On the Relativity in the Assessment of Blind Optimization Algorithms and the Problem-Algorithm Coevolution

Carlos D. Toledo-Suárez
Centro de Sistemas Inteligentes,
Instituto Tecnológico y de Estudios Superiores
de Monterrey, Campus Monterrey
Sucursal de Correos J, C.P. 64849
Monterrey, Nuevo León, México
ctoledo@exatec.itesm.mx

Manuel Valenzuela-Rendón
Centro de Sistemas Inteligentes,
Instituto Tecnológico y de Estudios Superiores
de Monterrey, Campus Monterrey
Sucursal de Correos J, C.P. 64849
Monterrey, Nuevo León, México
valenzuela@itesm.mx

Hugo Terashima-Marín
Centro de Sistemas Inteligentes,
Instituto Tecnológico y de Estudios Superiores
de Monterrey, Campus Monterrey
Sucursal de Correos J, C.P. 64849
Monterrey, Nuevo León, México
terashima@itesm.mx

Eduardo Uresti-Charre
Centro de Sistemas Inteligentes,
Instituto Tecnológico y de Estudios Superiores
de Monterrey, Campus Monterrey
Sucursal de Correos J, C.P. 64849
Monterrey, Nuevo León, México
euresti@itesm.mx

ABSTRACT

Considering as an optimization problem the one of knowing what is hard for a blind optimization algorithm, the usefulness of absolute algorithm-independent hardness measures is called into question, establishing as a working hypothesis the relativity in the assessment of blind search. The results of the implementation of an incremental coevolutionary algorithm for coevolving populations of tunings of a simple genetic algorithm and simulated annealing, random search and 20-bit problems are presented, showing how these results are related to two popular views of hardness for genetic search: deception and rugged fitness landscapes.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*Complexity Measures, Performance Measures*; I.2.8 [**Computing Methodologies**]: Artificial Intelligence—*Problem Solving, Control Methods, and Search*

General Terms

Algorithms, Experimentation, Measurement, Performance

Keywords

Blind search, algorithmic assessment, coevolution

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

1. INTRODUCTION

Following Wolpert and Macready's [16] notation, having a mapping $f : \mathcal{X} \rightarrow \mathcal{Y}$ from a finite set $\mathcal{X}$ with size $|\mathcal{X}|$ to a finite set of numerical values $\mathcal{Y}$ of size $|\mathcal{Y}|$, a combinatorial optimization problem consists of finding the member in $\mathcal{X}$ corresponding to the optimal in $\mathcal{Y}$. Letting

$$d_m \equiv \{(d_m^x(1), d_m^y(1)), \ldots, (d_m^x(m), d_m^y(m))\}$$

a sample of m pairs sorted as they are visited and $\mathcal{D}_m = (\mathcal{X} \times \mathcal{Y})^m$ the space of all m-size samples such that $d_m \in \mathcal{D}_m$ and the set of all possible samples with arbitrary size is $\mathcal{D} \equiv \cup_{m \geq 0} \mathcal{D}_m$.

A *blind optimization algorithm* a is defined as a mapping of sets from previously visited points to a new in $\mathcal{X}$ without including any extra information from the problem domain corresponding to f, this is

$$a : d \in \mathcal{D} \rightarrow \{x | x \in \mathcal{X}\}. \quad (1)$$

Defining $P(d_m^y | f, m, a)$ as the conditional probability to get d_m, Wolpert and Macready arrived at the conclusion that, for any pair of algorithms a_1 and a_2

$$\sum_f P(d_m^y | f, m, a_1) = \sum_f P(d_m^y | f, m, a_2) \quad (2)$$

suggesting that for any performance measure no algorithm is better than another when their performances are averaged on all possible discrete functions $\mathcal{F}$, under three conditions:

1. Algorithms do not visit members of $\mathcal{X}$ more than once: $a : d \in \mathcal{D} \rightarrow \{x | x \notin d^x\}$.
2. All mappings in $\mathcal{F}$ have $\mathcal{Y}$ as codomain: $\mathcal{F} = \mathcal{Y}^{\mathcal{X}}$.
3. The efficiency in search is not considered in the performance measurement.

Wolpert and Macready named this result the *No Free Lunch* (*NFL*) theorem, and together with the reactions it

raised, it has been the most important theoretical contribution concerning the assessment of optimization algorithms within the last years.

