A Diversity Study in Genetic Algorithms for Job Shop Scheduling Problems

Carlos A. Brizuela
Kyoto Institute of Technology
Matsugasaki, Sakyo-ku, Kyoto 606–8585 Japan
cbrizuel@si.dj.kit.ac.jp
+81–75–724–7467

Nobuo Sannomiya
Kyoto Institute of Technology
Matsugasaki, Sakyo-ku, Kyoto 606–8585 Japan
sanmiya@si.dj.kit.ac.jp
+81–75–724–7447

Abstract

This paper deals with the study of population diversity in Genetic Algorithms for Job Shop Scheduling Problems. A definition of population diversity at the phenotype level and a way to compute it are given. Two diversity oriented selection procedures for GA are proposed. Their performances in terms of diversity and solution quality are tested against a standard Genetic Algorithm. Relations between population diversity and algorithm accuracy are shown through numerical experiments.

1 Introduction

When using Genetic Algorithms (GA) to face dynamic scheduling problems or even in the case of static multi-objective optimization problems, it is necessary to have at each iteration a high diverse population. How high this diversity should be or whether it should be slowly or rapidly varying value, is not known. In fact, up to date very little has been said about system population diversity and its influence on algorithms performance.

System adaptability is one of the most important performance criterion when dealing with real world problems. In these real world problems, we usually track the behavior of population through the fitness function. When the problem representation is simple in the sense that the genotype-phenotype is an one-to-one like map, we can easily look for properties of the population through the genotype. However, when the representation does not have this feature we need perhaps to use extra information besides that provided by the fitness function to know more about the population behavior.

It is our belief that population diversity plays an important role in the adaptation mechanism of Genetic Algorithms. This belief is based on the fact that for a certain class of ecological system, simulation results of behavior suggest that adaptability to environmental variations strongly depends on system diversity as well as on other characteristics of the population (Sannomiya and Tian, 1998).

In the case of scheduling problems, so far as we know, nothing has been done in trying to answer the question of adaptability regarding the population diversity. This work is a first step of an attempt in trying to answer this question. Following in this direction, the first step is to have a clear definition of system diversity and a mechanism to control it. This work is aimed to achieve these goals.

One of the most challenging problem in real world optimization is the Job Shop Scheduling Problem (JSSP). In this work we deal with a very special case of this problem, the one that lasted more than twenty years to be solved (Muth and Thompson, 1963).

A very important problem when dealing with GA for JSSP is the time consuming tuning process for large size problems. Perhaps, the information provided by the diversity measure can help us to automate this tuning process and to choose the most appropriate population size.

The paper is organized as follows. Section 2 gives a brief explanation of the problem we consider and the way we want to face it. In section 3 a definition of population diversity as well as a way to compute it are given. Section 4 gives two proposed methods for keeping high diversity values. Section 5 explains the experimental results and finally section 6 states the conclusions.

2 Background

2.1 Problem Formulation

The Job Shop Scheduling Problem (JSSP) is among the toughest real world combinatorial optimization problems. It was shown to belong to the NP-hard class (Taillard, 1994).

Basically this problem consists of a set **J** of jobs and a set **M** of machines. Each job j ($1 \leq j \leq n = |\mathbf{J}|$) has $m = |\mathbf{M}|$ operations $(O_{j1}, O_{j2}, ..., O_{jm})$ that must be scheduled in a pre-defined machine sequence imposed by technological requirements. Each operation is processed on each machine only once. No operation may free its machine until it is finished. There is also a specified processing time $t(j,k)$ for each operation O_{jk} (job j in machine k). The problem is to find a sequence of jobs for each machine satisfying the technological restrictions and such that the finishing time of the last operation (completion time) on the latest job be minimized (Makespan minimization).

Many procedures have been proposed to solve this problem. Branch and Bounds methods provides the optimal solution for this problem but its computation becomes prohibitive for large size instances. Local Search approaches like those analyzed in (Aarts et al., 1994) and (Vaessens et al., 1996) can get good quality solutions without ensuring optimality.

