Oversized Populations and Cooperative Selection: Dealing with Massive Resources in Parallel Infrastructures

Juan Luis Jiménez Laredo[1] and Bernabe Dorronsoro[3] and Carlos Fernandes[2,4] and Juan Julian Merelo[4] and Pascal Bouvry[1]

[1] FSTC-CSC/SnT
University of Luxembourg, Luxembourg
e-mail: {juan.jimenez,pascal.bouvry}@uni.lu
[2] Laseeb
Technical University of Lisbon, Portugal
e-mail: cfernandes@laseeb.org
[3] Laboratoire d'Informatique Fondamentale de Lille
University of Lille, France
e-mail: bernabe.dorronsoro_diaz@inria.fr
[4] Geneura Lab
University of Granada, Spain
e-mail: jmerelo@geneura.ugr.es

Abstract. This paper proposes a new selection scheme for Evolutionary Algorithms (EAs) based on altruistic cooperation between individuals. Cooperation takes place every time an individual undergoes selection: the individual decreases its own fitness in order to improve the mating chances of worse individuals. On the one hand, the selection scheme guarantees that the genetic material of fitter individuals passes to subsequent generations as to decrease their fitnesses individuals have to be firstly selected. On the other hand, the scheme restricts the number of times an individual can be selected not to take over the entire population. We conduct an empirical study for a parallel EA version where *cooperative selection* scheme is shown to outperform binary tournament: both selection schemes yield the same qualities of solutions but *cooperative selection* always improves the times to solutions.

Keywords: Selection schemes, evolutionary algorithms, parallelization, execution times

1 Introduction

We seek after a more efficient exploitation of massively large infrastructures in parallel EAs by balancing *population size* and *selection pressure* parameters. To that aim, we assume platforms in which the number of resources can be always considered *sufficient* (see e.g. [5]), i.e. large enough to allow a parallelized population to eventually converge to problem optima. The challenging issue here

is to make an efficient use of such resources since a too large population size can be considered oversized: a parametrization error leading to unnecessary wastes of computational time and resources [8]. We show that, in the case of oversized populations, selection pressure can be increased to high values in such a way that computing time is minimized and the solution quality is not damaged. Hence, the population sizing problem can be redefined into a twofold question that we call the *selection dominance* criterion; a selection scheme A can be said to dominate other selection scheme B if:

a) any arbitrary population size P sufficient to B is always sufficient to A. In our case, we set up the sufficiency criterion to the algorithm performing with a success rate (SR) greater or equal to 0.98 ($SR \geq 0.98$).
b) the execution time due to the pair *(A,P)* is strictly smaller than the execution time due to *(B,P)*.

Under the selection dominance perspective, it is a good practice to tune the selection pressure to maximum values which still respect the sufficiency criterion. Nevertheless, by doing that we may fall in the following well-known dilemma: On the one hand, a high selection pressure will eventually make the algorithm converge faster but with the risk of losing diversity and getting stuck in local-optima. On the other hand, a low selection pressure will improve the success rate expectations at the cost of a worse parallel execution time.

In the ideal case, a selection operator should be able to self-regulate the selection pressure according to the problem features and the given population size. However, in this paper, we limit the scope of the research to demonstrate a new selection scheme that is better than binary tournament ($s2$) in the sense that $s2$ solutions are always dominated: binary tournament is shown to be outperformed in execution times while both selection schemes have equivalent sizing requirements, i.e. same population sizes are *sufficient* in both cases.

The new selection scheme –introduced in section 2– is inspired by reciprocal altruism, a simple form of natural cooperation in which the fittest individuals decrease their own fitnesses in order to allow breedings of less fitter ones. An experimental analysis is conducted in section 3. Finally, some conclusions and future lines of research are exposed in section 4.

2 Cooperative Selection

The design of new selection schemes is an active topic of research in EC. In addition to canonical approaches such as ranking, roulette wheel or tournament selection [3], other selection schemes have been designed to trade off exploration and exploitation [1] or to be able to self-adapt the selection pressure on-line [2], just to mention a few. Cooperation has been also considered in the design of co-evolutionary EAs [9] in which sub-populations represent partial solutions to a problem and have to collaborate in order to build up complete solutions. However, to the extent of our knowledge, there have been no attempts for designing selection schemes inspired by cooperation.

