the capability weights in each arrangement.

For example, the first entry of the table is for G88. There are eight computing nodes, the capability of the root is eight, and that of the leaf is one. The arrangement is descending order of the capability weights from the root to the leaf, that is, eight to one. The second entry is named as G81. There are eight nodes and the capability of the root is one. The arrangement is ascending order of the capability weight from the root to the leaf. G88 and G81 have the same number in "Total" since they have the same computing resources, but the arrangement is opposite.

Table 1: Computing capability of each node

Name	Computing capability								Total
G88	8	7	6	5	4	3	2	1	36
G81	1	2	3	4	5	6	7	8	36
G66	6	5	4	3	2	1	-	-	21
G61	1	2	3	4	5	6	-	-	21
G63	3	4	5	6	7	8	-	-	33
G68	8	7	6	5	4	3	-	-	33
G44	4	3	2	1	-	-	-	-	10
G41	1	2	3	4	-	-	-	-	10
G45	5	6	7	8	-	-	-	-	26
G48	8	7	6	5	-	-	-	-	26

These arrangements can be classified into two types, the ascending order and the descending order. G41, G45, G61, G63 and G81 belong to the former, G48, G44, G68, G66, and G88 the latter. These two arrangement types have the extreme characteristics and effect differently on cooperative evolution with the tree-based migration. Therefore, it is very important to investigate the behavior of the evolution process of these two extreme arrangement types.

3.2 Ascending vs Descending

First of all, let us observe the difference of the cooperative evolution between the ascending and descending arrangements. Here, we show all the solution curves of G88 and G81 when the migration interval is 50 generation.

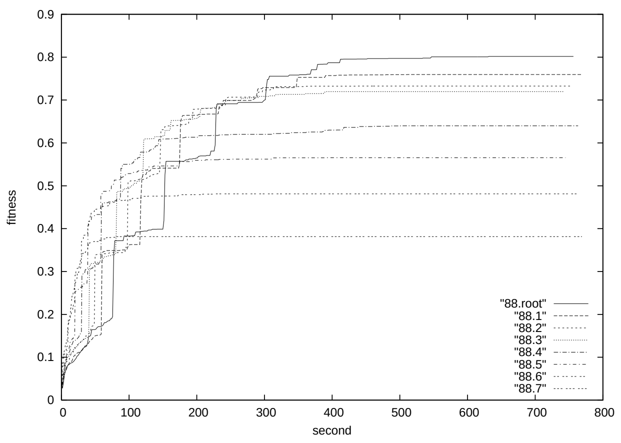

Figure 3: Solution curve of each node in G88

Figure 3 shows the elite (current best) solution curves of all the nodes in G88, and Figure 4 shows the ones in G81. In Figure 3, '88.root' corresponds to the curve of the root node, '88.1' is for the second node from the root, '88.7' represents the leaf node, and so on. For Figure 4, the naming of the curves is the same as Figure 3. From the two figures we can say that the root can obtain the best final solution among the final solutions in all the computing nodes, followed by '1','2','3','4','5','6', and '7'. That is, the upper the node, the better the final result. It is obvious that the capability of the computing node influences the rising time of the solution curve. The solution curves of nodes with weaker capability rise more slowly. From Figure 3, it is also obvious that the

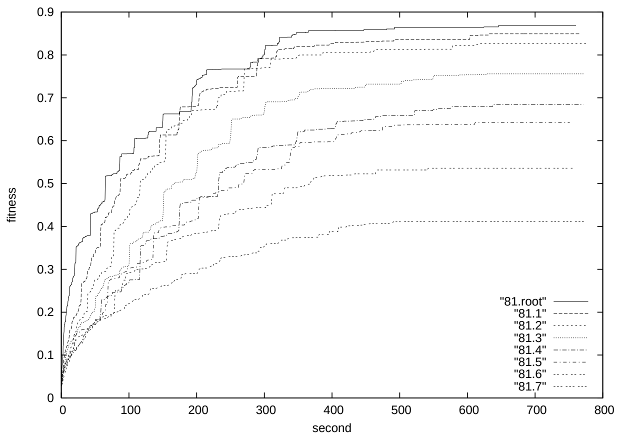

Figure 4: Solution curve of each node in G81

curves of upper nodes rise slower than ones of the lower nodes at the early stage (about 80 seconds before). This is because in G88 the upper nodes have weaker power than the lower nodes. However, they take over the lower nodes after receiving the elite solutions from their children. The upper nodes have the stronger capability in G81, thus in Figure 4 the solution curves of upper nodes rise earlier, and when elite solutions are transfered to the upper nodes, they can keep ahead. We also find the same properties in the other arrangements even though we just show the results of G88 and G81.

