Procedia Computer Science

Volume 51, 2015, Pages 2698–2702

ICCS 2015 International Conference On Computational Science

A Formal Method for Parallel Genetic Algorithms*

Natalia López, Pablo Rabanal, Ismael Rodríguez, Fernando Rubio

Facultad Informática. Universidad Complutense de Madrid. 28040 Madrid, Spain
natalia@sip.ucm.es,prabanal@fdi.ucm.es,isrodrig@sip.ucm.es,fernando@sip.ucm.es

Abstract
We present a formal model that allows to analyze non trivial properties about the behavior of parallel genetic algorithms implemented using multi-islands. The model is based on a probabilistic labeled transition system, that represents the evolution of the population in each island, as well as the interaction among different islands. By studying the traces these systems can perform, the resulting model allows to formally compare the behavior of different algorithms.

Keywords: Genetic Algorithms, Multi-islands, Formal Methods, Labeled Transition Systems

1 Formal Model

In this paper we define a *probabilistic labeled transition system* to allow the specification and systematic study of the parallelization of genetic algorithms (GA) [1]. We assume that the GA is parallelized by splitting the total population of individuals (chromosomes) into different subsets and mapping each subset to a different processor, where each subset will independently evolve for some number of iterations. Each of these divisions is a parallel *island*. However, after some iterations, chromosomes of different islands influence each other, which allows islands to converge to a common solution. The goal of our probabilistic formal method is modeling this type of evolutive parallel algorithm.

Definition 1.1. *A probabilistic labeled transition system (PLTS) is a tuple* $(S, \mathrm{Act}, \rightarrow)$ *where* S *is a set of states;* Act *is a set of actions; and* $\rightarrow \subseteq S \times (\mathrm{Act} \times [0,1]) \times S$ *is the set of transitions.* □

For convenience, we will use the notation $s \xrightarrow{\alpha}_p s'$, instead of $(s, (\alpha, p), s')$, to express that, with probability p, the state s evolves to state s' after performing the action α. In a GA, each state of a PTLS represents a population (a set of chromosomes), the actions represent the different ways that the GA can evolve, and the probability associated with the transition will denote the probability that the GA takes this transition instead of any other of the available transitions from the starting state of the transition. For the sake of simplicity, sometimes the transitions with probability 1 will be denoted without the probability.

*This research has been supported in part by the CICYT project TIN2012-39391-C04-04 and the CAM project S2013/ICE-2731.

 Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2015

doi:10.1016/j.procs.2015.05.389

Definition 1.2. *A chromosome (or individual) is defined as* $(c_1, c_2, \cdots, c_m) \in \mathbb{Z}^m$ *a m-tuple of integers that represents the individual genetic information. We denote by* Q *the set of all the possible individuals* $(x_1, x_2 \cdots$ *to range over* $Q)$*.*

A population is defined as a multiset $\{\!\{ x_1, \cdots, x_n \}\!\}$ *of individuals. We assume that* Pop *denotes the set of all the possible populations.* □

Definition 1.3. *A genetic algorithm state is defined by a tuple of elements* $GA = (\mathrm{pop}, \mathrm{nS}, \mathrm{nG})$ *where* $\mathrm{pop} = \{\!\{ x_1, \cdots, x_n \}\!\} \in \mathrm{Pop}$ *is the actual population of the GA;* nS *is the number of individuals of the population that will be selected in each selection stage; and* nG *is the number of remaining generations (iterations) to finish the execution of the GA. We use* S *to denote the set of states of a GA.* □

Once the states of a PLTS are defined, we must define the actions. The set of actions consists of:

- The *initialization action*, denoted by init, that produces an initial population.
- The *evolution action*, denoted by evol, that produces a new generation (a new population) based on two consecutive actions:
  - The *selection* function, denoted by $fitSel$, that applied to a population, returns a set of pairs. Each pair consists of a sorted population of individuals and the probability of selecting that sorted population. The order given will be used to choose the way the individuals of the population are crossed in the reproduction.
  - The *reproduction* function, denoted by *reprod*, that, applied to a pair of individuals, returns a multiset of individuals where the initial pair is included (the parents), and the other pairs are their offspring.
- The *stop* action, denoted by stop. This action will be performed to finish the evolution process, when no more actions can be performed. When it is executed, the state reached will be a special state where the only relevant part is the multiset of individuals (that will represent the population of solutions reached by the GA). The number of remaining generations will be 0.

Therefore the set of actions of a GA will be $\mathrm{Act} = \{\mathrm{init}, \mathrm{evol}, \mathrm{stop}\}$.

Before defining the rules of the evolution of a GA, we need to define a simpler PLTS to define the behavior of the *reproduction* action. This action also needs two auxiliary functions *crossOver* and *mutation*, defined below.

