Transgenetic Algorithm: A New Evolutionary Perspective for Heuristics Design

Elizabeth F. G. Goldbarg
Federal University of Rio Grande do Norte
Natal, RN, Brazil
+55 – 84 21353814
beth@dimap.ufrn.br

Marco C. Goldbarg
Federal University of Rio Grande do Norte
Natal, RN, Brazil
+55 – 84 21353814
gold@dimap.ufrn.br

Ligia B. Bagi
Federal University of Rio Grande do Norte
Natal, RN, Brazil
+55 – 84 21353814
ligiabagi@yahoo.com.br

ABSTRACT

Transgenetic algorithms are evolutionary computing techniques based on living processes where cooperation is the main evolutionary strategy. Those processes contain the movement of genetic material between living beings and endosymbiotic interactions. With the objective of having a better approximation between the proposed metaphor and the reality the algorithm also considers intracellular mechanisms of genetic information transposition and the quorum sensing, that is, the bacteria's ability for communicating and coordinating actions. To illustrate the application of a transgenetic algorithm to a difficult combinatorial optimization problem, an example is provided for the Traveling Purchaser Problem. The introduced approach is compared with two recent heuristics proposed for the same problem. The results of a computational experiment are reported and 9 new best solutions for benchmark instances are presented.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search – *heuristic methods.*

General Terms

Algorithms, Design.

Keywords

Transgenetic Algorithm, Horizontal Gene Transfer, Transposon, Quorum Sensing.

1. INTRODUCTION

Transgenetic algorithms are evolutionary algorithms whose metaphor is based on the endosymbiotic theory and on properties of the intracellular flow [11]. The endosymbiotic theory was popularized by Margulis [23] and states that a new organism can emerge from the fusion of two or more independent beings. The term "endosymbiosis" specifies the relationship between organisms which live one within another (symbiont within host). That theory suggests that competition is not the unique way to promote genetic improvement. She states that "Life did not take over the globe by combat, but by networking". The approach suggests also that sexual reproduction was originated on a variant of endosymbiosis and that this reproduction mean was a way to prevent the perpetuation of genetic alterations caused by microbial transcriptions.

Evidences for the Margulis' proposal have been provided by successive discoveries concerning evolutionary mechanisms that yield direct sharing of DNA among microorganisms and cells. Such mechanisms are very primitive. They were constituted to allow the occurrence of permanent alterations on the genetic code of cells and microorganisms, facilitating the emergence of jumps of fitness. A set of such mechanisms is called "horizontal gene transfer". Horizontal gene transfer is defined to be the movement of genetic material between bacteria other than by descent in which information travels through the generations as the cell divides. The horizontal transfer of functional genes between organisms is the theoretical foundation of the endosymbiotic origin of cellular organelles, as well as the basis of genetic therapies and the technology of genetic modification [28]. Those mechanisms by which genetic materials are exchanged are: transformation, transduction and conjugation. A vehicle for genetic exchange is the plasmid. Plasmids are mobile genetic particles, DNA rings that can be exchanged between certain cells. The transformation is a common mode of horizontal gene transfer in which foreign genetic material is transferred to a cell, resulting on genetic alteration. The transduction is a method in which the transport of DNA between organisms involves the mediation of viruses. Finally, conjugation is the method where DNA transference occurs between bacterial cells that are in physical contact. During bacterial evolution, the ability of bacteria to adapt to new environments most often results from the acquisition of new genes through horizontal transfer rather than by the alteration of gene functions through numerous point mutations.

Symbiont vectors are molecular structures, microorganisms or cells, able to act on the intra or extra-cellular flow. They can compose their information with the information of other molecular structures. The symbiosis is a complex concept and it may involve situations where mutual benefits are not clear [17]

GECCO'07, July 7–11, 2007, London, England, United Kingdom.
Copyright 2007 ACM 978-1-59593-698-1/07/0007...$5.00.

[21]. The endosymbiosis is directed to the symbiont absorption on the long run. As a result of that absorption the symbiont genome is reduced while the shared genetic material is augmented.

There are other possibilities that result on genetic alterations in microorganisms and cells, such as the mechanisms of transposition or transposons [34]. Transposons are DNA sequences that are part of other genetic elements such as chromosomes or plasmids. They move from one location to another inside the DNA of a given cell characterizing a transposition (permutation). The insertion sequence is encoded on a cellular enzyme, the Transposase. The transposon action comprises two distinct mechanisms of DNA edition. The first mechanism allows cutting and pasting DNA fragments. The second mechanism executes a copy and paste. The composition of those two mechanisms results on an effect that is similar to permute restricted DNA fragments. The permutations may occur exclusively on the information inside the chromosome or in a composition with the information of an environment (for instance, in composition with information encoded on plasmids). The transposons are intracellular flow vehicles.

