In V. Kurková, N. Steele, R. Neruda, M. Kárný (eds), Proceedings of the Fifth International Conference on Neural Networks and Genetic Algorithms (ICANNGA'01), pp. 236-239, Prague, Czech Republic, 22-25 April, Springer, 2001.

An Evolutionary Approach to the Zero/One Knapsack Problem: Testing Ideas from Biology

Anabela Simões[1,2], Ernesto Costa[2]
[1] Instituto Superior de Engenharia de Coimbra, Quinta da Nora, 3030 Coimbra, Portugal
[2] Centro de Informática e Sistemas da Universidade de Coimbra, Pinhal de Marrocos, 3030 Coimbra, Portugal
E-mail: abs@sun.isec.pt; ernesto@dei.uc.pt

Abstract

The transposition mechanism, widely studied in previous publications, showed that when used instead of the standard crossover operators, allows the genetic algorithm to achieve better solutions. Nevertheless, all the studies made concerning this mechanism always focused the domain of function optimization. In this paper, we present an empirical study that compares the performances of the transposition-based GA and the classical GA solving the 0/1 knapsack problem. The obtained results show that, just like in the domain of function optimization, transposition is always superior to crossover.

1 Introduction

Genetic Algorithms (GA) are a search paradigm that applies ideas from evolutionary biology (crossover, mutation, natural selection) in order to deal with intractable search spaces [5]. The basic iteration cycle of a GA proceeds on a population of individuals, each of which represents a search point in the space of potential solutions of a given optimization problem [2]. The standard GA starts with an initial population of individuals created at random. Then, this population evolves through time by a string manipulation process based in three genetic operators: reproduction, crossover and mutation. The power and success of GA is mostly achieved by the diversity of the individuals of a population. In the classical GA, diversity is maintained through the genetic operators crossover and mutation.

In nature, genetic diversity is caused and maintained by several mechanisms besides crossover and mutation. Some of those mechanisms are inversion, transduction, transformation, conjugation, transposition and translocation [3].

Several authors have already used some biologically inspired mechanisms besides crossover and mutation in evolutionary approaches. For instance, inversion [5], conjugation [4],[11], transduction [7], translocation [1] and transposition [8] were already used as the main genetic operators in the GA.

The goal of this paper is to enlarge the domain of application of the transposition mechanism, using it to solve different types of the 0/1 knapsack problem. The empirical study will focus the GA efficiency solving the 0/1 knapsack problem, first using the classical crossover operators and then, several variants of the transposition mechanism. The obtained results show that, in all the studied situations, transposition allows the GA to reach better solutions.

This paper is organized in the following manner. First, in section 2, we explain how transposition works in nature and we summarize previous work related to the transposition mechanism. In section 3, we introduce the 0/1 knapsack problem. Section 4 describes the characteristics of the experimental environment. In section 5, we present a summary of the obtained results using transposition and crossover. Finally, we present the relevant conclusions of the work.

2 Transposition

2.1 Biological Transposition

Transposition is characterized by the presence of mobile genetic units inside the genome, moving themselves to new locations or duplicating and inserting themselves elsewhere. These mobile units are called transposons [3]. The movement of the transposons (also known as jumping genes) can take place in the same chromosome or to a different one. In order to a transposable element to transpose as a discrete entity, it is necessary for its ends to be recognized. Therefore, transposons within a chromosome are flanked by identical or inverse sequences, some of which are actually part of the transposon.

The point into which the transposon is inserted requires no homology with the point where the transposon was excised. This is in evident contrast to classical recombination, and consequently, transposition is sometimes referred to as illegitimate recombination.

2.2 Computational Transposition

Previous publications studied three different variants of the transposition mechanism, all inspired in the biological process.

The first form of computational transposition was called simple transposition. After the selection of two

parents for mating, the transposon is formed in one of them. The insertion point is searched in the second parent and the same amount of genetic material is exchanged between the two chromosomes [8].
In order to come closer to the biological mechanism, we proposed a new form of transposition: tournament-based transposition. The two selected parents become competitors in a tournament and the transposon will be searched in the winner chromosome. The insertion point will be located in the loser parent. Only this individual will be changed: the transposon is inserted, replacing the same number of bits after the insertion point [9].
In nature, the transposition mechanism can also occur in the same chromosome. Based on this principle, we introduced the asexual form of transposition, where all the process operates in the same individual [10].

