Towards a Self-Stopping Evolutionary Algorithm Using Coupling From The Past

German Hernandez
Department of Computer and Systems Engineering
National University of Colombia
Bogota, Colombia
gjhernandezp@unal.edu.co

Kenneth Wilder
Department of Statistics
The University of Chicago
Chicago, IL 60637
wilder@galton.chicago.edu

Fernando Nino
Department of Computer and Systems Engineering
National University of Colombia
Bogota, Colombia
lfninov@unal.edu.co

Julian Garcia
Department of Computer and Systems Engineering
National University of Colombia
Bogota, Colombia
julian.garcia@ieee.org

ABSTRACT

In this paper a stopping criterion for a particular class of evolutionary algorithms is devised. First, a model of a generic evolutionary algorithm using iterated random maps is presented. The model allows the exploration of a connection between coupling from the past, and a stopping criterion for evolutionary algorithms. Accordingly, a method to stop a generic evolutionary algorithm is proposed. Some computational experiments are carried out to test the stopping criterion, using a modified version of coupling from the past. Empirical evidence is shown to support the suitability of the criterion.

Categories and Subject Descriptors

I.2.2 [**Artificial Intelligence**]: Automatic Programming—*Automatic analysis of algorithms*; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search —*Heuristic methods*

General Terms

Algorithms, Experimentation

Keywords

Evolutionary Algorithms, Stopping Criteria, Coupling from the Past, Optimization

1. INTRODUCTION

One of the main concerns of the evolutionary computation community is to find methods to determine when an evolutionary algorithm (EA) has found a good solution. Nevertheless, despite the wide range of applications of such algorithms, "there are few theoretical guidelines for determining when to terminate the search" [1].

In recent years, a lot of effort has been devoted to ground a theoretical basis for evolutionary computation. Particularly, significant efforts have been undertaken to analyse the long-term dynamics of evolutionary algorithms [4, 8]. Such studies have mainly focused on convergence theorems, using stochastic processes such as Markov chains[7]. However, work still needs to be done in order to determine how long it will take to an EA to find a good solution.

In general, an EA can be viewed as a Markov chain (MC) on the space of populations [4, 8]. Thus, it can be shown that, under certain conditions, there exists a stationary distribution that characterises the long-term behaviour of the EA. Once stationarity has been reached, the dynamics of the EA may be thought of as a sampling procedure from the stationary distribution. Particularly, it has been proved that, for a Generic Evolutionary Algorithm (GEA)[4], this distribution is supported on the set of populations whose individuals are optimizers of the fitness function. Accordingly, a straightforward stopping criterion for a GEA could be to check whether the equilibrium distribution has been reached or not. In this work a sampling procedure (Coupling from the Past[10, 9]), based on Montecarlo Markov Chains (MCMC) samplers will be used for this purpose.

In this work, a stopping criterion, for a particular class of EAs termed GEA's [4], will be proposed. Such algorithms will be regarded as MCMC samplers on the space of populations. It has been shown that the equilibrium distribution associated to a GEA, is supported on the set of populations made up of optimizers of the fitness function. Consequently, an interesting idea to explore in order to devise a self-stopping GEA, is to sample the associated equilibrium distribution using CFTP. The GEA should stop once it reaches stationarity, i.e., once the underlying population is made up of optimal individuals.

In previous work, a combinatorial approach established bounds on the number of generations required by an EA to explore all possible populations[1]. In contrast, this work uses a probabilistic approach based on sampling techniques,that allows a GEA to stop itself.

The rest of this paper is organized as follows. In section 2 some basics on coupling from the past are presented, and some fundamentals on iterated random maps are summarized. In section 3, a generic evolutionary algorithm is modeled by iterated random maps. In section 4, a self-stopping generic evolutionary algorithm, using CFTP, is presented. The results of some experiments carried out to test the proposed stopping criterion are reported in section 5. Finally, some conclusions are devised in section 6.

2. PRELIMINARIES

2.1 Coupling from the Past

MCMC samplers are intended to generate samples from an arbitrary probability distribution. The keystone of MCMC methods is to build a MC whose equilibrium distribution is the target distribution[5]. Accordingly, to sample an arbitrary distribution using an MCMC sampler, it is necessary to wait for an unknown *"burn-in"* time, until the equilibrium distribution is reached. Once the *"burn-in"* time has passed, samples can be considered to follow approximately the target distribution[5].