The transgenetic algorithms adapt the concepts of endosymbiosis and other properties of the intracellular flow for the computational context. They propose the execution of evolution by means of a process of information sharing among a population of endosymbionts within a host. The evolution of the population of symbionts and its integration with the host information is intermediate by horizontal gene transfer and transposition mechanisms. Those genetic exchange mechanisms are operated by means of agents named "transgenetic vectors". The transgenetic vectors mimic natural vehicles, such as plasmids and transposons. The evolution of the population of symbionts utilizes the information within the host's cytoplasm. That information is composed and transported by the transgenetic vectors. It is done in accordance with the "quorum sensing" paradigm, the ability of bacteria to communicate, which proposes the possibility to direct and guide the action of a microbial population in the cellular context.

The first work that presented the elements of a symbiotic approach in the computational context is due to Hillis [15]. The first work of the Evolutionary Computation that mimics the endosymbiosis was presented by Bull and Fogarty [5]. Kim *et al.* [19] proposed a co-evolutionary algorithm that introduced the concept of endosymbiotic individuals – formed by the union of two individuals. Other algorithms based on that metaphor where co-evolution or symbiosis were emphasized (not properly endosymbiosis) were presented by Tsujimura *et al.* [37], Pagie and Mitchell [27], Rosin and Belew [33], Husbands [16], De La Cal Marín *et al.* [7], Kim *et al.* [20], and Michaelian [24]. A correlated approach based upon evolutionary interactions between individuals of different natures was presented by Kubota *et al.* [22]. Horizontal gene transfer mechanisms inspired some genetic algorithms operators: transduction [22], conjugation [25] [26], and transformation [35]. Operators of genetic algorithms inspired in transposons are presented by Simões and Costa [36] and Chan *et al.* [6].

Section 2 presents the transgenetic algorithms. To illustrate the application of a transgenetic algorithm to an NP-hard Combinatorial Optimization problem, the proposed approach is applied to the Traveling Purchaser Problem in section 3. Section 4 presents the results of a computational experiment. Finally, some conclusions and remarks are presented in section 5.

2. THE TRANSGENETIC ALGORITHM

This paper presents an algorithm with fundamentals on the process of endosymbiosis, and also able to consider other genetic alterations that occur in the intracellular context. Besides employing the concept of agglutination of genetic information of more than one individual, the algorithm allows considering intracellular interactions of gene transposition. Once the metaphor is based on the evolution of primitive beings, crossover and mutation operations are not considered.

The recent discovery of microbiology that bacteria communicate is also a source of inspiration for the transgenetic algorithms. Bacterial communication is called "quorum sensing" because it is a density-dependent process that functions when a population is of sufficient size [38]. Quorum sensing systems are very widespread and modulate many processes in bacteria that are associated with humans, plants, animals, and which occur in the natural environment. It is known that quorum sensing also regulates a wide range of physiological processes and involves a variety of different signal molecules [18].

The evolutionary process of a transgenetic algorithm is done with the interaction of the information contained in a host and a population of symbiont chromosomes by means of the transgenetic vectors. A transgenetic vector, λ, is a pair $\lambda = (I, \phi)$, where I is an information string and ϕ is a manipulation method, $\phi = (p_1,...,p_s)$, p_j, j=1,...,s, are procedures that define the vector's action. The insertion of an information string, I, transported by a transgenetic vector, λ, on a chromosome C, will probably alter C and its fitness.

In analogy with the terminology employed by microbiologists, there are four types of transgenetic vectors: plasmid, virus, recombined plasmid and transposon. Table 1 summarizes the procedures that compose the manipulation methods of those transgenetic vectors.

Table 1. Procedures of transgenetic vectors

Procedures	Description
p_1 - Attack	Defines a criterion that establishes whether a chromosome C is susceptible to the information of a transgenetic vector λ A: C, $\lambda \rightarrow$ {false,true}
p_2 - Transcription	If $A(C, \lambda)$ = "true", the procedure defines how the information I is transferred from λ to C
p_3– Blocking/ Unblocking	Establishes a period of time (e.g. number of iterations) in which the transcribed information cannot be altered in C
p_4 - Identification	Identifies the positions in C that will be utilized to limit λ's operation
p_5 - Recombination	Identifies the origin and the length of two or more information strings and composes them in λ

A vector is called plasmid when its information string is translated on a genetic code – a DNA substring – and its method utilizes only procedures p_1 and p_2. If the vector's information string is a genetic code and the vector utilizes procedures p_1, p_2 and p_3, then λ is called virus. A vector is called transposon when its method utilizes procedures p_1, p_2 and p_4. Finally, a vector is called recombined plasmid when its information is a composition of two or more information strings and the vector utilizes procedures p_1, p_2, p_4 and p_5.

There are three interacting contexts in a transgenetic algorithm:

i) A population of chromosomes (cells or prokaryotes);

ii) A population of transgenetic vectors representing vehicles for transferring and editing the genetic information;

iii) The extra-cellular context, that is, the interior of the host, rich in information which can influence the symbionts.

The algorithm utilizes a host database with the information contained in the host's cytoplasm. It is a general repository of information. That base may contain information obtained *a priori* and information found out during the execution of the evolutionary process. The information may be encoded either on a genetic format, such as partial solutions of a given problem, or on an abstract format such as procedures or rules for genetic sharing.

