Optim Lett (2016) 10:309–324
DOI 10.1007/s11590-015-0852-0

ORIGINAL PAPER

OMEGA one multi ethnic genetic approach

Carmine Cerrone · Raffaele Cerulli · Manlio Gaudioso

Received: 7 February 2014 / Accepted: 8 January 2015 / Published online: 25 January 2015

Abstract The genetic algorithm (GA) is a quite efficient paradigm to solve several optimization problems. It is substantially a search technique that uses an ever-changing neighborhood structure related to a population which evolves according to a number of genetic operators. In the GA framework many techniques have been devised to escape from a local optimum when the algorithm fails in locating the global one. To this aim we present a variant of the GA which we call OMEGA (One Multi Ethnic Genetic Approach). The main difference is that, starting from an initial population, k different sub-populations are produced at each iteration and they independently evolve in k different environments. The resulting sub–populations are then recombined and the process is iterated. Our basic algorithmic scheme is tested on a recent and well-studied variant of the classic problem of the minimum spanning tree: the Minimum Labeling Spanning Tree problem. We compare our algorithm with several approaches drawn from the literature. The results are encouraging in view of future application of OMEGA to other classes of problems.

Keywords Genetic algorithm · Evolutionary algorithm · Minimum labeling spanning tree

C. Cerrone · R. Cerulli
Dipartimento di Matematica, Universitá di Salerno, 84084 Fisciano, SA, Italy
e-mail: cerronecarmine@gmail.com

R. Cerulli
e-mail: raffaele@unisa.it

M. Gaudioso (✉)
Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica,
Universitá della Calabria, 87036 Rende, CS, Italy
e-mail: gaudioso@deis.unical.it

1 Introduction

The idea that population evolution models can be fruitfully used to devise effective algorithms for solving numerical optimization problems is now widely accepted. The available methods let a population of candidate solutions to a given problem evolve, using operators inspired by natural genetic variation and natural selection. The approach was firstly introduced in 1954 in the seminal papers [2,19] and computer simulation of evolution of biological systems has been intensively studied since the early 1960s [10,11]. Genetic algorithms (GA), in particular, became popular thanks to the work of John Holland in the early 1970s [14], stemming from his studies on cellular automata. In particular Holland introduced a formal framework known as Holland's Schema Theorem.

In more recent years the interest of scientists coming from different fields for various evolutionary computation methods has grown dramatically, and it is quite hard to retrace the boundaries between GAs, evolution strategy, evolution programming, and other evolutionary approaches. Nowadays the term "Genetic Algorithm" describes something very far from Holland's original concept. Among the most famous GA variants, the Lemarcking evolution [13] and Memetic algorithms [9,16] play a crucial role. There are several variants of the basic GA models, and in most cases they are problem dependent (see [17] for a survey on several applications).

GA is a population-based search metaheuristic based on population evolution and genetic operators. Many methods have been developed to escape from local optima whenever the genetic algorithms fail to detect the global one. In general it appears quite hard to skip local optima while keeping the natural evolution scheme of a genetic algorithm. In this paper we introduce yet another variant of the GA which we call OMEGA (One Multi Ethnic Genetic Algorithm).

The approach is aimed at reducing the probability of remaining trapped at a local minimum. The main difference with standard methods is that, starting from any initial population, we produce in turn k different populations and we define k different evolution environments where the k-sub-population evolve independently. We propose several merging schemes to allow, from time to time, both communication and interaction among the different populations. The idea of letting k different populations simultaneously evolve is not new. In fact it is adopted in parallel GA schemes such as, e.g., the "Island Model" [23] (see also [22]). In our method, however, each population is characterized by its own fitness function.

In stating our method, we take into account some other characteristics of the evolutionary process, aimed at improving the flexibility of the classic genetic algorithm. We translate the biologic concepts of genetic isolation (a characteristic of populations with scarce genetic interchange) and genetic convergence (the process of acquisition of the same biological trait in unrelated populations) into an algorithmic approach. As for genetic isolation, we resort to the concept of speciation [1], the evolutionary process which gives rise to new biological species in connection with geographic isolation.

More specifically, speciation tells us that if a population is branched into two or more sub–populations which are forced to adapt to new environments, then the rise of new species will probably occur. On the other hand we must take into account that genetic convergence suggests that simple parallel evolution (which in the optimization parlance can be interpreted as a kind of multi–start process) probably leads to obtain fairly similar results.

To summarize, we modify three different components of classic GA: (i) the population (ii) the fitness function (iii) the chromosome representation. In particular, as previously mentioned, we generate several populations instead of just one. As for fitness function, we allow diversification of the fitness whenever a new population is considered (this is, in fact, the main difference between our approach and the "Island Model"). Finally the chromosome representation we adopt is conceived to obtain sound results even from the application of genetic operators to individuals coming from different populations.

The sequel of the paper is organized as follows. In Sect. 2 we introduce the basic elements of our approach. In Sect. 3 the characteristics of the chromosome we adopt are discussed, while in Sect. 4 we apply our algorithm OMEGA to the Minimum Labeling Spanning Tree (MLST) problem. Finally, in Sect. 5 we presents some final remarks.