Unfortunately, in practice, there is no certainty on what the "burn-in" period should be. However, Coupling from the Past (CFTP) addresses this problem; this technique can be used to determine the number of iterations a MCMC sampler needs to reach stationarity.

A MCMC sampler will have reached stationarity in infinite time, thus, if you started running the chain at time $t = -\infty$, the chain would reach equilibrium at time zero. Propp and Wilson[10] [6] [3] proved that using couplings, it is not necessary to run the chain from $t = -\infty$, but from a finite (but stochastic) time in the past. They developed an algorithm called "*Coupling From The Past* (CFTP)" which determines such time, allowing to obtain perfect samples from a MCMC sampler.

Coupling is a probabilistic technique that defines two or more random processes jointly on a common probability space, and based on their realizations, draws conclusions about their distributions. In order to simulate a finite MC, it is convenient to consider a function $\phi(X_n, u)$, which yields the next state of the chain given the present state X_n, using a uniform random number u in $[0, 1]$ to simulate the transition. Such function is termed *update function.* CFTP uses coupling by running several Markov chains using the same random number in the update function that determines the next state of the chains. Hence, the chains are said to be coupled since their transitions are being produced by the same sequence of random numbers. Rather than running the coupled chains from the present to the future, the CFTP algorithm runs the chains from a distant point in the past to the present, but that point in the past is at a finite distance, which is determined by the algorithm itself.

The CFTP algorithm works as follows: a MC for each possible state is run starting at time $-t$ in the past; the chains are run forward to time zero, using the same random number in the update function for all the chains at each time step. Clearly, if two chains end up in the same state at the same time, they will run together from that time on. When this happens, the chains are said to have coalesced[5]. Thus, the chains are started at all possible states, and all of the states have coalesced by time zero, the observations from time zero on will follow the stationary distribution[10]. If the chains have not coalesced by time zero, then, the algorithm runs the chains starting at an earlier time (e.g., $-2t$). Hence, the "burn-in" time is determined during the run of the algorithm itself.

For the sake of clarity, a simple CFTP example will be shown next(see figure 1). Consider a MC with state space $\{0, 1, 2, 3\}$, and transition probabilities $p_{ij} = \frac{1}{4}$ for $i, j = 0, ..., 3$. The MC has a uniform equilibrium distribution on $\{0, 1, 2, 3\}$, which will be sampled using CFTP. First, a transition for each initial state is simulated starting at time $t = -1$ (see Figure 1(a)). Given that the realizations of the MC's do not coalesce, the algorithm goes back in time and starts at $t = -2$, and simulates again a transition for each initial state, which in turn is composed with the transition simulated for $t = -1$ (see Figure 1(b)); since there is no coalescence again, the algorithm goes back in time and starts now from $t = -3$, in this case the composition yields a coalesced state at time $t = -0$, then it is guaranteed that from this point on the samples may be considered to be perfect (see Figure 1(c)).

2.2 Iterated Random Maps

An *iterated random map* (IRM) is a discrete dynamical system governed by a collection of functions chosen randomly in accordance to some probability distribution. Although such probability distribution may depend on the state of the system, here, it will be considered that the probability distribution is the same for all states.