The evolutionary process of transgenetic algorithms is accomplished by a structure of rules that simulate the natural mechanisms of quorum sensing. Those rules are called transgenetic rules. The control of the evolution of the population of chromosomes, transgenetic vectors, and the information of the host database is done by three classes of transgenetic rules.

Type 1 rules direct the construction of the information string *I* that is transported by the transgenetic vectors. Type 1 rules may utilize any type of knowledge stored in the host database. Type 2 rules define how the information *I* is transcribed in a chromosome – the operator utilized by λ. The transcription rules may evolve in conformity with the resistance shown by the chromosomes. A number of type 1 and type 2 rules may exist in an algorithm. Figure 1 shows that type 3 rules direct the whole process playing part of the role of the quorum sensing. Type 3 rules define which vectors are utilized, the number of chromosomes that are attacked at a given iteration, the number of vectors that are created, the stopping criterion, etc.

Figure 2 presents a general framework of a transgenetic algorithm. The transgenetic rules are not detailed in the algorithm description to avoid overloading the text. Initially, a population of chromosomes is generated. It is done in the same manner other evolutionary algorithms do. Then, the parameters of the transgenetic rules are loaded. The information of the host database can be obtained from theoretical or heuristic knowledge about the problem that is being tackled. It can be updated during the execution of the algorithm. The information sharing between the population of chromosomes and the transgenetic vectors is done until a stopping criterion be satisfied.

Transgenetic algorithms were proposed by Gouvêa [14] who applied them to the Quadratic Assignment Problem. Two initial contexts were proposed for those algorithms: an extra-intracellular context [10] and a purely intracellular context [9] where only plasmids were utilized as transgenetic vectors. The latter class of algorithms was called ProtoG. ProtoG algorithms were more promising than the former class being utilized to tackle a number of real world problems [2] [8] [12] [13]. Ramos *et al.* [31] applied logistic regression for parameter tuning of ProtoG algorithms. An application of ProtoG algorithms for the Travelling Salesman Problem is reported in the work of Ramos [30].

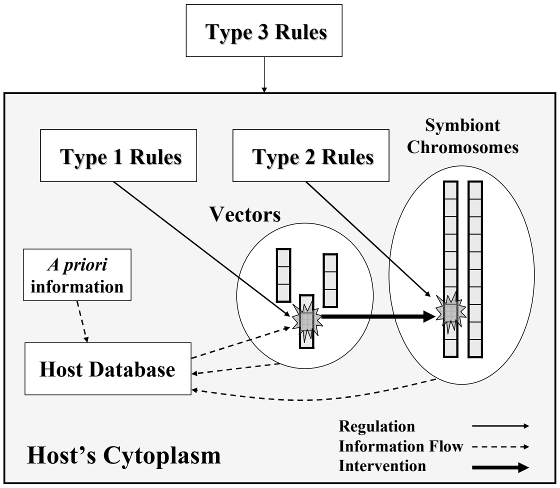

Figure 1. Transgenetic evolution

```
1. Generate an initial population
2. Load transgenetic rules (TR)
3. Load the host database (HB)
4. Repeat
5.    Generate extra-intracellular vectors
6.    Select chromosomes for manipulation
7.    Manipulate chromosomes in conformity with TR
8.    Update TR and HB
9. until a stopping criterion is satisfied
```

Figure 2. General framework of a transgenetic algorithm

3. A TRANSGENETIC ALGORITHM FOR THE TPP

The proposed approach is applied to the NP-hard problem named Traveling Purchaser Problem (TPP), a generalization of the Traveling Salesman Problem. In this variant there is a set of *m* markets, vertices of a graph *G*, and a set of *n* products that must be purchased. Each product is available, with different quantities, on a subset of markets and the unit cost of a product depends on the market where it is available. The objective of the purchaser is to buy all the products, departing and returning to a domicile (location v_0), with the least possible cost. The cost is defined as the summation of the weights of the edges in the tour plus the price paid to acquire the products. Thus, there is no need of including all the markets in the tour. The first work where the TPP is introduced as it is presently known is due to Ramesh [29]. The problem can be stated as follows. Given a domicile, v_0, a set of markets $M = \{v_1, v_2, ..., v_m\}$and a set of products $K=\{f_1, f_2, ..., f_n\}$, the problem is represented in a graph $G = (V,E)$ where $V = \{v_0\} \cup M$ and $E = \{[i,j]: v_i, v_j \in V, i<j\}$. A demand d_k is

assigned to each product f_k. The number of units of product f_k at market v_i is denoted by q_{ki} and M_k denotes the set of markets where the product f_k is available, $M_k \subseteq M$. The cost of product f_k at market v_i is denoted by b_{ki} and the cost of traveling from market v_i to market v_j is given by c_{ij}. The objective is to determine a minimum cost tour in G such that v_0 is the starting and the ending point and the f_k products are purchased, completely satisfying the demand. In this paper the uncapacitated version of the TPP (UTPP) is tackled by a Transgenetic Algorithm. For the UTPP it is assumed that if a product is available at a given market, its quantity is sufficient to satisfy the demand [3]. In this problem variant, it can be considered that d_k=1 and $q_{ki} \in \{0,1\}$, $1 \le k \le n$, $1 \le i \le m$.