2 The approach

The OMEGA approach takes inspiration from the biological concepts of genetic isolation and speciation; moreover it leans on the building-block hypothesis [12,14].

At the $t-th$ iteration of our scheme a population P_t is available. This population is considered as a biological species and we encourage speciation through appropriately splitting it into a fixed number of sub-populations.

They are in turn exposed to different environments in order to prevent convergent evolution. Differentiation of the environments is induced by slightly modifying the fitness function for each sub-population. Such process promotes generation of possibly different species.

Population P_{t+1} is then the union of the independently evolved sub-populations.

We present now the basic scheme of Omega. The proposed scheme is quite general and different implementations are possible for several steps, thus giving rise to a family of implementable algorithms.

By $\mathcal{G}(x, P, f)$ we indicate any evolutionary scheme, where x is the input instance, P is a population of solutions and f is the fitness function. Let f_0 be a given fitness function (the "main fitness" in the following), and let $F = \{f_1, f_2, \dots, f_n\}$ be a set of n fitness functions closely related to f_0. The basic scheme of the algorithm considers a unique population P_t at time $t = 0$ which evolves according to a given $\mathcal{G}$ with $f = f_0$. After this evolutionary step, the obtained evolved population $\hat{P}_t$ is split into k different sub-populations and each of them, in turn, evolves according to the given evolutionary scheme and to a fitness function f_j randomly selected from the set F. The evolved populations are finally merged into a unique population P_{t+1} and the process is iterated until a given stopping criterion is fulfilled (see Fig. 1).

The following algorithm will be referred to as the "Main Iteration", to emphasize that it embeds the basic GA iterative schema.

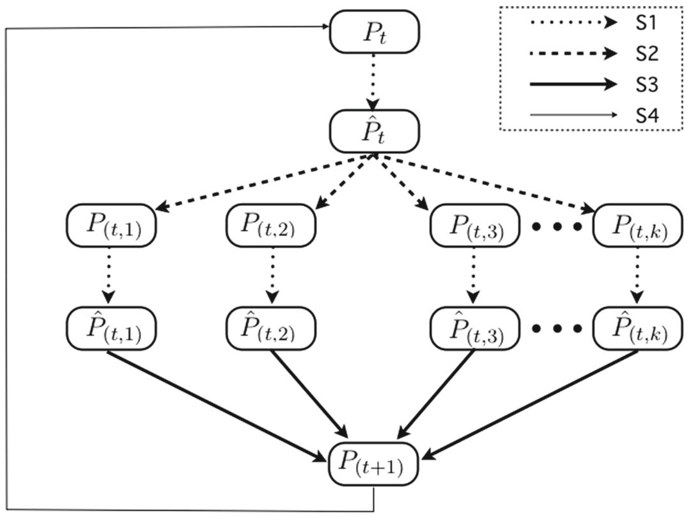


Fig. 1 Block diagram of OMEGA's Basic version

Input: problem instance x.
Output: a feasible solution of the given problem.

```
1. Initialization: Create $P_0$, the starting population for the problem. Set $t = 0$.
2. Repeat until a given stopping criterion is satisfied:
   (a) Execute $\mathcal{G}(x, P_t, f_0)$ to obtain the evolved population $\hat{P}_t$.
   (b) Split Operation: Split $\hat{P}_t$ in $P_{(t,1)}, \ldots, P_{(t,k)}$ populations
   (c) For each population $P_{(t,j)}$
         – Select a fitness function $f_j \in F$.
         – Execute $G(x, P_{(t,j)}, f_j)$ to obtain the evolved population $\hat{P}_{(t,j)}$
   (d) Merge Operation: Merge populations $\hat{P}_{(t,j)}$, $j = 1, \ldots k$, to obtain $P_{(t+1)}$.
   (e) $t := t + 1$
```

Now we discuss in details the issues related to the implementation of the various steps, dropping the subscript t for simplicity of notation. The *Split Operation* takes as input a given population P and returns a set of k sub-populations representing a partition of P.

The integer number $k \geq 2$ is given and, to get sub-populations of balanced size, the cardinality of each of them is set as follows. Letting $|P| = \left\lfloor \frac{|P|}{k} \right\rfloor k + q$, the cardinality of $(k - q)$ populations is set equal to $\left\lfloor \frac{|P|}{k} \right\rfloor$ and the cardinality of the remaining q populations is set equal to $\left\lfloor \frac{|P|}{k} \right\rfloor + 1$.

The population P is an input of the algorithm and $\mathcal{G}$, after a fixed number of iterations, returns the evolved population $\hat{P}$.

To fully exploit the characteristics of OMEGA, in selecting the "sub-routine" G it is important to pay attention to the fact that crossover takes place on couples of chromosomes that are possibly extracted from populations that have followed fairly different evolution processes. Thus the choice of the chromosome structure and the design of the crossover operator are to be devised so that a sound structure of the chromosomes is preserved. This aspect will be treated in the next sections.

