STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVI**, Number 2, 2011

A BRIEF ANALYSIS OF EVOLUTIONARY ALGORITHMS FOR THE DYNAMIC MULTIOBJECTIVE SUBSET SUM PROBLEM

IULIA COMŞA[1], CRINA GROŞAN[1], AND SHENGXIANG YANG[2]

Abstract. The paper investigate the behavior of evolutionary algorithms for solving multiobjective combinatorial problems in dynamic environments. Present work envisages the multiobjective subset sum problem which is known as an NP-hard problem [2]. Several test and analysis are performed in order to asses the advantages and to point out the disadvantages and drawbacks of these classes of algorithms.

1. Introduction and problem formulation

The paper aims at analyzing the behavior of two classes of evolutionary algorithms for a well known combinatorial optimization problem – subset sum-but in its multiobjective form (more than one sum is considered) and in its dynamic version (the set of elements and the sums change in time).

The idea that motivates the majority of work in dynamic evolutionary optimization is the reuse of information uncovered in the past (and, to a lesser extent, the prediction of future dynamics). In other words, most evolutionary approaches to dynamic optimization problems (DOP) attempt to reduce the computational complexity of the dynamic problem by "transferring knowledge from the past" [1]. The number of publications in the field of dynamic evolutionary computation has increased significantly in recent years: [3]-[8]. The majority of work is motivated by the presence of real-world problems that are inherently dynamic: solutions to such problems need to be re-optimised, as time goes by to ensure feasibility and satisfactory quality.

Even thought the research on DOPs dealing with a single objective to optimize id increasing, there is still no significant research dealing with situations

Received by the editors: April 1, 2011.

2010 *Mathematics Subject Classification.* 91D30.

1998 *CR Categories and Descriptors.* I.6.3 [**Simulation and Modeling**]: Applications; G.2.2 [**Discrete Mathematics**]: Graph Theory – *Graph Algorithms*.

Key words and phrases. dynamic environment, multiobjective optimization, NP-completeness.

in which more than one objective is present. This paper proposes a case study involving multiple objectives and various dynamics.

The two evolutionary methods are a standard genetic algorithm and a genetic algorithm which uses an external population (archive) to store all the nondominated solutions found so far during the search process at a time step. The motivation for the need of such an archive comes in the explanation of the results obtained by the standard algorithm.

There are numerous variations of the classical subset sum problem, which is an NP-complete decision problem that may be solved in pseudo-polynomial time. In this paper, we consider its NP-hard optimization variant: given a set of positive numbers N and a positive integer S, the task is to find a subset of N the sum of which is as close as possible to c, without exceeding it.

In the multiobjective case, the problem comes slightly modified: given a set of positive numbers N and m positive integers S_1, S_2, $\ldots$, S_m, find m disjoint subsets of N the sums of whose are as close as possible to any of the S_i, i=1, 2, ...m, without exceeding them.

In the multiobjective case we are interested in finding as many Pareto optimal solutions as possible.

We consider two dynamic situations: one in which Pareto set is static and the other one in which Pareto set is dynamic. Objectives values change at each time step in both situations.

2. Description of the algorithms

The chromosome was represented as a string of size equal to the items in the set and values from 0 to 1 in case of two objectives and from 0 to 2 in case of three objectives.

At each iteration, the old population was completely replaced, by repeatedly selecting two chromosomes, combining them with a probability P_c, and mutating them on every position with a probability P_m.

Tournament selection was used. To select a chromosome, two random chromosomes were taken from the population, and the winner was either the dominant chromosome or, if they were non-dominant, one was randomly selected.

One-point crossover was used. The crossover point was randomly generated and it was assured that at least a gene from every chromosome would be transferred into the child.

Mutation was done on every chromosome before adding it to the new population. Mutation was strong, meaning that the value of a gene before and after mutation was always different.

Every iteration was repeated for a number of steps, after which the objective sums were modified (therefore also the fitness function).

Since the population was completely replaced at every iteration and mutations were involved, there was little chance that all the solutions will be present in the population at the last iterations; even if a Pareto optimal solution was found, it would be probably soon replaced. Therefore, we considered a second algorithm which keeps a special set of Pareto optimal solutions found by the algorithm – an external archive - that was updated at every iteration with the new found solutions.

When comparing two chromosomes, we first used Pareto dominance: a solution dominates another when its fitness is better (lower) with respect to one objective and equal or better with respect to the other objectives. If none of the solutions was dominant, the chromosomes were non-dominant and therefore considered equally good.