A general framework of the algorithm proposed for the UTPP is shown on figure 3. Four input parameters are passed to the algorithm: *#sizeP*, the population size; *k*, the number of plasmids generated on each iteration; β, the number of iterations of the inner loop; and η, the total number of iterations.

```
1. Generate_population($P$={$C_1$,…,$C_{\#sizeP}$ })
2. Load the host database, HD
3.  j ←β;
4. repeat
5.  i ←1
6.   repeat
7.      u ← random(η)
8.      if (u ≥j) then
9.         Generate $k$ plasmids and choose the best one, λ
10.        for each individual $C$ of $P$
11.              $C'$← attack_plas($C$,λ)
12.     else
13.        Generate a transposon
14.        for each individual C of the population
15.              $C'$← attack_trans($C$,λ)
16.        if $C'$ is better than $C$ then
17.           $C$ ← $C'$
18.        if $C$ is better than the current best solution then
19.           Include $C$ in HD
20.           Remove the worse chromosome of HD
21.     i ←i+1
22.   until (β = i)
23.   j ←j+β
24. until  (j>η)
```

Figure 3. Transgenetic algorithm for the TPP

The chromosomes represent TPP solutions and are defined as a sequence of markets, beginning and ending at the domicile, v_0. The fitness is given by the cost of the tour represented in the chromosome plus the lowest costs of acquisition of all products on the markets of the tour.

At first, a population of chromosomes is generated. To construct a chromosome, random markets, with no repetition, are iteratively included until a feasible solution is built. Then the Lin and Kernighan algorithm version of Applegate *et al*. [1] is applied to optimize the tour.

The host database is organized with information to be utilized by the plasmids. It contains *a priori* information and information obtained during the evolutionary process. The *a priori* information is a Hamiltonian cycle of G obtained with the LK algorithm of Applegate *et al*. [1]. The information of the evolutionary process is represented by the four best current solutions.

Two types of vectors were utilized: plasmids and transposons. Mimicking the biological process, the transgenetic algorithm employs the information obtained in the host context at the beginning of the evolutionary process. Then, as the population evolves, endogenous information is increasingly privileged. Interactions among the population of chromosomes and the transgenetic vectors occur while a stop criterion is not satisfied. At each iteration, a vector λ, plasmid or transposon, is chosen with a probability depends on the evolutionary process stage. Once the symbiogenesis metaphor suggests that the infiltration of host information is more useful at the initial steps of a symbiotic evolutionary process, the likelihood of choosing a given type of vector varies during the iterations. At the beginning, extra-cellular information is privileged. Thus, plasmids attacks are more likely to occur. At the end of the process, the probability of transposon attacks is higher. The counter *j* controls that tendency, being initialized on step 3 and updated on step 23. Its effect is determined on the comparison of step 8.

If a plasmid is selected, *k* vectors are generated. They are evaluated in accordance with a criterion that will be described further. Then, the best of the *k* plasmids is chosen to attack the chromosomes, as shown on step 11. To form the plasmid's string, one element of the environmental data base is randomly chosen as the source of information, according to a uniform distribution. The string length, *r*, is also chosen randomly in the interval [3, $\lfloor m/8 \rfloor$]. An initial point of the selected element is randomly chosen, then starting on that point, *r* successive markets form the vector's string. One can observe that a genetic fragment transported by a plasmid corresponds to a path (of markets). At each iteration where the plasmid is chosen as the manipulation vector, $k = 30$ plasmids are generated. Those plasmids are evaluated in order to choose one of them, the best one, to manipulate the chromosomes. Once their strings are partial solutions, they are evaluated with basis on the summation of three parcels:

1. The weights of the edges of the correspondent path

2. The lowest prices of the products available on some market of the string

3. The highest prices of the products not available in any market of the string

The plasmid with the lowest associated value is chosen to attack all chromosomes of the current population.

The pseudo-code of the procedure *attack_plas*() is shown on figure 4. The input parameters are chromosome *C*, and the chosen plasmid λ. First, the procedure verifies which markets are simultaneously in the chromosome and in the plasmid and remove such markets from the chromosome (step 1). The loop checks the insertion of the string between each pair of markets of the cycle represented on *C*, preserving the best insertion regarding the cost of the tour (steps 3-7). The solution cost after inserting λ's string in *C* in position *j* is set to variable *c*. If that cost is better than the best cost of the tested insertions, the best infiltration position is kept in variable *index*. The transcription is done in the best position, that is the one with the lowest tour cost (step 8), and the markets where no products are purchased are removed (step 9). The Lin and Kernighan procedure is called to optimize the tour.

```
1.  C'← remove(C, λ)
2.  c_best ← ∞; index ← 1
3.  for each possible infiltration position j in C
4.      c = cost_infilt(C, j, λ)
5.      if (c < c_best) then
6.          c ← c_best;   index ← j
7.  end_for
8.  C'' ← transcription (λ,C, index)
9.  C'' ← remove_empty_market(C'')
10. C'' ← Lin_Kern(C'')
11. if C'' is better than C' then C'← C''
12. return(C')
```

Figure 4. Pseudo-code of procedure *attack_plas*

Figure 5 illustrates the infiltration process of a plasmid. Observe that the rectangles of figure 5 represent the markets and the sequence of rectangles maps the sequence of visits of a given tour. The traced arrows show the positions where it is possible to infiltrate the plasmid's string. The resultant tour after the plasmid's manipulation are (4,3,1,2,6,5), (2,4,3,1,6,5), (2,6,4,3,1,5) and (2,6,5,4,3,1). The infiltration position which results on a chromosome with the lowest cost is finally chosen.