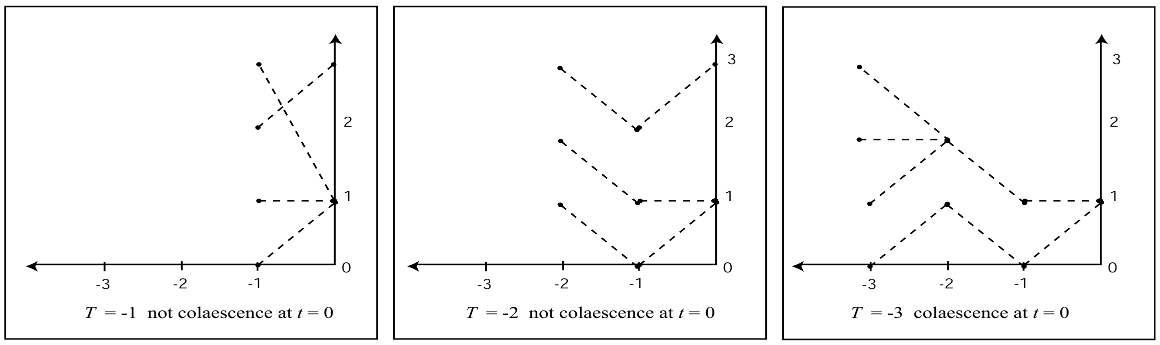
In this work, an IRM will be denoted as

$$\{X; f_1, f_2, ..., f_k; p_1, p_2, ..., p_k\},$$

where X denotes the state space, f_i are functions defined on X, and p_i is a probability associated to the function f_i, such that $p_1 + p_2 + ... + p_k = 1$. Accordingly, an IRM defines a random discrete dynamical system $\langle X, F \rangle$, where

$$F(\mathbf{x}) = \begin{cases} f_1(x) & \text{with probability} & p_1 \\ f_2(x) & \text{w.p.} & p_2 \\ \vdots & \vdots & \\ f_k(x) & \text{w.p.} & p_k. \end{cases}$$

Iterated random maps offer a method to represent and study a MC in terms of its steady state distribution [2]. Intuitively, a MC on the state space X can be constructed by applying iterating random functions on X. Therefore, the dynamics of the system is given by the composition of the functions f_i, randomly chosen according to μ, i.e., $X_0 = x_0, X_1 = f_{i_1}(x_0), X_2 = (f_{i_1} \circ f_{i_2})(x_1), ...$, where the values i_j are independent draws from the probability distribution μ[2].



-3
-2
-1
0
1
2
T = -1 not colaescence at t = 0
(a)
3
T = -2 not colaescence at t = 0
(b)
T = -3 colaescence at t = 0
(c)

Figure 1: A simple CFTP example

3. GENERIC EVOLUTIONARY ALGORITHMS AS ITERATED RANDOM MAPS

An EA can be thought of as a random dynamical system that describes the changes of a population, under the action of operations that emulate natural evolution (selection and variation) throughout time. The state space of such system is the set of all possible populations $\mathbb{P}$, and its dynamics is defined by a stochastic transition operator E. A population in $\mathbb{P}$ will be represented as an n-tuple of elements of I, termed *space of individuals*, where an individual can appear more than once.

A *Generic Evolutionary Algorithm* (GEA), introduced in [4], is an EA with three generic operators: *selection*, *local mutation* and *global mutation.*

Specifically, the dynamics of the GEA is given by a stochastic operator $E : \mathbb{P} \to \mathbb{P}$, defined as

$$E(\mathbf{x}) = \begin{cases} S(\mathbf{x}) & \text{with probability} \quad p_S \\ S \circ V_L^{\epsilon}(\mathbf{x}) & \text{w.p.} \quad p_L \\ S \circ V_G(\mathbf{x}) & \text{w.p.} \quad p_G. \end{cases}$$

where S is the selection operator, V_L^{ϵ} is the local mutation operator, and V_G is the global mutation operator. Thus, p_S, p_L and p_G are the probabilities of applying the evolutionary operators as specified above. Clearly, $p_S + p_L + p_G = 1$. Such operators will be described next.

The selection operator is an elitist selection operator that takes a population $\mathbf{x}$ in $\mathbb{P}$, and randomly produces a new population such that only the fittest individuals in $\mathbf{x}$ can be chosen to become part of the new population.

For the local mutation operator, the space of individuals is considered to be endowed with a meaningful metric. This operator introduces a "small" change in one individual. Given a population $\mathbf{x}$, $\mathbb{L}_{\epsilon}(\mathbf{x})$ will denote the set of populations that can be produced from $\mathbf{x}$ by replacing any individual in $\mathbf{x}$ with an individual that is at a distance less than or equal to ϵ (a real-valued parameter); Thus, the operator V_L^{ϵ} will choose a population in $\mathbb{L}_{\epsilon}(\mathbf{x})$ with uniform probability.

In contrast, the global mutation operator introduces a "strong" variation in one individual of the population. Given a population $\mathbf{x}$, let $\mathbb{G}(\mathbf{x})$ denote the set of populations that can be produced from $\mathbf{x}$ by replacing any individual in $\mathbf{x}$ with an arbitrary individual in I. The operator V_G applied to a population $\mathbf{x}$ produces a population in $\mathbb{G}(\mathbf{x})$, chosen with uniform probability.