3.3 Scaling of Computing Resources

We observed here how evolution process is influenced when we increase computing resources in the two types of arrangement, the ascending order and the descending order.

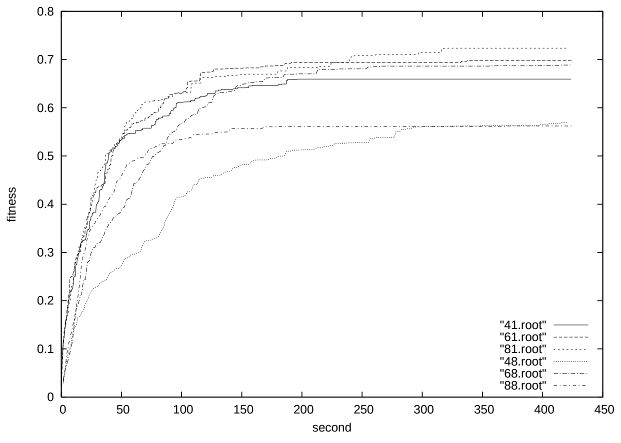

Figure 5: Root solution curves with migration interval 1

Figure 5 shows the elite solution curves of the root node for the six arrangements when the migration interval is 1, and Figure 6 depicts the same curves when the migration interval is 50. In the figures, 'xy.root' represents the elite solution curve of the root in Gxy. From Figure 5, we observed

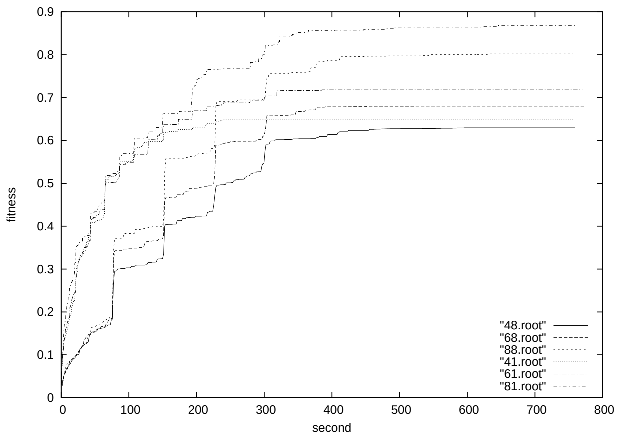

Figure 6: Root solution curves with migration interval 50

that in the ascending arrangements of capability weight, G61 obtained better results than G41, and G81 is better than G61. This means that adding weaker power nodes to the end of the ascending order can contribute to improve solution quality. However, we also find that G68 got better results than G48, but G88 obtained almost the same result as '48'. That is, adding computing resources to the end of the descending order do not straightforwardly contribute to evolution process, especially when the migration interval is short. This is because adding more powerful nodes to the leaf side brings much aggressive migration, especially when the migration interval is short. As we know, too aggressive migration may lead to excessively fast convergence, and the quality of solutions seems to be negatively affected [2].

From Figure 6, we find that increasing computing nodes can improve solution quality in these two arrangements when the migration interval is appropriately long.