Cooperative selection, in this early approach, is not more than a simple extension of the classical tournament selection operator. As in the latter, a set of randomly chosen individuals $\overrightarrow{s} = \{random_1(P), \ldots, random_s(P)\}$ compete for reproduction in a tournament of size s. The best ranked individual is then selected for breeding. The innovation of the new operator consists of each individual having two different measures for fitness. The first is the standard fitness function f which is calculated in the canonical way while the second is the cooperative fitness f_{coop} which is utilized for competing. Since we analyze the operator in a generational scheme context, at the beginning of every generation f_{coop} is initialized with the current fitness value (f) of the individual. Therefore, every first tournament within every generation is performed the same way as with the classical tournament selection. The novelty of the approach relies on the subsequent steps: after winning a competition of a tournament $\overrightarrow{s}$, the f_{coop} of the fittest individual is modified to be the average of the second and the third, which means that, in the following competitions, the winning individual will yield its position to the second. Since each f_{coop} is restarted with the f value every generation, it is likely that fitter individuals reproduce at least once per generation but without taking over the entire population. The details of the cooperative selection scheme are described in procedure 1.

Procedure 1 Pseudo-code of Cooperative Selection

procedure CooperativeSelection($\boldsymbol{s}$)

#1. Ranking step:
Competing individuals in $\overrightarrow{s}$ are ranked according to their cooperative fitnesses f_{coop}^{rank}
$rank(\overrightarrow{s}) \leftarrow \{f_{coop}^1, f_{coop}^2, f_{coop}^3, \ldots, f_{coop}^s\}$

#2. Competition step:
The individual with the highest cooperative fitness f_{coop}^1 is selected
$winner \leftarrow rank_1(\overrightarrow{s})$

#3. Altruistic step:
After being selected, the *winner* of the competition decreases its own fitness
$f_{coop}^1 \leftarrow \frac{f_{coop}^2 + f_{coop}^3}{2}$

return *winner*

end procedure

As in *tournament selection*, the only parameter to adjust in *cooperative selection* is the tournament size. We performed preliminary studies in order to tune such a parameter. In those experiments, we found out that a cooperative tournament size of 16 (*Coop s16*) is equivalent to binary tournament $s2$ in terms of selection pressure. Therefore, all experiments will be conducted for such a parameter value.

3 Analysis of results

In order to analyze the performance of the cooperative selection scheme, we conduct simple experiments in a master-slave GA [4] and tackle an instance of length $L = 200$ of the onemax problem [10]. The GA, in addition to be parallel, follows a 1-elitism generational scheme. Besides, only a simple point crossover was considered as breeding operator[1] and three different types of selection parametrization: two of them using tournament selection with tournament sizes of 2 ($s2$) and 16 ($s16$), and one using cooperative selection with a tournament size of 16 (*Coop s16*). Then every setting is analyzed for population sizes scaling so that the SR can be estimated and the sufficiency criterion met. Our method to estimate optimal population sizes starts with a population size of $P = 40$, doubling P in every step until $P = 20480$. Each parametrization is run independently 50 times so that a fair estimation of the success rate (SR) can be made.

For the sake of simplicity, we assume that each individual is sent for evaluation to a single processor so as to apply the following execution time metrics [6]:

- T_{sec}: is the sequential optimization time and refers to the number of function evaluations until the first global optimum is evaluated.
- T_{par}: is the parallel optimization time and accounts for the number of generations it takes to find the problem optimum.

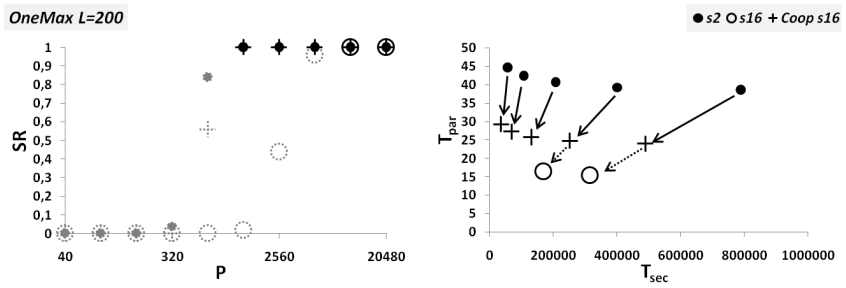
Figure 1(a) shows the scalability of the SR with respect to the population size for the three different selection operators. The SR scales in all cases with a *sigmoid* shape in which smaller population sizes perform poorly and larger ones reach the sufficiency $SR \geq 0.98$. The only remarkable difference between parametrizations rely on the transition phase of the sigmoid. Such transitions occur much earlier in $s2$ and *Coop s16* than in $s16$. This allows smaller populations to be sufficient for the formers while not for the latter.

In figure 1(b), the analysis of the trade offs T_{sec}/T_{par} shows that, whenever a given population size is sufficient in the three settings, the winning strategy is $s16$, i.e. the maximum selection pressure. Nevertheless, such results do not imply that $s16$ *dominates* $s2$ since $s16$ requires of larger population sizes to achieve better performance. *Coop s16*, however, outperforms parallel and sequential times of $s2$ while having the same population requirements. Therefore, it can be said that *Coop s16* dominates $s2$ under any setting.