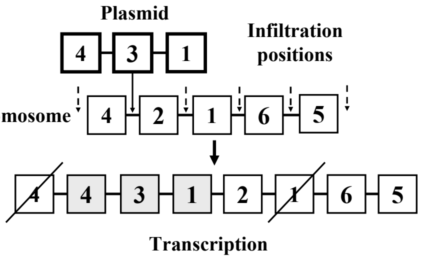

Figure 5. Transcription of a plasmid

The second vector utilized by the transgenetic algorithm is the transposon. The pseudo-code of procedure *attack_trans*() is shown on figure 6. The transposon's string is formed by indices that mark parcels of the chromosome to be rearranged by the vector's operator. In this work, this string is a pair indicating the initial and final positions of a sequence of markets in the chromosome. Those indices are randomly selected. A fragment of the chromosome is, then, determined by those two positions. The markets of that sequence are the elements of set X (step 1). The procedure verifies, iteratively, if each market of that sequence can be removed (steps 2-16). The market removal may result in a chromosome that represents an unfeasible solution. In this case, new markets will be added to the chromosome until feasibility is reached again (steps 7 and 12). That market addition is described straightforward. Let C be a chromosome and S be the set of markets that are not in C. A value is associated with each market of S. The value associated with a given market is calculated by the least increase its addition will bring to the tour cost of the solution represented in C plus the highest prices of the products that are still not purchased. The tour cost is calculated with the insertion of the considered market between two consecutive markets of C. All pairs of markets are considered. Given a chromosome, C, its tour cost, c, a pair of consecutive markets v_i, v_{i+1}, of C, and a market v_j not in C, the tour cost obtained with the insertion of v_j between v_i and v_{i+1} in C, is given by c plus the cost of edges $[i,j]$ and $[j,i+1]$ minus the cost of edge $[i,i+1]$. The resultant tour is optimized with the Lin and Kernighan procedure (steps 8 and 13). Markets where no products are purchased are removed (step 15). Each chromosome generated by a transposon's action is evaluated and the best configuration is preserved. If the manipulated chromosome is better than the original one concerning their fitness values, then the new individual replaces the old one in the population (steps 16 and 17 of figure 3). If the new individual represents a solution that is better than the best known solution, then the host database is updated with the inclusion of that chromosome and the removal of the worst one (steps 19 and 20 of figure 3).

```
1. X ← markets(C, λ)
2. for each market j ∈ X
3.     C'' ← C
4.     C'' ← remove_market( j, C'')
5.     C' ← remove_market( j+1, C'')
6.     if C'' is unfeasible then
7.         C'' ← include_new_markets(C'')
8.     C'' ← Lin_Kern(C'')
9.     if C'' is better than C then C← C''
10.    else
11.        if C' is unfeasible then
12.            C' ← include_new_markets(C')
13.        C' ← Lin_Kern(C')
14.        if C' is better than C then C← C'
15.    C ← remove_empty_market(C)
16. end_for
17. return(C)
```

Figure 6. Pseudo-code of the procedure *attack_trans*

4. COMPUTATIONAL EXPERIMENT

The purpose of the computational experiment was to evaluate the potential of the proposed evolutionary approach in tackling an NP-hard problem. The parameters $k = 30$, $\eta = 40$ and $\beta = 4$ were fixed after preliminary experiments. The tests were run on a Pentium IV 2.8 GHz, 512 MB of RAM, Ubuntu Linux operational system and gcc compiler.

In this experiment 89 instances with known optimal solutions and 51 instances where the optimal solution is not known were considered. Among the instances with known optimal solutions, *m* varies between 50 and 250, and *n* varies between 50 and 200.

A comparison of the transgenetic algorithm, TA, with the algorithms RL-SG [32] and BF [4] for those instances is shown in tables 2 and 3. Those results are presented in accordance with the format utilized in the papers of Boctor *et al*. [3] and Riera-Ledesma and Salazar-González [32]. The advantage of that format is that it allows examining the results in a compact table, since a great number of instances are tested. Classes of instances are defined by number of markets and products. Table 2(3) shows the results for the classes of instances grouped by number of markets (products). For example, in table 2 the column with *m* = 50 refers to all instances (with known optimal solution) with 50 markets. A number of independent runs are executed for each instance. Then the best solution found on those runs for each instance is kept. The result shown on that table is the average of the best solutions of the instances of a given class. Those results are shown in terms of percent difference from the optimal solution.