3.4 Chromosome Diversity

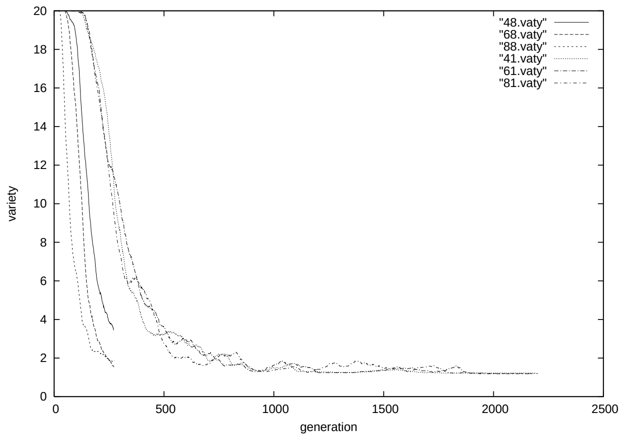

Figure 7: Root variety curves with migration interval 1

Let us see the variety curves of the root node for some arrangements by varying the migration interval. The vari-

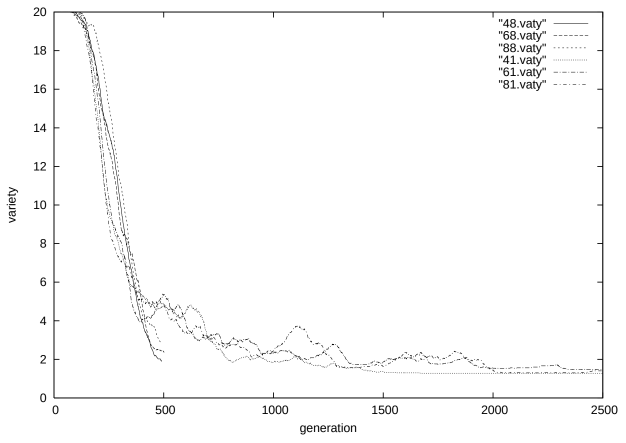

Figure 8: Root variety curves with migration interval 50

ety means here the number of different chromosomes in the node. At every generation we counted the variety of each node in the experiment. In Figure 7 and 8, 'xy.vaty' represents the root variety curve of Gxy. Note that the horizontal axis is 'generation' but not 'time'. Therefore, the curves of the descending arrangements stopped at earlier generations since the root nodes of the arrangements are faster than those of the ascending arrangements even though the computation time is almost the same.

Figure 7 and 8 show the variety curves when the migration interval is 1 and 50, respectively. From Figure 7, we can see that the ascending arrangements keep variety better than descending ones before about 500 generations. This is because in the descending groups, the evolution speed of lower nodes is faster than upper nodes and the migration speed to the root is faster compared to the evolution speed at the root. When the migration interval is shorter, they become more evident. That is, frequent migrations should lead to fast convergence.

Table 2: Average chromosome variety of ascending groups

	G41		G61		G81	
gen_span	mig1	mig50	mig1	mig50	mig1	mig50
1∼300	17.41	16.11	17.13	16.52	16.75	15.87
300∼600	4.14	5.17	4.91	4.97	4.23	5.08
600∼900	2.08	2.97	1.98	3.10	1.90	2.98

However, Figure 8 shows different feature from Figure 7. In order to see more clearly the chromosome variety in the ascending arrangements, we show the average chromosome variety of the arrangements in Table 2 when the migration interval is 1 and 50. 'gen_span' means the generation span, '1∼300' row represents the average chromosome variety from generation 1 to 300, and the same for the rest. The chromosome variety of interval 50 is worse before 300 generations than that of interval 1, however, it becomes better after 300 generations, and the difference between those two migration intervals increases when the generation comes to 900. This can be explained as follows: in the ascending arrangements, lower nodes evolve chromosomes slower than upper nodes, so migration chromosomes are transfered much more slowly compared to the evolution of chromosomes at the root. When the migration interval is long, this situation can be evident. That is, the root looses the chromosome variety at earlier stage (before about 300) because of too passive migration.

However, migration data come later and contribute some how to improve the variety. We can find in Figure 8 that continuous up-and-down appear later in the ascending groups when the migration interval is 50.

Thus, from these two figures, we can conclude that too aggressive migration reduces chromosome variety in the descending arrangements and passive migration for the ascending arrangements also leads to less variety of chromosomes at the early stage.