3 The chromosome

The ability of the Genetic Algorithms to produce high quality solutions is often related to the fact that they implicitly and efficiently implement the building block hypothesis (BBH): an abstract adaptive mechanism that performs adaptation by recombining "building blocks". A description of this mechanism is given in [12]. Although the debate on the building block hypothesis is still ongoing, our approach relies on it.

3.1 OMEGA chromosome definition

The presence of different populations that evolve together does not imply any difference between OMEGA and any GA as far as the life cycle of a single population is concerned. On the other hand, when elements of different populations (species) are mixed up into a unique population, it is crucial to resort to a strategy that ensures compatibility among individuals coming from different populations.

The selection of the chromosome structure has to be made aiming at improving the crossover compatibility between two individuals. A possible approach to achieve it is to define a structure that it is likely to be imported into the new generation. Using this approach the chromosome is no longer a representative of the solution set, instead it is a data set containing information useful to generate a solution.

Consider, for example, the problem of finding a degree constrained spanning tree [4,5] of a graph $G(V, E)$ (refer to figure 3.1). The vertex set V has cardinality $|V|$ and, once all edges have been numerated, we can use as chromosome c any integer vector of size greater than or equal to $|V|-1$, whose entries are edge identifiers, with possible repetitions. Obviously any chromosome c does not necessarily represent a spanning tree of G but it is possible to devise a procedure that, starting from c, provides, in a non-ambiguous way, a feasible solution (a spanning tree). Hence the chromosome represents a "suggestion" for the spanning tree creation procedure. Obviously by mixing different populations with elements very far from each other, we are sure that the "suggestions" arriving from both the parents are taken into account in the child. The above ideas will be put in action in the sequel.

3.2 OMEGA and blocks

We have previously introduced a possible definition of the OMEGA chromosome. The structure is motivated by the need of improving the crossover compatibility between chromosomes coming from different sub-populations.

Now, taking inspiration from the building block theory [14], we describe a possible way to modify the chromosome structure, aiming at preserving some substructure information after the crossover operation. In fact we propose to split the chromosome into sub-components, called blocks, characterized as follows:

- The size of the block is not necessarily constant.
- The crossover operator is not enabled to modify a block, it can just recombine parents' blocks.
- Mutation is the only operator enabled to modify a block by adding/removing elements [single component (gene) of the block].

Our aim is to represent, by means of blocks, promising "solution slices" to be possibly kept in view of crossover. In Fig. 3 we show three possible OMEGA chromosomes, with the related block structure, for the graph in Fig. 2 (in Fig. 3, a block corresponds to a "column" of elements).

In next section we apply OMEGA to solve the Minimum Labeling Spanning Tree (MLST) problem.

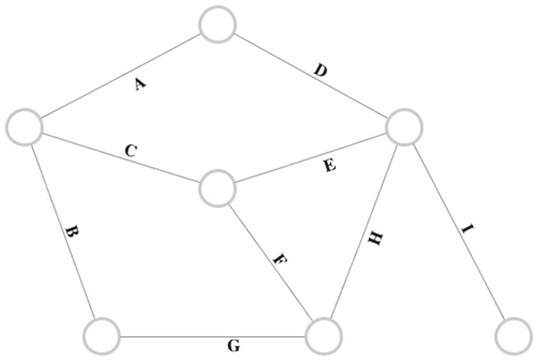
4 The Minimum Labeling Spanning Tree (MLST) problem

Given a connected, undirected graph whose edges are labeled, MLST problem looks for a spanning tree with the minimum number of distinct labels. We chose MLST as a sample problem both because it is a widely studied one (and thus comparison to other methods is viable) and because it is particularly suitable for a plain description of how our approach works in practice.

4.1 Problem definition

We are given an undirected labeled graph $G = (V, E, L)$, where V is the set of vertices and E is the set of edges, each of them being labeled by exactly one label extracted from the set of distinct labels L. In the following we denote by $l(e) \in L$ the

Fig. 2 $G(V, E)$



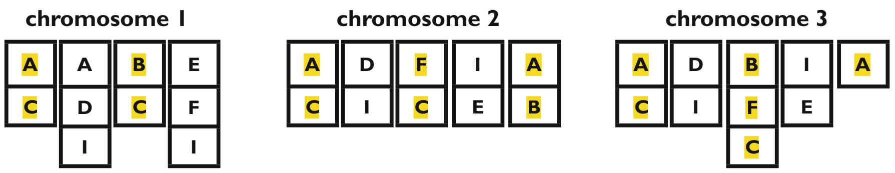


Fig. 3 OMEGA's chromosome examples

(unique) label associated with the edge e, $e \in E$. We look for a spanning tree whose edges exhibit the minimum number of distinct labels. The problem was introduced by Chang and Leu [7], who proved its NP-hardness by reduction from the minimum cover problem. Since then, several researchers have introduced heuristics for MLST problem. Some references are [3,6,15,18,20,21]. In particular Genetic Algorithms for MLST problem are presented in [18,20,21].

4.2 Omega for MLST problem

To describe OMEGA as applied to the MLST problem we need to introduce the appropriate chromosome definition and to specify the crossover, mutation and population splitting operators, together with the set of different fitness functions.