In [4] it was shown in that when this algorithm is used to solve an optimization problem, its dynamics converges to a set of populations made up of optimizers of the fitness function $\mathcal{F} : I \to \mathbb{R}$.

The iteration of a GEA on a population $\mathbf{x}$ is modeled by $E(\mathbf{x})$. This iteration can be simulated using the update function $\phi_E : \mathbb{P} \times [0,1]^s \to \mathbb{P}$, where s is the number of uniform random numbers required to simulate an iteration of the algorithm. In other words, given a population $\mathbf{x}$, the iteration produces a new population $\mathbf{y} = \phi_E(\mathbf{x}, \mathbf{W})$, where $\mathbf{W}$ is a vector of s independent uniform random numbers in $[0, 1]$ needed to run an iteration of the GEA. Accordingly, an iteration of the GEA at time step t can be thought of as the result of the application of a random map $\psi_t = \phi_E(\cdot, W_t) : \mathbb{P} \to \mathbb{P}$ on the current population. Notice that the random map ψ_t is implicitly defined once W_t is set.

Consequently, the discrete dynamical system $\langle \mathbb{P}, E \rangle$ defined by a GEA can be also described as an IRM

$$\{\mathbb{P}; \psi_1, \psi_2, ..., \psi_L; p_1, p_2, ..., p_L\}$$

Notice that there is no need to define the correspondent p_i probabilities, since the simulation will use uniform random numbers that implicitly define the particular random map ψ_i to be used at each iteration. Details about functions ψ_i are discussed next.

A pseudocode for the GEA is presented next.

```
GEA()
1  $t \leftarrow 0$
2  $\mathbf{x}_t \leftarrow$ Generate_initial_population()
3  while stopping criterion not met
4     do $\psi_t \leftarrow$ Create_random_map()
5        $\mathbf{x}_{t+1} \leftarrow \psi_t(\mathbf{x}_t)$
6        $t \leftarrow t + 1$
7  return $\mathbf{x}_t$
```

Notice that the function *create_random_map()* is reduced to generate the random vector $\mathbf{W}$, and returns $\psi_t = \phi_E(\cdot, W_t)$. The array $\mathbf{W}$ is composed of $n+3$ random numbers, one of this random numbers is used to choose the operator applied; n of this random numbers are used to simulate selection (using each one of them to pick at ramdom one of the best elements in the current population to be part of the next population), and the two remaining random numbers are used to simulate mutations (one of them is used to pick the element that is mutated, and the other one is used to pick at random one of the possible mutations of that element).

4. STOPPING A GEA USING CFTP

GEAs as optimization problem solvers may be seen as MCMC samplers of a probability distribution supported on the set of populations composed of optimal solutions. Therefore, using CFTP, a GEA should be stopped once it reaches optimal solutions.

Although, in this research, a stopping criterion will be applied to GEAs, similar stopping criteria based on CFTP could be applied to other EAs as long as it can be shown that the stationary distribution of a MC associated to an EA is supported on optimizers of the problem.

In general, CFTP requires to simulate as many MC realizations as states in the state space. Thus, to run CFTP on the MC associated to the GEA, it is necessary to run as many MC realizations as elements in the population space ($|\mathbb{P}| = |I|^n$). Since the size of the space of populations is usually very large, this may be computationally intractable.

In this work, a modified version of CFTP, termed *sampled CFTP* is used. Instead of running a MC realization for each possible population, sampled CFTP runs a small number r of MC realizations; where r is small in comparison with the size of the state space . The r initial states of the MCs are uniform independent samples from the state space (in this case, the population space). The algorithm will stop once the r MC realizations coalesce.

Next, sampled CFTP is summarized. The parameters of the algorithm are r, the number of MC realizations, and n, the number of individuals in a population. In the pseudocode below, $P[1 \cdots r]$ is an array of populations, $\psi[1 \cdots r]$ is an array of random maps, and T is the time in the past where the MC realizations are started.

```
SAMPLED_CFTP_GEA($n, r$)
 1  $P[1 \cdots r] \leftarrow$ Generate_initial_populations()
 2  $\psi[1] \leftarrow$ Create_random_map()
 3  $T \leftarrow 1$
 4  while the MC realizations do not coalesce
 5    do for $t \leftarrow 1 \cdots T$
 6         do for $j = 0 \cdots r$
 7              do $P[j] \leftarrow \psi[t]\big(P[j]\big)$
 8
 9       for $t \leftarrow (T+1) \cdots (2*T)$
10         do $\psi[t] \leftarrow$ Create_random_map()
11
12       $T \leftarrow 2*T$
13       return $P[1]$
```

In practical terms, the self-stopping criterion is applied by running several realizations of a GEA in parallel, using the same random values in all iterations. The algorithm should stop when the underlying populations are the same for all the realizations.