Definition 1.4. *Let* $Pop_2 \subseteq Pop$ *be the set of multisets of individuals with multiplicity 2 (that is, each multiset contains either two different individuals or two instances of the same individual) and let* $act = \{reprod\}$ *be a set with a single action. We define the PLTS of the reproduction of pairs of individuals as the tuple* $(Pop, act, \longrightarrow)$ *where* $\longrightarrow \subseteq Pop_2 \times (act \times (0, 1]) \times Pop$ *is described in Table 1.* □

The defined PLTS will use a function $crossOver : Q \times Q \longrightarrow \mathcal{P}(Pop \times [0, 1])$ that will define how the crossover will be done in the GA. Depending on how this function is defined, the function will return a different number of children. Besides, a function $mutation : Q \longrightarrow \mathcal{P}(Q \times [0, 1])$ will define how to carry out mutation in the GA. This function will be applied to an individual and it will return a set of pairs of the form $Q \times [0, 1]$ that will denote that the former individual

$$\frac{(\emptyset,p)\in crossOver(x_1,x_2)}{\{\!|\ x_1,x_2\ |\!\} \xrightarrow{reprod}_{p} \{\!|\ x_1,x_2\ |\!\}} \quad \text{(Reprod1)}$$

$$\frac{(\{\!|\ y_1,\cdots,y_r\ |\!\},p)\in crossOver(x_1,x_2),\ (z_i,q_i)\in mutation(y_i),1\leq i\leq r}{\{\!|\ x_1,x_2\ |\!\} \xrightarrow{reprod}_{p\cdot\Pi_{1\leq i\leq r} q_i} \{\!|\ x_1,x_2,z_1,\cdots,z_r\ |\!\}} \quad \text{(Reprod2)}$$

Table 1: Rules for the reproduction of a pair of individuals

mutates into some individual with the associated probability. In particular, we will consider that $(y, p) \in mutation(x)$ if and only if x mutates into y with probability p.

The rules that appear in Table 1 describe the reproduction of two individuals, where the crossover and the mutation are performed. Each pair of individuals has two options: either crossing over or not. In the latter case, there is nothing else to do. However, if individuals are crossed, the progeny can also either suffer a mutation or not. The first rule of the table represents the case where there is no crossing. In this case, the individuals remain the same. The second rule represents the reproduction of two individuals, where the crossover is the former action and, after that, some mutations can be suffered by the offspring.

The evolution of the population represents the evolution of the whole set of individuals of the population. The PLTS used to describe the behavior of a whole population is defined as follows.

Definition 1.5. *The genetic algorithm evolution is denoted by a PLTS* $(S, \mathrm{Act}, \rightarrow)$ *where* S *is the set of states of the genetic algorithm,* Act *is the set of actions* $\{\mathrm{init}, \mathrm{evol}, \mathrm{stop}\}$*, and the transition* $\rightarrow \subseteq S \times (\mathrm{Act} \times (0, 1]) \times S$ *is defined by the rules in Table 2.* □

The value $nMaxGen$, appearing in the first rule, represents the maximum number of allowed generations (it has to be previously fixed). In the last two rules, the predicate termination(s, nS, nG) will return `true` if the state satisfies the optimality criterion in the reached solutions or if the number of remaining generations is 0, and it will return `false` in the other case. This predicate, $nMaxGen$, and the functions $mutation : Q \longrightarrow \mathcal{P}(Q \times (0, 1])$ and $crossOver : Q \times Q \longrightarrow \mathcal{P}(Pop \times (0, 1])$, previously presented, will be defined depending on the specific GA that will be modeled.

In Table 2, the first rule represents the initialization of the GA. The function init : $\mathcal{P}(\mathrm{Pop} \times (0, 1])$ will produce a set of pairs of population and probability $\{(\mathrm{pop}_1, p_1), \cdots, (\mathrm{pop}_n, p_n)\}$ where $\mathrm{pop}_i \in \mathrm{Pop}$, $p_i \in (0, 1]$, and $\sum_{i=1}^{n} p_i = 1$. Each pair (pop, p) of the set represents that the population pop will be created with probability p. Therefore, this first rule represents that, from the initial state, the process can evolve into a initial population *pop* given by the function init with probability p, considering that $(pop, p) \in$ init. Besides, the number of remaining generations is set to the maximum, that is, $nMaxGen$. After each evolution step (last rule), this number decreases by 1. The second rule represents the situation in which no more evolutions can be performed because either the state satisfies the required criterion or because the remaining generations are 0. In both cases, the only action able to be performed is stop.

