A Parameter-Less Genetic Algorithm

•••

Matthew Flammia, Queens College, Professor Goldberg

Abstract

This paper is concerned with the creation and performance of a genetic algorithm, where the end-user does not need to define parameters such as population size, crossover probabilities, selection and mutation rates, etc.

If done correctly, this parameter-less GA could be applied to more fields by less knowledgeable people. The way the parameter-less GA comes up with the optimal parameters is based on previous research on optimal parameters, schema theorem, building block mixing, and genetic drift. Overall, the parameter-less GA should outperform hard-coded regular GA algorithms with preset parameters.

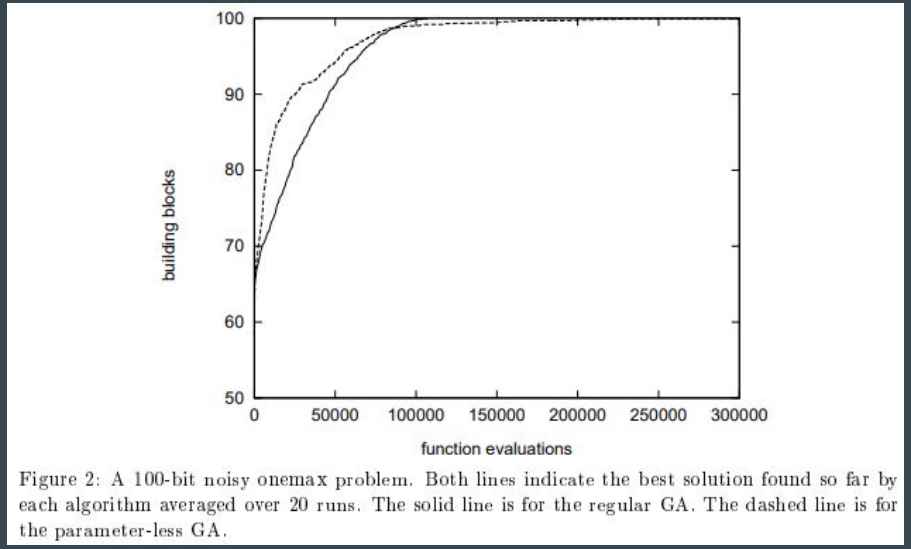

Figure 2: A 100-bit noisy onemax problem. Both lines indicate the best solution found so far by each algorithm averaged over 20 runs. The solid line is for the regular GA. The dashed line is for the parameter-less GA.

Introduction

A genetic algorithm requires certain parameters to be defined before beginning its process, these parameters include population size, crossover rates, mutation rates, etc. All these parameters are decided based on the goal to accomplish and are different for every application of a GA.

To make GA more applicable to a wider audience, having a GA define its own parameters would mean that the end-user only needs to define what the goal of the GA should be, and a way of checking for that goal.

On the right is an example of some parameters a user might need to consider.

Factor	Level 1	Level 2	Level 3	Level 4
x_1 Population Size	50	100	150	200
x_2 Selection Method	SUS	T(r=.60)	T(r=.75)	T(r=.90)
x_3 Elite Selection %	0	2	4	8
x_4 Reboot Proportion	0	0.1	0.2	0.4
x_5 # Crossover Points	0	1	2	3
x_6 Mutation Rate	Adaptive	0.001	0.0055	0.01
x_7 Precision Scaling	1/2	1	2	4

Table 1: Seven factors (control parameters) and four levels (parameter values) per factor investigated in the experiment.

Background

This problem is the result of over complication. When Holland first introduced genetic algorithms the idea was that GA's could be applied to a large variety of problems. Due to the complexities of GA parameter tweaking, this was never actually seen in practical use. The parameterless GA strives to show that the intensive part of making a GA can be taken away from the end-user and therefore open GA's to a wider audience with less technical knowledge.

Parameters are so inconsistent that they are never hard coded, opting for a configuration file instead. That way users can tweak them to get the appropriate settings, which can take substantial time and testing. The parameter-less GA strives to take that hassle away from the end-user and accomplish it automatically.

John Holland
1929-2015

The problem

Unoptimized GA's due to mistakes in the parameter setup can cause the overall performance to drop significantly. The parameter-less GA is able to define its own parameters dynamically and intelligently, allowing it to come to better solutions faster. By eliminating the error caused by the users unoptimized parameters, the parameter-less GA is both better and easier.

