Comparison between Genetic Algorithms with Exaptation and Case Injected Genetic Algorithm in dynamic job shop problems

Luis Torres-Treviño

Corporación Mexicana de Investigación en Materiales
Oceania 190, Fracc. Saltillo 400
Phone (844)-411-32-00
Saltillo, Coahuila, Mexico.
ltorres@comimsa.com.mx

Abstract. It is propose new evolutionary algorithms with exaptive properties to tackle dynamic job shop problems. Exaptation is a new theory with two implicit procedures of retention and reuse of old solutions. The retention of a solution involves some kind of memory and the reuse of a solution implies the adaptation of the solution to the new problem. The first algorithm proposed applies seeding techniques to reuse a solution and the second algorithm proposed uses memory with seeding techniques to retain and reuse solutions respectively. Both algorithms are compared with a simple genetic algorithm with random initialization (RIGA) and the Case Injected Genetic Algorithm (CIGAR). The test include two selection mechanism, the tournament selection of size two and the selection proposed by Eshelman.

Track category: Evolutionary Scheduling and Routing

1 Introduction

In recent years the optimization of dynamic problems has become a growing field of research. The real-world problems are not static, they exist in a dynamic environment and it is necessary to modify the current solution when a change is detected. It is necessary evolutionary algorithms that do not re-start in every change; these algorithms must take advantage of the population information to obtain a valid solution in a short time. It is expected a similar solution when there are a minimum change of the problems and a dissimilar solution when there are an important change of the problem.

Some examples of dynamic problems are job shop problems where there are changes in due time, changes in the number of machines, processing times, etc., and these changes imply a re-schedule in the job shop. Learning in dynamic environments is a desirable quality for mobile robots. Navigation represents a simultaneous problem of path planning and movement to the goal along the path. Finally consider an company where in some periods there is high demand

Advances in Artificial Intelligence and Applications
Research in Computer Science 32, 2007, pp. 249–257

Received 15/06/07
Accepted 31/08/07
Final version 30/09/07

of products and in another periods there is low demand. Usually it is necessary an optimization algorithm to manage and control sources efficiently with this dynamic demand. All of these dynamic problems can be modeled with variables, the optimization function and a set of constrains. Every one of them can changes through time [1]. Many authors have suggested some extension in the simple genetic algorithm to tackle these dynamic problems. Branke has suggested the following categories to group the algorithms proposed [2, 3]:

- Evolutionary algorithms that detect every change in the environment. If it is detected some change, then new individuals are injected into the population to increase diversity.
- Evolutionary algorithms which have an implicit memory. These algorithms use double or more complex representations (diploid, haploids) [4, 5]. In a given moment just one representation is active.
- Evolutionary algorithms which have an explicit memory to store useful information of the past and it is recalled when the dynamic problem returns to a similar situation presented in the past [6, 7].
- Evolutionary algorithms which avoid every time the convergence. Genetic algorithms with sharing and random immigrants are examples of these kinds of algorithms.
- Evolutionary algorithms that use a multiple subpopulations to search the optimum solution or to search a new one.

In the following sections it will be presented a comparison of two new evolutionary algorithms inspired in exaptation, versus the simple genetic algorithms with random initialization (RIGA), and the Case Injected Genetic Algorithm (CIGAR) proposed by Sushil J. Louis to solve dynamic job shop problems. Section II reviews the exaptation theory. In section III it is proposed two genetic algorithms inspired in exaptation theory. Section IV reports some experiments and results of the comparisons. Finally, the conclusions and future work will be give in the last section.

2 Exaptation

Gould and Vrba [8, 9] proposed the term exaptation, which refers to a trait that current provides fitness, but originally arose for some other reason. Every entity (species) tries to survive in a continuous non-static environment. The entity has traits which lets survive in the environment. Some traits are useful because provide high fitness but another ones do not provide fitness, they are usually useless. Some traits may have evolved in one context of the environment but later, such a trait may be co-opted for use in a different role. In other words, the exaptation is a change in the function of an old trait to solve a new problem similar or very different to the original one.

It is possible to describe three procedures in exaptation. First, when there is a change in the environment it is detected a set of possible traits with high fitness. Second, it is possible to reuse useful traits with high fitness and adapt

them to the new environment. The third procedure retains a useful trait for future references; however, the useless traits do not disappear completely, they are stored as redundant or useless structures.