The ad hoc conditions that sustain the *NFL* theorem are polemic because identifying and/or attaining them in real world algorithmic implementations is impractical [15]. On the other hand the complexity resulting from ignoring these conditions and considering a more general theoretical approach to algorithmic assessment suggests the abandonment of the hope of getting a general analytic result of the kind of *NFL* theorem, as pointed humorously by Goldberg:

> The larger lesson here is the existence of what might be called the *NFL Theorem for Theorems in Complex Systems*. This theorem —really a conjecture— says that there is no free lunch with respect to theorem proving in complex systems science and engineering in the sense that it is not possible to say anything definitive or profound about a complex system without an appropriately complex proof [3, pp. 75,76]

Remembering that optimization algorithms appeared as feasible alternatives to problems that are analyticly hard to attack, is it possible to consider as an optimization problem the one of knowing what is hard for a blind optimization algorithm, without assuming the conditions sustaining the *NFL* theorem? The purpose of this article is to highlight the connection between the practical accomplishment of such an endeavor, the unfeasibility of getting meaningful absolute algorithm-independent hardness measures, and the possibility of achieving the coevolution of blind search algorithms and problems.

The remainder of the article is organized as follows: Section 2 shows the reasons for establishing the relativity in the assessment of blind search as a work hypothesis. Section 3 describes the implementation of an incremental coevolutionary algorithm (ICA) for coevolving populations of tunings of a simple genetic algorithm and simulated annealing, random search and 20-bit problems. Section 4 points the difference between our approach and previous related work. Section 5 shows how the results of the implementation are related to two popular views of hardness for genetic search: deception and rugged fitness landscapes. Section 6 concludes this article.

2. RELATIVITY IN THE ASSESSMENT OF BLIND OPTIMIZATION

For a finite set of numerical values $\mathcal{Z}$, suppose there is a mapping $w : \mathcal{F} \to \mathcal{Z}$ that assigns a difficulty evaluation to any problem f, therefore seen as an optimization problem it would allow finding easy or hard problems in $\mathcal{F}$. Is it possible to get a measure w independent of the algorithms used to optimize $\mathcal{F}$? Is it possible to find an absolute measure for the hardness of a problem? It is very improbable such a measure exists, because finding it would be a simple way of talking about $\mathcal{F}$'s complexity [3].

To modify the previous statement, w could be seen as the performance measure Φ for an algorithm a, so by optimizing $\mathcal{F}$ it could be possible to answer the question of knowing what is easy or hard for a. This view faces the difficulty that comparing the numerical performance measure of a blind search algorithm on two problems by itself does not say which one is harder. This suggests it is also improbable to find an absolute measure for what is easy or hard for a blind optimization algorithm alone.

PROOF. Focusing in the maximization case suppose d_m is a sample of algorithm a from problem f_1 whose global maximum is M_1 and e_m a sample of algorithm a from problem f_2 whose global maximum is M_2, then it is possible for $M_1 - \Phi(d_m^y) < M_2 - \Phi(e_m^y)$ to hold and at the same time $\Phi(e_m^y) > \Phi(f_m^y)$ if $M_2 > M_1$, what holds in the case $\mathcal{F}$ includes functions whose codomains are taken from $\mathcal{Y}$'s power set, a case more general than the one stated in the definition of *NFL* theorem, because for an authentic blind search M_1 nor M_2 values are known.

□

These difficulties lead us to adopt the next working hypothesis:

HYPOTHESIS 1. *Considered as an optimization problem, it is only possible to know what is easy or hard for a blind optimization algorithm in relation to the performance of another algorithm.* ***There is not an absolute way to assess a blind search.***

To understand this hypothesis take ${}^1d_m$ as a sample of algorithm a_1 and ${}^2d_m$ as a sample of algorithm a_2 both from problem f_1, ${}^1e_m$ as a sample of algorithm a_1 and ${}^2e_m$ as a sample of algorithm a_2 both from problem f_2, in the maximization case if $\Phi({}^1d_m^y) - \Phi({}^2d_m^y) > \Phi({}^1e_m^y) - \Phi({}^2e_m^y)$ holds it means problem f_1 was easier for a_1 than f_2 in relation to a_2, independently of the specific values of global maxima for both problems.

This optimization view for the assessment of algorithms does not present the restriction $a : d \in \mathcal{D} \to \{x | x \notin d^x\}$ sustaining *NFL* theorem, so it is possible to use a performance measure that takes into account the efficiency of a search process.

Best-found-so-far curve when algorithm a is faced to problem f is defined as

$$\Omega_a^f(i) = \frac{\sum_{j=1}^{n} \max_{k \le i} {}^j d_m^y(k)}{n} \tag{3}$$

where $i = 1, \ldots, m$ and ${}^j d_m^y$ is the j-th from n m-size samples. In few words this is a curve resulting from averaging the best-found-so-far value at position i, where $i = 1, 2, \ldots, m$, from n samples.

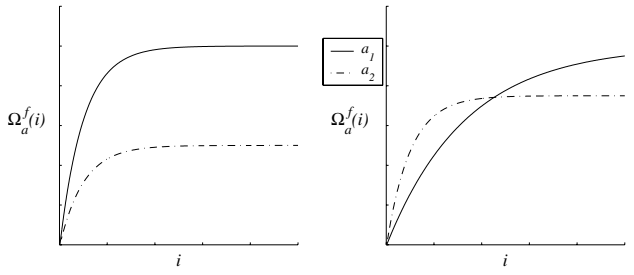

Figure 1: Examples of best-found-so-far curves for two algorithms a_1 and a_2 for n large.

