Fate Agent Evolutionary Algorithms with Self-Adaptive Mutation

Arthur Ervin Avramiea
VU University Amsterdam
a.e.avramiea@student.vu.nl

Giorgos Karafotias
VU University Amsterdam
g.karafotias@vu.nl

A.E. Eiben
VU University Amsterdam
gusz@cs.vu.nl

ABSTRACT

Fate Agent EAs form a novel flavour or subclass in EC. The idea is to decompose the main loop of traditional evolutionary algorithms into three independently acting forces, implemented by the so-called Fate Agents, and create an evolutionary process by injecting these agents into a population of candidate solutions. This paper introduces an extension to the original concept, adding a mechanism to self-adapt the mutation of the Breeder Agents. The method improves the behaviour of the original Fate Agent EA on dynamically changing fitness landscapes.

Categories and Subject Descriptors

I.2.8 [**Problem Solving, Control Methods, and Search**]: Heuristic methods

Keywords

Distributed EAs, parameter control, dynamic problems

1. INTRODUCTION

Evolutionary computation implemented through Fate Agents is a new approach introduced in [2]. The motivational vision is grounded in Adaptive Collective Systems, for example in swarm robotics or, more generally, in distributed systems of autonomous agents, e.g. wireless sensor networks, smart devices, smart vehicles, etc. The main idea is to make such a collective system adaptive by injecting a force that can adjust the behavioural policies (a.k.a. controllers) of the individual units.

Such systems need to be able to adjust their initial pre-deployment settings to unexpected circumstances and they should be able to cope with changes. Using evolutionary techniques the swarm can be naturally considered as a population where each individual is (the controller of) a robotic unit. The "only" challenge is to add selection and variation operators that work on these individuals. To this end, it is important that the evolutionary process (i) is decentralised (hence scaleable), (ii) can self-calibrate its own parameters on the fly (needs no user tuning), and (iii) can cope with changes (can re-calibrate its parameters).

The Fate Agents approach attempts to achieve this by perceiving the controllers of the robots as individuals of a population and it "evolutionarizes" this system by adding three types of Fate Agents to the population. These agents implement the three principal evolutionary operators parent selection, reproduction, and survival selection, acting on the original individuals (robot controllers) *and on the Fate Agents themselves*.

Our work addresses the classic theme of adaptivity and parameter control in EC [6]. The Fate Agents EA is a spatially structured EA [8], [1], however, unlike most spatial EAs, it does not rely on a centralised "oracle". The combination of spatial structure and parameter control has been studied in [5] and [4]. Our system can also be related to meta-evolution [3], in particular the so-called local meta-evolutionary approaches [7] (not the meta-GA lookalikes).

2. FATE AGENTS

Our Fate Agents EA is situated in a (virtual) space where agents move and interact. The evolving population consists of passive Candidate Solutions and active Fate Agents that embody EA operators. Candidate Solution agents only carry a genome representing a solution to the given problem. Fate Agents embody evolutionary operators and evolve themselves because they act not only upon candidate solutions but also upon each other. They have a limited range of perception and action, thus, the evolutionary process is fully distributed as there is no central authority but different parts of the space are regulated by different agents. A Fate Agent is evaluated by the fitness of the fittest candidate solution in its range.

There are three types of Fate Agents. *Cupids* and *reapers* realise parent and survival selection respectively using tournaments. Their genome consists of tournament sizes and selection probabilities for each agent type. *Breeders* perform variation (recombination and mutation). In our initial experiments with numeric optimisation, breeders used Gaussian mutation[1] and their genome consisted of three values: the mutation step sizes for candidate solutions, cupids and reapers. A step size for mutating breeders was not included; breeders act differently upon themselves and upon other agent types. If the breeders' mutation step size was evolved within their genome it would be used to mutate itself leading to a positive feedback loop and exploding values. Instead, breeders are mutated using the Evolution Strategies' self-adaptation rule $x' = x \cdot e^{\tau N(0,1)}$, where τ is the *learning rate* constant.

3. ADAPTIVE BREEDERS LEARNING RATE

The constant learning rate for breeders introduces inflexibility to the otherwise self-regulated Fate Agents EA. Breeders are the source of adaptivity but they themselves have a constant rate of

[1] And averaging recombination that has no parameters.

mutation. This can be most restrictive when solving dynamic problems, especially in the case of cataclysmic changes in the environment when the population has to quickly respond to the new situation. For this reason we introduce the Adaptive Breeders Learning Rate(ABLR) mechanism. It affects the learning rate τ_{cs} used to mutate the step size for candidate solutions in the breeders' genomes.

Let F_t be the global best fitness at generation t and $G = F_t - F_{t-K+1}$ the overall growth for the past K generations. When $G \leq 0$, we increase τ_{cs} by a factor ℓ: $\tau_{cs}^{t+1} = \tau_{cs}^{t} \cdot \ell$. This results in an increased mutation rate for candidate solutions (σ_{cs}) for the newly created breeders since $\sigma'_{cs} = \sigma_{cs} \cdot e^{\tau_{cs} N(0,1)}$. When G becomes again positive τ_{cs} is reset it to its initial value τ_{init} and the mutation rates for all the breeders are reset to small random values in order to promote converge after the exploratory phase.

The value of K poses a typical exploration/exploitation tradeoff: if too small the ABLR mechanism will be activated too often not giving a chance for exploitation while if too high the exploration phases will be too long and waste time. Through initial experiments we have a found a balance for $K = 20$. The other default values for the ABLR settings are $\tau_{init} = 0.25$ and $\ell = 1.25$.