We show another example in Figure 9 to confirm our consideration above. In Figure 9, 'xy_mig$z$.vaty' represents the the variety curve of the root node in Gxy with migration interval z. We can see that the variety of the root in G88 becomes better, but the variety of the root in G81 becomes worse before about 300 generations when we increase the migration interval from 1 to 50.

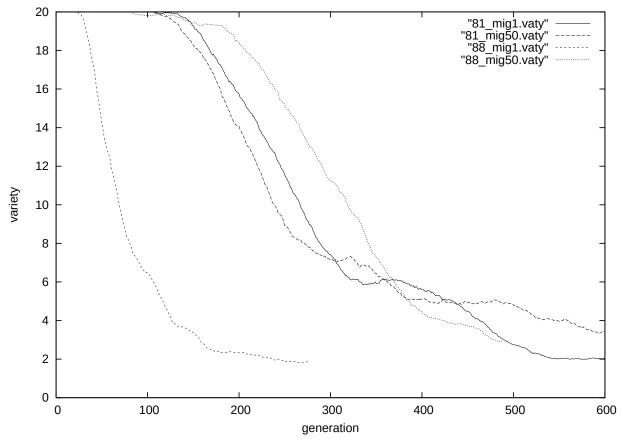

Figure 9: Root variety curves with different migration interval

3.5 Migration Frequency

The migration does not always take place at every migration interval since it depends on the timing of the current best updating. So here we investigate the migration frequency which represents how many times the migration occurs during the evolution process. Longer migration interval should lead to lower migration frequency. However, there are differences between the ascending and descending arrangements even though they are with the same migration interval. Under a certain migration interval, if the migration frequency is higher, it means that the best fitness is updated more frequently according to our migration condition.

Table 3, 4, 5, 6, and 7 show the number of chromosomes received at each node when the migration interval is 10, 30 and 50. "0" represents the root node, "1" represents the next one, the same for the rest. The migration is performed by communication on a child-parent pair. Therefore, the num-

Table 3: Receive times of G81 and G88

	mig10		mig30		mig50	
node	G81	G88	G81	G88	G81	G88
0	21	9	12	5	9	4
1	21	10	11	6	8	4
2	19	10	10	5	7	4
3	17	11	9	5	7	4
4	17	12	9	6	6	4
5	14	13	8	6	6	4
6	12	16	7	8	5	5
sum	121	81	66	41	48	29
diff	40		25		19	

Table 4: Receive times of G61 and G66

	mig10		mig30		mig50	
node	G61	G66	G61	G66	G61	G66
0	20	10	11	5	8	4
1	19	11	11	5	8	4
2	17	12	10	6	7	4
3	16	13	9	6	6	4
4	14	16	8	8	5	5
sum	86	62	49	30	34	21
diff	24		19		13	

ber of receiving equals the number of migrations. The tables do not include the leaf node since the node does not receive any chromosome in their migration, they can be a sender. 'sum' means the sum of all nodes. 'diff' represents the difference between the two arrangements. We find a common property such that the migration frequency increases from the bottom to the root in the ascending arrangements, but decreases in the descending ones. Another observation is such that the sum of the received chromosomes in the ascending arrangement is bigger than the sum of those in the descending ones.

Let us look at G81 and G88. The leaf node of G88 is the most powerful resource node among the arrangement, it evolves fastest and can send data frequently, however, the leaf node of G81 is the weakest, the migration frequency should be low. Thus node 6 (the parent of the leaf) of G88 receives data more frequently than node 6 of G81.

However we find that node 5 of G81 has more number of receiving than node 5 of G88 even though the power of the child (node 6) in G81 is weaker than that in G88. Why node 5←6 pair of G81 performs migration more frequently than that of G88? The essential difference between both is the direction of the migration, that is, from a weaker node to a stronger one or from a stronger node to a weaker one. The migration frequency becomes higher when the best solution is updated more frequently. From the results, we can say that the migration from a weaker node to a stronger node can make the node evolve more efficiently. In case of the migration from a stronger node to a weaker node, a child should send a much better solution to its parent compared to the average fitness value of its parent, it seems to be not always so good for the cooperative evolution.