The initial population of a genetic algorithm is random. When the simple genetic algorithm (SGA) solves a problem, it takes several cycles to get an optimum or an individual with high fitness. In the last generation the population has individuals that are very similar between them, it means, there are a low diversity. If there is a change of the environment (i.e. change the objective function) and some individuals are useless maybe some of them have useful traits. These traits are components of the phenotype so there are some genes to represent them. If the SGA runs again with a changed in the environment and the same population (without a random initialization), the useful traits may arise and they can let to get a new optimum quickly. In the SGA it is possible to reuse the last population if the function does not change too much. If the change is important then the reuse of the last population can be useless. It is necessary to modify the SGA in order to get some features of exaptation and it can be used to solve dynamic problems.

3 Genetic algorithms with exaptation and CIGAR

The first algorithm proposed is the SGA with seeding techniques; this algorithm is inspired in exaptation because it is reused some structures when the algorithm detects any change in the optimization function. If this change is detected then some variations or neighborhoods of the best solution found are injected into the actual population. The variations or neighborhoods replace a percent of the total population. It could be that some components (genes) of the best individual can be reused to improve the fitness in the new optimization function. The neighborhoods of the solution can give the appropriate solution if the problem changes slightly. The variations (mutations) of the solution can give a key to improve the fitness if the problem changes abruptly. This algorithm can be classified as an approach that detect a change and it is applied an injection.

The second algorithm is inspired in exaptation too and it is implemented from the point of view of learning by analogy; this learning mechanism has a memory of useful solutions of the past, a storing procedure of solutions, a search mechanism and a modification procedure. The second algorithm applies similar procedures: A recognition procedure where it is used the evaluation of the best individual to be compared again the best evaluation found in the past. If there is a degradation of the past solution then there is a change in the objective function. Storing procedure saves the best individual found into the memory. This avoids to store the same individual in the memory two times. The procedure is simple; first, it locates the most similar individual of the memory to the best individual found and if the best individual found has better evaluation than the individual of the memory then the best individual is stored into the memory. In other case, there is not change in the memory. This procedure tries to apply the exaptive

property of retention of solutions. The modification procedure is based in seeding techniques [10].

The algorithms implemented are the Randomly Injected Genetic Algorithm (RIGA), the Case Injected Genetic Algorithm (CIGAR), the SGA with seeding (SGAS) and the SGA with seeding and memory (SGASM). All the algorithms reuse the last population where it is initialized just at the beginning of the run. The RIGA is a simple genetic algorithm with random initialization and this is shown below. P is the population, F_E retains the evaluation value of every individual in the population. I is an auxiliary variable and it retains the best individual of the population per generation. The evaluation value of the Individual I is saved in e.

Simple Genetic Algorithms

1) $P \leftarrow$Random initialization
2) $F_E \leftarrow$ Evaluation(P)
3) Get the best individual I and its evaluation e from F_E
4) $P \leftarrow$ Selection(P, F_E)
5) $P \leftarrow$ Reproduction(P) (crossover and mutation)
6) If end condition is not satisfied, then go to step (2)
7) End

The CIGAR proposed by Sushil Louis consists of a case based reasoning system (CBRS) with a genetic algorithm. The components of each case are: the best solution, the number of generation in which it was stored, the value of evaluation, and the description of the problem to detect similarities. CBRS can place solutions of examples to have some reference in the search. The system initiates in random form and solutions are injected when some cases are similar to the new problem. In each certain number of generations, solutions of the CBRS are injected into the population of the genetic algorithm when the problem is similar to one already solved. The system looks for best the individuals of the population and it is stored in the memory of the CBRS, replacing the most similar solution but with smaller value of evaluation. The process is repeated in certain number of generations; for example, solutions in generations 1, 25, 50 and 100 can be stored and be injected. There are four forms in CIGAR to inject solutions, the first form consists of injecting solutions that are similar to the best-found solution; the second form consists of injecting solutions that are similar to the worse found solution. In the third form, solutions of CBRS are injected and chose according to a probability factor that is inversely proportional to the Hamming distance between the solution stored in the CBRS and the best solution found in the population. In the fourth form, solutions of the reasoning system are injected according to a probability factor that is directly proportional to the Hamming distance between the cases of the reasoning system and a solution of the population. In all the cases the injected solutions replace the worse individuals of the population. Lately the author has used his system CIGAR without the mechanism to detect similarities because it implies a high cost or a difficulty to determine a metric for certain problem. In was replaced by the fitness evaluation [12–14].

