Combinatorial Optimization by Gene Expression Programming: Inversion Revisited

Cândida Ferreira

Gepsoft, 37 The Ridings,
Bristol BS13 8NU, UK
candidaf@gepsoft.com
www.gene-expression-programming.com/author.asp

In J. M. Santos and A. Zapico, eds., *Proceedings of the Argentine Symposium on Artificial Intelligence*, pages 160-174, Santa Fe, Argentina, 2002.

Combinatorial optimization problems require combinatorial-specific search operators so that populations of candidate solutions can evolve efficiently. Indeed, several researchers created modifications to the basic genetic operators of mutation and recombination in order to create high performing combinatorial-specific operators. However, it is not known which operators perform better as no systematic comparisons have been done. In this work, a new algorithm that explores a new chromosomal organization based on multigene families is used. This new organization together with several combinatorial-specific search operators, namely, inversion, gene and sequence deletion/insertion, and restricted and generalized permutation, allow the algorithm to perform with high efficiency. The performance of the new algorithm is empirically compared on the 13- and 19-cities tour traveling salesperson problem, showing that the long abandoned inversion operator is by far the most efficient of the combinatorial operators. The efficiency and potentialities of the new algorithm are further demonstrated by solving a simple task assignment problem.

1. Introduction

Gene expression programming (GEP) is a multigenic genotype/phenotype system encoding expression trees linked by a particular linking interaction (Ferreira 2001). In its simplest representation (head length $h = 0$ and maximum arity $n = 0$), GEP is equivalent to the canonical genetic algorithm (GA), in which each gene consists of only one terminal. Such simple chromosomal organization was used to solve the 11-multiplexer problem, where the one-element expression trees encoded in each gene were posttranslationally linked by the Boolean function if (*x*,*y*,*z*) (Ferreira 2001). To solve combinatorial problems, however, another kind of linking interaction is required. For instance, in the traveling salesperson problem (TSP) the linking consists obviously of the distance between the cities represented by two adjacent genes.

The TSP represents a classic optimization problem and good, traditional approximation algorithms have been developed to tackle it down (see, e.g., Papadimitriou and Steiglitz 1982 for a review). However, to evolutionary computists, the TSP serves as the simplest case of a variety of combinatorial problems which are of enormous relevance to industrial scheduling problems (Bonachea *et al*. 2000; Hsu and Hsu 2001; Johnson and McGeoch 1997; Katayama and Narihisa 1999; Merz and Freisleben 1997; Reinelt 1994). Indeed, several evolution inspired algorithms used the TSP as a battleground to develop combinatorial-specific search operators such as: *alternating edge crossover* (Grefenstette *et al*. 1985), *subtour chunks crossover* (Grefenstette *et al*. 1985), *heuristic crossover* for adjacency representation (Grefenstette *et al*. 1985; Jog *et al*. 1989; Lin 1965; Suh and van Gucht 1987), *partially-mapped crossover* (Goldberg and Lingle 1985), *cycle crossover* (Oliver *et al*. 1987), *order crossover* (Davis 1985; Oliver *et al*. 1987), *order* and *position based crossover* (Syswerda 1991), *heuristic crossover* for path representation (Grefenstette 1987; Liepins *et al*. 1987), *genetic edge recombination crossover* (Whitley *et al*. 1989, 1991), *maximal preservative crossover* (Mühlenbein *et al*. 1988), *voting recombination crossover* (Mühlenbein 1989), *displacement mutation* (Herdy 1991; Michalewicz 1992), *exchange mutation* (Ambati *et al*. 1991; Banzhaf 1990; Michalewicz 1992; Oliver *et al*. 1987; Syswerda 1991), *repeated exchange mutation* (Ambati *et al*. 1991; Beyer 1992), *insertion mutation* (Fogel 1988; Michalewicz 1992; Syswerda 1991), *simple inversion mutation* (Grefenstette 1987; Holland 1975), *inversion mutation* (Fogel 1990, 1993), and *scramble mutation* (Ulder *et al*. 1990). Note that, in some cases, operators are not named exactly as in the original work, as this nomenclature was recently proposed by Larrañaga *et al*. (1998) in an attempt to classify the overwhelming number of combinatorial search operators. More recently, other operators such as *edge assembly crossover* (Nagata and Kobayashi 1997) and *inver-over operator* (Tao and Michalewicz 1998) were developed.

Despite or due to the huge number of combinatorial-specific operators, little work has been done on the compara-

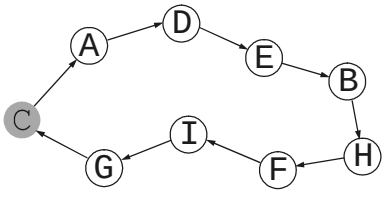
tive performance of the different operators, although different forms of crossover have been compared (Grefenstette *et al*. 1985; Oliver *et al*. 1987; Whitley *et al*. 1989, 1991). In this work, the performance of simple inversion mutation is compared with insertion mutation and exchange mutation on the difficult 19-cities tour TSP, with simple inversion mutation astoundingly outperforming insertion and exchange mutation. Furthermore, poorly performing operators such as displacement mutation or repeated exchange mutation, could only be compared using an easier tour of only 13 cities. Thus, insertion and displacement mutation, two closely related operators in terms of implementation, are compared on the 13-cities tour TSP. Exchange mutation and repeated exchange mutation are also compared using the shorter tour. Moreover, the potentialities of inversion are further demonstrated by solving a simple task assignment problem where a new chromosomal organization based on multigene families is used.