In practical applications, a commonly used method is to perform r independent runs of an EA, and then choose the best amongst the r found solutions. Two points are worth to be emphasized in order to support the practical relevance of the proposed approach: The probability to find a good solution is increased by coupling the r GEA instances; and, the procedure requires less computational resources by generating less random numbers than in the case of the r independent runs.

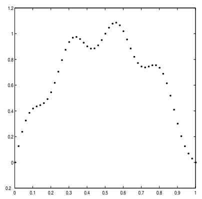


Figure 2: $\mathcal{F}(u) = 1 - 4(u - \frac{1}{2})^2 + \frac{1}{10} sin(8\pi u)$.

Notice that, in principle, this modified version of CFTP does not guarantee perfect samples, i.e., it does not assure that optimal solutions will be always obtained. However, it is hypothesized that sampled CFTP will stop the GEA at good solutions. Although there is no theoretical proof of this hypothesis, strong evidence that it may be valid is shown by the experiments that were perform, which will be described in detail in the next section.

5. EXPERIMENTS

This section reports the experimental results of the proposed sampled CFTP GEA. Since this work intends to explore a theoretical approach to serve as a basis to a practical EA stopping criterion, a set of toy optimization problems were chosen to test the potential of the proposed algorithm. However, work still needs to be done to mature the technique in order to make it more suitable to solve real-world problems.

For each optimization problem, the average coalescence time (number of iterations that the algorithm requires to reach a population composed only of optimal solutions) and the experimental distribution of the coalescence time are presented. Also, the estimated experimental probability that the algorithm reaches a population of this type is reported.

The general setting of the optimization problem considered is as follows: maximize a function $\mathcal{F} : I \to I$, where

$$I = \{0.0, 0.02, 0.04, ..., 0.98, 1\}$$

(I has size 50). The metric on I is $d_I(i, j) = |i - j|$, and $\mathbb{P} = I \times I$. In all cases, the parameters of the GEA were $p_S = 0.5, p_L = 0.3, p_G = 0.2$ and $\epsilon = 0.01$.

Four objective functions were considered, (three one dimensional and one two dimensional functions). Different population sizes (n) and number of MC realizations (r) were considered in each experiment. For each objective function and some specific combinations of r and n, the algorithm was run 1000 times.

5.1 Experiment 1

In this experiment, a sampled CFTP GEA is used to maximize the function $\mathcal{F}(u) = 1 - 4(u - 1/2)^2 + 0.1 \sin(10\pi u)$, $\mathcal{F}(u)$ is shown in figure 2. Notice that this function has several local maxima. In this case, the values considered for r were $10, 20$ and 30, and for n, $8, 16$ and 32. Thus, a total of 9000 runs of the algorithm were performed.

r	n	$\widehat{\overline{T}}$	$\widehat{p}$
10	8	394.642	1.0
20	8	438.192	1.0
30	8	435.296	1.0
10	16	726.912	1.0
20	16	771.512	1.0
30	16	772.864	1.0
20	32	1424.192	1.0
10	32	1460.352	1.0
30	32	1427.2	1.0

Table 1: Results of experiment 1.

In table 1, $\widehat{\overline{T}}$ is the average coalescence time of the 1000 independent runs of the algorithm, and $\widehat{p}$ is an estimation of the probability of success, i.e., the probability that the algorithm stops at a population composed of solutions to the optimization problem. Notice that, in all cases, the algorithm stopped at an optimum solution.

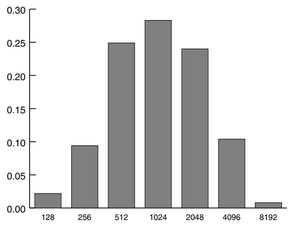


Figure 3: Experimental probability distribution function of T for the 1000 simulations in experiment 1; $r = 30$, $n = 32$. The values of T are powers of 2.