The CIGAR algorithm is shown below. In these experiments was used the fitness value as a metric to detect similarities and it is injected the solutions of the memory I_M (extracted) that are similar to the best-found solution of the population [16–18]

Cased Injected Genetic Algorithm
1) $P \leftarrow$ Random initialization
2) $M \leftarrow$ Random initialization or empty memory
3) $F_E \leftarrow$ Evaluation(P)
4) Get the best individual I and its evaluation e from F_E
5) $P \leftarrow$ Selection(P, F_E)
6) $P \leftarrow$ Reproduction(P) (crossover and mutation)
7) Every 25 generations save the best solution I into the memory M (case based)
8) Every 25 generations extract the solution I_M (see text above) from M and inject it in P
9) If end condition is not satisfied, then go to step (3)
10) End

The genetic algorithm with seeding injects neighbors of the best individual I when it detects a change in the objective function.

Genetic algorithm with seeding techniques
1) $P \leftarrow$ Random initialization
2) Get the best past evaluation e_a
3) $FE \leftarrow$ Evaluation(P)
4) Get the best individual I and its evaluation value e from F_E
5) If $e < e_a$ then apply seeding of 50% of neighbor and variations of I in P
6) $P \leftarrow$ Selection (P, F_E)
7) $P \leftarrow$ Reproduction (P) (crossover and mutation)
8) Includes Elitism injecting I in P
9) If end condition is not satisfied, then go to step (2)
10) End

The SGA with memory and seeding has a memory to save the best solution found. The algorithm uses a SGA with seeding of neighborhoods and variations of 50% (25% each one). The rest is random. The condition to make a seeding is when the evaluation value of the memory e_m is greater than the best past evaluation value e_a. When the memory detects a small evaluation in the individuals of the memory this is an indication that the function changes and there is an unknown solution, so it is necessary to find a new one considering only the information of the memory [11].

Genetic algorithm with memory and seeding techniques

1) $P \leftarrow$ Random initialization
2) $M \leftarrow$ Random initialization or empty memory
3) Get the best past evaluation value e_a
4) $FEM \leftarrow$ Evaluation(M)
5) Get the best individual I_m and its evaluation value e_m from F_{EM}
6) If $em > e_a$ then applies seeding of 50% of Neighborn and variations of I_m in P
7) $FE \leftarrow$ Evaluation(P)
8) Get the best individual I and its evaluation value e from F_E
9) $P \leftarrow$ Selection(P, F_E)
10) $P \leftarrow$ Reproduction(P) (crossover and mutation)
11) Every k generations saves in memory M the best individual I
12) If end condition is not satisfied, then go to step (3)
13) End

It is important to describe the injection mechanism of the genetic algorithms with exaptation. The procedure requires the injection of variations and neighborhoods that represent sequence solution. The generation of neighborhoods use a method called one step look-back interchange [19]. It is necessary a sequence of operations and the list of machines where the operations were allocated. The method first locate a job k that was assigned in machine m (called position p_1) and locate another operation which has the same machine m (called position p_2). In the next step it is interchanged the operations between position $p1$ and $p2$ so we have a new sequence. The solutions variation is a simpler procedure where it is selecting two random positions of operations and it is interchanged the operations among them.

4 Experimentation and results

This experiment is inspired in Sushil Louis works [15]. The problem is to minimize the makespan; that means, the total elapsed time between the beginning of the first task and the completion of the last task. In the problem of dynamic JSP there are 10 jobs and 10 machines; each job comprises a set of operations or task. The operations must each be done in different machines in a given job-dependent order.

A machine does not process two different jobs at once and different operations of the same job are not simultaneously being processed on different machines. The sequence of operations or tasks represents a chromosome of size 100 (10 jobs x 10 operations per job). The enconding follow the scheme proposed by Fang [20].

It is generated with a uniform random initialization 10 jobs with 10 operations. Each job has a due time between 20 and 500; the processing time of every

job is between 4 and 12 time units and the job-dependent order has a uniform random initialization. This is called the basic problem.