4.2.1 Chromosome definition

We introduce first two parameters $\mathcal{W}$ and $\mathcal{H}$, aimed at bounding the chromosome size. In particular $\mathcal{W}$ represents the width of the chromosome, that is the maximum number of blocks, while $\mathcal{H}$ represents its height, corresponding to the maximum number of elements for each block. The value of the parameters $\mathcal{W}$ and $\mathcal{H}$ is related to $|V|$. In the numerical experiments section we will define a possible choice.

Each element (cell) of the chromosome is associated with an edge, characterized by a specific label. In Fig. 4 a chromosome composed of $\mathcal{W}$ blocks, with block size ranging between one and $\mathcal{H}$ is presented. The procedure in Fig. 5 takes as input a chromosome and returns a spanning tree, which is obtained by solving a standard Minimum Weight Spanning Tree (MST) problem on the given graph G, where each edge is assigned a weight $w(e)$ depending on the chromosome structure.

In particular the algorithm builds the integer vector $Cost$ of size $|L|$. $Cost(l)$, $l = 1, \ldots, L$, represent the opposite of the total number of occurrences of label l in the chromosome. Then, a weight $w(e)$ equal to $Cost(l(e))$ is associated with each edge $e \in E$. The weights $w(e)$, $e \in E$, are grouped into set W.

Fig. 4 OMEGA's chromosome structure

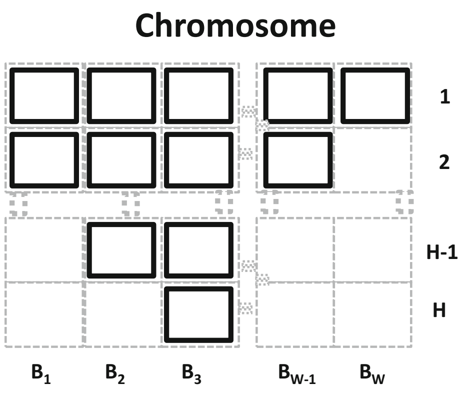


Fig. 5 Spanning tree generator

```
TreeCreator
INPUT: $G(V, E, L)$ , chromosome $c$
OUTPUT: $Sol \in E$ Spanning tree of $G$

1: $Cost \leftarrow 0$
2: for all edge $e \in c$ do
3:     $Cost(l(e)) \leftarrow Cost(l(e)) - 1$
4: end for
5: for all edge $e \in E$ do
6:     $w(e) \leftarrow Cost(l(e))$
7: end for
8: $Sol \leftarrow MST(G(V, E, W))$
9: return $Sol$
```

Fig. 6 Input graph

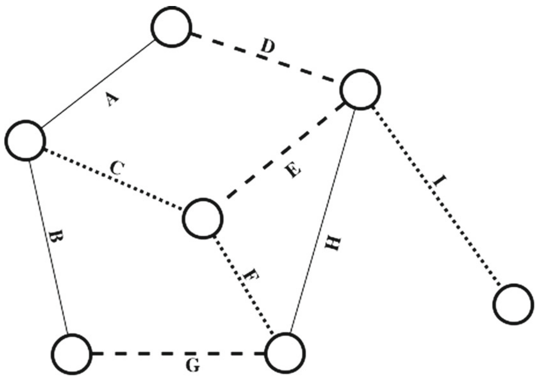
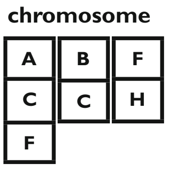


Fig. 7 Chromosome



In the following we present an example to show how the procedure produces a spanning tree starting from the input chromosome. We use as input the graph of Fig. 6, where a label is associated with each edge.

$$G(V, E, L), |V| = 7, E = \{A, B, C, D, E, F, G, H, I\}, L = \{0, 1, 2\}.$$

We remark that the label set L is composed of three different labels. The set of couples edge-label is:

$$\{(A, 0), (B, 0), (C, 1), (D, 2), (E, 2), (F, 1), (G, 2), (H, 0), (I, 1)\}$$

and the chromosome represented in Fig. 7 is

$$c = \{\{A, C, F\}, \{B, C\}, \{F, H\}\}$$

The procedure TreeCreator assigns the weights $w(e)$ as follows:

$$\{(A, -3), (B, -3), (C, -4), (D, 0), (E, 0), (F, -4), (G, 0), (H, -3), (I, -4)\}$$

and the resulting graph is shown in Figs. 8 and 9. The MST procedure generates the spanning tree of Fig. 10, where two different labels are present.

4.2.2 The crossover

The crossover operator takes as input two chromosomes $c1$, $c2$ and produces as output the chromosome c3.

The procedure randomly selects half of the blocks, respectively, of $c1$ and $c2$, and inserts them into the new chromosome $c3$. In Fig. 11 an example referred to problem previously considered is shown.