The results shown for TA correspond to 200 independent executions for each instance. The stopping criterion for the TA was to find the optimal solution or a maximum of 200 iterations.

The results shown by RL-SG and BF are reported in the works of Riera-Ledesma and Salazar-González [32] and Bountoux and Feillet [4], respectively. The runtimes of RL-SG were obtained on a PC Celeron 500 MHz, and BF ran on a Pentium IV 2 GHz.

Table 2. Results for instances grouped by number of markets

Method		***m***				
		50	100	150	200	250
RL-SG	Gap	0.07	0.14	0.03	0.32	0.06
	T(s)	3	10	14	19	25
BF	Gap	0	0	0.08	0.02	0.01
	T(s)	2	20	172	232	154
TA	Gap	0	0	0.01	0	0
	T(s)	4	25	44	43	64

Table 3. Results for instances grouped by number of products

Method		***n***			
		50	100	150	200
RL-SG	Gap	0.07	0.24	0.10	0.08
	T(s)	5	13	20	21
BF	Gap	0	0.05	0	0.03
	T(s)	37	154	96	165
TA	Gap	0	0	0	0.01
	T(s)	12	37	39	50

No significance statistical tests could be done in order to compare the three algorithms, once both algorithms, RL-SG and BF, could not be implemented in their original forms and the details of their computational tests are not fully available. Therefore, a comparison is done with the published results.

Among the nine groups of instances shown on tables 2 and 3, TA finds all the optimal solutions for seven of them and a percent gap of 0.01 for classes *m* = 150 and *n* = 200. RL-SG does not find gap zero for any instance class. BF finds all the optimal solutions of four classes. Regarding quality of solution, TA is superior to RL-SG in all groups of instances and is superior to BF in five groups. Those algorithms did not outperformed TA in any group of instances. BF outperforms RL-SG in eight of the nine groups.

Table 4. Results for instances with *m* < 300

m	***n***	**Id**	**BF**		**TA**	
			Sol	**T(s)**	**Sol**	**T(s)**
200	150	4	2419	1216.92	2419	23.98
200	200	4	2344	527.03	2344	99.19
250	100	1	1301	33.84	1301	143.19
250	100	4	1673	10.23	1673	3.55
250	100	5	1641	550.24	1641	1.84
250	150	4	1836	45.24	1836	2.27
250	150	5	1531	21.1	1531	151.43
250	200	2	**2785**	1137.65	2786	246.31
250	200	3	1924	281.88	1924	16.45
250	200	4	2116	83.83	2116	3.06
250	200	5	1797	930.03	1797	38.97

Table 5. Results for instances with *m* = 300

m	***n***	**Id**	**BF**		**TA**	
			Sol	**T(s)**	**Sol**	**T(s)**
300	50	1	1477	160	1477	1.5
300	50	2	813	116.01	813	1.41
300	50	3	1117	20	1117	1.46
300	50	4	1176	2.11	1176	1.44
300	50	5	1257	276	**1256**	1.57
300	100	1	1035	55.54	1035	2.29
300	100	2	**1179**	617.22	1180	3.98
300	100	3	1498	103.42	1498	2.25
300	100	4	1749	312.16	1749	37.49
300	100	5	1774	2.74	1774	2.27
300	150	1	1457	756.71	1457	98.66
300	150	2	1656	483.32	1656	3.02
300	150	3	2485	663.24	**2484**	6.34
300	150	4	1801	95.93	1801	8.17
300	150	5	1816	309.25	1816	41.16
300	200	1	1815	488.15	**1803**	575.39
300	200	2	1791	1918.52	**1790**	627.73
300	200	3	2442	2852.05	**2437**	184
300	200	4	1815	2946.79	1815	113.82
300	200	5	2022	1577.83	**2014**	605.39

Tables 4, 5 and 6 show a comparison of the TA and the BF for the remaining 51 instances where no optimal solution is known. The results of the RL-SG are not reported because it does not present better solutions than the other two algorithms for any instance. The results shown for TA correspond to the best solution found in

5 independent executions for each instance. The stopping criterion was a maximum of 200 iterations. The results shown by BF are reported by Bountoux and Feillet [4]. From those 51 instances, the proposed algorithm found new best solutions for 9 benchmark instances. BF reports the best known solutions for 4 benchmark instances. Both algorithms find the same best solutions for the remaining instances.

Table 6. Results for instances with *m* = 350

m	*n*	Id	BF		TA	
			Sol.	**T(s)**	**Sol.**	**T(s)**
350	50	1	723	46.04	723	1.7
350	50	2	736	25.71	736	13.02
350	50	3	942	6	942	1.82
350	50	4	805	379.39	805	5.01
350	50	5	**1125**	26.35	1225	1.67
350	100	1	1317	1698.48	1317	229.99
350	100	2	962	155.48	962	2.37
350	100	3	796	839.65	796	2.43
350	100	4	1059	13.94	1059	9.14
350	100	5	1566	464.86	1566	41.76
350	150	1	**1457**	1986.42	1459	319.67
350	150	2	1315	159.12	1315	16.31
350	150	3	2553	257.69	2558	597.74
350	150	4	1239	595.85	1239	3.06
350	150	5	2288	8.93	2288	229.27
350	200	1	1503	1033.39	**1498**	25.34
350	200	2	1374	3085.09	**1369**	56.07
350	200	3	1873	368.66	1873	59.05
350	200	4	1385	122.24	**1356**	32.88
350	200	5	2336	2385.65	2336	204.53

5. CONCLUSION

This paper presented an efficient evolutionary method that makes no use of crossover and mutation mechanisms in order to promote genetic information sharing or diversity in an evolutionary computing process. Inspired on the endosymbiotic serial theory and other intracellular interactions, the proposed approach is based upon powerful biological mechanisms that, in nature, constitute the most successful survival strategy of the living beings.