The last rule represents the GA evolution. This evolution is possible only if the number of remaining generations is greater than 0 and a state with the required optimality has not been reached (it is checked by the *termination* function). In each evolution process, a selection of the best individuals in terms of fitness, a crossover between individuals, and the mutation of some of them will be done. Assuming that $lists(A)$ denotes the set of all possible sorted lists of

$$\frac{(\text{pop},p)\in\text{init}}{(\emptyset,nS,0)\xrightarrow{\text{init}}_p(\text{pop},nS,nMaxGen)} \qquad \frac{\text{termination}(\text{pop},nS,nG)}{(\text{pop},nS,nG)\xrightarrow{\text{stop}}(\text{pop},nS,0)}$$

$$\frac{\neg termination(\{\!|\, x_1,\cdots,x_n \,|\!\}, nS, nG) \ \wedge\ ([z_1,\cdots,z_{nS}],p)\in \text{fitSel}(\{\!|\, x_1,\cdots,x_n \,|\!\}, nS) \wedge 0<i\leq nS\ div\ 2, \{\!|\, z_{2i-1}, z_{2i} \,|\!\} \xrightarrow{reprod}_{p_i} \{\!|\, y_{i1},\cdots,y_{in_i} \,|\!\}}{(\{\!|\, x_1,\cdots,x_n \,|\!\},nS,nG)\xrightarrow{evol}_{(\Pi p_i)\cdot p}(\{\!|\, y_{11},\cdots,y_{1n_1},\cdots,y_{r1},\cdots,y_{rn_r} \,|\!\},nS,nG-1)}$$

Table 2: Evolution of the GA rules

$$\frac{s_1\xrightarrow{evol}_p s_1',\ s_2\xrightarrow{evol}_q s_2'}{s_1\|s_2\xrightarrow{evol}_{p\cdot q} s_1'\|s_2'} \qquad \frac{s_1\xrightarrow{evol}_p s_1',\ \text{termination}(s_2)}{s_1\|s_2\xrightarrow{evol}_p s_1'\|s_2} \qquad \frac{\text{termination}(s_1),\ s_2\xrightarrow{evol}_q s_2'}{s_1\|s_2\xrightarrow{evol}_q s_1\|s_2'}$$

$$\frac{\text{termination}(s_1),\ \text{termination}(s_2),\ \neg\text{iteration}(s_1,s_2)}{s_1\|s_2\xrightarrow{stop}s_1\|s_2} \qquad \frac{\text{termination}(s_1),\ \text{termination}(s_2),\ \text{iteration}(s_1,s_2)}{s_1\|s_2\xrightarrow{evol}_r s_1''\|s_2''}$$

where $s_1=(pop_1,nS,nG_1),\ s_2=(pop_2,nS,nG_2),\ (pop_1',pop_2',r)\in\text{exch}(pop_1,pop_2)$
$s_1''=(pop_1',nS,nMaxGen),\ s_2''=(pop_2',nS,nMaxGen)$

Table 3: GA parallel evolution rules

individuals of the set A (in which each individual can appear any number of times), the function $\text{fitSel}:\text{Pop}\times\mathbb{N}\longrightarrow\mathcal{P}(lists(\text{Pop})\times(0,1])$ will assign a probability to each way of sorting the individuals of a given population considering their fitness values. In particular, the algorithm will consider that the pairs of parents to be crossed will be pairs of consecutive individuals of the list: the first one with the second one, the third with the fourth, and so on. Some individuals could appear more than once if they are chosen more than once to be parents. So, we have $\text{fitSel}(\{\!|\, x_1,\cdots,x_n \,|\!\},nS)=\{(listpop_1,p_1),\cdots,(listpop_m,p_m)\}$ where $\sum_{i=1}^m p_i=1$. All lists will contain nS individuals. Therefore, the function will return a set of pairs of the list of individuals and probability, where $(listpop,p)\in\text{fitSel}(\{\!|\, x_1,\cdots,x_n \,|\!\},nS)$ represents that with probability p the list $listpop$ will be chosen (which will contain nS instances of individuals).

Once the GA evolution is defined, we formalize a similar evolution process for independent populations that evolve in *parallel*. For the sake of simplicity, initially we present a case where two independent islands exist. First of all, such islands evolve in parallel without any communication, until they reach local termination states. When this happens, both populations interchange some individuals, and then the independent evolution of both islands restarts.

The result of the parallel composition of two systems defined by the PLTSs $A_1=(S_1,\text{Act},\rightarrow_1)$ and $A_2=(S_2,\text{Act},\rightarrow_2)$ is a new PLTS $A_1\|A_2=(S_1\times S_2,\text{Act},\rightarrow)$. Instead of representing their composed states with the form of a pair $(s_1,s_2)\in S_1\times S_2$, we use $s_1\|s_2$ to represent them. The transitions in $\rightarrow$ are inferred by the rules given in Table 3.