2.2 Genetic Algorithms Approach

There have been a number of studies on applying GA to JSSP (see (Davis, 1985), (Yamada and Nakano, 1995), (Kobayashi et al., 1995), (Shi et al., 1996), (Yamada and Nakano, 1996), (Shi et al., 1997), (Yamada and Nakano, 1997), (Gen and Cheng, 1997)). Their methodologies differ mainly in the problem representation approach or in the way they perform the genetic operations. Among the most successful ones we can cite (Yamada and Nakano, 1995) and (Yamada and Nakano, 1996). For these cases, the encoding uses knowledge of the problem in order to build neighborhoods to be used in genetic operations.

In this work we deal with a job-based like representation as used in (Shi et al., 1997). The idea is to have a matrix $\mathbf{A} := [a_{ij};\ i = 1, ..., n,\ j = 1, ..., m]$ whose elements are job numbers to be scheduled in the $a(1,1), a(1,2), ..., a(1,m), a(2,1), ..., a(n,m)$ order, following the technological restriction. The main characteristic of this representation is that the matrix columns are forced to be permutations of job numbers. This fact makes it easy and computationally cheap to perform genetic operations without worrying about feasibility of the generated schedule. The disadvantage is that there could be cases where the optimal solution can not be expressed by this representation. Despite this disadvantage we still use this representation for two reasons; first we use the same encoding in both; our proposed methods and in the standard GA. Secondly, we care about diversity and not too much about the optimality of solution.

The standard GA is composed of the basic evolution operators like selection, crossover and mutation (see (Davis, 1985), (Gen and Cheng, 1997) and (Yamada and Nacano, 1997)). Now, let us describe the standard GA we will use in this paper.

Algorithm 1. Standard GA.

> **Step 1**. Set $k = 0$. Generate an Initial Population Pop[k] of G individuals.
> **Step 2**. Using RWS select two individuals from Pop[k] for genetic operations.
> **Step 3**. Do Crossover (with probability Pc) and mutation (with probability Pm).
> **Step 4**. If the total number of selected individuals is smaller than G, then go to Step 2; otherwise go to Step 5.
> **Step 5**. Set $k = k + 1$. Construct the new generation Pop[k] of G individuals.
> **Step 6**. If the stop criterion expires then stop, otherwise go to Step 2.

Here, G is the population size. As it was described previously a matrix description is used as a representation method in this algorithm. Roulette Wheel Selection (RWS) is used as selection procedure and one point crossover with embedded mutation (Shi et al., 1997) for the other genetic operations. Elitism is used in this algorithm as well as in our proposed methods. We believe that almost all conventional Genetic Algorithms for the JSSP will have similar behavior as the one just described, at least, in terms of population diversity and solution quality. Here, the main idea is to use this algorithm as a reference to compare with our proposed methods.

3 Computation of Population Diversity

One of the most difficult points to understand when dealing with GA applied to JSSP is the genotype-phenotype mapping. Due to the complex constraints usually involved in problems like the JSSP, the representation methods are complex and do not allow direct view of what is happening at the phenotype level. For instance, in job-based representation a high diversity measure for the genotype (code space) will not necessarily imply a high diversity measure of the phenotype (schedules). Problems like these motivate us to define diversity at the phenotype level.

In this section we give an explicit way to compute the population diversity of the phenotype, in the case of JSSP. We see the necessity of doing so due to the abuse in use of the word!H diversity !Iwithout giving a clear definition of it. First we define the difference between two given schedules a and b.

Definition 1. The difference $dif(a,b)$ between two given schedules a and b, for a JSSP of n-jobs and m-machines, is given by the sum over all machines of the number of differently sequenced jobs on each machine,

i. e.,

$$dif(a,b) = \sum_{i=1}^{m}\sum_{j=1}^{n} \partial_{ab}(i,j), \tag{1}$$

where $\partial_{ab}(i,j) =$

$$\begin{cases} 0 \text{ if job } j \text{ in machine } i \text{ of schedule } a \text{ (i.e., } (j,i,a)\text{) is immediately followed by the same job that follows } (j,i,b). \\ 1 \text{ otherwise.} \end{cases}$$

Here, we consider that (in all machines) the last job is followed by the same fictitious job. Thus, if the maximum number of jobs for a given machine is n then the maximum number of differently sequenced jobs for that machine will be n.