It is generated 50 problems picking randomly the processing time of the operations (40% of the total operations) from the basic problem and the processing time has a random modification adding or resting up to 4 units of time.

For the experiment all the evolutionary algorithms has a crossover probability of 0.9 and mutation rate of 0.0725, it is used a swap mutation and a greedy crossover [21], a total population size of 100 individuals. The CIGAR memory has the capacity to save 50 solutions (cases) and the SGASM memory has the capacity to store 10 solutions. It is used a selection tournament of size 2. Every generation involves 100 evaluations per generation. All the algorithms solve the same problems. It is reported the best solution found per algorithm per problem.

It is used two selection procedures. The selection procedure proposed by Eshelman [22] was used by Sushil Louis in his CIGAR system. Every plot illustrates the performance to minimize the total cost of makespan in every problem through time. The figure 1 shows all the algorithms for the dynamic JSP problem with selection proposed by Eshelman. The plot suggests a better performance of the genetic algorithms with seeding and memory.

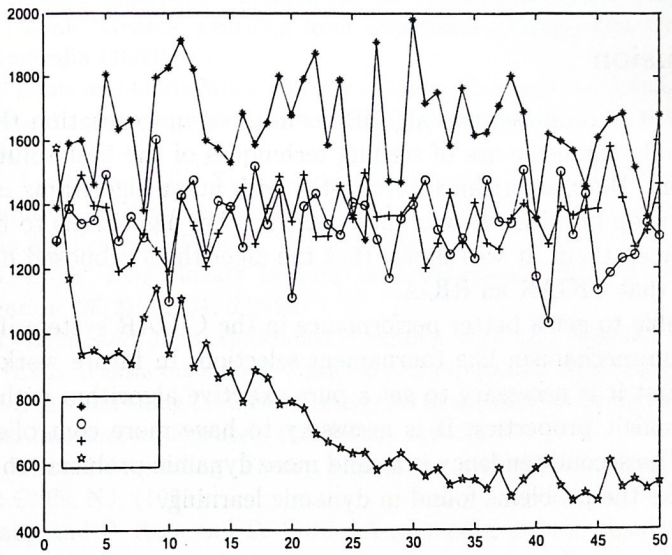

Fig. 1. Performance of several approaches in every problem using selection proposed by Eshelman.

It was changed the selection procedure and it is used the tournament selection of size two in all algorithms. The figure 2 shows a better performance of the genetic algorithms that use exaptation.

RIGA has a poor performance because do not use the last information and take the every problem as a new one.

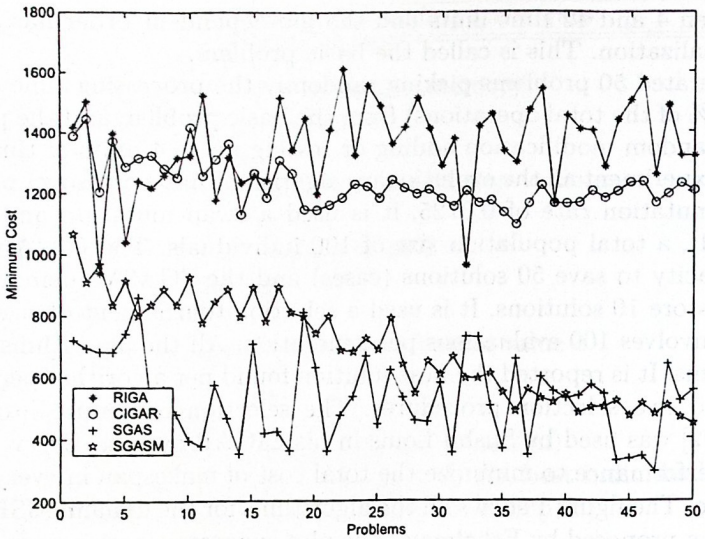

Fig. 2. Performance of several approaches in every problem using tournament selection of size 2.

5 Conclusion

In this paper it is proposed two algorithms inspired in exaptation theory. The algorithms make intensive use of seeding techniques of the best solution useful at the moment. Both algorithms were tested with other algorithms in a simple dynamic job shop problem. It was implemented other algorithms to make comparisons between them. It was shown that the algorithms proposed have better performance that CIGAR an RIGA.