2. Multigene families and scheduling problems

As stated previously, the chromosomal organization used to solve combinatorial problems is very simple and consists of multigenic chromosomes composed of one-element genes, where each gene codes for a one-element expression tree (ET) (Ferreira 2001). Furthermore, one-element genes may be organized in multigene families (MGFs), in which a particular class of terminals or tasks is gathered. Such chromosomes composed of MGFs are very useful to evolve solutions to combinatorial problems, as different classes of terminals/items can be included in each MGF. For instance, the different cities in the traveling salesperson problem may be encoded in a multigene family, where each gene codes for a city. Consider the simple chromosome below, composed of one MGF with nine members:

```
CADEBHFIG
```

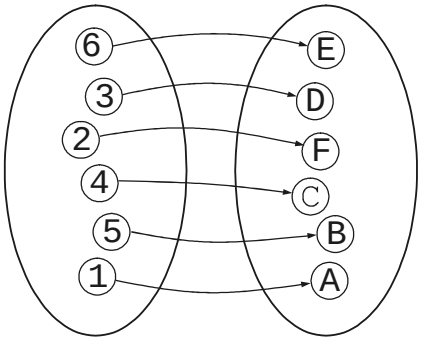
where each element represents a city. In this case, the expression of this chromosome consists of the spatial organization of the one-element ETs, for instance, the following tour for the traveling salesperson problem (the starting and finishing point is in gray):



For optimization problems with *N* classes of terminals, multigenic chromosomes composed of *N* multigene families are used. For instance, the chromosome below composed of two MGFs with six members, was designed to evolve solutions to the simple task assignment problem of section 4.2 (multigene families have different shades):

```
012345012345
632451EDFCBA
```

Its expression consists of the orderly interaction of the members of each MGF with one another as shown below:



3. Combinatorial-specific operators: Performance and mechanisms

Combinatorial problems can not be solved using the genetic operators of mutation and recombination of the basic gene expression algorithm as these operators would generate useless individuals containing MGFs with repeated elements on the one hand, and missing certain elements on the other. Indeed, in combinatorial problems, the elements of a multigene family must all be present and cannot be represented more than once. Therefore, special search operators must be created so that genetic variation could be introduced without creating invalid structures.

In this section, five genetic operators are described: inversion, gene and sequence deletion/insertion, and restricted and generalized permutation. These combinatorial-specific operators allow the introduction of genetic variation without disrupting both the structure and balance of multigene families. Note that these operators have been used by other researchers, but I took the liberty to change their names whenever the name previously given could cause confusion or does not reflect the GEP context of genes and MGFs. However, the correct references to the original names are given below.

Before proceeding with the description of their mechanisms, it is useful to compare their performances (Figure 1). The problem chosen to make this analysis is the TSP of section 4.1 with 19 cities using population sizes *P* of 100 and evolutionary times *G* of 200 generations. The 19 cities were arranged in a rectangle so that the shortest tour is 20. Therefore, the performance can be rigorously determined in terms of success rate, which is evaluated over 100 identical runs. As Figure 1 clearly demonstrates, the best operator is by far inversion, followed by gene deletion/insertion, whereas restricted permutation is extremely limited.

3.1. Inversion

The inversion operator, in its mechanism, corresponds basically to the inversion operator firstly described by Holland

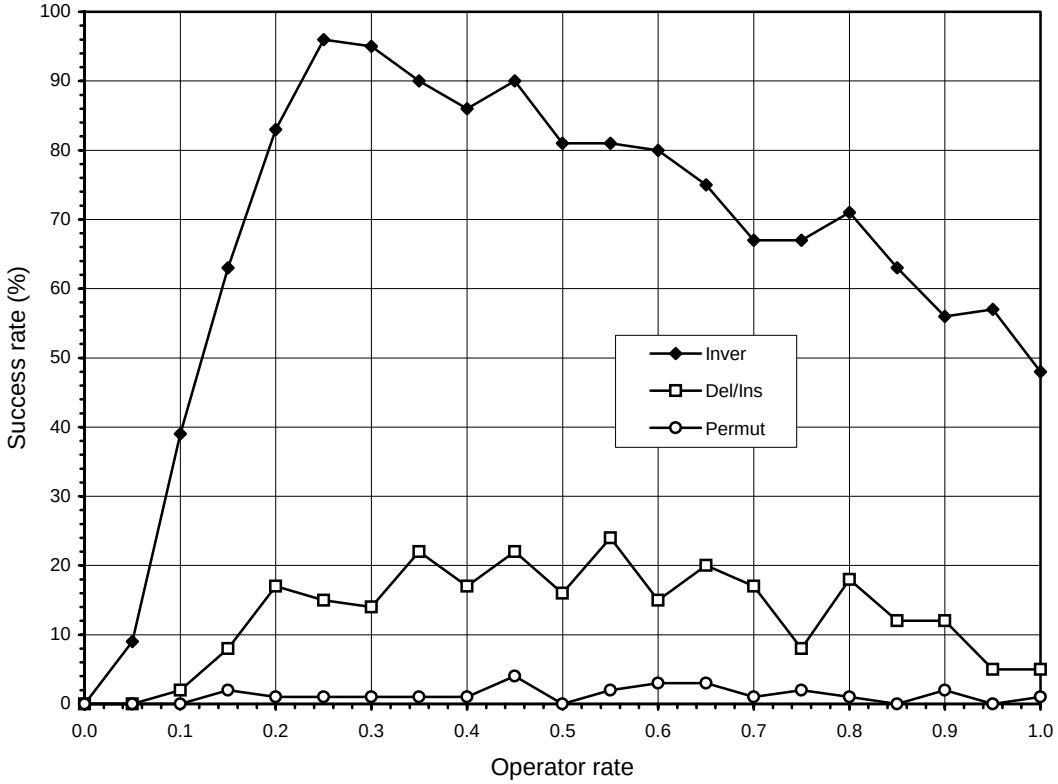


Figure 1. Comparison of inversion (Inver), gene deletion/insertion (Del/Ins), and restricted permutation (Permut) on the traveling salesperson problem with 19 cities. For this analysis $P = 100$ and $G = 200$. The success rate was evaluated over 100 identical runs.