For example, let us suppose we have 3-jobs 3-machines job shop schedules defined as:

M1: 1 2 3

M2: 3 2 1

M3: 3 1 2

for schedule a and,

M1: 1 2 3

M2: 2 3 1

M3: 2 3 1

for schedule b.

Here, **M1** represents machine 1 and **M1**:1 2 3 indicates the job number sequence in machine 1, i.e., job 1 is scheduled first, followed by job 2 and this must be followed by job 3.

We can see that **M1** in a and b, has the same job sequence, thus there is not difference between a and b in **M1**. For **M2** we see that the two first jobs are sequenced differently, while the last jobs are followed by the same fictitious job. In this case the difference for **M2** is 2. In **M3** all the jobs are sequenced differently, resulting in a difference of 3. Therefore, we have a total difference between schedules a and b of $dif(a,b)$=5.

The disjunctive graph distance may also be used to measure the difference between two given schedules. Since there are many combinatorial problems that can be represented as disjunctive graph problems, this distance can be well exploited.

Definition 2. The population diversity of G individuals for a JSSP with n jobs and m machines and difference $dif(a,b)$ between schedules a and b is given by

$$Div = \frac{2}{mnG(G-1)} \sum_{i=1}^{G-1}\sum_{k=1}^{G-i} dif(i, i+k). \tag{2}$$

Notice that this definition is independent of both the population size and the problem size. This is because we take into account the total number of schedule difference computations ($G(G$-1$)/2$), and the maximum number of different job sequences for any given instance (mn).

With this definition we have a quantitative view of what is happening at the phenotype level, avoiding any wrong judgment at the genotype level. For example, if we try to estimate the diversity through any other means, let us say objective value variance, there could be cases where this variance and the real population diversity are poorly correlated, leading to a wrong judgment of what is happening at the population level.

We can see that in this setup of diversity there is a maximum population size $Gmax$ for which a maximum diversity value is achieved. For instance, if we have two completely different schedules then $Div = 1$. In the same way for 3,4,...,$Gmax$ completely different schedules we will have $Div = 1$. If the number of schedules is greater than $Gmax$ we will have $Div < 1$.

Now, it could be interesting to see whether there is any relation among population size (G), diversity (Div), and $Gmax$. To study this we need to compute $(mn)Div$ for the same problem instance, and different G's. This analysis may help us to find the appropriate value of G to use.

4 Diversity Oriented Selection

We know that population diversity (for a given G) is affected by selection, crossover and mutation. To what degree each of these operations affects the diversity measure is still an open problem. A study on how mutation and recombination operations are related (for simple cases), was presented in (Spears, 1998). More general results are far still from being known.

There have been a number of propositions for crossover and mutation operations which are expected to generate a highly diverse population. However, such propositions for selection strategies have received less consideration.

Even though we can modify diversity by varying the mutation rate we choose the selection procedure as a high level diversity generator while still keeping the mutation rate as a low level diversity generator. In this way we will have higher degree of freedom for varying the population diversity.

The idea of the proposition originates in a simulation result of fish behavior affected by a narrow trap (Sannomiya and Tian, 1998). According to the simulation result, a fish school enters a trap and can get out of the trap if a certain condition holds. Since the space of the trap is narrow, the school tries to escape from the trap as soon as possible. Then it is considered that small escaping time leads to highly adaptive behavior of the fish school. It is also concluded in (Tian et al., 1999) that the adaptability of the fish behavior arises from a certain range of system diversity. Thus we lay down that the escaping time in fish behavior corresponds to the objective function (Makespan) for JSSP. On the basis of this analogy selection algorithms and the concept of neighborhood are proposed with reference to the fish behavior simulation.