Figure 3 shows the experimental distribution function of T. Notice that it is "approximately normal", which is the expected result according to Central Limit Theorem, given that in each one of the 1000 runs, an independent identically distributed observation of the random variable T is obtained.

5.2 Experiment 2

Now, the function considered is $\mathcal{F}(u) = |sin(4\pi u)|$ (see figure 4). Again $\mathcal{F}$ has several local maxima. As in experiment 1, the values considered for r were $10, 20$ and 30, and for n $8, 16$ and 32.

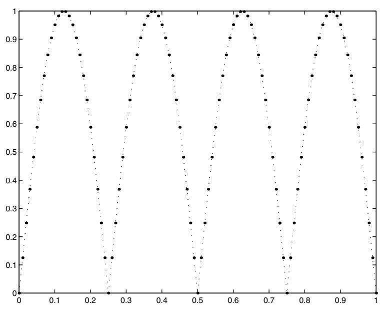


Figure 4: $\mathcal{F}(u) = |sin(4\pi u)|$.

As in the previous experiment, the algorithm stopped at an optimum solution in all cases (see table 2).

r	n	$\widehat{\overline{T}}$	$\widehat{p}$
10	8	1402.236	1.0
10	16	3069.714	1.0
10	32	6524.672	1.0
20	8	1507.032	1.0
20	16	3108.096	1.0
20	32	6470.528	1.0
30	8	1518.504	1.0
30	16	3099.648	1.0
30	32	6808.128	1.0

Table 2: Results of experiment 2.

5.3 Experiment 3

In this case, the function to be maximized is $\mathcal{F}(u) = 1 - 4(u - \frac{1}{2})^2 + \frac{1}{10}sin(32\pi u)$, which is shown in figure 5. Now, a larger space of individuals $I = \{0.0, 0.001, 0.002, ..., , 1\}$ is considered; the size of I is 1000. Thus, in this experiment, the combinations of values of r and n considered were: $n = 8$ and $r = 10, 20, 30$; $n = 16$ and $r = 10, 20, 30, 40, 50$, and $n = 32$, $r = 10, 20$.

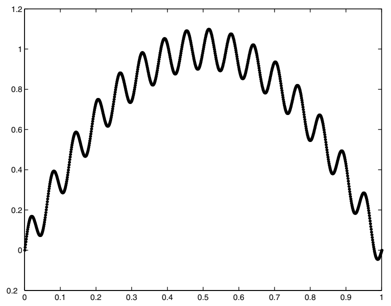


Figure 5: $\mathcal{F}(u) = 1 - 4(u - \frac{1}{2})^2 + \frac{1}{10}sin(32\pi u)$.

r	n	$\widehat{\overline{T}}$	$\widehat{p}$
10	8	1945.036	0.771
20	8	2491.91	0.891
30	8	2924.982	0.933
10	16	2372.318	0.851
20	16	2811.68	0.934
30	16	2915.412	0.963
40	16	2938.97	0.973
50	16	2972.856	0.984
10	32	2521.528	0.923
20	32	2813.184	0.966

Table 3: Results of experiment 3.

It is important to emphasize that, on average, sampled CFTP GEA required less than 3000 iterations to find an optimal solution with very high probability (see table 3).

5.4 Experiment 4

In this experiment, the two dimensional function $\mathcal{F}(u, v) = \big(2 - 4(u - .5)^2 + 0.1sin(32\pi u)\big)$ - $\big(4(v - 0.5)^2 + 0.1sin(32\pi v)\big)$ is maximized (see figure 6). Now, the space

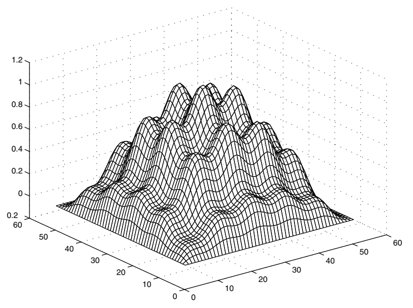


Figure 6: $\mathcal{F}(u,v) = \big(2 - 4(u-\frac{1}{2})^2 + \frac{1}{10}sin(32\pi u)\big) + \big(- 4(v-\frac{1}{2})^2 + \frac{1}{10}sin(32\pi v)\big)$.