(1975). In the classification proposed by Larrañaga *et al*. (1998) it received the more complicated designation of "simple inversion mutation". In the MGFs context of GEP the inversion operator works as follows: it randomly selects the chromosome, the multigene family to be modified, the inversion points in the MGF, and then inverts the sequence between these points. Each chromosome can only be modified once by this operator.

Consider the chromosome below composed of two multigene families, each containing 13 members:

```
012345678901201234567890 12
bhfgkicmjlaedMD CLEBF GKJHIA        (3.1)
```

Suppose genes 2 and 6 in MGF_2 were chosen as inversion points. Then the sequence between these points is reversed, obtaining:

```
012345678901201234567890 12
bhfgkicmjlaedMD FBELC GKJHIA        (3.2)
```

Note that, with inversion, the whole multigene family can be inverted if the first and last gene were chosen as inversion points. For instance, the inversion of MGF_2 in chromosome (3.1) gives:

```
012345678901201234567890 12
bhfgkicmjlaedAIHJKGFBELCDM          (3.3)
```

Note also that this operator allows small adjustments like, for instance, the permutation of two adjacent genes. For instance, if genes 7 and 8 in FMG_1 of chromosome (3.3) were chosen as inversion points, these genes are permuted, obtaining:

```
012345678901201234567890 12
bhfgkicjmlaedAIHJKGFBELCDM          (3.4)
```

As Figure 1 emphasizes, inversion is the most powerful of the combinatorial-specific genetic operators, causing populations to evolve with great efficiency when used as the only source of genetic diversification. Indeed, this operator alone produces better results than when combined with gene deletion/insertion or permutation.

The high performance obtained by GEP inversion is surprising and deserves a careful inspection. Perhaps for historical reasons, inversion was abandoned by researchers in the early development of GAs (see, e.g., Goldberg 1989 and Mitchell 1996). Decisive for this outcome was, most certainly, Bagley's (1967) disappointment with inversion while trying to conciliate inversion with homologous recombination (see Goldberg 1989 for a detailed narrative). Obviously, inversion disrupts homology and homologous recombination ceases to work. Unfortunately, Bagley persisted with recombination and did not try inversion alone.

Furthermore, the astounding results obtained for inversion are better appreciated if we compare them with attempts

to solve the 19-cities tour TSP by GAs (Haupt and Haupt 1998). These researchers could not find the shortest route using population sizes of 800 for 200 generations. As shown in Figure 1, GEP not only finds the shortest route using inversion but also using gene deletion/insertion or restricted permutation using population sizes eight times smaller than the ones used by the GA. Moreover, if inversion alone is doing the search, GEP finds the shortest route in 96% of the runs.

3.2. Gene deletion/insertion

The gene deletion/insertion operator is the second in importance of the analyzed combinatorial-specific operators (see Figure 1 above). This operator corresponds to the "insertion mutation" operator in the classification proposed by Larrañaga *et al*. (1998). Here, the designation "gene deletion/insertion" was chosen for three reasons: 1) to reflect the fact that the inserted element is previously deleted; 2) to emphasize that only one gene is deleted/inserted at a time; and 3) to distinguish this operator from the closely related operator "sequence deletion/insertion" described below (section 3.4).

The gene deletion/insertion operator randomly selects the chromosome, the multigene family to be modified, the gene to transpose, and the insertion site. Each chromosome can only be modified once by this operator.

Consider the chromosome below composed of two multigene families, each with 13 members:

```
012345678901201234567890 12
ifg habdecjklmKLHCIGDFEJMBA          (3.5)
```

Suppose gene 3 in MGF_1 was selected to transpose to site 7 (between genes 6 and 7). Then gene 3 is deleted in the place of origin and inserted between genes "d" and "e", obtaining:

```
012345678901201234567890 12
ifgabd hecjklmKLHCIGDFEJMBA          (3.6)
```

The deletion/insertion of genes when combined with more powerful operators such as inversion, might be useful to make finer adjustments. However, for all the problems analyzed in this work, the performance was higher if inversion worked alone.

3.3. Restricted permutation

The restricted permutation operator appears as the "exchange mutation" operator in Larrañaga *et al*. (1998). It allows two genes occupying any positions within a particular multigene family to trade places. This operator might also be useful to make finer adjustments when combined with inversion, but it performs poorly if used as the only source of genetic variation (see Figure 1 above).

The restricted permutation operator randomly chooses the chromosome, the multigene family to be modified and the genes to be exchanged. Each chromosome is only modified once by this operator.