On the right is an example of a parameter-less GA compared to a regular GA in the Bounded Deceptive(4.3) problem. The dashed line is the parameter-less GA and the solid line is a regular GA

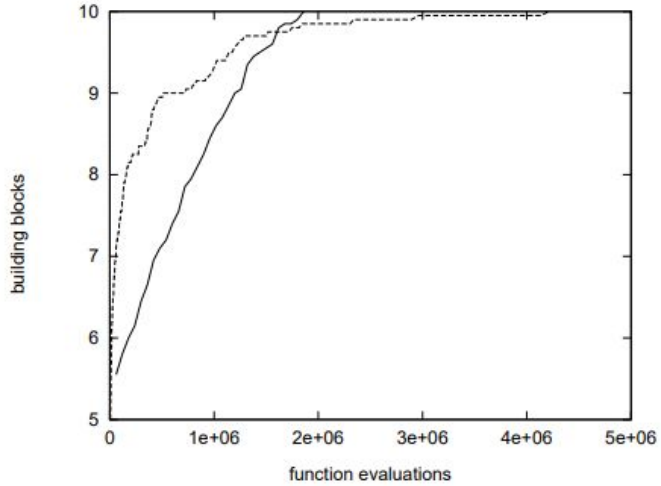

Figure 3: 10 copies of a 4-bit trap function. Both lines indicate the best solution found so far by each algorithm averaged over 20 runs. The solid line is for the regular GA. The dashed line is for the parameter-less GA.

Literature Review

Applications of this type of genetic algorithm have been used along with other improvements, such as in the paper Parameter-less Optimization with the Extended Compact Genetic Algorithm and Iterated Local Search by Cl´audio Lima and Fernando Lobo.

This paper focuses on a parameterless optimization framework, using extended compact genetic algorithms and iterated local searches. When combining all these algorithms, the performance improves significantly.

On the right is the results of some tests using each type of GA from that paper.

Table 1. Mean and standard deviation of the number of function evaluations spent to find the target solution for the tested problems. The number of runs (R_{ts}) in which a target solution was found was also recorded.

		SGA1	SGA2	ECGA	ILS	ILS+ECGA
Onemax	mean	2,990	1,256	13,735	451	451
	std. dev.	±189	±258	±5,371	±65	±65
	R_{ts}	20	20	20	20	20+0
Unimodal Himmelblau	mean	2,019	1,750	3,731	1,400	3,174
	std. dev.	±790	±497	±2,290	±1,385	±2,766
	R_{ts}	16	20	20	20	14+6
Four-peaked Himmelblau	mean	2,414	2,850	5,205	2,593	4,990
	std. dev.	±750	±668	±2,725	±3,002	±3,432
	R_{ts}	14	20	20	20	12+8
10-variable Rastrigin	mean	1,555,300	570,000	149,635	>2,000,000	275,170
	std. dev.	±306,600	±87,240	±85,608	—	±87,472
	R_{ts}	3	20	20	0	0+20
Bounded Deceptive	mean	—	741,000	15,388	>2,000,000	31,870
	std. dev.	—	±95,416	±3,417	—	±15,306
	R_{ts}	0	20	20	0	0+20

Experimentation

The simplest experiment done is called Onemax(4.1). This experiments goal is to acquire a string of all ones. In a regular GA, the parameters used are tournament size 2, crossover probability of 1, and an initial population size of 100.

The population size was decided using the population sizing equation:

$$N = \frac{-2^k \sqrt{\pi m \sigma_{BB}^2}}{2d} \ln \alpha$$

Where N is our population size, k is our building block size, m is the number of building blocks, σ^2bb is the building blocks fitness variance, and d is the building blocks fitness signal.

Experimentation Results

This figure shows how the parameter-less GA performed compared to a regular GA in the Onemax experiment. The dashed line represents our parameter-less GA while the solid line represents a regular GA.

We see that a completely optimized GA performs faster than the parameter-less GA, but this outcome was expected.

More importantly, we see that in the beginning, the parameter-less GA was performing better than the regular GA.

