Math Oracles: A New Way of Designing Efficient Self-adaptive Algorithms

Gabriel Luque
E.T.S.I. Informática, University of Málaga,
29071 Málaga (España)
gabriel@lcc.uma.es

Enrique Alba
E.T.S.I. Informática, University of Málaga,
29071 Málaga (España)
eat@lcc.uma.es

ABSTRACT

In this paper we present a new general methodology to develop self-adaptive methods at a low computational cost. Instead of going purely ad-hoc we define several simple steps to include theoretical models as additional information in our algorithm. Our idea is to incorporate the predictive information (future behavior) provided by well-known mathematical models or other prediction systems (the *oracle*) to build enhanced methods. We show the main steps which should be considered to include this new kind of information into any algorithm. In addition, we actually test the idea on a specific algorithm, a genetic algorithm (GA). Experiments show that our proposal is able to obtain similar, or even better results when it is compared to the traditional algorithm. We also show the benefits in terms of saving time and a lower complexity of parameter settings.

Categories and Subject Descriptors

D.2.10 [**Software Engineering**]: Design—*Methodologies*; F.2 [**Theory of Computation**]: Analysis of Algorithms

General Terms

Algorithms

Keywords

methodology, mathematical oracles, self-adaptive techniques

1. METHODOLOGY

The development of self-* algorithms which adapt their behavior to the specific characteristics of the problem is currently a very hot topic in Computer Science. Most of the proposed techniques in this domain change their dynamics using the current status of the method and some historical data. But there exists a number of mathematical studies which could provide us some information about the expected future behavior of the method which can be used to modify its dynamics. We define a *mathematical oracle* as a mathematical description of the behavior of a search technique that allows a prediction on the near future steps of the technique. We assume that this prediction is done in the oracle in an approximate, imperfect way, and that it can be used in a black-box form.

GECCO'13 Companion, July 6–10, 2013, Amsterdam, The Netherlands.
ACM 978-1-4503-1964-5/13/07.

The main goal of this work is then to propose a methodology to use the information provided by a mathematical tool allowing to predict the behavior of algorithms, what will allow us to build enhanced versions of them. The main decisions and steps which should be considered by the developer are:

1) *Offering to the oracle.* First, we have to gather the input data latter offered to fed the oracle. This information depends on the mathematical oracle which we are using. Generally, it is related to the evolution of the search, such as improvement speed, number of steps performed, ..., or in the case of population-based techniques some statistical indicator of the population (such as diversity, entropy, standard deviation, ...).

2) *Praying the oracle.* Now, since we have the data needed by the mathematical oracle (got in the previous phase), we could apply it to obtain the new predictive information. The next decision is when the mathematical oracle is to be applied, e.g. we could apply it in every step of the algorithm. However, the decision should be an elaborated one since some algorithmic steps are usually needed to see if the last change is positive or not.

3) *Oracle's response.* When we use the mathematical oracle, we get some prediction about the expected behavior (with some inaccuracy) of the technique in the next steps (convergence, diversity, movement in search space, ...).

4) *Pray and work.* The final phase is how to use this information to change our algorithms. Usually we will have arbitrary potential oracles which provide information only about the convergence time, the improvement speed, or similar. We have to decide if this trend is the adequate one for the future run of our algorithm and, otherwise, take some actions (change parameters or operators of the method systematically) to approach this prediction to the expected one.

2. CASE STUDY

This section shows how this general methodology can be instantiated for a concrete algorithm and a specific mathematical oracle. We design a family of self-adaptive genetic algorithms using some predictive information coming from a mathematical model of takeover time [1] . We are going to propose several alternatives to each step of this methodology to illustrate the many interesting future research lines that opens from our idea.

1) *Offering to the oracle.* Our theoretical oracle can estimate some values related to the convergence of the GA. To calculate these values it needs the proportion of the optimal solution in the populations of the last steps of the algorithm.

Here, we face the first challenge, since this value is always 0 (a real method has got no copies of the optimal value until maybe its last steps). We propose three alternative definitions of this concept: (a) proportion of individual close to the optimal one (αOpt model), (b) proportion of individuals close to the current best found solution ($\alpha bestFS$ model), and (c) use the standard deviation (*stdDev* model). The two last models can be used on any arbitrary problem, while the first one needs to know the optimal solution.