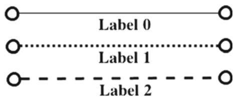


Fig. 8 Labels

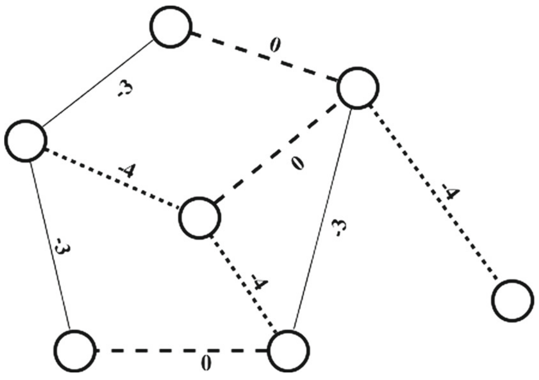


Fig. 9 MST graph

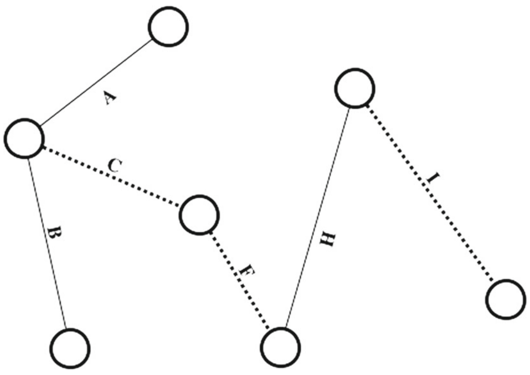


Fig. 10 Spanning tree

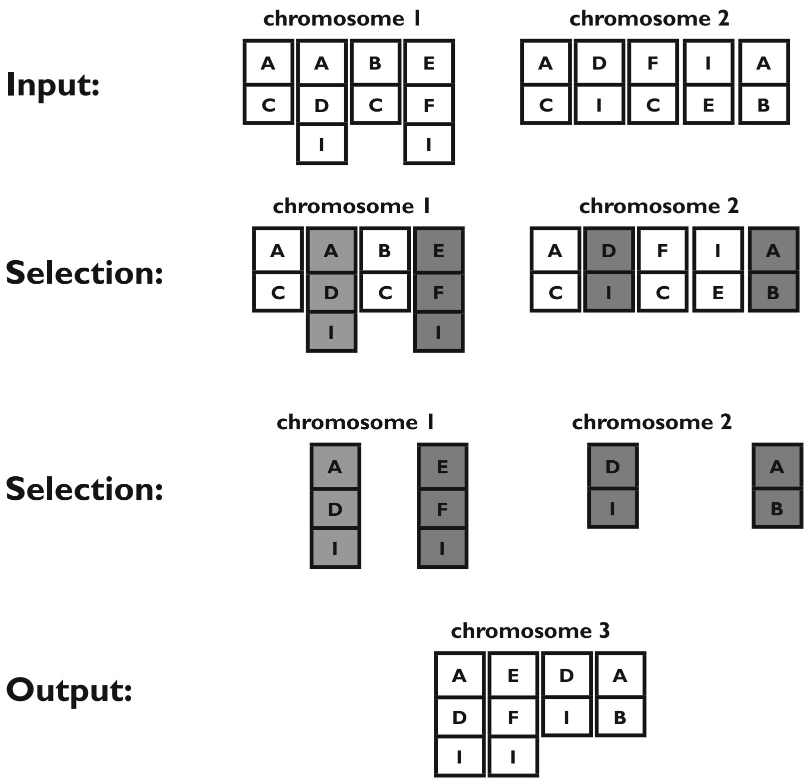


Fig. 11 Crossover

4.2.3 The mutation

OMEGA embeds three different mutation schemes referred as, respectively, MUT-ADD, MUT-DEL and MUT-CHANGE. They work as follows.

- MUT-ADD randomly selects a block in the chromosome and if its size is not equal to $\mathcal{H}$, a new element (an edge in the MLST example) is added into block in the last position. The choice of the element to be added is problem-dependent. For the MLST problem, such choice has been restricted to a randomly selected edge among those sharing a vertex with any of the edges of the block.
- MUT-DEL first randomly selects a block in the chromosome, then, if the size is non-zero, a randomly selected cell is removed from the block.
- MUT-CHANGE randomly selects both a block and an element in it. For the MLST problem the selected edge is replaced by a new one, also in this case it is randomly picked from among those sharing a vertex with any of the edges of the block.

In our implementation the mutation operator selects one of the above described schemes according to given probabilities. MUT-CHANGE is used whenever it is impossible to use either MUT-ADD or MUT-DEL.

4.2.4 Population splitting

The Splitting Population Function (SPF) is used to partition the original population into k sub-populations of (possibly) equal size. This procedure is totally random, and the cardinalities of the resulting sub–populations differ at most of 1.

4.2.5 The fitness functions

A fitness function f associates a value with each chromosome and it is strongly problem-dependent. In the considered MLST problem, the natural fitness function is the number of different labels present in the spanning tree generated by the TreeGenerator procedure.