This comparison did not yield very good results on the second large data set (dynamic Pareto set), as it found roughly about 70% of the Pareto optimal solutions. Therefore, we further compared the non-dominant solutions using sum of fitness values with respect to every objective, considering better the chromosome which minimized this sum. The results improved (which was also influenced by crossover and mutation rate change), more than 90% of the Pareto optimal solutions being found.

3. Experiments

3.1. **Algorithm parameters.** Experiments are performed on two data sets, a small and a large one.

We use a crossover rate of 0.6 and a mutation rate of 1/n, n being the number of items in the set. The experiments on the large data set showed that a mutation rate two times lower and a higher crossover rate (0.75) would produce better results.

We used a population of 50 chromosomes for the small data set and of 100 for the large data set. We iterated 20 times at every sum change (time step) for the small data set and 50 times for the large set. This means that, in the case of the large data set, 5000 individuals (not necessarily distinct) were generated out of the 177147 possible (since the large data set consisted of 11 elements).

3.2. **Numerical tests.** We performed 2 tests for the DOP with 2 and 3 objectives respectively: one test for static and one for dynamic Pareto set. Each test considers two data sets. One set is a small set and the other one is larger. All the results presented here are averaged over 10 different runs.

The small data set for the dynamic Pareto set consists of the items: 1, 2, 3, 4, 5, 30, with the initial sums (4, 25) which are further modified (at each time step) into (5, 24), (6, 23) and so on.

The large data set for the dynamic Pareto set consists of the items: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 100, and the initial sums are 6 and 65 (and then they modify into (7, 64), (8, 63) and so on).

The small data set for the static Pareto set is formed by the items: 1, 2, 3, 20, 21, 80, with the initial sums (6, 70) which modify with the time steps into (7, 69), (8, 68) and so on.

The large data set for the static Pareto set is formed by the items: 1, 2, 3, 4, 5, 30, 31, 32, 135, 150, 200 and the initial sums (15, 118) which modify into (16, 117), (17, 116) and so on at each time step.

We call the data sets small or large based not the item set but on the number of Pareto solutions they generate.

It can be easily observed that for the small set both algorithms – standard genetic algorithm (GA) and GA using archive are able to find a number of Pareto solutions (see Figure 1 (for dynamic Pareto set and Figure 2 (for static Pareto set)). It is by far obvious that the algorithm incorporating archive is able to find a number of solutions close to the real number of Pareto optimal solutions (as generated by a backtracking algorithm for this simple test).

Figure 1. Comparison of standard GA and GA with archive using a small data set for the dynamic Pareto set case.

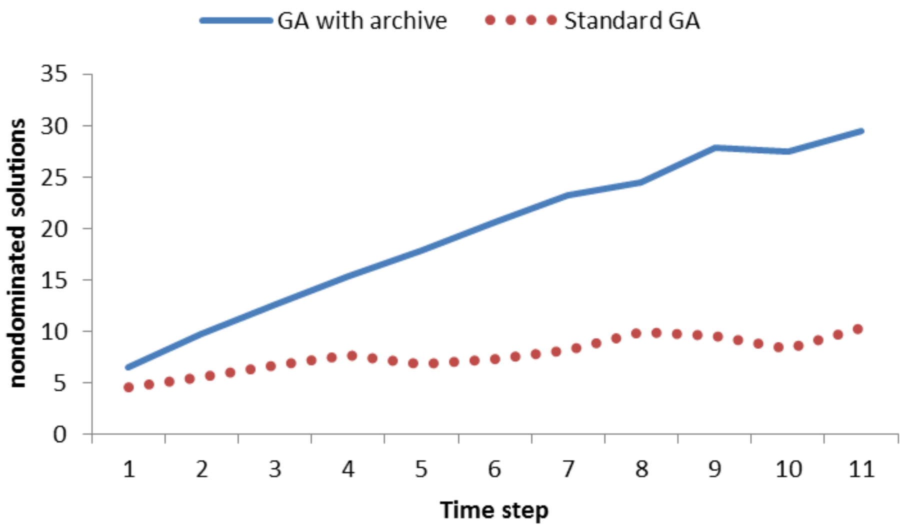
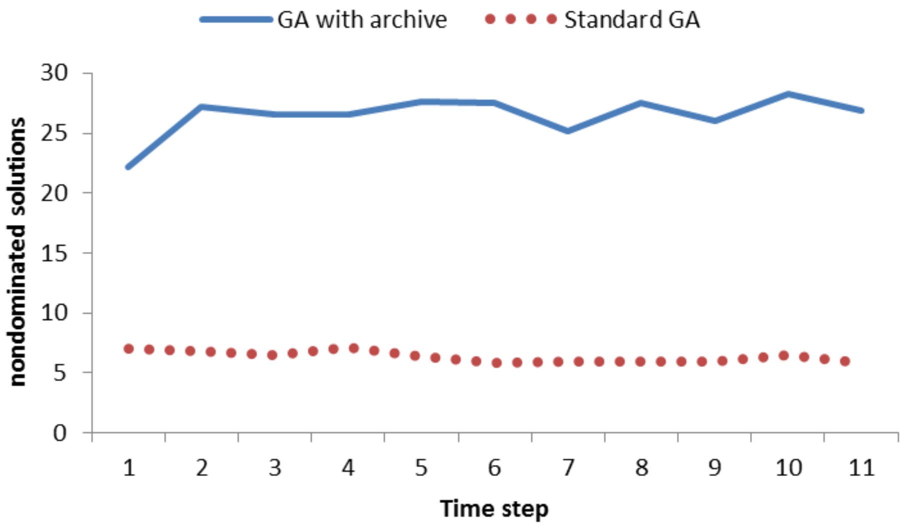