Consider another chromosome composed of two multigene families, each with 13 members:

```
012345678901201234567890 12
ikmfghdeljcabLJIHGCDBKMFAE           (3.7)
```

Suppose genes 5 and 9 in FMG_2 were chosen to be exchanged. Then the following chromosome is formed:

```
012345678901201234567890 12
ikmfghdeljcabLJIHGMDBKCFAE           (3.8)
```

Restricted permutation, when used at small rates and in combination with inversion, might be useful to make finer adjustments. However, for the problems analyzed in this work, when permutation is used in conjunction with inversion the success rate slightly decreases.

3.4. Other search operators

The gene deletion/insertion operator introduced in section 3.2 permits only the deletion/insertion of genes, i.e., the smallest sequence composed of only one element. Another operator can be easily implemented that deletes/inserts sequences of varied length. This operator was named "sequence deletion/insertion", and corresponds to the "displacement mutation" operator in the classification proposed by Larrañaga *et al*. (1998). The deletion/insertion of sequences of different lengths might appear more advantageous than the deletion/insertion of genes, but experience shows the opposite (see Figure 2 below). In fact, this operator produces results which are even worse than the restricted permutation operator in the traveling salesperson problem with 19 cities (compare with Figure 1 above). In fact, an identical analysis done with this operator showed that the sequence deletion/insertion is incapable of solving the 19 cities TSP using population sizes of 100 individuals for 200 generations. Thus, an easier version of the TSP with 13 cities (with the cities also placed in a rectangle so that the shortest tour is 14) was chosen in order to make comparisons between gene deletion/insertion and sequence deletion/insertion (Figure 2). For this analysis, a population size of 100 individuals and an evolutionary time of 200 generations were used and the success rate was also evaluated over 100 identical runs.

The other operator to be analyzed here is an extension to the restricted permutation operator of section 3.3. Recall that that kind of permutation operator exchanges only a pair of genes per chromosome, i.e., the restricted permutation rate p_{rp} is evaluated by $p_{rp} = N_C/P$, where N_C represents the number of chromosomes modified. A more generalized version of this operator can be easily implemented where a different number of genes in a chromosome can trade places with other genes according to a certain rate. More formally, the generalized permutation rate p_{gp} is evaluated by $p_{gp} = N_G/(C_L \cdot P)$, where N_G represents the number of genes modified and C_L

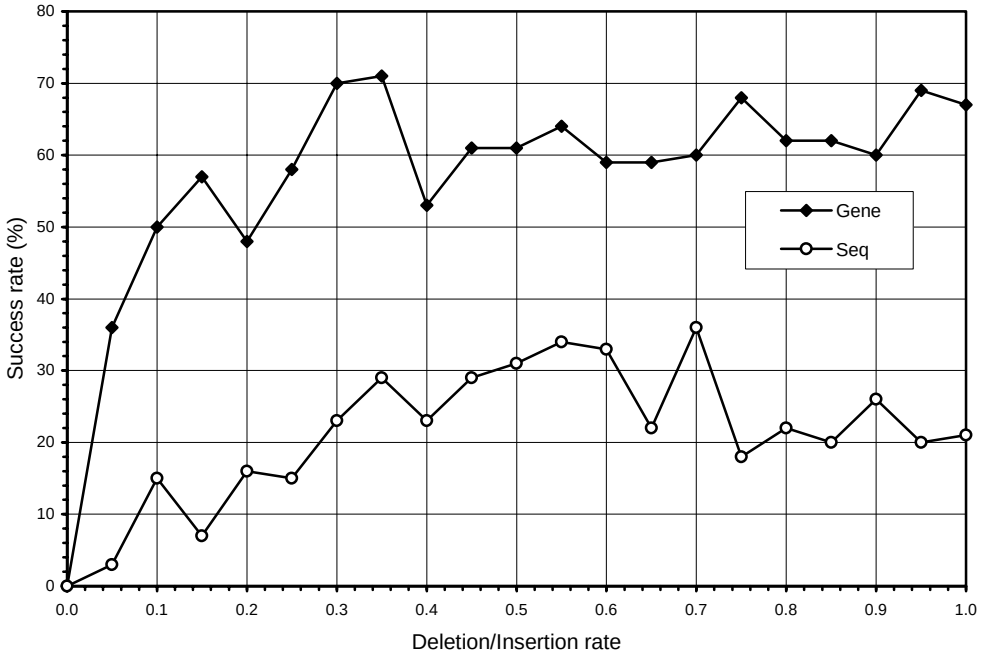


Figure 2. Comparison of gene deletion/insertion (Gene) with sequence deletion/insertion (Seq) on the traveling salesperson problem with 13 cities. For this analysis $P = 100$ and $G = 200$. The success rate was evaluated over 100 identical runs.

the chromosome length. This operator was named "generalized permutation" and corresponds to the "exchange repeated mutation" operator in Larrañaga *et al*. (1998). Again, a more generalized permutation might appear more efficient than the restricted permutation described above, but experience shows that restricted permutation is slightly better (see Figure 3 below). For instance, in the TSP with 19 cities (see Figure 1 above), this operator performed worse than restricted permutation and, in fact, was incapable of finding a perfect solution. The results obtained for the simpler version of the TSP with 13 cities are shown in Figure 3. In this analysis the restricted and generalized permutation are compared using populations of 100 individuals evolving for 200 generations, i.e., exactly the same values of P and G used to solve the much more complex TSP with 19 cities of Figure 1.