The area below a best-found-so-far curve is a measure that, besides including the best value found in a sample,

comprises the temporal evolution of the search, its efficiency, and since it is built using many samples it reduces fortuitous effects during the measurement of the performance of an algorithm. The performance measure to maximize for algorithm a_1 in relation to algorithm a_2 in the implementation to show in the next sections of this article is:

$$\sum_{i=1}^{m} \left(\Omega^f_{a_1}(i) - \Omega^f_{a_2}(i) \right), \qquad f \in \mathcal{F}, \tag{4}$$

which is the difference between the areas below their best-found-so-far curves.

A problem whose measure (4) is greater than zero tends to benefit algorithm a_1 over a_2, so for easy reference it will be called *tendentious problem*.

3. INCREMENTAL COEVOLUTION

Suppose an algorithm that brings as output a tendentious problem is implemented. Someone could argue he cannot object the search process but that the performance measure was skewed, because of the way algorithms were tuned. Since in practice most algorithms count with a set of parameters that determine their behavior, we face the additional optimization problem of finding those tunings bringing the best ones. Is there any algorithm capable of including the optimization of parameters together with the performance measure (4)?

One of the main motivations for the work presented in this article was answering this question affirmatively through evolutionary computation, specificly implementing an incremental coevolutionary algorithm (ICA)[11]. A coevolutionary algorithm seems a natural answer for the multiple optimization problem stated above because it implies the competition between populations of problems and tunings, in which as time passes members from those populations become better contenders. Specificly there are three populations involved: 1) a population of a_1's tunings, 2) a population of a_2's tunings, and 3) a population of problems. In the case we search for problems making a_1 look better, the relation between populations 1) and 2) and that between 2) and 3) will be competitive, whereas it will be cooperative between 1) and 3).

ICA's distinctive features are:

- It is non-generational, with populations acting as a memory that helps them to avoid oscillating around previously visited zones.
- Individuals in each population compete against a random sample of individuals from the oponent populations.
- The fitness of every individual at time t is an explicit measure whose adjustement is inspired in the one of Learning Classifier Systems [4, 12]. Taking S as the fitness of an invividual, its adjustement for time $t+1$ is made through the equation

  $$S(t+1) = S(t) + \text{Reward} - \text{Taxation}.$$

  The evaluation of the reward reflects the comparison against the competitors and, together with the taxation, is calculated in such a way it is bounded, making the fittest individuals tend to a stable state, there

(**Initialize populations**)
Generate random $\mathbb{A}$ —population of a_1's tunings—
Generate random $\mathbb{B}$ —population of a_2's tunings—
Generate random $\mathbb{P}$ —population of problems—

(**Initialize fitness**)
for-each $A \in \mathbb{A}$
  $S_A \leftarrow (1/10)(R_{\mathbb{A}}/T_{\mathbb{A}})$
for-each $B \in \mathbb{B}$
  $S_B \leftarrow (1/10)(R_{\mathbb{B}}/T_{\mathbb{B}})$
for-each $P \in \mathbb{P}$
  $S_P \leftarrow (1/10)(R_{\mathbb{P}}/T_{\mathbb{P}})$

(**Main cycle**)
repeat
  (**Competition cycle**)
  for $c \leftarrow 1$ **to** $\#_c$
    $A \leftarrow$ tuning selected randomly from $\mathbb{A}$
    $B \leftarrow$ tuning selected randomly from $\mathbb{B}$
    $P \leftarrow$ tuning selected randomly from $\mathbb{P}$
    Calculate $\Omega^P_{a_1}$ with n m-size samples of a_1 tuned by A running on P
    Calculate $\Omega^P_{a_2}$ with n m-size samples of a_2 tuned by B running on P
    $E_A \leftarrow (1/m)\sum_{i=1}^{m} \Omega^P_{a_1}(i)$
    $E_B \leftarrow (1/m)\sum_{i=1}^{m} \Omega^P_{a_2}(i)$
    $S_A \leftarrow R_{\mathbb{A}} \tanh(2E_A) \tanh(10\ T_{\mathbb{B}} S_B / R_{\mathbb{B}}) + (1 - T_{\mathbb{A}}) S_A$
    $S_B \leftarrow R_{\mathbb{B}} \tanh(2E_B) \tanh(10\ T_{\mathbb{A}} S_A / R_{\mathbb{A}}) + (1 - T_{\mathbb{B}}) S_B$
    $S_P \leftarrow R_{\mathbb{P}}(1/2)(1 + \tanh(20\ (E_A - E_B))) + (1 - T_{\mathbb{P}}) S_P$
  end-for c