Table 5: Receive times of G41 and G44

	mig10		mig30		mig50	
node	G41	G44	G41	G44	G41	G44
0	19	12	9	6	7	4
1	17	13	9	6	6	4
2	16	16	8	8	5	5
sum	52	41	26	20	18	13
diff	11		6		5	

Table 6: Receive times of G63 and G68

	mig10		mig30		mig50	
node	G63	G68	G63	G68	63	G68
0	19	12	10	6	7	5
1	17	13	9	7	7	5
2	17	13	9	7	6	5
3	14	14	8	7	6	5
4	12	16	7	8	5	5
sum	79	68	43	35	31	25
diff	11		8		6	

Table 7: Receive times of G45 and G48

	mig10		mig30		mig50	
node	G45	G48	G45	G48	G45	G48
0	17	13	9	7	6	5
1	14	15	8	8	6	5
2	12	15	7	7	5	5
sum	43	43	24	22	17	15
diff	0		2		2	

We also observed the difference between G45 and G48 is smaller than that of G41 and G44, and the difference between G63 and G68 is smaller than G61 and G66. This is because the power ratios in these groups are smaller than the other groups. For example, in groups '48' and '45', the biggest power ratio is 8 to 5, while in groups '44' and '41', it is 4 to 1. When the computing power is not much different from each other, the arrangement of computing resources should give very small influences.

3.6 Solution Quality

In order to investigate the solution quality, we depict two curves: the elite solution curves of the root node for G81 and G88 in Figure 10 and 11. In Figure 10, '81_ncom.best' represents the best solution curve of G81 when the migration interval is infinite, that is, no data are migrated between any pair of nodes. '81_mig$x$.root' represents the elite solution curve of the root in G81 when the migration interval is x. For Figure 11, the naming of curves is the same as Figure 10. From these two figures, we find that no migration type, shown as xy_ncom.best, is the worst, thus we can say that the migration improves solution quality obviously.

We also observe that longer migration interval leads to slower convergence. Comparing all the migration intervals except infinity, we find that the case of migration interval 1 gets the worst result, and the solution quality becomes better when increasing the migration interval. However, solution quality becomes again worse when we increase more. That is, there exists an appropriate migration interval for each case.

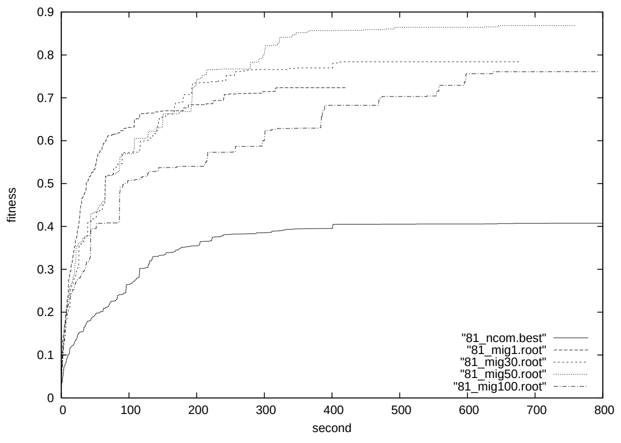

Figure 10: Root solution curves of G81

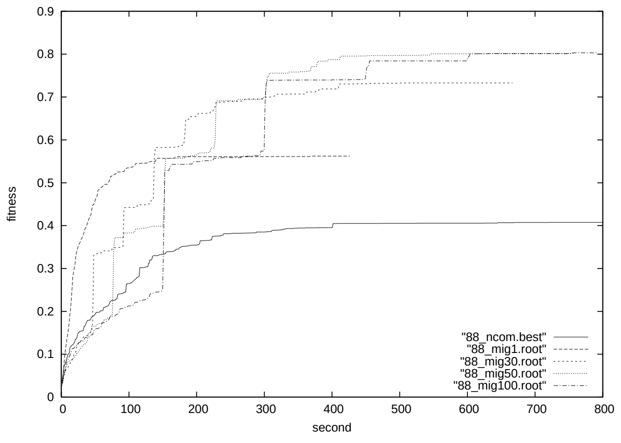

Figure 11: Root solution curves of G88

Table 8 shows the final solution values for all the arrangements. Each pair of continuing two entries from the top has the same total power, but different order arrangement, for example, G81 and G88, G61 and G66, and so on. In Table 8, we show the fitness values of the best individual at the end of evolution. Note that they are the average values of 50 trials. '1' in the first row represents the migration interval is 1, the same for the rest. 'Ave' represents the average of all the migration intervals except infinity.