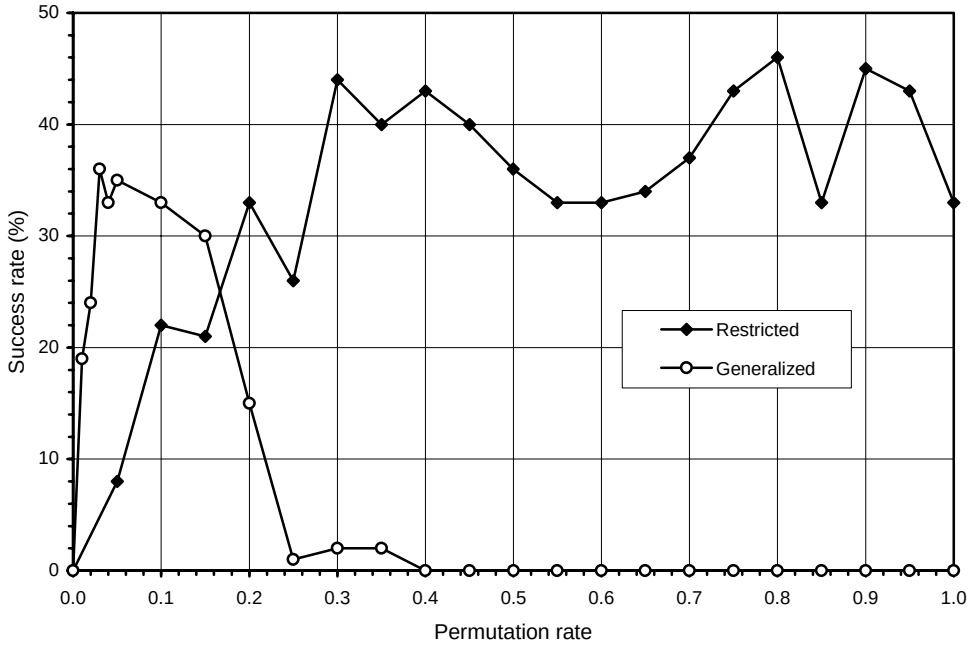


Figure 3. Comparison of restricted permutation with generalized permutation on the traveling salesperson problem with 13 cities. For this analysis $P = 100$ and $G = 200$. The success rate was evaluated over 100 identical runs.

4. Using inversion to solve scheduling problems

The first problem of this section is the already mentioned TSP with 19 cities. And it has already been shown that this problem requires only a multigene family consisting of the genes representing the 19 cities the salesperson should visit.

The second problem is a task assignment problem and requires two different multigene families, one containing the agents and the other the tasks assigned to the agents.

4.1. The traveling salesperson problem

For the TSP with 19 cities there are 19! = $1.21645.10^{17}$ combinations to search. If the starting point is fixed (and consequently the ending point), the number of possible combinations is halved to $6.0823.10^{16}$. Furthermore, choosing a configuration where all the cities lie in a rectangle so that the shortest tour is 20, allows the rigorous evaluation of the performance of the algorithm in terms of success rate.

Obviously, the tour length cannot be used directly as a measure of fitness as the shorter the tour the fitter the individual. Thus, each generation, the fitness f_i of an individual program i in generation g is evaluated by the formula:

$$f_i = T_g - t_i + 1 \qquad (4.1)$$

where t_i is the length of the tour encoded in i, and T_g is the length of the largest tour encoded in the chromosomes of the current population. This way, the fitness of the worst individual of the population is always 1. As usual in GEP, individuals are selected according to fitness by roulette-wheel selection and each generation the best individual is cloned unchanged into the next generation (simple elitism). The parameters used per run are summarized in the first column of Table 1.

The results obtained by GEP are astounding if we compare them with the performance of GAs to solve the 19-cities tour TSP. As a comparison, Haupt and Haupt (1998) could not solve this problem using population sizes of 800 for 200 generations. As shown in Figure 1 above and in the first column of Table 1, GEP not only is capable of solving this problem using populations of only 100 individuals and for the same 200 generations, but also is capable of finding the shortest route in practically all runs (in 96% of the runs, in fact).

It is worth emphasizing that only inversion was used to create genetic variation. And, indeed, the presence of other genetic operators, namely gene deletion/insertion and restricted permutation, decreases slightly the success rate and therefore were not used. It seems that these operators are unnecessary for finer adjustments whenever inversion is doing the search.

Figure 4 shows the progression of average and best tour for a successful run of the experiment summarized in the first column of Table 1. Note that the evolutionary dynamics for combinatorial problems is similar to the dynamics characteristic of GAs (see, e.g., Goldberg 1989). In these dynamics the plot for average fitness closely accompanies the plot for best fitness and the oscillatory pattern on average fitness is less pronounced than in GEP dynamics (Ferreira 2002). The dynamics obtained here support the idea that simple replicator systems are fundamentally different from genotype/phenotype systems where a complex expression already exists. Indeed, an extremely simple expression takes place to express fully the chromosomes used to solve the TSP: in fact, the chromosome itself is the phenotype or the solution.

Table 1
Parameters for the traveling salesperson problem with 19 cities (TSP) and for the task assignment problem (TAP).