The survival strategy privileges the incorporation of information that comes from distinct types of biological agents such as chromosomes, plasmids and transposons. Intensification and diversification are obtained by means of the interaction of those agents. The idea of environment, here thought as a host's cytoplasm, is also considered, once the history of the evolutionary process can be utilized on the improvement of certain transgenetic vectors and, consequently, on the improvement of the population.

An example of application of the proposed approach is provided for the uncapacitated version of the Traveling Purchases Problem, an NP-hard optimization problem. The approach was very efficient in solving instances of a known benchmark. The proposed algorithm was compared with two other recent heuristics and shown a superior performance, exhibiting 9 new best solutions for the investigated instances.

Future works in transgenetic algorithms will consider conjugation mechanisms and implementations with the recombined plasmid. Transgenetic algorithms are being developed for the protein folding problem, the prize collecting Steiner tree problem, the multi-criteria minimum spanning tree and the multi-criteria traveling purchaser problem.

6. ACKNOWLEDGMENTS

This research was partially funded by CNPq and the program PRH-22 of the Petroleum National Agency.

7. REFERENCES

[1] Applegate, D., Bixby, R., Chvatal, V., and Cook, W. *Finding Tours in the TSP*. Technical Report TR99-05. Department of Computational and Applied Mathematics: Rice University; 1999.

[2] Barboza, A.O. *Simulação e Técnicas da Computação Evolucionária Aplicadas a Problemas de Programação Linear Inteira Mista*. Ph.D. Thesis, Universidade Tecnológica Federal do Paraná, 2005.

[3] Boctor F.F., Laporte, G., and Renaud, J. Heuristics for the traveling purchaser problem. *Computers and Operations Research* 30, 2003, 491–504.

[4] Bontoux, B., and Feillet, D. Ant colony optimization for the traveling purchaser problem. *Computers and Operations Research*; 2007, to be published.

[5] Bull, L., and Fogarty, T.C. Artificial symbiogenesis. *Artificial Life* 2, 1995, 269–292.

[6] Chan, T-M., Man, K-F., Tang, K-S., and Kwong, S.A. Jumping gene algorithm for multiobjective resource management in wideband CDMA. *Computer Journal* 48, 6, 2005, 749-768.

[7] De La Cal Marín, E.A., and Ramos, L.S. Supply estimation using coevolutionary genetic algorithms in the Spanish electrical market. *Applied Intelligence* 21, 2004, 7–24.

[8] Goldbarg, E.F.G., Castro, M.P., and Goldbarg, M.C. A transgenetic algorithm for the gas network pipe sizing problem. In *Computational Methods*, Liu, G.R., Tan, V.B.C., Han, X. (eds.), Springer, 2006, 893-904.

[9] Goldbarg, E.F.G., and Goldbarg, M.C. ProtoG: A computational transgenetic algorithm. In *Proceedings of the Metaheuristic International Conference* (MIC 2001), 2001, 625-630.

[10] Goldbarg, M.C., and Goldbarg, E.F.G. Extra-intracellular transgenetic algorithm. In *Late Breaking at the Genetic and Evolutionary Computation Conference* (GECC0 2001), AAAI, 2001, 115-121.

[11] Goldbarg, M.C., and Goldbarg, E.F.G. Transgenética computacional: Uma aplicação ao problema quadrático de alocação, *Pesquisa Operacional* 22, 3, 2002, 359-386.

[12] Goldbarg, M.C., Goldbarg, E.F.G., and Medeiros Neto, F.D. Piston pump mobile unity tour problem: An evolutionary view. In *Late Breaking at the Genetic and Evolutionary Computation Conference* (GECC0 2002), AAAI, 2002, 185-192.

[13] Goldbarg, M.C., Medeiros Neto, F.D., and Goldbarg, E.F.G. An evolutionary approach for the optimal configuration of cogeneration systems. In *Proceedings of the MCO'04 Fifth International Conference on Computer Sciences*, Hermes Science Publishing, 2004, 336-346.

[14] Gouvêa, E.F. *Transgenética Computacional: Um Estudo Algorítmico*. Ph.D. Thesis, Universidade Federal do Rio de Janeiro, 2001.

[15] Hillis, D.W. Co-evolving parasites improve simulated evolution in an optimization procedure. *Physica D* 42, 1999, 228–234.