More formally, once the tree is associated with any chromosome c, we can define the binary vector $z(c)$, whose components $z_l(c), l = 1, \ldots, |L|$ are defined as follows,

$$z_l(c) = \begin{cases} 1 \text{ if there exists in the spanning tree at least one edge with label } l \\ 0 \text{ otherwise,} \end{cases}$$

and we define the main fitness function f_0 as follows

$$f_0(c) = \sum_{l \in L} z_l(c).$$

On the other hand OMEGA is characterized by population splitting, accompanied by diversification of the fitness function, which leads to the definition of specific fitness functions, one for each sub–population. In our approach we define a weight vector $\beta^{(j)} \in \mathcal{R}^{|L|}$ for each sub–population P_j, $1 \leq j \leq k$ and the corresponding fitness function $f_j(c)$ for the chromosomes belonging to population P_j is:

$$f(c, \beta^{(j)}) = \sum_{l \in L} \beta_l^{(j)} z_l(c)$$

4.3 Pseudocode

The pseudocode in Fig.12 describes, with reference to the MLST problem, the basic genetic algorithm which is called by OMEGA whenever a population or sub–population evolution is implemented (the sub–population index j is dropped for sake of readability).

4.4 Computational results

As previously mentioned we have tested our method on several instances of MLST problem. The following parameters have been adopted.

- Chromosome width $\mathcal{W} = \frac{|V|}{10}$;
- Block height $\mathcal{H} = 3$;

```
BASIC-GA
INPUT: $G(V, E, L)$ , Population $P$ and weight vector $\beta$
OUTPUT: Evolved Population $\hat{P}$

 1: for fixed number of iteration do
 2:     for all $c$ in $P$ do
 3:         select any $c' \in P$ such that $f(c', \beta) \leq f(c, \beta)$
 4:         $P \leftarrow (P \setminus \{c\}) \cup \{crossover(c, c')\}$
 5:     end for
 6:     for all $c$ in P do
 7:         $P \leftarrow (P \setminus \{c\}) \cup \{mutation(c)\}$
 8:     end for
 9: end for
10: return $P$
```

Fig. 12 Basic genetic algorithm

Table 1 OMEGA's computational results

n	l	d	MVCA	VNS	SA	RTS	PILOT	OMEGA
20	20	0.8	2.6	2.4	**2.4**	**2.4**	**2.4**	**2.4**
20	20	0.5	3.5	3.2	**3.1**	**3.1**	3.2	**3.1**
20	20	0.2	7.1	6.9	**6.7**	**6.7**	**6.7**	**6.7**
30	30	0.8	**2.8**	**2.8**	**2.8**	**2.8**	**2.8**	**2.8**
30	30	0.5	**3.7**	**3.7**	**3.7**	**3.7**	**3.7**	**3.7**
30	30	0.2	8	7.8	**7.4**	**7.4**	7.5	**7.4**
40	40	0.8	**2.9**	**2.9**	**2.9**	**2.9**	**2.9**	**2.9**
40	40	0.5	3.9	3.9	3.9	4	**3.7**	**3.7**
40	40	0.2	8.6	8.3	**7.4**	7.9	7.7	**7.4**
50	50	0.8	**3**	**3**	**3**	**3**	**3**	**3**
50	50	0.5	4.4	4.1	4.2	4.2	**4**	4.1
50	50	0.2	9.2	9.1	8.7	8.8	**8.6**	**8.6**
100	100	0.8	3.3	3.1	4	3.4	**3**	**3**
100	100	0.5	5.1	5	5.2	5.1	4.7	**4.6**
100	100	0.2	11	10.7	10.7	10.9	10.3	**10.1**

- Population cardinality $|P| = 100$;
- Mutation probabilities: MUT-ADD = 0.50, MUT-DEL = 0.30, MUT-CHANGE = 0.20;
- Number of iterations for each execution of Basic GA = 100;
- The weight $\beta^{(j)}$, $1 \leq j \leq k$ is randomly selected in the set $\{1, 2, \ldots, 10\}$.

The overall procedure stops whenever no improvement in the objective function is achieved after 20 executions of "Main Iteration". Comparison has been made with the results obtained by several other heuristics and metaheuristics methods, such as MVCA [15], Variable Neighborhood Search (VNS) [8], Simulated Annealing (SA), Reactive Tabu Search (RTS), and the Pilot Method (PILOT) [6]. We have used the same parameters setting as in the cited articles. The benchmark test problems, together with the numerical results of the above mentioned algorithms are taken from [6].

In Table 1 the first three columns show the parameters characterizing the different instances (n, number of vertices, l, number of labels and d, graph density). The remaining columns give the results of the tested algorithms.

All the reported values are average values over 10 different instances of the objective function (number of distinct labels) at the stop. The instances set is composed of 150 instances. In each row of the table in the figure it is reported in bold the best result obtained. In Fig. 13 we report the number of times (in percentage) each algorithm has performed best. Table 2 presents comparison of the (average) computation time, which, in turn, is expressed as a function of the size of the input instance in Fig. 14.