FIGURE 2. Comparison of standard GA and GA with archive using a small data set for the static Pareto set case.



In the case of large dataset, it can be noticed form Figures 3 and 4 (corresponding to dynamic and static Pareto set respectively) that the standard GA manages to increase the number of nondominated solutions as it approaches the final time steps in the case of dynamic Pareto set. It still remains a significant gap between standard GA and GA with archive. It is interesting that the number of nondominated solutions increases in standard GA in the situation in which the Pareto set is dynamic and not when it is static.

For the three objectives case we consider same 11 time steps as with the previous experiments. In this case, Pareto domination relationship among solutions will return most of the solutions as nondominated among them. This situation worsens with the increase in the number of objectives (4 or more). The convergence to the real Pareto from is much slower

The data set used is composed from the items: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 100 and the initial sums are (6, 65, 100) which then modify into (7, 64, 101), (6, 63, 102) and so on.

Figure **??** show the results obtained by the algorithm by averaging the objectives values for each of nondominated solutions found at the end of each time step. Minimum, maximum and average values among them are displayed in the graph. It cannot be observed a linear evolution towards the end of the

FIGURE 3. Comparison of standard GA and GA with archive using a large data set for the dynamic Pareto set case.

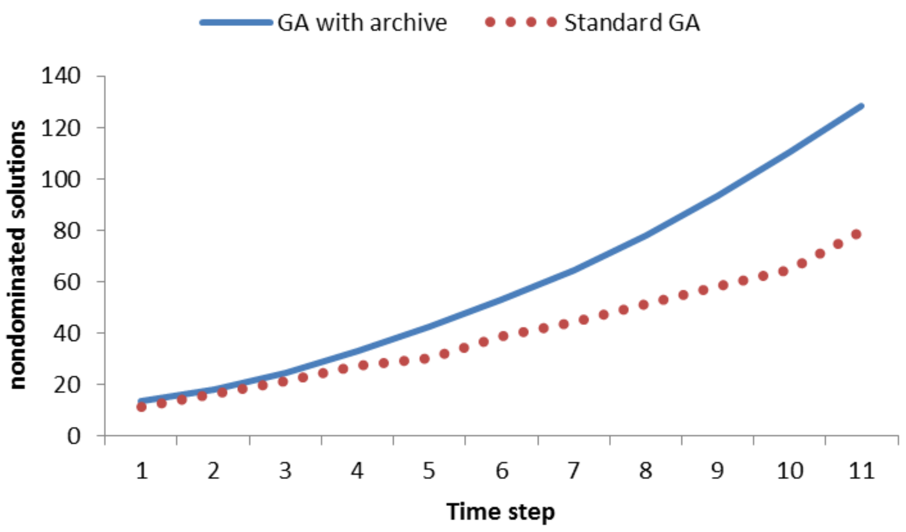
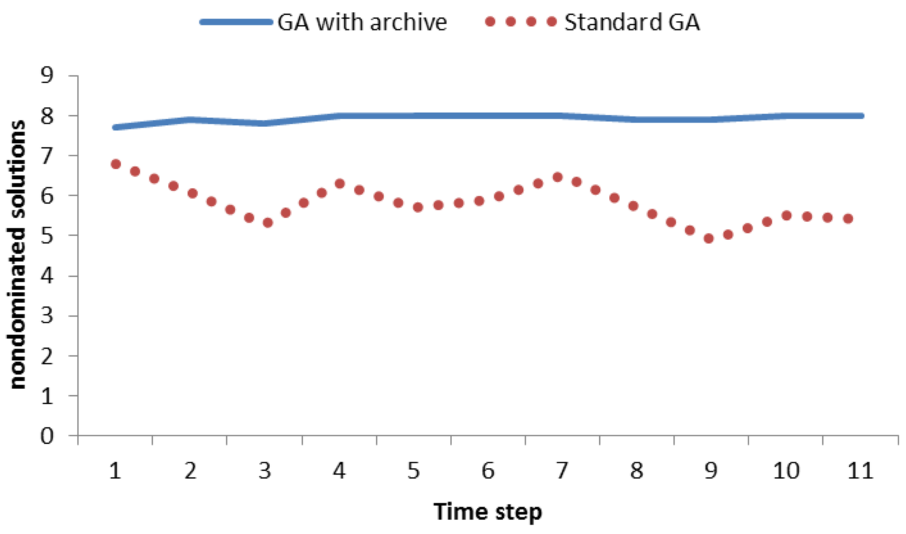
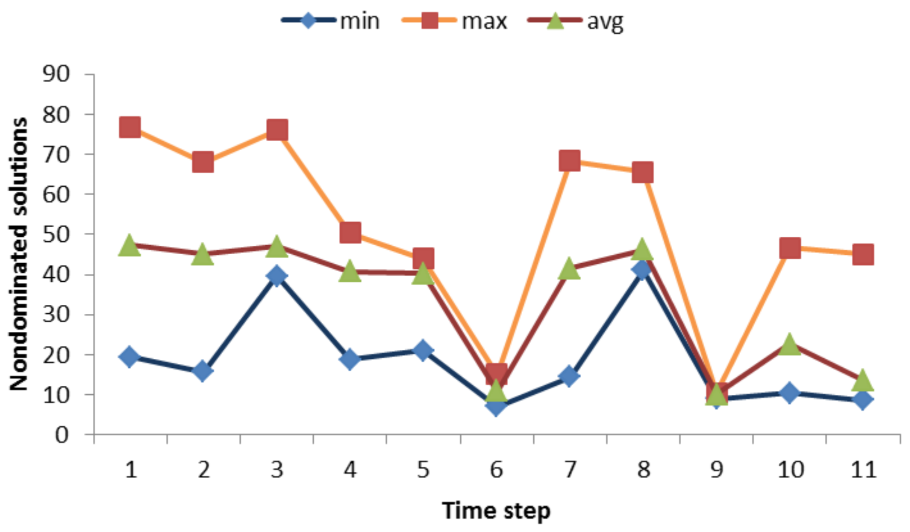