It is possible to get a better performance in the CIGAR system if it is used other selection mechanism like tournament selection. In future work there are two routes, first it is necessary to get a pure exaptive algorithm with retention and reuse implicit properties. It is necessary to have more controlled seeding techniques. The second tendency is to find more dynamic problems to test these algorithms like the problems found in dynamic learning.

References

1. Trojanowski, K. and Michalewicz, Z. Evolutionary algorithms for non-stationary environments. In: Proc. of 8th Workshop: Intelligent Information systems, ICS PAS Press, (1999) 229–240
2. Branke, J. Evolutionary approaches to dynamic optimization problems; a survey. In: GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems, (1999) 134–37
3. Branke, J. Evolutionary approaches to dynamic optimization problems - Updated survey. In: GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems (2001), 27–30

4. Goldberg, D. Genetic algorithms in search, optimization, and machine learning. Addinson - Wesley. Pub. Co., Reading MA (1989).
5. Dasgupta, D. and McGregor, D. Nonstationary function optimization using the structured genetic algorithm. In: R. Männer and B. Manderick, editors, Parallel Problem Solving from Nature, Elsevier Science Publisher (1992) 145–154
6. Louis, Sushil and Johnson, J.: Robustness of Case-Initialized Genetic Algorithms, FLAIRS-99, Orlando, FL, AAAI Press, (1999)
7. Connie Loggia Ramsey and John J. Grefenstette: Case-Based Initialization of Genetic Algorithms. Proc. of the Fifth Int. Conf. on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA. (1993) 84–91.
8. S. J. Gould and S. Vrba: Exaptation: A missing term in the science of form, Paleobiology, volume 8 (1982) 4–15
9. Stephen Jay Gould: A crucial Tool for an Evolutionary Psychology, Journal of Social Issues, volume 47, (1991) 43–65
10. Torres-Treviño, L. Sistemas exaptivos: retención y reutilización de conocimiento en algoritmos evolutivos. Tesis Doctoral. Instituto Tecnológico de Estudios Superiores de Monterrey. Campus Monterrey (2004)
11. Torres, L. GA with Exaptation: New Algorithms to Tackle Dynamic Problems. Lecture Notes in Computer Science, 2972, (2004) 746-753
12. Sushil J. Louis and Judy Johnson: Solving Similar Problems Using Genetic Algorithms and Case-Based Memory, Proceedings of the Seventh International Conference on Genetic Algorithms, Morgan Kauffman, San Mateo, CA, (1997) 283–290
13. Sushil J. Louis: Genetic Learning from Experience, Accepted in CEC-03, Camberra, Australia (2003)
14. Sushil J. Louis and Chris Miles. Playing to learn: Case-injected genetic algorithms for learning to play compuer games. IEEE Transactions on Evolutionary Computation, 9(6). 2005
15. Sushil J. Louis and Zhijie Xu: Genetic algorithms for Open Shop Scheduling and Re-Scheduling, M. E. Cohen and D. L. Hudson, ISCA 11th Int. Conf. on Computers and their Applications, Raleigh, NC (1996) 99–102
16. Sushil J. Louis. Evolutionary learning from experience. Journal of Engineering Optimization, 26(2), (2004) 237-247
17. Sushil J. Louis and John McDonnell. Learning with case injected genetic algorithms. IEEE Transactions on Evolutionary Computation, 8(4), (2004) 316-328
18. Sushil J. Louis and Gong Li. Case injected genetic algorithms for traveling salesman problems. Information Sciences, 122, (2000) 210-225.
19. Pinedo, Michael Scheduling: Theory, algorithms, and systems, Prentice Hall, Englewood Cliffs, NJ, (1995)
20. H. L. Fang and P. Ross and D. Corne: A promising genetic algorithm approach to job shop scheduling, rescheduling, and open shop scheduling problems, Proc. of the Fifth Int. Conf. on Genetic Algorithms, Morgan Kaufmann, San Mateo, CA., (1993) 375–382
21. G. E. Liepins and M. R. Hilliard and M. Palmer and M. Morrow: Greedy genetics, In Proceedings of the Second International Conference on Genetic Algorithms, Lawrence Erlbaum, (1987) 90–99
22. L. J. Eshelman: The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination, Foundations of genetic algorithms 1, (1991) 265–283.