The first rule in Table 3 defines the parallel evolution of two populations. The following two rules represent the situation when one population can evolve independently while the other one has just reached its local termination condition. In this case, the island that can evolve will do it, meanwhile the other island waits until both of them find their local termination condition. The fourth rule represents the situation when both populations have finished their independent evolutions. In such situation, both can continue their evolution once they exchange some of

their individuals and they restart their evolution process independently again. The exchange and the restarting can be performed only if the predicate $iteration(s_1, s_2)$ is fulfilled. This predicate controls the finalization of the parallel evolution of the two islands.

The function *exch* is defined as follows:

Definition 1.6. *Let* $pop_1, pop_2 \in \text{Pop}$. *The function* $\text{exch} : \text{Pop} \times \text{Pop} \longrightarrow \mathcal{P}(\text{Pop} \times \text{Pop} \times (0, 1])$ *is defined as* $\text{exch}(pop_1, pop_2) = \{(pop_1^1, pop_2^1, p_1), \ldots, (pop_1^m, pop_2^m, p_m)\}$ *where* $\sum_{k=1}^{m} p_k = 1$ *and, for all* $1 \leq k \leq m$, $pop_1^k \cup pop_2^k = pop_1 \cup pop_2$ *and* $|pop_1^k| + |pop_2^k| = |pop_1| + |pop_2|$. □

Next we define the behavior of a GA, this time considering the effect of *several* steps of execution together. A sequence of execution steps will be called *trace*.

Definition 1.7. *Let us consider the PLTS of a genetic algorithm evolution* $E = (\text{S}, \text{Act}, \longrightarrow)$. *Let* $s' \in \text{S}$ *be a state and let* $\rho \in \text{Act}^*$ *be a sequence of actions.* $\rho = \alpha_1 \cdots \alpha_r$ *with* $\alpha_1, \cdots, \alpha_r \in \text{Act}$.

We say that E produces the trace $(\emptyset, nS, 0) \stackrel{\rho}{\Longrightarrow}_p s'$ *if there exist* $s_1, \cdots, s_r \in \text{S}$ *and* $p_1, \cdots, p_r \in (0, 1]$ *such that* $\Pi_{i=1}^{r} p_i = p$ *and* $(\emptyset, nS, 0) \xrightarrow{\alpha_1}_{p_1} s_1 \xrightarrow{\alpha_2}_{p_2} \cdots s_{r-1} \xrightarrow{\alpha_r}_{p_r} s_r = s'$.

A trace is complete if $\alpha_r = \text{stop}$ *and* $s' = (pop, nS, 0)$ *with* $pop \in \text{Pop}$. □

By making the probability of a trace be the multiplication of the probabilities of each step of the trace, our model allows to calculate the probability of a GA to obtain a population through some given path. Moreover, by adding the probabilities of all the traces that reach the same population, we can obtain the probability of a GA to finalize its execution producing a given population. Using the same idea, we can add the probability of all the populations where the best individual is a given one, and compute the probability of a GA to return, as a final result, a given individual.

Definition 1.8. *Let us consider the PLTS of a genetic algorithm evolution* $E = (\text{S}, \text{Act}, \longrightarrow)$. *We define the probability of E to perform an execution to reach a population* $pop \in Pop$ *as:* $prob_{pop} = \sum \{\!|\, p \mid \exists\, \rho : (\emptyset, nS, 0) \stackrel{\rho}{\Longrightarrow}_p (pop, nS, 0) \,|\!\}$.

Let $best : \text{Pop} \longrightarrow Q$ *be a function such that, for any* $pop \in Pop$, $best(pop)$ *is the individual with the best fitness value in pop. We define the set of final populations where an individual* $q \in Q$ *is the best solution as:* $pops_q = \{pop | q = best(pop) \ \wedge\ \exists\, \rho, p : (\emptyset, nS, 0) \stackrel{\rho}{\Longrightarrow}_p (pop, nS, 0)\}$.

We define the probability of the PLTS E to return at the end of its execution the solution q as $\sum_{pop \in pops_q} prob_{pop}$. □

The aim of this formal model is to study how to prove that a particular set of conditions (not shown due to lack of space) is enough to ensure that an evolutionary algorithm will reach the same optimality when solving two different NP-complete problems, which is a formal continuation of our previous empirical studies about it [2]. These properties compare the behavior of the algorithm for two given problems applying a *polynomial reduction* between both, which necessarily exists, since both of them are NP-complete. Proving this property helps us to know in advance whether some strategy will obtain good optimality results when facing a new NP-complete problem, thus reducing the effort needed to adjust GAs empirically.

References

[1] D.E. Goldberg. *Genetic Algorithms in Search, Optimisation and Machine Learning.* Addison-Wesley, 1989.

[2] P. Rabanal and I. Rodríguez. Using polynomial reductions to test the suitability of metaheuristics for solving np-complete problems. In *SAC 2013*, pages 194–199. ACM Press, 2013.