Because the ABLR mechanism can create high mutation rates, candidate solution gene boundaries are enforced using a bouncing approach to avoid having big parts of their population getting stuck at the limits of the search space instead of exploring.

Finally, in order for the ABLR mechanism to properly facilitate recovery in dynamic environments we imposed a limitation on the reapers' behaviour. We have previously noticed that reapers eradicate themselves when the EA converges to an optimum, thus, making recovery from a subsequent change impossible (since no space can be made for new individuals). For this reason, here reapers are allowed to kill an agent only if this agent's type comprises at least 10% of the total mixed population within the reaper's range.

4. EXPERIMENTS AND RESULTS

We conducted experiments using drastically changing environments to test if the ABLR mechanism improves the Fate Agents EA's ability to cope with change. We used three scenarios A to C where the fitness landscapes undergo complete transformations. Each scenario is divided into five epochs of length 250000 evaluations. During each epoch an entirely different fitness function is used but this function remains the same during a specific epoch. Thus each epoch ends with a cataclysmic change of the environment and there are four epochs (second to fifth) where the algorithm is evaluated after such a change. For scenario A, the Fletcher & Powell function was used with five different matrices for the five epochs.[2] For scenarios B and C the BBOB 2013 benchmark suite[3] was employed.

In table 1 we have compared the performance of the original algorithm and the algorithm with the adaptive breeder learning rate mechanism for the 3 scenarios. The results convincingly demonstrate the advantage of the ABRL method: for all three scenarios, the performance after a change is much higher. The differences can be small (one or two cases), but typically they are large, even one or two orders of magnitude (in eight cases).

To further assess the ABLR's quality we compared the expected performance of the algorithms when starting with a converged[4] population to the expected performance when starting with a fresh random population. The ABLR enhanced algorithm starting with a converged population achieved performance not significantly worse than when starting with a random population for eight out of 12 problems. The original algorithm achieved that only once.

[2]These matrices are available via the webpages of the authors.
[3]http://coco.gforge.inria.fr/doku.php?id=bbob-2013
[4]Converged to a very different previous landscape.

Table 1: Comparison of the original Fate Agent EA and the new version using the ABLR. Mean Best Fitness for each scenario/epoch calculated over the 40 runs is shown. All problems are maximisation: larger values are better

Scn	Algthm	Epo 1	Epo 2	Epo 3	Epo 4	Epo 5
A	Original	0.2285	0.001	0.0255	0.0008	0.0003
	ABLR	0.1963	0.1029	0.2371	0.0521	0.0083
B	Original	0.4631	0.1262	0.9994	0.0654	0.5963
	ABLR	0.4644	0.183	0.9998	0.6301	0.9172
C	Original	0.6158	0.0596	0.0361	0.298	0.024
	ABLR	0.5383	0.5667	0.4394	0.5685	0.1589

5. CONCLUSIONS

We aimed at improving the Fate Agents EA's ability to cope with changes. To this end, we redefined the working of the reproduction agents by adding the ABLR mechanism that makes the mutation operator they use self-adaptive. For an experimental assessment we used synthetic fitness landscapes and defined 3 scenarios composed of 5 epochs with cataclysmic changes between them.

The experimental results showed that the Fate Agents EA with the ABLR mechanism performs better than the original exhibiting much higher best fitnesses after each epoch. Furthermore, the ABLR enhanced algorithm very often makes a "full" recovery as opposed to the original that almost never does.

6. REFERENCES

[1] E. Alba and B. Dorronsoro. *Cellular Genetic Algorithms*. Springer, Berlin, Heidelberg, New York, 1st edition, 2008.

[2] J. Bim, G. Karafotias, S. K. Smit, A. E. Eiben, and E. Haasdijk. It's fate: A self-organising evolutionary algorithm. In C. A. C. Coello, V. Cutello, K. Deb, S. Forrest, G. Nicosia, and M. Pavone, editors, *PPSN*, volume 7491–7492 of *Lecture Notes in Computer Science*, pages 185–194. Springer, 2012.

[3] B. Freisleben. Meta-evolutionary approaches. In T. Bäck, D. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*, pages 214–223. Institute of Physics Publishing, Bristol, and Oxford University Press, New York, 1997.

[4] Y. Gong and A. Fukunaga. Distributed island-model genetic algorithms using heterogeneous parameter settings. In *IEEE Congress on Evolutionary Computation*, pages 820–827, 2011.

[5] V. Gordon, R. Pirie, A. Wachter, and S. Sharp. Terrain-based genetic algorithm (TBGA): Modeling parameter space as terrain. In W. Banzhaf *et al*, editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-1999)*, pages 229–235. Morgan Kaufmann, San Francisco, 1999.

[6] G. Karafotias, M. Hoogendoorn, and A. E. Eiben. Parameter control in evolutionary algorithms: Trends and challenges. *IEEE Transactions on Evolutionary Computation*, to appear, 2014.

[7] A. Samsonovich and K. De Jong. Pricing the 'free lunch' of meta-evolution. In H.-G. Beyer and U.-M. O'Reilly, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2005)*, pages 1355–1362. ACM, 2005.

[8] M. Tomassini. *Spatially Structured Evolutionary Algorithms: Artificial Evolution in Space and Time (Natural Computing Series)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.