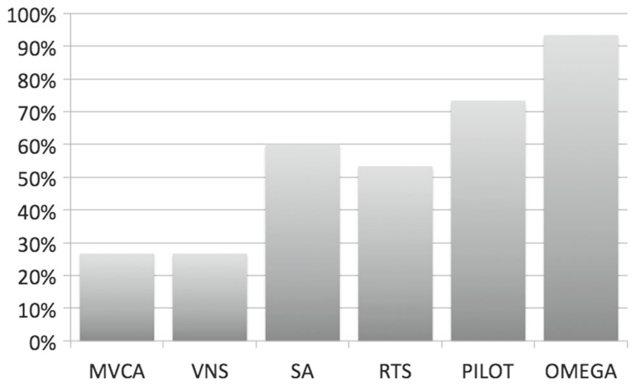


Fig. 13 Best solution percentage

Table 2 Computational times (s)

n	l	d	MVCA	VNS	SA	RTS	PILOT	OMEGA
20	20	0.8	0.01	1.67	0.69	10.00	0.11	1.95
20	20	0.5	0.01	1.92	0.79	10.00	0.13	1.50
20	20	0.2	0.02	1.69	0.91	10.00	0.26	0.93
30	30	0.8	0.03	5.58	2.44	10.00	0.46	4.02
30	30	0.5	0.04	7.22	1.67	10.00	0.59	2.92
30	30	0.2	0.07	8.55	2.15	10.00	1.24	1.78
40	40	0.8	0.07	14.04	3.37	10.00	1.32	7.02
40	40	0.5	0.09	19.14	3.14	10.00	1.60	4.97
40	40	0.2	0.19	26.76	2.59	10.00	3.56	3.00
50	50	0.8	0.14	28.52	6.63	10.00	2.95	10.48
50	50	0.5	0.20	41.81	5.04	10.00	3.80	7.71
50	50	0.2	0.39	64.52	5.25	10.00	9.11	4.14
100	100	0.8	1.25	258.93	53.29	50.00	37.19	45.95
100	100	0.5	1.76	381.55	27.66	50.00	50.86	30.27
100	100	0.2	3.24	710.55	19.36	50.00	122.42	16.08

The good behavior of OMEGA on the large instances is apparent. The Table 3 reports more details on the computational results of OMEGA; in particular, for each scenario, the minimum and the maximum value of the objective function is reported, together with the average value (also in Table 1). In Table 3 we compare OMEGA with GA, that is the OMEGA with only one population, is reported in bold the worst result obtained. In the experiments we have given to GA a maximum execution time equal to the one provided to OMEGA, multiplied by the number of populations. Such comparison helps to assess the impact of our multi-ethnic schema on a standard algorithm. In fact OMEGA improves the quality of the solution for several of the larger instances.

5 Conclusions

In this paper we introduced a variant of the GA which we called OMEGA. This new approach is aimed at reducing the probability of remaining trapped at a local minimum. The main difference with standard methods is that we produce *k* different populations and we define *k* different evolution environments where the k-sub-population evolve independently using its own fitness function. We report some preliminary test of our approach on a recent and well-studied variant of the classic problem of

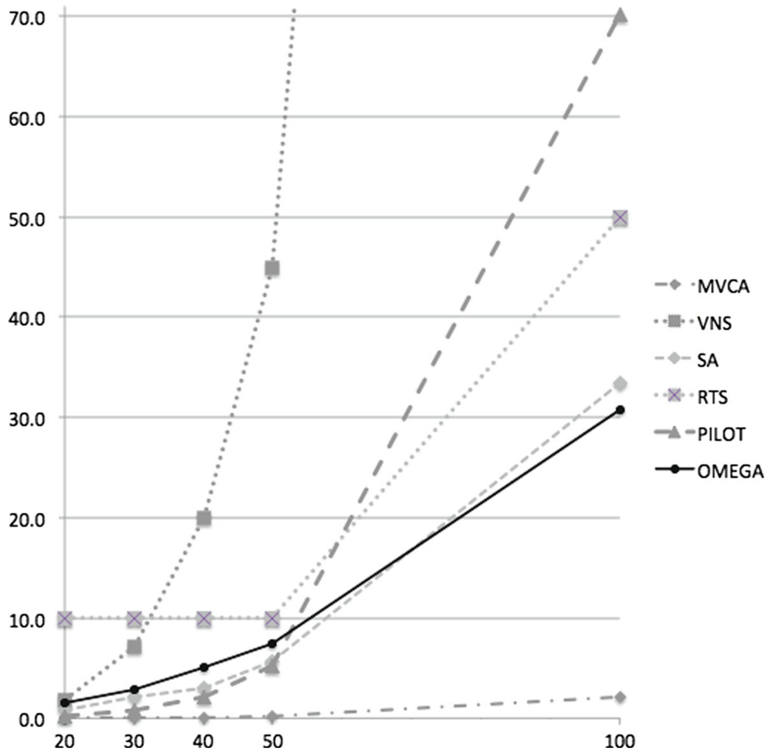


Fig. 14 Computational time (s) vs. instance size. (Averaged values over the three different densities)