Comparing G81 and G88, we find that G81 gets better results than G88 when the migration interval is short, like '1','10','30', or '50'. It is also obvious to see that G81 is much better than G88 when the migration interval is '1'. However, G88 gets better results than G81 when the migration interval is '100'. We find the similar property for the other two pairs (G41 and G44, G61 and G66) also. The difference between G48 and G45, and that of G63 and G68 are smaller than the other pairs. This is because the power ratios in these two pairs are small.

From the above data, we can say that migration interval affects considerably the solution quality. Too short migration interval leads to fast convergence, and the final solution is negatively affected, especially for the descending arrangement. Longer migration interval improves the solution quality for the descending arrangement greatly, and even some time makes it perform better than the ascending groups. However, it does not mean the descending groups can always get better results than the ascending groups when the migration interval is long. For any arrangement, there is an appropriate migration interval.

Table 8: Final solution values of all groups

	Migration Interval							
Name	1	10	30	50	100	150	∞	Ave
G81	0.72	0.75	0.78	0.87	0.76	0.84	0.41	0.79
G88	0.56	0.72	0.73	0.80	0.80	0.79	0.41	0.74
G61	0.70	0.69	0.80	0.72	0.69	0.71	0.41	0.72
G66	0.55	0.69	0.68	0.73	0.71	0.69	0.41	0.68
G41	0.66	0.65	0.68	0.65	0.69	0.66	0.38	0.66
G44	0.49	0.61	0.57	0.64	0.63	0.59	0.38	0.59
G48	0.57	0.59	0.61	0.63	0.61	0.61	0.41	0.60
G45	0.57	0.52	0.65	0.68	0.62	0.71	0.41	0.63
G68	0.69	0.70	0.70	0.68	0.72	0.71	0.41	0.70
G63	0.63	0.68	0.75	0.83	0.71	0.80	0.41	0.73

3.7 Verification of the other function optimization problems

We tested our algorithm also on Schwefel function, Griewank function and Rastrigin function [3]. We show results of Schwefel function here, it is a minimization problem.

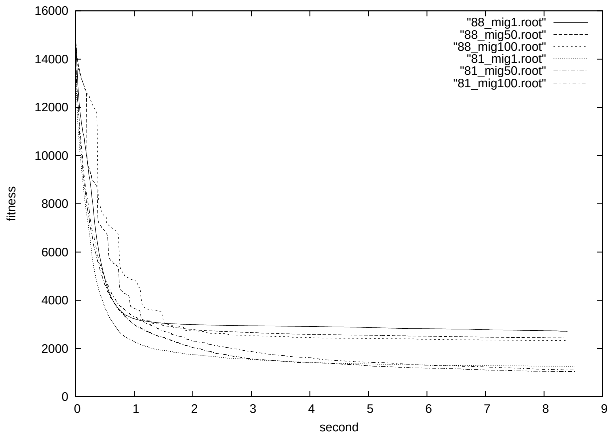

Figure 12: Root solution curves of G81 and G88

Figure 12 shows the elite solution curves of the root node for G81 and G88 when the migration interval is 1, 50 and 100 generation. We find that the root of G81 get much better results than the root of G88. We confirm that too short migration interval affects negatively the final solution quality, especially for the descending groups.

For chromosome variety and migration frequency, we get similar properties with Gaussian functions. For Griewank and Rastrigin functions, we observed similar results.

3.8 Discussion

In this experiment, we investigated the influence of the arrangements of heterogeneous computing resources on cooperative evolution in distributed parallel GAs. However, in this paper we focused on two simple cases of the arrangements, that is, the descending and ascending order arrangements.

In the experimental evaluation, we confirm that in any pattern of arrangements, our parallel GA on heterogeneous computing resources evolve cooperatively. However, we also observe that these two basic arrangements influence cooperative evolution differently.