3 The 0/1 Knapsack Problem

The well-known single-objective 0/1 knapsack problem (KP) is defined as follows: given a set of items (*n*), each with a weight *W[i]* and a profit *P[i]*, with *i=1,..,n*. The goal is to determine the number of each item to include in the knapsack so that the total weight is less than some given limit (*C*) and the total profit is as large as possible.

4 Empirical Study

In our empirical analysis, we implemented the KP varying some parameters as suggested in [6]. We used several instances for the KP, taking in consideration several aspects such as, the algorithm used for evaluation of the individuals, the number of items, the correlation between the weights and the profits and the capacity of the knapsack. All those aspects will be detailed in the next sections. We will also refer to the parameters of the genetic algorithm used to solve the problem.

4.1 Algorithms Used in the Individual's Evaluation

We used two types of algorithms in the evaluation of the individuals (**x**) of the population: algorithms based on penalty functions (*Ap[i]*) and algorithms based on repair methods (*Ar[i]*), where *i* is the index of a particular algorithm in each class [6].
Using algorithms based on penalty methods, a binary vector of length *n* represents a solution **x** for the problem each element of **x** can be 0 or 1. If **x**[i]=1 then the item *i* was selected for the knapsack. The fitness ***f(x)*** for each binary string is determined as:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \mathbf{x}[i].P[i] - Pen(\mathbf{x}) \qquad (1)$$

The penalty function *Pen(x)* is zero to all feasible solutions and greater than zero otherwise.
There are many possibilities for assigning the penalty value. We consider three cases where the growth of the penalty function is logarithmic, linear and quadratic [6].
When using algorithms based on repair methods, the evaluation of the binary string **x** is determined as:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \mathbf{x}'[i].P[i] \qquad (2)$$

where **x'**[i] is the repaired version of the original chromosome **x**. We used two different repair methods to obtain **x'**:
Ar[1] (random repair): the item to be removed from the knapsack is selected at random.
Ar[2] (greedy repair): all the items in the knapsack are ordered in the decreasing order of their profit to weight ratios. The deleted item is the last one.
Another aspect to consider in the repair-based algorithms is the percentage of repaired chromosomes to be replaced in the original population. Such replacement rate may vary between 0% and 100%. In our experiments, we used two replacement rates 100% (replace all) and 5%. We chose the 5% rate to test the 5% rule proposed by Orvosh et al. [6].

4.2 Parameters Used in the Evaluation Algorithms

For each algorithm previously described, the empirical study analyzed the influence of the variation of several parameters such as the number of items in the knapsack, the relation between weights and profits and the capacity of the knapsack. In the following sections, we will detail how these parameters were changed in the experiments.
Concerning the number of items, we used three different values: n=100, n=250 and n=500.
Since the difficulty of the problem is affected by the correlation between profits and weights, we considered three different ways of randomly generate those vectors:
Uncorrelated: both vectors *W[i]* and *P[i]* are generated at random, using a uniform distribution. W[i]=(uniformly) *rand* ([1..*v*]); P[i]=(uniformly) *rand* ([1..*v*]).
Weakly correlated: vector *W[i]* is created at random, but *P[i]* is created with some correlation with *W[i]*. *W[i]*= (uniformly) *rand* ([1..*v*]); *P[i]=W[i]*+ (uniformly) *rand* ([-*r*..*r*]).
Strongly correlated: W[i]=(uniformly) *rand* ([1..*v*]); *P[i]=W[i]* + *r*.
The values used for the parameters *v* and *r* were *v*=10 and *r*=5 (see [6]).

We used two types of capacity for the knapsack: restrictive knapsack capacity: $C_1 = 2.v$ and average knapsack capacity: $C_2 = 0.5\sum_{i=1}^{n} W[i]$.