2) *Praying the oracle.* In this case the oracle only needs information on the previous 10-15 generations to gather enough information to get an accurate prediction. Since the execution of the GA to find the optimal solution in the tested benchmark is much larger than this value, applying the mathematical oracle every 15 generations seems to be reasonable. But for a more comprehensive study, we will test the next values: 1, 15, 30, 45, and 100.

3) *Oracle's response.* In the previous section we stated that the oracle can facilitate two types of information: the takeover time and the expected convergence speed. In this case study, since we know that these values are not fully accurate, we have decided to use only the convergence speed. According to some preliminary results we have defined three possible states for this value: the convergence is too fast, it is too slow, and the convergence is the expected one. Of course, a more fine-grained classification could be used, but this is a good first approach, very intuitive and general.

4) *Pray and work.* In this step we have to decide if some change should be applied to the search method according to the information provided by the oracle. Also, we have to indicate what actual actions should be performed. To maintain the algorithm as simple as possible, we perform some actions only if the predicted convergence speed is too fast or to slow. In our proposed self-adaptive technique, when the predicted convergence speed is too slow, we will decrease the intensity of the mutation, while if the predicted convergence value is too fast, we will increase the intensity of the mutation. Please notice that this is done not in a pre-programmed form or in an ad-hoc manner directed by the researched, but controlled automatically by a math oracle.

3. EMPIRICAL VALIDATION

In this section we discuss the design of a set of experiments in order to observe the performance of the self-* models proposed in the previous section and compare them against themselves and also against a canonical GA. Our goal is not only to show that this idea works and requires a low cost, but also that the resulting techniques are actually accurate and able of solving actual problems. To test the different models we chose two well-known problems: the MAX-SAT problem (nine instances) and the TSP problem (five instances).

Table 1 shows the results of applying the statistical Wilcoxon test confronting each of the models against a traditional GA. The column tagged with $\%>$ contains the percentage of independent runs in which our self-* algorithm is statistically better than the canonical GA. The column $\%\geq$ has a similar meaning, but it also includes the cases in which there are no statistical difference among the techniques.

Several conclusions can be obtained from Table 1. The first one is that almost any of the proposed alternatives to design a self-* GA by using a (simple) oracle is embarrassingly better than a traditional GA in most of the instances. We observe that using the αOpt obtains the best results

Table 1: Advantage of using oracles (Wilcoxon test)

self-adaptive GA		MAXSAT + TSP	
Approx. scheme	period	$\%>$	$\%\geq$
αOpt	1	14%	36%
	15	64%	78%
	30	**78%**	**100%**
	45	71%	**100%**
	100	14%	14%
$\alpha bestFS$	1	21%	21%
	15	64%	71%
	30	71%	86%
	45	**78%**	93%
	100	21%	28%
stdDev	1	7%	21%
	15	21%	36%
	30	28%	43%
	45	28%	36%
	100	14%	21%

Table 2: Overhead provoked by the different periods

1	15	30	45	100
58%	11%	6%	2%	1%

while the $\alpha bestFS$ also offers quite good results and it has the advantage that it does not require any additional information of the problem. On the contrary, the model based on the standard deviation obtains quite poor results.

The period is a very important factor. The results show that it is not beneficial to ask the oracle at every algorithmic step (the mathematical oracle needs several generations to observe the effects of the changes performed). However, a large value (100) is neither good for the search, since the oracle services are ignored for too long. The best value usually is around 30 generations for these two problems.

We will also analyze the overhead provoked by each studied model. The use of self-tuning techniques means an increase in the computation time of the algorithm (due to the additional steps) that it is too often neglected in self-* articles. In Table 2 we only show the average overhead provoked by the period, since we observed that it is main factor in the increasing of the runtime (it controls how many petitions are performed to the oracle). We show the percentage in which the run time of an average generation measured without/with the oracle. We can observe that the overhead decreases when the period value is larger. This is an expected result since when the period is increased the number of additional operations needed by the algorithm is smaller. We can also notice that the overhead is quite low in all cases (with the exception of period one), and as we saw before the self-adaptive models outperform the traditional GA, requiring just a few generations to find the optimum. These two factors (small overhead and a lower number of generations) yield a global execution time of self-adaptive models similar to the traditional GA (or even smaller, in some cases).

Acknowledgments

Authors acknowledge funds from the Spanish Ministry of Sciences under contract TIN2011-28194.

4. REFERENCES

[1] G. Luque and E. Alba. Analyzing the behaviour of population-based algorithms using Rayleigh distribution. In *PPSN XII*, pages 417–427, 2012.