  (**One step of a GA in* $\mathbb{A}$*)
  Select parents (A_1 and A_2) proportionally to S_A
  Create A_0 applying *crossover* with probability $cp_{\mathbb{A}}$
  $S_{A_0} \leftarrow (S_{A_1} + S_{A_2})/2$
  Replace tuning with worst fitness by A_0
  Apply *mutation* to $\mathbb{A}$ with small probability $mp_{\mathbb{A}}$

  (**One step of a GA in* $\mathbb{B}$*)
  Select parents (B_1 and B_2) proportionally to S_B
  Create B_0 applying *crossover* with probability $cp_{\mathbb{B}}$
  $S_{B_0} \leftarrow (S_{B_1} + S_{B_2})/2$
  Replace tuning with worst fitness by B_0
  Apply *mutation* to $\mathbb{B}$ with small probability $mp_{\mathbb{B}}$

  (**One step of a GA in* $\mathbb{P}$*)
  Select parents (P_1 and P_2) proportionally to S_P
  Create P_0 applying *crossover* with probability $cp_{\mathbb{P}}$
  $S_{P_0} \leftarrow (S_{P_1} + S_{P_2})/2$
  Replace problem with worst fitness by P_0
  Apply *mutation* to $\mathbb{P}$ with small probability $mp_{\mathbb{P}}$

until termination criteria met

Table 1: Incremental coevolutionary algorithm.

comes some t when for their fitnesses $S(t+1) \approx S(t)$ holds.

Table 1 shows the steps for the implementation of an ICA with a population $\mathbb{A}$ of a_1's tunings, $\mathbb{B}$ of a_2's tunings and $\mathbb{P}$ of problems —suppose we search for a tendentious problem that benefits a_1 over a_2. $R_{\mathbb{A}}, R_{\mathbb{B}}$ and $R_{\mathbb{P}}$ represent the maximum rewards given to individuals in each population according to their performance when competing, and $T_{\mathbb{A}}, T_{\mathbb{B}}$ and $T_{\mathbb{P}}$ their maximum taxations. E_X is the area below the best-found-so-far curve for algorithm whose tuning is X divided by m —the size of samples— and S_X is the fitness of tuning X involved in competition, where $X = \{A, B\}$.

The phenotype of every problem is the result of multiplying two functions g and h, such that $f = g(\mathcal{X}_1) \times h(\mathcal{X}_2)$, where g and h are both codified as follows by a couple of vectors r and θ that constitute its genotype:

$$\gamma = -\tan^{-1}\left[\sum_{i=1}^{N} r_i \cos\left(\sum_{j=1}^{i}\sum_{l=1}^{j}\theta_l\right) \Big/ \sum_{i=1}^{N} r_i \sin\left(\sum_{j=1}^{i}\sum_{l=1}^{j}\theta_l\right)\right], \tag{5}$$

$$g_k = \rho \sum_{i=1}^{k} r_i \sin\left(\gamma + \sum_{j=1}^{i}\sum_{l=1}^{j}\theta_l\right), \qquad x_k = k, \tag{6}$$

where $0 < r_i \leq 1$ and $-\alpha \leq \theta_i \leq \alpha$ ($\alpha > 0$) are real numbers vectors of N size and $\rho = 1/(\max g - \min g)$. Figure 2 shows an example of the way these equations act upon r and θ.

For the implementation made $N = |\mathcal{X}_1| = |\mathcal{X}_2| = 10^3$, which means f is a surface with 10^6 points approximately equivalent to a 20-bit search space, where the first ten point toward a position on $\mathcal{X}_1$ axis and the next ten point a position on $\mathcal{X}_2$.

Algorithms whose tunings form populations $\mathbb{A}$ and $\mathbb{B}$ are a simple generational genetic algorithm (GA) with tournament selection and simulated annealing (SA). Tunings forming populations A and B are 32-bit binary vectors codifying the possible values for their parameters, as shown by Table 2 and 3.

Crossover operator [2, 5] is used in three ways in the implementation:

- In the ICA to create a new P .
- In the ICA to create new A and B.
- In the GA involved in the coevolution.

Parameter	Bits	Range
Population size	[1 , 8]	[50 , 250]
Tournament size	[9 , 16]	[2 , 10]
Crossover probability	[17 , 24]	[0.75 , 1]
Mutation probability	[25 , 32]	[0 , 0.1]

Table 2: Coding of GA's tuning.

Calling *allele* every member in the alphabet of a genotype, and *chromosome* a vector of alleles in which single point crossover is applied, every individual in $\mathbb{P}$ is made of four chromosomes with real alleles, two for the r and θ parts of g and two for the r and θ parts of h. Every individual in $\mathbb{A}$ and $\mathbb{B}$ is made by a binary 32-bit chromosome, whereas every individual in the GA is made by two binary 10-bit chromosomes, one for $\mathcal{X}_1$ and another for $\mathcal{X}_2$.