It is still difficult, at this stage, to make a one-to-one analogy between the fish school behavior and the Genetic Algorithms. But, the idea of diversity could be exploited to analyze whether or not the hypothesis of strong relation between diversity and adaptability (in fish behavior) also holds for Genetic Algorithms. We have to pay special attention to the concepts of adaptability if we want to establish a complete analogy between these two systems. This is because the concepts of adaptability for the systems may not be so easily related.

The first step to study the diversity will be to generate methods that produce a high diverse population at small mutation rates. Our two proposed selection algorithms to achieve this goal are as follows.

4.1 Method A

As we have just explained, we can affect population diversity by modifying any of the three genetic operations. For our case we choose the selection operation as the means to increase the diversity measure. In our proposed method we basically choose N individuals from the initial population of G ($N<G$). Around each of these sampled individuals we construct a neighborhood of size M (including one copy of the sampled individual). From each neighborhood of size M we choose M individuals, using the RWS method to do this. Of the MN generated individuals we choose at random G individuals. Finally we use the standard RWS to choose two individuals for genetic operations and continue as in the standard GA (from Step 3 of Algorithm 1). We describe all this in the following way.

Algorithm 2. Diversity Oriented Selection I (DOS I)

Step 1. Set $i = 1$.

Step 2. Select at random one individual I[i] from Pop[k].

Step 3. Construct a neighborhood of M elements around I[i]; {I[i], J1[i], ..., JM-1[i]}.

Step 4. Apply RWS to each neighborhood {I[i], J1[i], ...,JM-1[i]} to get M individuals.

Step 5. If $i < N$ then set $i = i + 1$ and go to Step 2; otherwise go to Step 6.

Step 6. From the new MN individuals select randomly G of them.

Step 7. From these G individuals select two by RWS for genetic operations.

Step 8. Continue as in the standard GA.

It is noted that Step 2 of Algorithm 1 corresponds to M=1 and N=G for Algorithm 2.

4.2 Method B

In order to increase diversity we intentionally modify Step 7 in the previous algorithm so that instead of using RWS selection to choose two individuals for recombination we use a random selection procedure. We describe this in the following algorithm.

Algorithm 3. Diversity Oriented Selection II (DOS II)

Step 1. Set $i = 1$.

Step 2. Select at random one individual I[i] from Pop[k].

Step 3. Construct a neighborhood of M elements around I[i]; {I[i], J1[i], ..., JM-1[i]}.

Step 4. Apply RWS to each neighborhood {I[i], J1[i], ...,JM-1[i]} to get M individuals.

Step 5. If $i < N$ then set $i = i + 1$ and go to Step 2; otherwise go to Step 6.

Step 6. From the new MN individuals select randomly G of them.

Step 7. From these G individuals select two randomly for genetic operations.

Step 8. Continue as in the standard GA.

It is noted that for both methods N individuals I[i]; i = 1,2,...,N are selected randomly at each generation k. Thus, the neighborhood obtained by the proposed methods is just like a bubble generating on the surface of a hot spring. In this sense the model for these methods may be called Bubble Model, in contrast with Island Model or Stepping-Stone Model which are proposed for parallel genetic algorithms.

Next we describe how to construct the neighborhood in Step 3 of Algorithms 2 and 3.

4.3 Neighborhood Generation

There are many ways of constructing a neighborhood for the JSSP, see (Gen and Cheng, 1997) and (Yamada and Nakano, 1997). From the results in (Yamada and Nakano, 1996) we can see that a neighborhood that exploits the structure of the problem (i.e., a neighborhood constructed at the phenotype level) gets better results, in terms of solution quality. However, in our case the neighborhood is constructed at the genotype level, keeping in mind that what we are behind is an increase in population diversity. In the algorithm below we call I[i] to each of the N selected individuals. Jk[i] represents the kth-individual in the neighborhood of I[i]. We consider each individual to have an associate matrix **A** which defines it. That is to say, J[k].a(l,r) is the (l,r)-entry of the matrix **A** corresponding to individual Jk[i] in the neighborhood of individual I[i]. The same goes for the matrix corresponding to I[i].