4.3 Parameters of the Genetic Algorithm

We implemented the standard GA as described in [2]. The GA used, each time, a different genetic operator: 1-point crossover, 2-point crossover, uniform crossover, simple transposition, tournament-based transposition and asexual transposition. The population consisted in 100 binary strings. We used roulette-wheel as the selection method and an elitism rate of 10%. Mutation and transposition/crossover rates were fixed in 5% and 65%, respectively. The number of generations was 1000. Each experiment was repeated 25 times. When using transposition, the used flanking sequence length was always computed through the appropriated heuristic proposed in previous work [10].

5 The Results

Our empirical study was very extensive, but due to lack of space we will present the results obtained using the penalty algorithm *Ap[1]* and the repair method *Ar[1]*.
To measure the performance of the different algorithms we used the best solution found within the 1000 generations. The results synthesized in the following tables are mean values of the 25 experiments. The best solutions are marked in bold. The genetic operators used in the tables are identified by Cx1, Cx2, CxU, AT, ST and TT for 1-point, 2-point, uniform crossover, asexual, simple and tournament-based transposition, respectively.

5.1 Results Obtained Using the Ap[1] Algorithm

Using the penalty-based algorithms, when the capacity of the knapsack was Restrictive (*C1*), no valid solutions were found by the GA. Using the Average capacity (*C2*), the GA using tournament-transposition achieved better results in all the situations. Concerning the remaining genetic operators, the other forms of transposition were always better than the crossover operators. The only exception was uniform crossover that had similar performances to asexual transposition in some particular cases. The main observation is that the results are influenced by the correlation of the data set. As we increase this correlation, the obtained results become more similar.
Table 1 shows the obtained results using the penalty algorithm *Ap[1]*. Using the penalty algorithms *Ap[2]* and *Ap[3]* the transposition-based GA had similar behavior.

Table 1. Results Using the Penalty Algorithm Ap[1]

Correl.	Items	Capac.	Genetic Operator					
			Cx1	Cx2	CxU	AT	ST	TT
Uncorr	100	C2	478	500	518	518	534	**562**
	250	C2	1159	1258	1237	1356	1276	**1401**
	500	C2	2291	2332	2440	2460	2480	**2606**
Weak	100	C2	584	622	636	668	641	**681**
	250	C2	1481	1524	1539	1549	1555	**1587**
	500	C2	2819	2802	2817	2878	2822	**2887**
Strong	100	C2	1023	1028	1012	1051	1055	**1055**
	250	C2	2471	2491	2485	2523	2573	**2578**
	500	C2	4654	4654	4656	4659	4663	**4770**

5.2 Results Obtained Using Ar[1] Algorithm

With this algorithm, we implemented two strategies (replace-all and replace 5%) but we will present the results achieved using the "replace-all" strategy. We choose to present these results, because they are inferior to those obtained with the replace 5% strategy (for the transposition mechanism). Using the *Ar[1]* algorithms, once again, transposition achieved better solutions. In particular, the tournament-based and simple transposition were the mechanisms that performed better. Table 2 synthesizes the achieved results.

Table 2. Results Using the Repair Algorithm Ar[1] and the "replace-all" Strategy

Correl.	Itens	Capac.	Genetic Operator					
			Cx1	Cx2	CxU	AT	ST	TT
Uncorr	100	C1	516	524	524	524	**525**	524
		C2	341	372	377	385	**395**	**395**
	250	C1	1343	1347	1348	1350	**1353**	1352
		C2	860	931	896	943	**981**	**981**
	500	C1	2511	2518	2515	2539	2540	**2541**
		C2	1860	1873	1869	1882	1880	**1883**
Weak	100	C1	633	633	633	634	634	**635**
		C2	351	358	344	376	**392**	386
	250	C1	1557	1559	1560	1560	1563	**1564**
		C2	930	950	955	967	966	**973**
	500	C1	2878	2885	2884	2776	2892	**2907**
		C2	1899	1907	1879	1917	**1972**	1939
Strong	100	C1	1030	1033	1037	1039	1039	**1041**
		C2	549	548	558	570	573	**578**
	250	C1	2346	2527	2531	2538	2539	**2540**
		C2	1400	1407	1416	1417	1422	**1428**
	500	C1	4438	4835	4839	4860	4846	**4895**
		C2	2829	2843	2809	2843	2857	**2861**

5.3 Using Smaller Population with Transposition

Previous work in the function optimization domain showed that, with 50 individuals, transposition always outperformed the standard crossover operators with 50, 100 and 200 individuals. In order to conclude about the influence of the population size solving the 0/1 KP, we executed some experiments using the three transposition mechanisms with only 50 individuals. These experiments focused the penalty algorithms with 100 items in the knapsack. Once again, transposition with smaller populations allowed the GA to achieve, in general, better solutions than when using crossover. Table 3 reports the obtained results.