Parameter	Bits	Range
Markov chain length	[1 , 10]	[5 , 25]
Initial temperature	[11 , 20]	[10^3 , 10^6]
Temperature decrement constant	[21 , 32]	[0.75 , 0.99]

Table 3: Coding of SA's tuning.

In the case of SA every new neighbor is generated by changing a bit in $\mathcal{X}_1$ and another in $\mathcal{X}_2$ chosen randomly. ICA was run also with a random search (RS) taking the place of a_1, consisting in sampling uniformly pairs of positions on $\mathcal{X}_1$ and $\mathcal{X}_2$. Because in this case its population of tunings is virtual, the fitness' updating of tunings in $\mathbb{B}$ is accomplished by

$$S_B \leftarrow R_{\mathbb{B}} \tanh(2E_B) + (1 - T_{\mathbb{B}})S_B.$$

4. RELATION TO PREVIOUS WORK

Genetic algorithms have been previously proposed as test case generators for benchmarking purposes. These proposals differ from ours in three main points:

- The test cases evolved are instances of problems for which information about bounds on the best possible performance is included in the genetic algorithm, allowing to measure the fitness of instances according to the performance of a single algorithm [8, 6, 13].
- When fitness is measured as the difference of performance between two algorithms, distance to global optima is used as stop criteria for search [10, 9].
- The need of evolving the tunings of algorithms, as a way of avoiding a possible bias in the search of test cases and as a first approach to the evolution of algorithms, is absent in all previous work. Ignoring this need is a way to leave an open opportunity for someone to argue that the results you get depend mainly on the way you tune algorithms.

5. RESULTS

Table 4 shows the parameters used in the implementation in all cases.

The length of samples was determined in such a way that divided by the size of every problem brings a fraction — 2×10^{-3}— that is difficult to reach in practice, where search spaces tend to be astronomical in size, with affordable execution time. The algorithm was written in C++ and executed in a personal computer with Pentium(R)4 CPU, 2.4GHz, 448 MB RAM. Average execution time was approximately two hours.

Figures 3 to 6 show the best found so far curves obtained using the fittest tunings and the graph for the best tendentious problem where GA beats SA, SA beats GA, RS beats GA, and RS beats SA . Measure $(E_A - E_B)/Optimum$ is the fraction of the area below the optimum occupied by the difference between the area below a_1's curve minus a_2's. Five thousand samples were employed to plot the curves.

Problems where SA won are differ from those where it lost mainly by allowing it to travel longer distances before finding

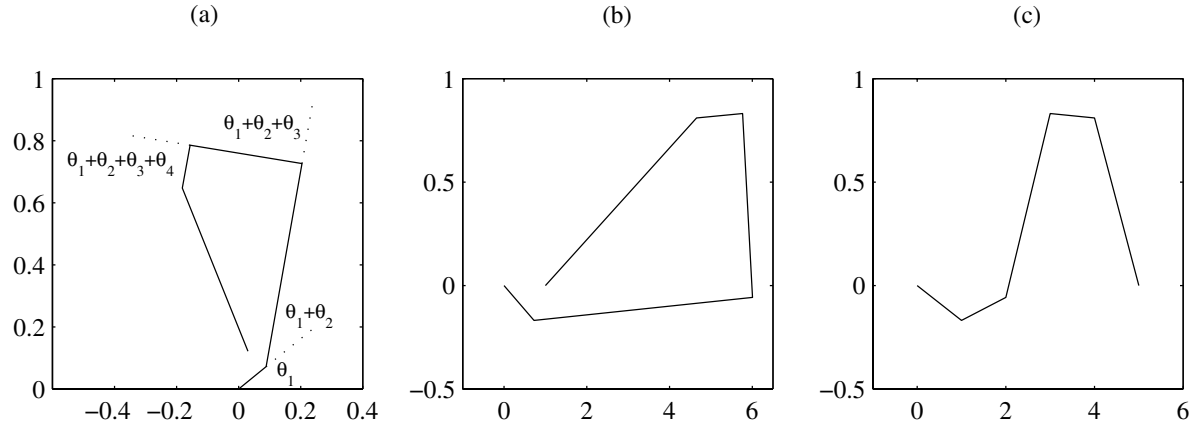

Figure 2: Example of the decoding of functions from θ and r, for N=6: (a) the length of segment i is r_i and $\sum_{j=1}^{i} \theta_j$ is the angle between $i-1$ and i , (b) rotation, (c) scaling and reordering of domain, getting a 1 to 1 function.