Algorithm 4. Neighborhood Generation.

```
Step 1. Set $i = 1$.
Step 2. For $p$=1 to $M$ do J$p$[$i$]=I[$i$].
Step 3. Set $k$=1.
Step 4. Set J$k$[$i$].$a$[1,$m$]=I[$i$].$a$[($k$+1),$m$], Set J$k$[$i$].$a$[($k$+1),$m$]=I[$i$].$a$[1,$m$].
Step 5. If $k$<$M$-1 then set $k$=$k$+1 and go to Step 4; otherwise go to Step 6.
Step 6. If $i$<$N$ then set $i$=$i$+1 and go to Step 2; otherwise go to Step 7.
Step 7. End.
```

5 Experimental Results

After having described these three selection methods let us take a look of the diversity behavior for each case. Figure 1 shows the behavior of diversity against generations for the famous FT10X10 benchmark JSSP (Muth and Thompson, 1963). The diversity measure given by (2) is computed every 10 generations. Ten trials for each algorithm were performed and the diversity average value is displayed for each case. The population size G was 201. The sampling size and neighborhood size for methods A and B were $N = 40$ and $M = 5$, respectively. Crossover probability Pc and mutation probability Pm were fixed (for all experiments) to 0.9 and 0.05, respectively.

We can appreciate in Figure 1 that as we expect the diversity for the starting generations is high and it decreases as generations increase. This behavior is because we start with a randomly generated population. In the process of evolution (for the standard GA) top ranked individuals are copied to next generations with high probability. This fact makes each generation less and less diverse, until the minimum steady value diversity is reached. This minimum value depends on the mutation rate.

In method A (see Figure 1) the diversity is higher than in the standard GA because the N individuals are selected randomly from the original population G, i. e., all the individuals have the same probability to participate in the neighborhood generation and, in this way there is high probability of generating different neighborhoods that contribute to the increase in diversity. However, in the standard GA no neighborhood is generated, then only the fittest individuals have high probability to survive.

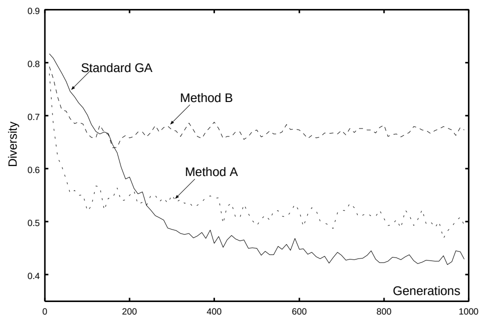

Figure 1. Diversity for FT10X10 problem. Pop. size=201, N=40, M=5.

In method B (see Figure 1), a high steady diversity value can be seen. This is mainly because in the selection procedure, different individuals surviving the first RWS (Step 4, Algorithm 3), have the same probability to pass to the next generations.

Figure 2 shows the behavior of the objective value (Makespan) against generations. It is observed from the figure that, comparing to the standard GA, the quality of solution for both methods, is decreased in about the same order as the diversity is increased. In this case we can see a trade-off between diversity and accuracy among the methods; the more diverse the less accurate.

Figure 3 shows the relations between diversity and mutation rate for fixed neighborhood and sample sizes. We can appreciate that our design goal is achieved, i.e., the two proposed methods generate higher diversity values for Pm up to 0.2 approximately for method B and up to 0.1 approximately for method A. The mutation rate for this experiment is varied from 0.0 to 1.0. Diversity values are measured over the last generations and the average value over 10 trials is displayed.

In order to study the effect of neighborhood size on diversity and accuracy we carried out the following experiments. The neighborhood size (M) in Algorithm 2 and 3 is varied from 3 to 10 keeping the sampling size N to constant values (40, 60 and 80). The diversity measure is computed for generations 970, 980 and 990. The average value for these generations over 10 trials is shown in Figure 4 for method A and in Figure 5 for method B, respectively. The average value for the Makespan over the same number of trials is shown in Figure 6 for method A and in Figure 7 for method B, respectively. Finally, the same experiments are repeated but in this case with a population size of 301 individuals. Figures 8 and 9 for diversity and Figures 10 and 11 for Makespan, show the corresponding results.