Table 3. Results using Transposition with 50 Individuals

Correl.	Penalty alg.	Nº Items	Genetic Operator					
			Cx1	Cx2	CxU	AT	ST	TT
			Pop = 100			Pop = 50		
Uncorr	Ap[1]	100	478	500	518	516	521	**536**
Weak	Ap[1]	100	584	622	636	636	637	**651**
Strong	Ap[1]	100	1023	1028	1012	1030	1029	**1046**
Uncorr	Ap[2]	100	349	358	359	366	360	**408**
Weak	Ap[2]	100	340	343	351	366	360	**378**
Strong	Ap[2]	100	556	557	558	558	**559**	**559**
Uncorr	Ap[3]	100	357	354	354	371	344	**379**
Weak	Ap[3]	100	346	346	347	347	348	**350**
Strong	Ap[3]	100	559	560	559	560	**562**	**562**

6 Conclusions

The goal of this paper was to use a biologically inspired genetic operator called transposition (already tested in the function optimization domain) in a different domain. The select problem was the 0/1 KP, several types of knapsacks were implemented and different genetic operators were used to solve it: the standard crossover operators (1-point, 2-point and uniform) and several variants of the transposition mechanism (asexual, simple and transposition-based).

The obtained results showed that, in all the implemented algorithms, the transposition mechanisms allowed the GA to achieve higher performances. We also reduced the size of the population for 50 individuals when using transposition and, even so, the GA obtained better results than when using crossover with 100 individuals. Those results reinforce our conviction that transposition is a powerful genetic operator alternative to crossover.

Acknowledgements

This work was partially financed by the Portuguese Ministry of Science and Technology under the Program Praxis XXI and by Coimbra Polytechnic.

References

[1] De Falco, I., Iazzetta, A., Tarantino, E., Della Cioppa, A.: On Biologically Inspired Mutations: The Translocation. In Late Breaking Papers at the 2000 Genetic and Evolutionary Computation Conference (GECCO'2000), pp. 70-77, Las Vegas, USA, 8-12 July 2000.

[2] Goldberg, D. E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Publishing Company, Inc, 1989.

[3] Gould, J. L., Keeton, W. T.: Biological Science. W. W. Norton & Company 1996.

[4] Harvey, I.: The Microbial Genetic Algorithm. Submitted as a Letter to Evolutionary Computation. MIT Press, 1996.

[5] Holland, J. H.: Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence. 1st MIT Press edition, MIT Press 1992.

[6] Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. 3rd Edition Springer-Verlag 1999.

[7] Nawa, N., Furuhashi, T., Hashiyama, T., Uchikawa, Y.: A Study of the Discovery of Relevant Fuzzy Rules Using Pseudo-Bacterial Genetic Algorithm. IEEE Transactions on Industrial Electronics, 1999.

[8] Simões, A., Costa, E.: Transposition: A Biologically Inspired Mechanism to Use with Genetic Algorithms. In the Proceedings of the Fourth International Conference on Neural Networks and Genetic Algorithms (ICANNGA'99), pp. 612-619. Springer-Verlag 1999.

[9] Simões, A., Costa, E.: Transposition versus Crossover: An Empirical Study. Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., and Smith, R. E. (eds.), Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'99), pp. 612-619, Orlando, Florida USA, CA: Morgan Kaufmann 1999.

[10] Simões, A., Costa, E.: Using Genetic Algorithms with Asexual Transposition. In D. Whitley, D. Goldberg, E. Cantú-Paz, L. Spector, I. Parmee, H. Beyer. (eds.), Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2000), pp. 323-330, Las Vegas, USA, CA: Morgan Kaufmann 2000.

[11] Smith, P.: Conjugation - A Bacterially Inspired Form of Genetic Recombination. In Late Breaking Papers of the First International Conference on Genetic Programming. Stanford University, California, 1996.