Parameter	Description	Value
m	Length of samples	2000
n	Amount of samples to calculate E_A y E_B	10
$\#_c$	Number of competitions per iterarion	50
$\|\mathbb{P}\|$	Size of the population of problems	100
$\|\mathbb{A}\| = \|\mathbb{B}\|$	Size of population of tunings	25
$cp_{\mathbb{P}}$	Crossover probability for population of problems	1
$cp_{\mathbb{A}} = cp_{\mathbb{P}}$	Crossover probability for populations of tunings	1
$mp_{\mathbb{P}}$	Mutation probability for population of problems	0.05
$mp_{\mathbb{A}} = mp_{\mathbb{P}}$	Mutation probability for population of tunings	0.05
$T_{\mathbb{P}}$	Maximum taxation for problems	0.01
$T_{\mathbb{A}} = T_{\mathbb{B}}$	Maximum taxation for tunings	0.01
$R_{\mathbb{P}}$	Maximum reward for problems	0.1
$R_{\mathbb{A}} = R_{\mathbb{B}}$	Maximum reward for tunings	1

Table 4: ICA's parameters used in all cases.

local optima. Problems where SA lost are characterized by having their local optima more distributed, with increasing values as you approach to the global optimum in Hamming space. In these kind of problems both GA and RS have the advantage of sampling them without stopping, whereas SA has to decide more often if continuing after finding local optima.

It was found that all the tendentios problems where GA was involved are deceptive in the highest possible order: 20 [14]. Variables that deception theory considers relevant to measure the hardness for a genetic search, including the amount and order of deceptions, variances and scaling in the evaluation of highest order schemata [3], do not seem to give determinant information to justify the cases where the GA was beaten.

The only variable that was confirmed to be relevant was high multimodality, as the flatness in problems depicted in Figure 4 and 5 testifies. This property showed up in all the problems where the GA was beaten, and calls into question the usefulness of associating ruggedness to GA-hardness when assuming our working hypothesis.

Rugged fitness landscapes [7] —see Figure 7— have been previously called into question within biology as an adequate model for speciation, based on the assertion that the birth of a new species should not necessarily imply a temporal reduction in fitness. How can a population evolve from a converged local peak to another through an adaptive valley without deleterious effects?

As an alternative *holey* landscapes [1] have been proposed, consisting in flat multidimensional functions with unitary fitness and multiple holes with null fitness. What this apparently trivial model aims to represent is that it is not proper to assume different species having different aptitudes but simply that there are good combinations of genes corresponding to apt individuals —fitness 1— and bad ones corresponding to non apt individuals —fitness 0, letting genetic drift to perform the exploration.

An important prediction made by holey landscapes is that viable genotypes tend to group, to be connected by chains of small genetic steps. These groupings are called *connected components*.

As can be seen in Figure 4 and 5 problems where GA was beaten resemble holey landscapes. For the problem where SA won the 351 global optima form a connected component. What this comparison suggests is that a way to beat a GA is to ensure genetic drift takes over the search process.

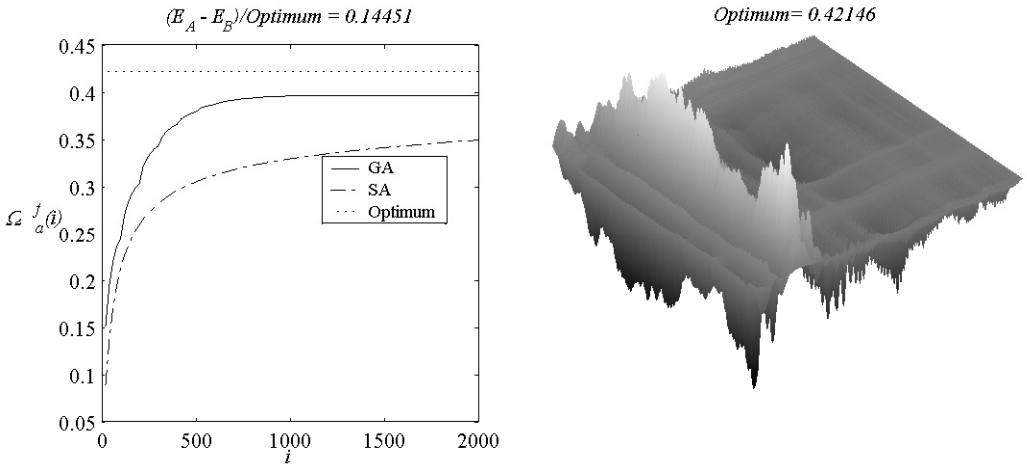

Figure 3: Best-found-so-far curves obtained using fittest tunings and the graph for the best tendentious problem where GA beats simulated annealing.

6. CONCLUSIONS

The results presented in the last section are an evidence of the connection existent between the accomplishment of the coevolution of optimization algorithms and problems, and assuming the relativity in the assessment of blind search.

The discrepancies found between the analysis of the tendentious problems where the GA was involved and popular paradigms about GA hardness highlights the difference implied by this assumption. The fact that the order of deception and ruggedness did not show to be determinant features in problems where a local search algorithm (SA) performed better than GA, calls into question their significance as determinant measures of algorithmic competence. Should the concept of "GA-competence" be re-formulated in the light of the working hypothesis introduced by this paper? This remains as an open question.