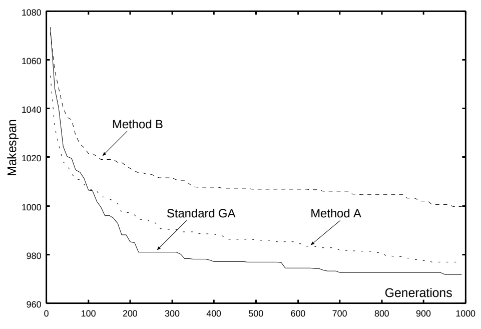

Figure 2. Makespan for FT10X10 problem. Pop. size=201, N=40, M=5.

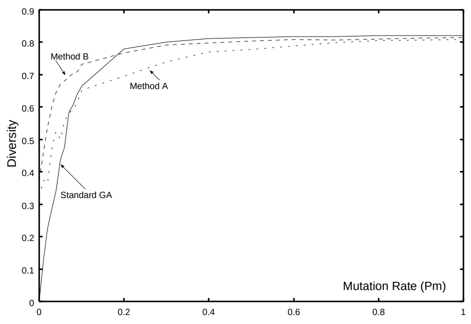

Figure 3. Diversity variation with Pm. Pop. size=201, N=40, M=5.

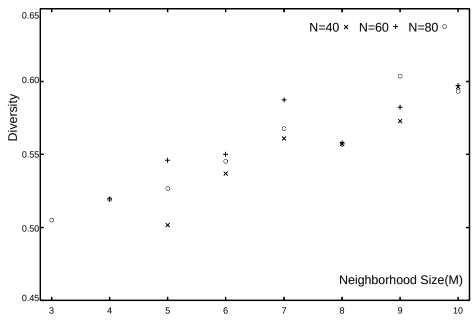

Figure 4. Variation of diversity with neighborhood size for Method A. Pop. size=201.

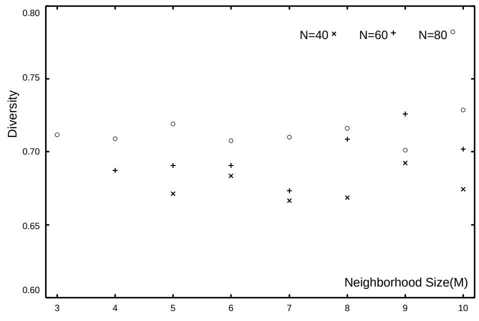

Figure 5. Variation of diversity with neighborhood size for Method B. Pop. size=201.

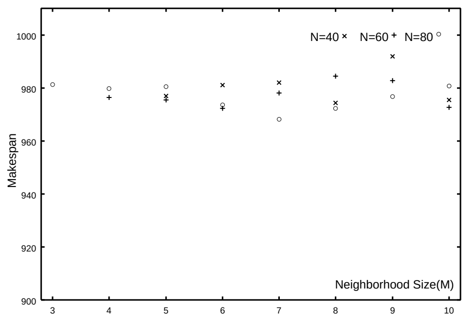

Figure 6. Variation of makespan with neighborhood size for Method A. Pop. size=201.

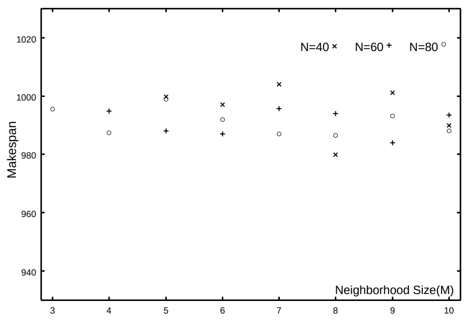

Figure 7. Variation of makespan with neighborhood size for Method B. Pop. size=201.