	TSP	TAP
Number of runs	100	100
Number of generations	200	50
Population size	100	30
Number of multigene families	1	2
Number of genes per multigene family	19	6
Chromosome length	19	12
Inversion rate	0.25	0.30
Success rate	96%	69%

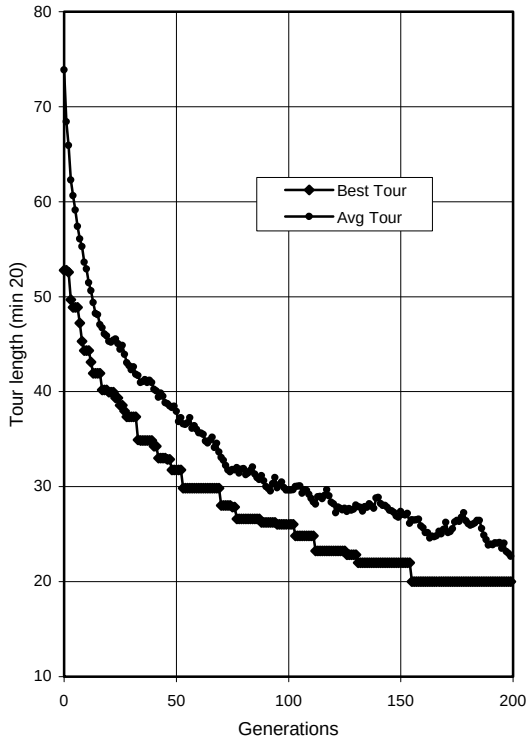


Figure 4. Progression of average tour of the population and the best tour for a successful run of the experiment summarized in the first column of Table 1 (TSP with 19 cities).

4.2. The task assignment problem

The task assignment problem (TAP) of this section is the toy problem chosen by Tank and Hopfield (1987) in their *Scientific American* article to illustrate the workings of Hopfield networks on combinatorial cost-optimization problems.

In TAP there are *n* tasks that must be accomplished by using only *n* workers. Each worker performs better at some tasks and worse at others and obviously some workers are better than others at certain tasks. The goal is to minimize the total cost for accomplishing all tasks or, stated differently, to maximize the overall output of all the workers as a whole.

Suppose we had to shelve *n* book collections in a library using *n* shelving assistants. Each assistant is familiar with the subject areas to varying degrees and shelves the collections accordingly. The data or fitness cases in the task assignment problem consist of the rates at which books are shelved per minute (Figure 5).

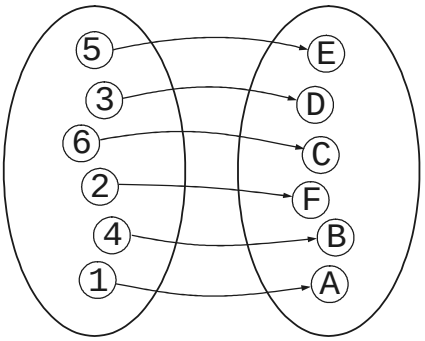
For this simple six-by-six problem there are already 6! = 720 possible assignments of assistants to book collections. The best solution has the highest sum of rates for the chosen assistants. For the particular set of fitness cases (see Figure 5), the best possible solution is known and corresponds to $f_{max} = 44$.

This kind of toy problem is very useful for comparing the performance of different algorithms and, here, the potentialities of inversion are further tested in the context of chromosomes composed of more than one multigene family. Indeed, the task assignment problem is solved very efficiently using only inversion as the source of genetic variation and two MGFs: one to represent the assistants (represented by 1-6) and another to represent the book collections (represented by A-F). The parameters used per run and the success rate for this problem are shown in the second column of Table 1. Again, selection was made by roulette-wheel sampling coupled with elitism. Note that this problem was efficiently solved using extremely small populations of only 30 individuals evolving for a short period of 50 generations.

Figure 6 shows the progression of average and best fitness of a successful run. By generation 16 an individual with maximum fitness was found:

```
012345012345
536241EDCFBA
```

which corresponds to the best assignment of 44 (see also Figure 5 above):



It is worth noticing that the evolutionary dynamics shown in Figure 6 is no longer of the type expected for a GA. In fact, it has all the characteristics of GEP dynamics with its oscillatory pattern in average fitness and a considerable gap between best and average fitness (Ferreira 2002). This kind of dynamics is to be expected due to the higher complexity required to express the chromosomes composed of two multigene families encoding not only the MGFs' members but also specific interactions between them.

	A	B	C	D	E	F
1	10	6	1	5	3	7
2	5	4	8	3	2	6
3	4	9	3	7	5	4
4	6	7	6	2	6	1
5	5	3	4	1	8	3
6	1	2	6	4	7	2

Figure 5. The task assignment problem. Each assistant (1-6) should be assigned to one collection of books (A-F) based on the rates at which books are shelved per minute (fitness cases). Shaded squares show the best assignment with the largest sum of shelving rates, 44.

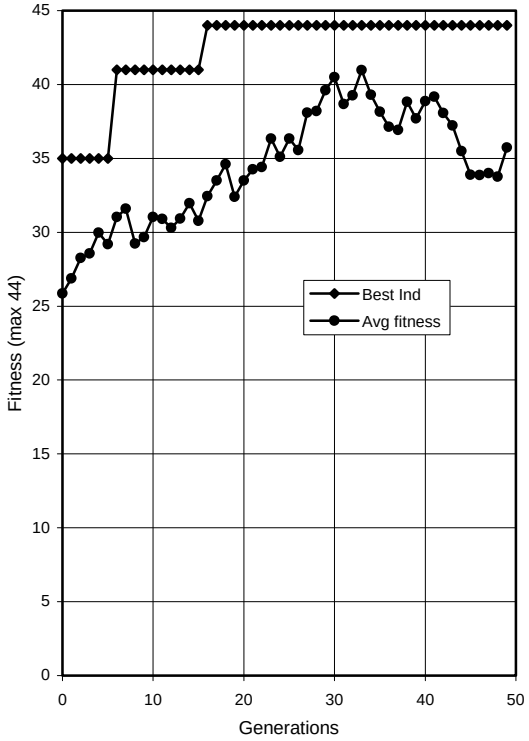


Figure 6. Progression of average fitness of the population and the fitness of the best individual for a successful run of the experiment summarized in Table 1, column 2 (TAP).