The sacrifice implied by turning the problem of knowing what is easy or difficult for an algorithm into an optimization problem is that it cannot have the status that a mathematical proof could have, but it is very probable such a demostration will never be found. The approach of this conversion is impartial since problems are generated by an evolutionary process and carries an implicit weighing of the efficiency of an algorithm.

Main objections to this approach may be focused on its implementation, but the fact that it was possible to find tendentious problems using the one depicted shows its sufficiency.

On the contrary to NFL theorem in the depicted work assertions about an infinitude of problems are not made, but an impartial method to find tendentious problems is introduced.

7. ACKNOWLEDGMENTS

This research was supported by ITESM under the Research Chair CAT-010.

8. REFERENCES

[1] S. Gavrilets. Evolution and speciation on holey adaptive landscapes. *Trends Ecol. Evol.*, 12(8):307–312, August 1997.

[2] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, Reading, MA, 1989.

[3] D. E. Goldberg. *The Design of Innovation. Lessons from and for Competent Genetic Algorithms*. Kluwer Academic Publishers, Boston, 2002.

[4] J. Holland. *Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems*. Morgan Kauffman, San Mateo, CA, 1986.

[5] J. H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, Ann Arbor, 1975.

[6] E. L. Johnson. Genetic algorithms as algorithm adversaries. In L. Spector et al., editors, *Genetic and Evolutionary Computation Conference*, pages 1314–1321. Morgan Kaufmann, 2001.

[7] S. A. Kauffman. *The Origins of Order. Self-Organization and Selection in Evolution*. Oxford University Press, New York, 1993.

[8] A. P. Kosoresow and M. P. Johnson. Finding worst-case instances of, and lower bounds for, online algorithms using genetic a lgorithms. In R. I. McKay and J. Slaney, editors, *Advances in Artificial Intelligence*, volume 2557 of *LNAI*, pages 344–355. Springer, 2002.

[9] W. B. Langdon and R. Poli. Evolving problems to learn about particle swarm and other optimisers. In *Congress on Evolutionary Computation*, pages 81–88. IEEE, 2005.

[10] W. B. Langdon, R. Poli, O. Holland, and T. Krink. Understanding particle swarm optimisation by

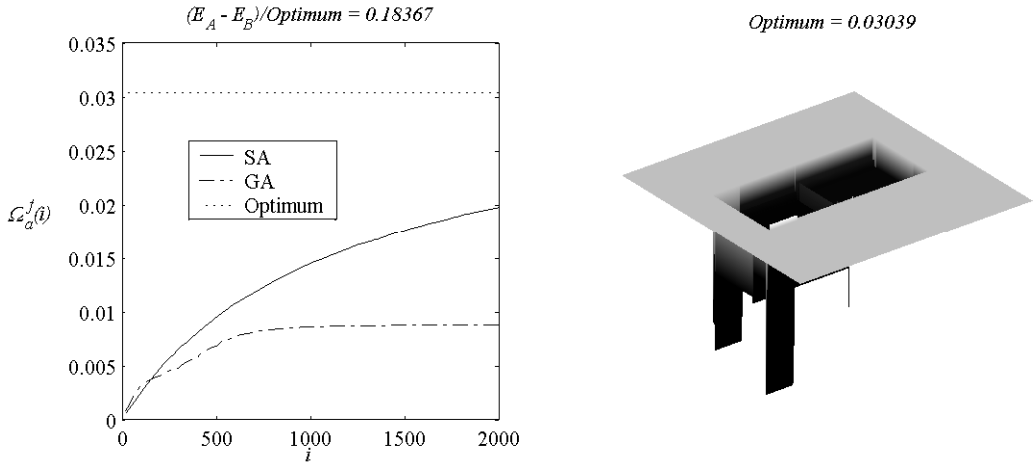

Figure 4: Best-found-so-far curves obtained using fittest tunings and the graph for the best tendentious problem where simulated annealing beats GA. The problem has 351 global optima.