Table 3 OMEGA vs. GA

n	l	*d*	OMEGA			GA		
			Min	Avg	Max	Min	Avg	Max
20	20	0.8	2	2.40	3	2	2.40	3
20	20	0.5	3	3.10	4	3	3.10	4
20	20	0.2	5	6.70	8	5	6.70	8
30	30	0.8	2	2.80	3	2	2.80	3
30	30	0.5	3	3.70	4	3	3.70	4
30	30	0.2	6	7.40	8	6	7.40	8
40	40	0.8	2	2.90	3	2	2.90	3
40	40	0.5	3	3.70	4	3	3.70	4
40	40	0.2	7	7.40	8	7	**7.80**	**9**
50	50	0.8	3	3.00	3	3	3.00	3
50	50	0.5	4	4.10	5	4	**4.20**	5
50	50	0.2	8	8.60	10	8	**8.90**	10
100	100	0.8	3	3.00	3	3	**3.10**	**4**
100	100	0.5	3	4.60	5	3	**4.80**	**6**
100	100	0.2	8	10.10	12	**9**	**10.80**	**13**

the minimum spanning tree: the Minimum Labeling Spanning Tree problem. Future research will be aimed at testing our approach on different classical combinatorial optimization problems to highlight the advantages of our approach compared to standard techniques.

References

1. Baker, J.M.: Adaptive speciation: the role of natural selection in mechanisms of geographic and non-geographic speciation. Stud. Hist. Philos. Sci. Part C Stud. Hist. Philos. Biol. Biomed. Sci. **36**(2), 303–326 (2005)
2. Barricelli, N.: Numerical testing of evolution theories. Acta Biotheoretica **16**(1–2), 69–98 (1962)
3. Captivo, M., Clímaco, J., Pascoal, M.: A mixed integer linear formulation for the minimum label spanning tree problem. Comput. Oper. Res. **36**(11), 3082–3085 (2009)
4. Carrabs, F., Cerulli, R., Gaudioso, M., Gentili, M.: Lower and upper bounds for the spanning tree with minimum branch vertices. Comput. Optim. Appl. **56**(2), 405–438 (2013)
5. Cerrone, C., Cerulli, R., Raiconi, A.: Relations, models and a memetic approach for three degree-dependent spanning tree problems. Eur. J. Oper. Res. **232**(3), 442–453 (2014)
6. Cerulli, R., Fink, A., Gentili, M., Voss, S.: Metaheuristics comparison for the minimum labelling spanning tree problem. In: Golden, B.L., Raghavan, S., Wasil, E.A. (eds.) The next wave on computing, optimization, and decision technologies, pp. 93–106. Springer, Berlin (2005)
7. Chang, R.S., Shing-Jiuan, L.: The minimum labeling spanning trees. Inf. Process. Lett. **63**, 277–282 (1997)
8. Consoli, S., Moreno-Prez, J.A., Mladenovic, N.: Intelligent variable neighbourhood search for the minimum labelling spanning tree problem. Electron. Notes Discrete Math. **41**, 399–406 (2013)
9. Cordeau, J.F., Gaudioso, M., Laporte, G., Moccia, L.: A memetic heuristic for the generalized quadratic assignment problem. INFORMS J. Comput. **18**(4), 433–443 (2006)
10. Fraser, A.: Simulation of genetic systems by automatic digital computers. I. Introd. Aust. J. Biol. Sci. **10**, 484–491 (1957)
11. Fraser, A.: Computer models in genetics. McGraw-Hill, New York (1970)

12. Goldberg, D.E.: Genetic algorithms in search. Optimization and machine learning. Addison-Wesley Longman Publishing Co., Inc., Boston (1989)
13. Grefenstette, John J. Lamarckian learning in multi-agent environments. Navy Center For Applied Research in Artificial Intelligence Washington DC, (1995)
14. 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)
15. Krumke, S., Wirth, H.: On the minimum label spanning tree problem. Inf. Process. Lett. **66**(2), 81–85 (1998)
16. Moscato, P., Norman, M.G.: A memetic approach for the traveling salesman problem implementation of a computational ecology for combinatorial optimization on message-passing systems. Parallel Comput. Transput. Appl. **1**, 177–186 (1992)
17. Miettinen, K., Mäkelä, M.M., Neittaanmäki, P., Periaux, J. (eds.): Evolutionary algorithms in engineering and computer science. Wiley, Chichester (1999)
18. Nummela, J., Julstrom, B.: An effective genetic algorithm for the minimum-label spanning tree problem. In: GECCO ’06 Proceedings of the 8th annual conference on Genetic and evolutionary computation, pp. 553–558 (2006)
19. Reed, J., Toombs, R., Barricelli, N.: Simulation of biological evolution and machine learning. I. Selection of self-reproducing numeric patterns by data processing machines, effects of hereditary control, mutation type and crossing. J. Theoret Biol. **17**, 319–342 (1967)
20. Xiong, Y., Golden, B., Wasil, E.: A one-parameter genetic algorithm for the minimum labeling spanning tree problem. IEEE Trans. Evolut. Comput. **9**(1), 55–60 (2005)
21. Xiong, Y., Golden, B., Wasil, E.: Improved heuristics for the minimum label spanning tree problem. IEEE Trans. Evol. Comput. **10**(6), 700–703 (2006)
22. Whitley, D., Rana, S., Heckendorn, R.B.: The island model genetic algorithm: on separability, population size and convergence. J. Comput. Inf. Technol. **7**, 33–48 (1999)
23. Latter, B.D.H.: The island model of population differentiation: a general solution. Genetics **73**(1), 147–157 (1973)