5. Conclusions

In this work, a new chromosomal organization consisting of multigene families was described. Multigene families contain members of a particular class of terminals and are, therefore, very useful for solving scheduling problems.

Furthermore, special combinatorial search operators were created so that both the chromosomal organization and the composition of multigene families could be fully exploited to solve combinatorial problems. Indeed, all the genetic modifications made by these combinatorial-specific operators always result in valid chromosomes, i.e., chromosomes in which both the structure of multigene families and the balance of its members are maintained. This is obviously a prerequisite for a good genetic operator. However, this is no guarantee that this kind of operator will be highly effective and, in fact, some perform better than others. The results presented here show that inversion astoundingly surpasses other combinatorial-specific operators such as the moderately performing gene deletion/insertion and restricted permutation, and the poorly performing sequence deletion/insertion and generalized permutation. And, no doubt, surpasses all the modified forms of crossover especially tailored to deal with combinatorial problems.

In addition, it was also shown that solutions to scheduling problems are better found if the search is exclusively done by inversion. Indeed, mixing inversion with other operators, albeit combinatorial-specific, results in a decrease in performance.

The performance of inversion was further evaluated on two scheduling problems: the difficult TSP with 19 cities that required only one multigene family and the task assignment problem that required two multigene families. In both cases, the new algorithm performed with high efficiency, solving, with less resources and with almost maximum success rate (96%), a problem the GA could not solve (TSP with 19 cities).

Bibliography

Ambati, B. K., J. Ambati, and M. M. Mokhtar, 1991. Heuristic Combinatorial Optimization by Simulated Darwinian Evolution: A Polynomial Time Algorithm for the Traveling Salesman Problem. *Biological Cybernetics*, 65: 31-35.

Bagley, J. D., *The Behavior of Adaptive Systems which Employ Genetic and Correlation Algorithms*, Doctoral Dissertation, University of Michigan, 1967.

Banzhaf, W., 1990. The "Molecular" Traveling Salesman Problem. *Biological Cybernetics*, 64: 7-14.

Beyer, H.-G., Some Aspects of the "Evolution Strategy" for Solving TSP-Like Optimization Problems Appearing at the Design Studies of the 0.5 $TeVe^{+}e^{-}$-Linear Collider. In R. Männer and B. Manderick, eds., *Parallel Problem Solving From Nature II*, 461-470, Amsterdam, North-Holland, 1992.

Bonachea, D., E. Ingerman, J. Levy, and S. McPeak, An Improved Adaptive Multi-Start Approach to Finding Near-Optimal Solutions to the Euclidean TSP. In D. Whitley, D. Goldberg, E. Cantu-Paz, L. Spector, I. Parmee, and H.-G. Beyer, eds., *Proceedings of the Genetic and Evolutionary Computation Conference*, 143-150, Las Vegas, Nevada, Morgan Kaufmann, 2000.

Davis, L., Applying Adaptive Algorithms to Epistatic Domains. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 162-164, 1985.

Ferreira, C., 2001. Gene Expression Programming: A New Adaptive Algorithm for Solving Problems. *Complex Systems*, 13 (2): 87-129.

Ferreira, C., Mutation, Transposition, and Recombination: An Analysis of the Evolutionary Dynamics. In H. J. Caulfield, S.-H. Chen, H.-D. Cheng, R. Duro, V. Honavar, E. E. Kerre, M. Lu, M. G. Romay, T. K. Shih, D. Ventura, P. P. Wang, Y. Yang, eds., *Proceedings of the 6th Joint Conference on Information Sciences, 4th International Workshop on Frontiers in Evolutionary Algorithms*, pages 614-617, Research Triangle Park, North Carolina, USA, 2002.

Fogel, D. B., 1988. An Evolutionary Approach to the Traveling Salesman Problem. *Biological Cybernetics*, 60: 139-144.

Fogel, D. B., A Parallel Processing Approach to a Multiple Traveling Salesman Problem Using Evolutionary Programming. In L. Canter, ed., *Proceedings of the Fourth Annual Parallel Processing Symposium*, 318-326, Fullerton, CA, 1990.

Fogel, D. B., 1993. Applying Evolutionary Programming to Selected Traveling Salesman Problems. *Cybernetics and Systems*, 24: 27-36.

Goldberg, D. E. and R. Lingle, Alleles, Loci and the Traveling Salesman Problem. In J. J. Grefenstette, ed., *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, Lawrence Erlbaum, 154-159, 1985.

Goldberg, D. E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.

Grefenstette, J. J., R. Gopal, B. Rosmaita, and D. van Gucht, Genetic Algorithms for the Traveling Salesman Problem. In J. J. Grefenstette, ed., *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, 160-165, Lawrence Erlbaum, 1985.

Grefenstette, J. J., Incorporating Problem Specific Knowledge into Genetic Algorithms. In L. Davis, ed., *Genetic Algorithms and Simulated Annealing*, 42-60, Los Altos, CA, Morgan Kaufmann, 1987.

Haupt, R. L. and S. E. Haupt, *Practical Genetic Algorithms*. Wiley-Interscience, 1998.