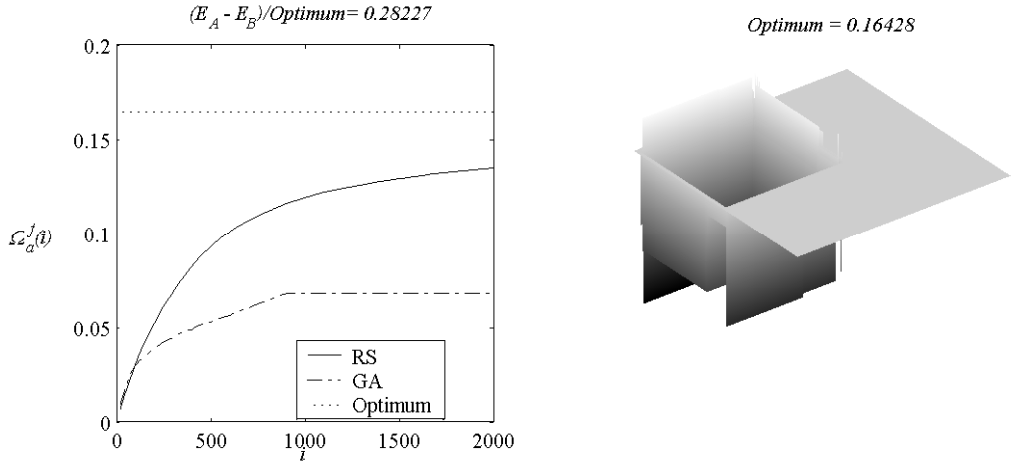

Figure 5: Best-found-so-far curves obtained using fittest tunings and the graph for the best tendentious problem where random search beats GA. The problem has two global optima.

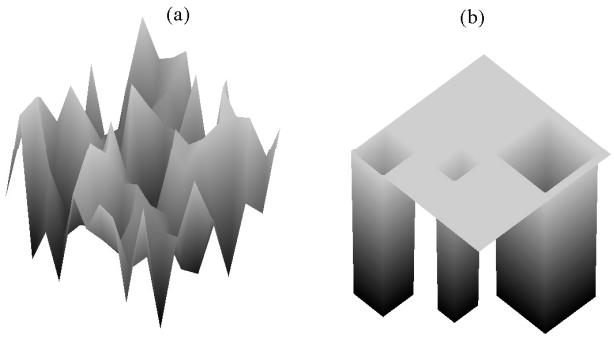

Figure 7: Comparison between a rugged landscape (a) and a holey landscape (b).

evolving problem landscapes. In *Swarm Intelligence Symposium*, pages 30–37. IEEE, 2005.

[11] R. A. Palacios-Durazo and M. Valenzuela-Rendón. Similarities between co-evolution and learning classifier systems. In *Genetic and Evolutionary Computation Conference GECCO (2004)*, pages 561–572, 2004.

[12] M. Valenzuela-Rendón and E. Uresti-Charre. A nongenerational genetic algorithm for multiobjective optimization. In *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 658–665. Morgan Kauffman, 1997.

[13] J. van Hemert. Evolving combinatorial problem instances that are difficult to solve. *Evolutionary Computation*, pages 433–462, 2006.

[14] D. Whitley. Fundamental principles of deception in genetic search. In G. J. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 221–241. Morgan Kaufmann, San Mateo, CA, 1991.

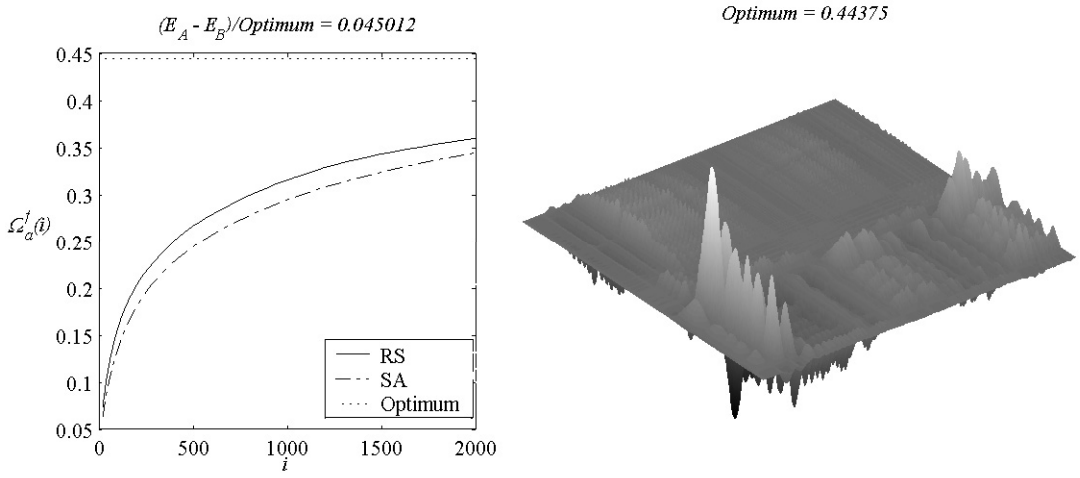

Figure 6: Best-found-so-far curves obtained using fittest tunings and the graph for the best tendentious problem where random search beats simulated annealing.

[15] D. Whitley and J. Watson. *A 'No Free Lunch' Tutorial*. Department of Computer-Science, Colorado State University, Fort Collins Colorado, 2004.

[16] D. H. Wolpert and W. G. Macready. No Free Lunch Theorems for Search. Technical Report SFI-TR-95-02-010, Santa Fe Institute, Santa Fe, NM, 1995.