We find that chromosome variety is influenced by the migration interval. Especially for the descending groups, migration interval affects its chromosome variety greatly. Migration frequency is also different in the descending and ascending groups. Migration from weaker nodes to stronger nodes contributes largely to cooperative evolution.

From our experimental analysis of chromosome variety and migration frequency, we can conclude that they are important factors in solution quality because less variety leads to premature convergence, and receiving much better solutions (comparing to the average fitness value of the parent) may make the parent similar to its child.

Therefore, we can summarize the experimental results as follows: The migration is useful for parallel GAs but it should be carefully designed if computing resources are heterogeneous. Because of the imbalance of evolution speed between a child and a parent, migration may lead to a negative influence on the evolution process. One possible solution is to employ the ascending order arrangement of computing resources. The migration can contribute more safely since the migration is performed from a weaker node to a stronger one in this arrangement. This case is harder to receive negative influence to the evolution process than the case of the descending order arrangement. For the descending order arrangement, we need to avoid aggressive migration more carefully.

We obtained similar results for other test instances with the same problem size. That is, even though the concrete values are different, the feature of the effects of the arrangements of computing resources are almost the same. Because of the space limitation, we omit the numerical results for the other test problems in this paper.

4. CONCLUDING REMARKS

This paper evaluated a parallel GA on the line topology of heterogeneous computing resources. Evolution process of parallel GAs was investigated on two types of arrangements of computing resources: the ascending order and descending order arrangement of computing capability. Their differences in chromosome variety, migration frequency and solution quality are investigated. The effects of increasing computing resources are also clarified. The results in this paper may facilitate implementation of parallel GAs in grid computing environments.

As future works, we need to investigate cases of other arrangements on the line topology and to analyze the cooperative evolution theoretically based on a probabilistic model. And also we will try to implement our parallel GA in a real grid environment.

5. REFERENCES

[1] B.Yuan and M.Gallagher. On building a principled framework for evaluating and testing evolutionary algorithms: A continuous landscape generator. *Congress on Evolutionary computation*, pages 451–458, 2003.

[2] Cantu-Paz, D. E, W., and E. J. Migration policies and takeover times in parallel genetic algorithms. *Proceeding of the Genetic and Evolutionary Computation Conference*, pages 525–532, 1999.

[3] D.Whitley, K. Mathias, S. Rana, and J. Dzubera. Building better test functions. *International Congress on Genetic Algorithms*, 1995.

[4] I. Foster and C. Kesselman. The grid: Blueprint for a new computing infrastructure. *Morgan Kaufmann*, 1998.

[5] W. Gropp, E. Lusk, and R. Thakur. Using mpi-2. *Using MPI-2*, 1999.

[6] G.Shao, F.Berman, and R.Wolski. Master/slave computing on the grid. *"Proceedings of the 9th Heterogeneous Computing Workshop*, pages 3–16, 2000.

[7] H.Imade, R.Morishita, I.Ono, and M.okamoto. A framework of grid-oriented genetic algorithms for large-scale optimization in bioinformatices. *2003 Congress on Evolutionary Computation*, pages 623–630, 2003.

[8] H.Kita, I.Ono, and S.Kobayashi. Theoretical analysis of the unimodal normal distribution crossover for real-coded genetic algorithms. *Proc. 1998 IEEE Intr. Conf. on Evolutionary Computation*, pages 529–534, 1998.

[9] H.Satoh, I.Ono, and S.Kobayashi. A new alternation model of genetic algorithms and its assessment. *J. of Japanese Society for Artificial Intelligence*, 12(5):734–744, 1997.

[10] I.Ono and S.Kobayashi. A real-coded genetic algorithm for function optimization using unimodal normal distribution crossover. *Proc. 7th ICGA*, pages 246–253, 1997.

[11] Y.Gong, M.Nakamura, T.Matsumura, and K.Onaga. A distributed parallel genetic local search with tree-based migration on irregular network topologies. *IEICE transactions, on Fundamentals of Electronics, Communications and Computer Sciences*, E87-A(6):1377–1385, JUNE 2004.