Herdy, M., Application of the Evolutionstrategie to Discrete Optimization Problems. In H.-P. Schwefel and R. Männer, eds., *Parallel Problem Solving from Nature*, *Lecture Notes in Computer Science*, 496: 188-192, Springer-Verlag, 1991.

Holland, J. H., *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, University of Michigan Press, 1975 (second edition: MIT Press, 1992).

Hsu, W. W. and C.-C. Hsu, The Spontaneous Evolution Genetic Algorithm for Solving the Traveling Salesman Problem. In L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, eds., *Proceedings of the Genetic and Evolutionary Computation Conference*, 359-366, San Francisco, California, Morgan Kaufmann, 2001.

Jog, P., J. Y. Suh, and D. van Gucht, The Effects of Population Size, Heuristic Crossover and Local Improvement on a Genetic Algorithm for the Traveling Salesman Problem. In J. Schaffer, ed., *Proceedings of the Third International Conference on Genetic Algorithms*, 110-115, Los Altos, CA, Morgan Kaufmann, 1989.

Johnson, D. S. and L. A. McGeoch, The Traveling Salesman Problem: A Case Study. In E. H. L. Aarts and J. K. Lenstra, eds., *Local Search in Combinatorial Optimization*, 215-310, Wiley & Sons, New York, 1997.

Katayama, K. and H. Narihisa, Iterated Local Search Approach Using Gene Transformation to the Traveling Salesman Problem. In W. Banzhaf, ed., *Proceedings of the Genetic and Evolutionary Computation Conference*, 321-328, Morgan Kaufmann, 1999.

Larrañaga, P., C. M. H. Kuijpers, and R. H. Murga, Tackling the Traveling Salesman Problem: Representations and Operators, *Technical Report*, 1998.

Liepins, G. E., M. R. Hilliard, M. Palmer, and M. Morrow, Greedy Genetics. In J. J. Grefenstette, ed., *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference*, 90-99, Lawrence Erlbaum, 1987.

Lin, S., 1965. Computer Solutions on the Traveling Salesman Problem. *Bell Systems Techn. J.*, 44: 2245-2269.

Merz, P. and B. Freisleben, Genetic Local Search for the TSP: New Results. In T. Bäck, Z. Michalewicz, and X. Yao, eds., *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation*, 159-164, Piscataway, NJ, IEEE Press, 1997.

Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, Springer Verlag, Berlin, 1992 (3rd edition: 1996).

Mitchell, M., *An Introduction to Genetic Algorithms*, MIT Press, 1996.

Mühlenbein, H., M. Gorges-Schleuter, and O. Krämer, 1988. Evolution Algorithms in Combinatorial Optimization. *Parallel Computing*, 7: 65-85.

Mühlenbein, H., Parallel Genetic Algorithms, Population Genetics and Combinatorial Optimization. In J. Schaffer, ed., *Proceedings of the Third International Conference on Genetic Algorithms*, 416-421, Los Altos, CA, Morgan Kaufmann, 1989.

Nagata, Y. and S. Kobayashi, Edge Assembly Crossover: A High-Power Genetic Algorithm for the Traveling Salesman Problem. In T. Bäck, ed., *Proceedings of the 7th International Conference on Genetic Algorithms*, 450-457, Morgan Kaufmann, 1997.

Oliver, I. M., D. J. Smith, and J. R. C. Holland, A Study of Permutation Crossover Operators on the Traveling Salesman Problem. In J. J. Grefenstette, ed., *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference*, 224-230, Lawrence Erlbaum, 1987.

Papadimitriou, C. H. and K. Steiglitz, *Combinatorial Optimization*, Prentice Hall, 1982.

Reinelt, G., The Traveling Salesman: Computational Solutions for TSP Applications. *Lecture Notes in Computer Science*, 496: 188-192, Springer-Verlag, Berlin, Germany, 1994.

Suh, J. Y. and D. van Gucht, Incorporating Heuristic Information into Genetic Search. In J. J. Grefenstette, ed., *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference*, 100-107, Lawrence Erlbaum, 1987.

Syswerda, G., Schedule Optimization Using Genetic Algorithms. In L. Davis, ed., *Handbook of Genetic Algorithms*, 332-349, Van Nostrand Reinhold, New York, 1991.

Tank, D. W. and J. J. Hopfield, 1987. Collective Computation in Neuronlike Circuits. *Scientific American*, 257 (6): 104-114.

Tao, G. and Z. Michalewicz, Inver-Over Operator for the TSP. In A.-E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, eds., *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature, Lecture Notes in Computer Science*, 1498: 803-812, Springer Verlag, 1998.

Ulder, N. L. J., E. H. L. Aarts, H.-J. Bandelt, P. J. M. van Laarhoven, and E. Pesch, Genetic Local Search Algorithms for the Traveling Salesman Problem. In *Parallel Problem Solving from Nature*, 106-116, Springer Verlag, 1990.

Whitley, D., T. Starkweather, and D. Fuquay, Scheduling Problems and Traveling Salesman: The Genetic Edge Recombination Operator. In J. Schaffer, ed., *Proceedings of the Third International Conference on Genetic Algorithms*, 133-140, Los Altos, CA, Morgan Kaufmann, 1989.

Whitley, D., T. Starkweather, and D. Shaner, The Traveling Salesman and Sequence Scheduling: Quality Solutions Using Genetic Edge Recombination. In L. Davis, ed., *Handbook of Genetic Algorithms*, 350-372, Van Nostrand Reinhold, New York, 1991.