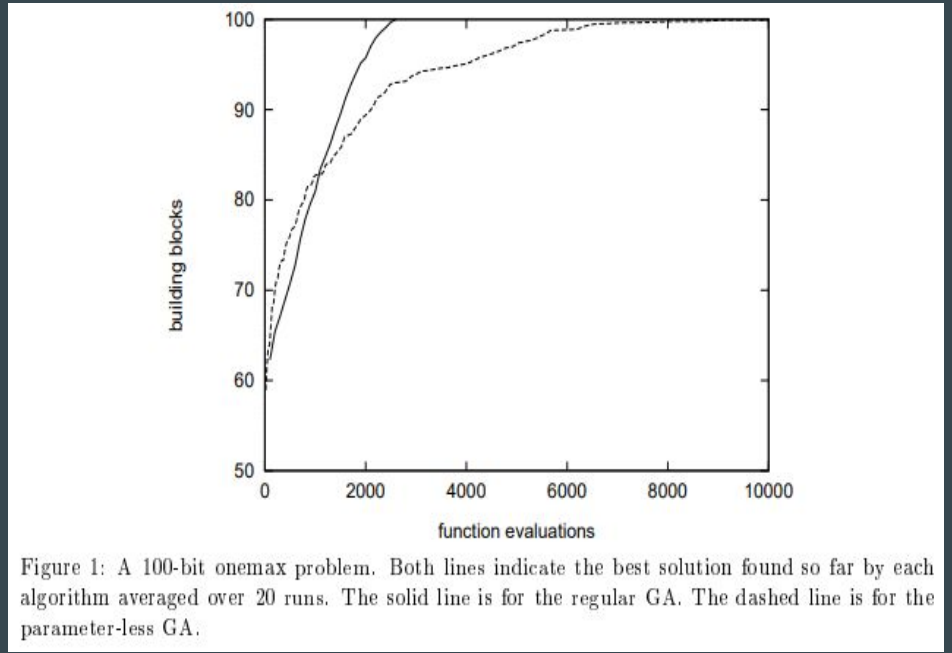

Figure 1: A 100-bit onemax problem. Both lines indicate the best solution found so far by each algorithm averaged over 20 runs. The solid line is for the regular GA. The dashed line is for the parameter-less GA.

Result Discussion

Based on the experiment results you would assume that the parameter-less GA is worse than the regular GA, but this is not what is to be taken away. The parameter-less GA was able to acquire a solution on its own without the predetermined parameters from a person. This core concept can be applied to more challenging problems, where the parameter definition is more complicated.

One such example is the noisey onemax problem, which is similar to before except noise can influence the fitness value. In this figure on the right, we see the parameter-less GA outpacing the regular GA.

The parameter-less GA is overall shown to scale better than the regular GA as complexity of the problem increases, and predetermined parameters begin to stagnate.

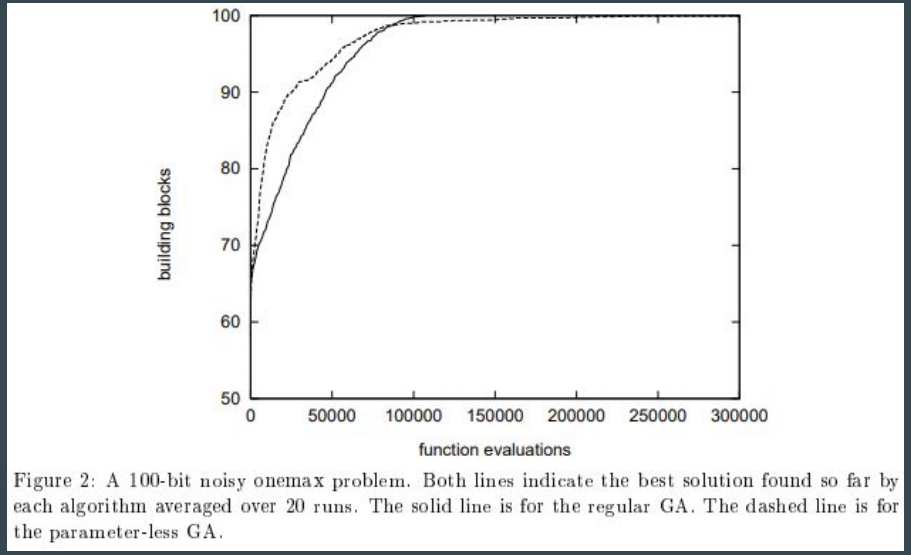

Figure 2: A 100-bit noisy onemax problem. Both lines indicate the best solution found so far by each algorithm averaged over 20 runs. The solid line is for the regular GA. The dashed line is for the parameter-less GA.

Conclusion

We see that it is possible to apply a parameter-less GA to various problems, and how on problems with complicated parameters, it can outperform a standard GA. The act of showing that it can find a solution is the main subject, as proving it works means it can be applied elsewhere.

The scalability of the parameter-less GA also shows that it can be applied to existing GA problems that are too complex for user-defined parameters.

Taking away the complexity that can make or break a GA allows for more widespread usage of GA's in other fields where previously it was deemed too difficult to implement.

Future Research

The next step in this would be to make some kind of GA that requires no programming from the end user at all. The end user can simply enter in their goal, and the program will automatically locate the most optimal result. Technology like this is soon to come, especially as machine learning continues to progress, allowing machines to make more accurate decisions based on analog inputs.

The more reasonable step though is to now work on improving the other algorithms within a GA, such as the optimal crossover function and other genetic operators. Once a machine can determine all of the best options for a given problem, the field will open up to more widespread use.

Acknowledgement

Thanks to my Jellycat espresso for listening to me read this powerpoint out loud to myself 9 times.

Also thanks to Peter for proofreading this.

Bibliography

Harik, G. R., & Lobo, F. G. (1999, January). A parameter-less genetic algorithm. In GECCO (Vol. 99, pp. 258-267).

Lima, C. F., & Lobo, F. G. (2004). Parameter-Less Optimization with the Extended Compact Genetic Algorithm and Iterated Local Search. Lecture Notes in Computer Science, 1328–1339. https://doi.org/10.1007/978-3-540-24854-5_127