From these simulation results we can see that the neighborhood size affects population diversity when we use a fittest selection procedure (method A). This means that selection level diversity control can be achieved. However, the measure is not much influenced when dealing with a low selective procedure (method B). If we take a close look at Figures 4 and 8 we can see that the diversity has an increasing tendency as M (the neighborhood size) increases. This is because the main contribution to the population diversity is due to neighborhood diversity, thus the bigger M the higher the diversity. In the case of method B (Figures 5 and 9) the main contribution to diversity is done by the random selection process for individuals to participate in recombination and mutation. Due to this fact the neighborhood size slightly affects the population diversity.

The increase in population size contributes to the increase in population diversity for both methods. The reason for this is simple; the bigger our universe is, the higher the probability to have a diverse population.

Now, it can be also appreciated that for method B the increasing in sampling size N increases the diversity (Figures 5 and 9). For method A such behavior does not appear (Figures 4 and 8).

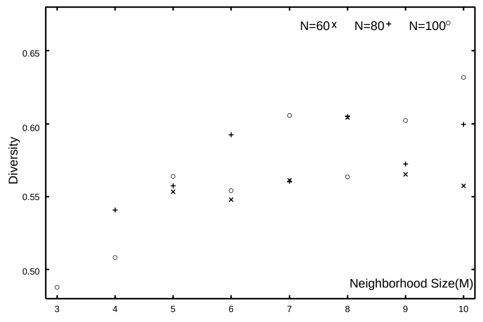

Figure 8. Variation of diversity with neighborhood size for Method A. Pop. size=301.

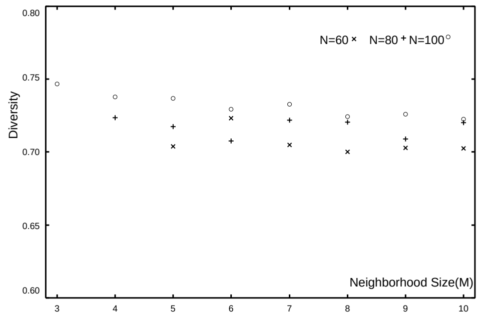

Figure 9. Variation of diversity with neighborhood size for Method B. Pop. size=301.

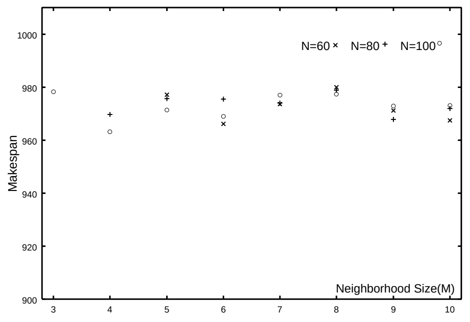

Figure 10. Variation of makespan with neighborhood size for Method A. Pop. size=301.

In the case of Makespan behavior (for different M) we did not have a clear pattern as in the diversity case for any of the methods.

6 Conclusions

A clear definition of population diversity and a way to compute it have been given. We use this measure to analyze two proposed DOS algorithms, called Bubble Model. These algorithms can keep a high diversity values at each generation when comparing to the standard GA method. There is a clear trade-off between accuracy and diversity among the three methods.

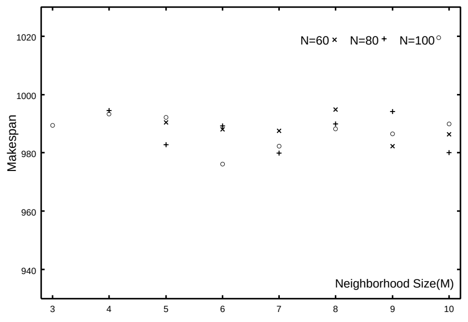

Figure 11. Variation of makespan with neighborhood size for Method B. Pop. size=301.

In Method A a good accuracy and selection level diversity control is achieved. The selection level diversity control is paid with higher computational cost, when comparing with the standard GA.