4 Conclusions and Future Works

In this paper, we have proposed the *cooperative selection* scheme, an extension of tournament selection that, by implementing an altruistic behavior in winners of the competitions, is able to outperform binary tournament. Without assuming

[1] We follow here the selectorecombinative approach of Lobo and Lima [7] for studying the scalability of the population size.



(a) *Scalability of the success rate (SR) as a function of the population size (P). Marks in bold show a* $SR \geq 0.98$.

(b) *Tradeoff between* T_{sec} *and* T_{par} *for* $SR \geq 0.98$ *and different population and tournament sizes.*

Fig. 1. Scalability of a master-slave GA tackling an instance of the onemax problem with size L = 200. Parameters s2 and s16 represent different tournament sizes of 2 and 16 respectively and *Coop s16* stands for cooperative selection with a tournament size of 16. Arrows in sub-figure *(b)* indicate a *"dominated by"* relationship between selection schemes using equal population sizes. Some circles for $s16$ are missing in *(b)* as this setting does not yield *sufficiency* ($SR \geq 0.98$) for population sizes smaller than 10240.

any knowledge on the problem domain, *cooperative selection* is shown to outperform the utilization of parallel resources in a simple test case: given identical population sizes (i.e. same computing platform), *cooperative selection* saves computational efforts with respect to binary tournament and requires of less parallel execution time to yield the same quality in solutions.

As a future work, we plan two major lines of research. The first is the straightforward application of *cooperative selection* to massively parallel EAs as in the case of GPU-based EAs or as in volunteer-computing-based EAs. The second line of research is related to high-dimensional real-parameter optimization. Here we think that *cooperative selection* could perform well since, on the one hand, typical frameworks for benchmarking usually impose restrictions on time and, on the other hand, problems with high-dimensionality require of large amounts of resources in order to minimize optimization errors.

Acknowledgments

This work was supported by the Luxembourg FNR Green@Cloud project (INTER/CNRS/11/03) and by the Spanish Ministry of Science Project (TIN2011-28627-C04). B. Dorronsoro acknowledges the support by the Fonds National de la Recherche, Luxembourg (AFR contract no 4017742).

References

1. Enrique Alba and Bernabé Dorronsoro. The exploration/exploitation tradeoff in dynamic cellular genetic algorithms. *IEEE Trans. Evolutionary Computation*, 9(2):126–142, 2005.
2. A. E. Eiben, M. C. Schut, and A. R. De Wilde. Boosting genetic algorithms with self-adaptive selection. In *In Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1584–1589, 2006.
3. Agoston E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. SpringerVerlag, 2003.
4. Daniel Lombraña Gonzalez, Juan Luis Jiménez Laredo, Francisco Fernández de Vega, and Juan Julián Merelo. Characterizing fault-tolerance of genetic algorithms in desktop grid systems. In Peter I. Cowling and Peter Merz, editors, *EvoCOP*, volume 6022 of *Lecture Notes in Computer Science*, pages 131–142. Springer, 2010.
5. Juan Luis Jiménez Laredo, A. E. Eiben, Maarten van Steen, and Juan Julián Merelo Guervós. Evag: a scalable peer-to-peer evolutionary algorithm. *Genetic Programming and Evolvable Machines*, 11(2):227–246, 2010.
6. Jörg Lässig and Dirk Sudholt. General scheme for analyzing running times of parallel evolutionary algorithms. In Robert Schaefer, Carlos Cotta, Joanna Kolodziej, and Günter Rudolph, editors, *Parallel Problem Solving from Nature, PPSN XI*, volume 6238 of *Lecture Notes in Computer Science*, pages 234–243. Springer Berlin / Heidelberg, 2010.
7. Fernando Lobo and Claudio Lima. Adaptive population sizing schemes in genetic algorithms. In Fernando Lobo, Claudio Lima, and Zbigniew Michalewicz, editors, *Parameter Setting in Evolutionary Algorithms*, volume 54 of *Studies in Computational Intelligence*, pages 185–204. Springer Berlin / Heidelberg, 2007.
8. Fernando G. Lobo and David E. Goldberg. The parameter-less genetic algorithm in practice. *Inf. Sci. Inf. Comput. Sci.*, 167(1-4):217–232, December 2004.
9. Mitchell A. Potter and Kenneth A. De Jong. A cooperative coevolutionary approach to function optimization. In *Proceedings of the 3rd International Conference on Parallel Problem Solving from Nature*, pages 249–257. Springer-Verlag, 1994.
10. J. David Schaffer and Larry J. Eshelman. On crossover as an evolutionarily viable strategy. In Richard K. Belew and Lashon B. Booker, editors, *ICGA*, pages 61–68. Morgan Kaufmann, 1991.