of individuals is $I \times I$, with I the space of individuals considered in experiments 1 and 2, therefore, the size of I is 2500. The distance d between individuals (a,b) and (c,d) is given by $d_I\left((a,b),(c,d)\right) = |a-c| + |b-d|$. In this experiment, the combinations of values of r and n considered were: $n = 10, 20$ and $r = 20, 30, 40, 50$; and $n = 30$ and $r = 50$.

r	n	$\widehat{\overline{T}}$	$\widehat{p}$
20	10	8069.12	0.594
20	20	10835.56	0.722
30	10	12688.96	0.716
30	20	13816.32	0.857
40	10	14806.08	0.771
40	20	13667.2	0.884
50	10	13250.6	0.817
50	20	15772.16	0.942
50	30	14831.36	0.970

Table 4: Results of experiment 4.

As in experiment 3, on average, sampled CFTP GEA stopped in less than 3000 iterations at an optimal solution with high probability (see table 4).

The experiments above showed that the proposed method is succesful in all cases. It is also important to emphasize that the sample size is small in comparison to the size of the population space, and that the probability that the algorithm stops at optimal solutions gets higher as the number of realizations of the GEA increases.

6. CONCLUSIONS

In this work, a model for a GEA based on iterated random maps was developed. In addition, a self-stopping criterion for a GEA was proposed. It is based on a modified version of CFTP, called sampled CFTP, which deals with the problem of running as many MC realizations as states in the state space. In contrast to CFTP, it only tracks a small number of MC realizations.

In practice, the self-stopping criterion is applied by running simultaneously several realizations of a GEA, using the same random values in all iterations. The algorithm should stop when the underlying populations are the same for all the realizations.

Even though, there is not theoretical proof that sample CFTP GEA, applied to solve an optimization problem, will stop at optimal solutions, the experimental results of this work are encouraging. In all the experiments, the stopping criterion was successful, i.e., the algorithm stopped at optimal solutions with high probability, which may be an up to standard level from the practicioneers point of view.

From the experiments, it can be seen that the probability that the algorithm stops at optimal solutions becomes higher as the number of realizations of the GEA increases.

In further work, sampled CFTP GEA should be studied in a more theoretical fashion, specifically a theoretical proof of its effectiveness should be explored. Also, additional work is required to elucidate the relationship between parameters such as: size of the population, size of the population space and size of the sample used to start the sampling CFTP procedure.

Particularly, a mechanism to determine an appropriate sample size should be devised. Notice that in this work, population samples are chosen according to a uniform distribution, hence, other sampling procedures are worth further studies. Finally, CFTP GEA should be tested on more complex high-dimensional benchmark problems.

7. REFERENCES

[1] H. Aytug and G. J. Koehler. New stopping criterion for genetic algorithms. Technical report, Charlotte NC and Gainesville FL, 4 1996.

[2] P. Diaconis and D. Freedman. Iterated random functions. *SIAM Review*, 41(1):45–76, Mar. 1999.

[3] J. Fill. An interruptible algorithm for perfect sampling via markov chains. *The Annals of Applied Probability*, 8(1):131–162, 1998.

[4] G. Hernandez, F. Nino, D. Dasgupta, and J. Garcia. On geometric and statistical properties of a generic genetic algorithm. In *Proceedings of the Congress on Evolutionary Computation (CEC-04)*, Portland, OR, 2004. IEEE Press.

[5] J. Liu. *Monte-Carlo strategies in scientific computing.* Springer-Verlag, New York, 2001.

[6] J. Propp and D. Wilson. Coupling from the past: a user's guide. In *Microsurveys in Discrete Probability, D. Aldous and J. Propp (eds)*, volume 41, pages 181–192. American Mathematical Society, 1998.

[7] G. Rudolph. Finite Markov chain results in evolutionary computation: A tour d'horizon. *Fundamenta Informaticae*, 35(1–4):67–89, 1998.

[8] M. D. Vose. *The simple genetic algorithm: foundations and theory.* MIT Press, Cambridge, MA, 1999.

[9] D. B. Wilson and J. G. Propp. Exact sampling with coupled markov chains and applications to statistical mechanics. Jan. 1996.

[10] D. B. Wilson and J. G. Propp. How to get an exact sample from a generic Markov chain and sample a random spanning tree from a directed graph, both within the cover time. In *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 448–457, Atlanta, Georgia, 28–30 Jan. 1996.