For method B (a low selective procedure) the neighborhood size slightly affects the diversity measure. It means that small value of M is enough to keep a high diversity measure (i.e., cheap computational cost).

In any case, there was not a clear relation between accuracy and neighborhood size as there was between accuracy and diversity among different methods.

The diversity measure will play a crucial role in the study of relations between population diversity and algorithm adaptability. We see a potential applicability of both the diversity measure and the proposed methods in areas such as re-scheduling or multi-objective optimization. Future research is planned to verify this hypothesis and to apply this idea to real world problems. More computational experiments are needed in order to compare our proposed Bubble Model with previously proposed models.

It is important to say that concepts related to adaptability should be clearly established before attempting to do any further comparison or study. Future work is aimed to establish such definitions.

Acknowledgements

This work is partly supported by the Research for the Future Program JSPS-RFTF97I00102 of the Japan Society for the Promotion of Science.

References

E. H. L. Aarts, J. K. Lenstra, N. L. J. Ulder (1994). A Computational Study of Local Search Algorithms for Job Shop Scheduling. *ORSA Journal on Computing* **6**(2):118-125.

D. Applegate and W. Cook (1991). A Computational Study of the Job Shop Scheduling Problem. *ORSA Journal on Computing* **3**(2):149-156.

L. Davis (1985). Job Shop Scheduling with Genetic Algorithms. *Proceedings of the International Conference on Genetic Algorithms and Their Applications* 136-140. Pittsburgh, PA.

M. Gen and R. Cheng (1997). Genetic Algorithms and Engineering Design. John Wiley & Sons, NY, USA.

S. Kobayashi, I. Ono, and M. Yamamura (1995). An Efficient Genetic Algorithm for Job Shop Scheduling Problems. *Proceedings of 6th International Conference of Genetic Algorithms*, 506-511.

J. F. Muth and G. L. Thompson (1963). Industrial Scheduling. Prentince-Hall, Englewood Cliffs, N. J.

N. Sannomiya and Y. J. Tian (1998). The Effect of System Diversity on Group Decision and Behavior: An Idea Based on the simulation result of Fish Behavior Model. *Proc. of The Second Japan-Australia Joint Workshop on Intelligent and Evolutionary Systems*, 1-9. Kyoto.

G. Shi, H. Iima and N. Sannomiya (1996). A New Encoding Scheme for Solving Job Shop Problems by Genetic Algorithm. *Proceedings of 35th IEEE Conference on Decision and Control* **4**:4395-4400. Kobe.

G. Shi, H. Iima and N. Sannomiya (1997). Comparison of Two Genetic Algorithms in Solving Tough Job Shop Scheduling Problems. *Trans. IEE of Japan* **117-C**(7):856-864.

W. M. Spears (1998). The Role of Mutation and Recombination in Evolutionary Algorithms. Ph.D. Dissertation. George Mason University.

E. D. Taillard (1994). Parallel Taboo Search Techniques for the Job Shop Scheduling Problem. *ORSA Journal on Computing* **6**(2):108-117.

Y. Tian, N. Sannomiya and H. Nakamine 1999. A Simulation Study on Adaptability to Enviromental Variations Based on Ecological Systems. *Proceedings of American Control Conference*, ACC 99. San Diego, (to appear).

R. J. M. Vaessens, E. H. L Aarts and J. K. Lenstra (1996). Job Shop Scheduling by Local Search. *INFORMS Journal on Computing* **8**(3):302-317.

T. Yamada and R. Nakano (1995). A Genetic Algorithm with Multi-Step Crossover for Job-Shop Scheduling Problems. *First IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA '95)*, 146-151. Shieffield, UK.

T. Yamada and R. Nakano (1996). A Fusion of Crossover and Local Search. *IEEE International Conference on Industrial Technology (ICIT '96)*, 426-430. Shaghai.

T. Yamada and R. Nakano (1997). Genetic Algorithms in Engineering Systems. IEE Control Engineering Series 55, 136-140. A. M. S. Zalzala and P. J. Fleming, Editors.