[16] Husbands, P. Distributed coevolutionary genetic algorithms for multi-criteria and multi-constraint optimisation. In *AISB Workshop on Evolutionary Computing*, Fogarty, T. (ed.), Selected Papers, Springer-Verlag, Lecture Notes in Computer Science 865, 1994, 150–165.

[17] Jeon, K.W. *Prokaryotic Symbionts of Amoebae and Flagellates in The Prokaryotes*. In The Prokaryotes, A Handbook on the Biology of Bacteria, Volume 7: Proteobacteria: Delta, Epsilon Subclass, Dworkin, M., Falkow, S., Rosenberg, E., Schleifer, K-H., and Stackebrandt, E., (eds.), Springer New York, 2006.

[18] Joint, I., Downie, J.A., and Williams, P. Bacterial conversations: Talking, listening and eavesdropping. An introduction, *Philosophical Transactions of The Royal Society B***,** 2007, available at http://www.journals.royalsoc.ac.uk.

[19] Kim, J.Y., Kim, Y., and Kim, Y.K. An endosymbiotic evolutionary algorithm for optimization. *Applied Intelligence* 15, 2001, 117–130.

[20] Kim, Y.K., Park, K., and Ko, J. A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling. *Computers and Operations Research* 30, 8, 2003, 1151-1171.

[21] Kirby Jr., H. *Organisms Living on and in Protozoa*. In Protozoa in Biological Research, Calkins, G.N.; Summers, F.M. (eds.), Columbia Univ. Press, New York, 1941.

[22] Kubota, N., Shimojima, K., and Fukuda, T. Virus-evolutionary genetic algorithm -coevolution of planar grid model. In *Proceedings of the Fifth IEEE International Conference on Fuzzy Systems* (FUZZIEEE '96) 1, 1996, 8-11.

[23] Margulis, L. *Symbiosis as a Source of Evolutionary Innovation: Speciation and Morphogenesis*. The MIT Press, 1991.

[24] Michaelian, K. A symbiotic algorithm for finding the lowest energy isomers of large clusters and molecules. *Chemical Physics Letters* 293, 1998, 202-215.

[25] Muhlenbein, H., and Voigt, H-M. Gene pool recombination in genetic algorithms. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, Eshelman, L. (ed.), Morgan Kaufmann: San Mateo, 1995, 104-113.

[26] Nawa, N., Furuhashi, T., Hashiyama, T., and Uchikawa, Y. A study of the discovery of relevant fuzzy rules using pseudo-bacterial genetic algorithm. *IEEE Transactions on Industrial Electronics 46, 6,* 1999, 1080-1089.

[27] Pagie, L., and Mitchell, M. A comparison of evolutionary and co-evolutionary search. *International Journal of Computational Intelligence and Applications* 2, 1, 2002, 53-69.

[28] Pierce, S.K., Massey, S.E., Hanten, J.J., and Curtis, N.E. Horizontal transfer of functional nuclear genes between multicellular organisms. *Biological Bulletin* 204, 2003, 237-240.

[29] Ramesh, T. Traveling purchaser problem. *Opsearch* 18, 1981, 78–91.

[30] Ramos, I.C.O. *Abordagens utilizando a Metodologia Estatística na Busca de Soluções para o Problema do Caixeiro Viajante e na Avaliação de Performance de Algoritmos:Um Estudo Aplicado à Transgenética Computacional*, Ph.D. Thesis, Universidade Federal do Rio Grande do Norte, 2005.

[31] Ramos, I.C.O, Goldbarg, M.C., Goldbarg, E.F.G., and Dória Neto, A.D. Logistic regression for parameter tuning on an evolutionary algorithm. In *Proceedings of the Congress on Evolutionary Computation* (CEC 2005), Edinburg, 2005, 1061-1068.

[32] Riera-Ledesma, J., and Salazar-González, J.J. A heuristic approach for the traveling purchaser problem. *European Journal of Operational Research* 162, 2005, 142–152.

[33] Rosin, C.D., and Belew, R.K. New methods for competitive coevolution, *Evolutionary Computation* 5, 1, 1997, 1–29.

[34] Shapiro, J. A. Transposable elements as the key to a 21st century view of evolution, *Genetica* 107, 1999, 171-179.

[35] Simões, A., and Costa, E. On biologically inspired genetic operators: Transformation in the standard genetic algorithm, In *Proceedings of the Genetic and Evolutionary Computation Conference* (GECCO-2001), 2001a, 584-591.

[36] Simões, A., and Costa, E. An evolutionary approach to the zero/one knapsack problem: Testing ideas from biology. In *Proceedings of the Fifth International Conference on Neural Networks and Genetic Algorithms* (ICANNGA' 2001), Springer-Verlag, 2001b, 22-25.

[37] Tsujimura, Y., Mafune, Y., and Gen, M. Effects of symbiotic evolution in genetic algorithms for the job-shop scheduling. In *Proceedings of the 34 Hawaii International Conference on Systems Science* 3, 2001, 3026.

[38] Waters, C., and Bassler, B. Quorum sensing: Cell-to-cell communication in bacteria, *Annual Review of Cell and Developmental Biology* 21, 2005, 319-346.