FIGURE 4. Comparison of standard GA and GA with archive using a large data set for the static Pareto set case.



dynamic process which clearly shows the algorithms need some improvements for an increased number of objectives.

FIGURE 5. Behavior of standard GA for 3 objectives subset sum problem.



4. CONCLUSIONS

The paper briefly analysis the behaviors of two types of genetic algorithms for the multiobjective dynamic version of the subset sum problem. Some of the conclusions and findings of this study are as follows:

The algorithm preserving all the nondominated solutions found so far during the search process approximates better the Pareto frontier.

Pareto nondominance relationship might not always be a relevant comparison measure among the solutions and additional information might be required.

Algorithms require improvements and extra information if the number of objectives is increased (to 3 or more criteria).

REFERENCES

[1] J. Branke, Evolutionary optimization in dynamic environments. Kluwer, Dordrecht, 2002.

[2] M.R. Garey, D. S. Johnson, Computers and Intractability; A Guide to the Theory of NP-Completeness, W. H. Freeman & Co. New York, NY, USA, 1990.

[3] S. Khuri, T. Back, J. Heitkotter, Evolutionary Approach to Combinatorial Optimization Problems, *Proceedings of the 22nd Annual ACM Computer Science Conference*,pp. 66-73, ACM Press, 1994.

[4] A.M.L. Liekens, Evolution of finite populations in dynamic environments. PhD thesis, Technische Universitat Eindhoven, 2005.

[5] R.W. Morrison, Designing evolutionary algorithms for dynamic environments, Springer, Berlin, 2004

[6] P. Rohlfshagen, X. Yao, Dynamic Combinatorial Optimisation Problems: An Analysis of the Subset Sum Problem, *Soft Computing*, DOI: 10.1007/s00500-010-0616-9, 2011.

[7] K. Weicker, Evolutionary algorithms and dynamic optimization problems. Der Andere Verlag, 2003.

[8] C.O. Wilke, Evolutionary dynamics in time-dependent environments. PhD thesis, Ruhr-Universitat Bochum, 1999.

[1] Department of Computer Science, Babes-Bolyai University, Cluj-Napoca, Romania

E-mail address: iulia.comsa1@gmail.com

E-mail address: cgrosan@cs.ubbcluj.ro

[2] Department of Information Systems and Computing, Brunel University, London, UK

E-mail address: Shengxiang.Yang@brunel.ac.uk