Algorithm configuration using GPU-based metaheuristics

Roberto Ugolotti, Youssef S.G. Nashed, Pablo Mesejo, Stefano Cagnoni

Algorithm Configuration using GPU-based Metaheuristics

Roberto Ugolotti, Youssef S.G. Nashed, Pablo Mesejo, Stefano Cagnoni
Department of Information Engineering, University of Parma, Italy
{rob_ugo, nashed, pmesejo, cagnoni}@ce.unipr.it

ABSTRACT

In this paper, a GPU-based implementation of Differential Evolution (DE) and Particle Swarm Optimization (PSO) in CUDA is used to automatically tune the parameters of PSO.

The parameters were tuned over a set of 8 problems and then tested over 20 problems to assess the generalization ability of the tuners. We compare the results obtained using such parameters with the 'standard' ones proposed in the literature and the ones obtained by state-of-the-art tuning methods (`irace`). The results are comparable to the ones suggested for the standard version of PSO (SPSO), and the ones obtained by `irace`, while the GPU implementation makes tuning time acceptable.

To the best of our knowledge, this is the first time that a general purpose library of GPU-based metaheuristics is used to solve this problem, as well as being one of the few cases where DE and PSO are both used as tuners.

Categories and Subject Descriptors

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

Keywords

Parallel Computing; GPGPU programming; Automatic Parameter Configuration; Evolutionary Algorithm; Swarm Intelligence

1. INTRODUCTION

One of the main problems to be tackled when applying Evolutionary Algorithms (EAs) and Swarm Intelligence (SI) techniques is the strong influence of parameter selection on their performance.

In this paper, we use the GPU implementation of two real-valued population-based optimization techniques (Particle Swarm Optimization (PSO) [2] and Differential Evolution (DE) [4]) to automatically configure another metaheuristic (PSO). This SI method is tuned using a benchmark of 8 different optimization problems (also implemented on GPU).

The main contribution of this work is three-fold: (i) the first documented use, to the best of our knowledge, of GPU-based metaheuristics devoted to the automatic configuration of algorithms, (ii) a comparison with the results obtained using the parameters selected by the `irace` software package [3], and using a classically and well-established set of manually selected parameters [1]), and (iii) the use of metaheuristics not only to tune numerical parameters but also to make some qualitative decision about the design of the algorithm to tune, by choosing categorical/nominal parameters (as, for instance, the PSO topology).

Figure 1: Representation of a set of PSO parameters inside the tuner. The first four elements represent numerical parameters (either integer or real-valued numbers), while the last three contribute to the decision about a nominal parameter (in this case topology).

2. PROPOSED METHOD

The implementation of the metaheuristics employed in this paper has been presented in [5]. The code used in this research is available at http://sourceforge.net/p/libcudaoptimize [6].

When DE or PSO are used as tuner, they work as any other optimizer. The only difference lies in fitness evaluation and how fitness values are stored and used. Each particle in a tuner represents a set of parameters of the optimizer to tune (in this case PSO). This optimizer is tested T times on a set of F functions. The resulting fitness is then computed based on the average and standard deviation of the best fitness obtained over all functions; that is a $2 \times F$ array of values. A set of parameters is better than another if it obtains better results in the majority of the F functions. In case of a tie, the winner is selected comparing the sum of Welch's t test values over the N functions under consideration:

$$R = \sum_{i=1}^{N} \frac{(\overline{X}^i{}_a - \overline{X}^i{}_b)}{\sqrt{\frac{{s^i}_a{}^2 + {s^i}_b{}^2}{T}}}$$

where a and b are the two sets of parameters (or the two optimizers), $\overline{X}^i$ and s^i are the mean and standard deviation obtained by each optimizer on function i. If $R < 0$, the optimizer a is better (since we are looking for the minima of the functions), otherwise b wins. During the tuners' execution, optimizers are compared pairwise while, at the end of the optimization process, the surviving particles are compared with one another in a full tournament to choose the winner.

PSO and DE are real-valued optimization methods, so the problem of representing nominal parameters needs to be addressed. Our choice has been to represent each nominal value by a vector with as many values as the options available and choose the parameter corresponding to the element with the highest value. For instance, one of the parameters we want to optimize in PSO is topology. We can

Table 1: Results (average and standard deviation) obtained on the complete dataset of 20 functions (see [5] for the complete definition) and using the best configuration obtained by all tuners. All functions were optimized using 30 dimensions and 30 runs per function. Due to lack of space, only the functions that showed statistically significant differences are included.

	DE		PSO		irace		Standard	
	Avg	Std	Avg	Std	Avg	Std	Avg	Std
f_4	2.57e-045	1.43e-045	0.00e+000	0.00e+000	0.00e+000	0.00e+000	9.34e-046	8.50e-046
f_7	5.86e+000	3.56e-001	6.35e+000	4.42e-001	6.31e+000	4.30e-001	6.17e+000	4.08e-001
f_{10}	4.43e+003	1.63e+003	4.07e+003	2.09e+003	3.98e+003	1.52e+003	9.27e+003	3.10e+003
f_{13}	2.11e+001	1.50e+000	1.38e+001	8.51e-001	1.55e+001	2.64e+000	2.10e+001	4.45e+000
f_{19}	1.46e-001	6.37e-001	4.25e-001	6.54e-001	4.71e-001	7.74e-001	5.34e-001	9.68e-001

Table 2: Best automatically-tuned parameter values found by DE, PSO, and irace, in 10 runs of every tuner, and 10 of PSO with standard parameters. See [5] for a definition of the symbols.

Param.	DE	PSO	irace	SPSO [1]
C_1	0.953	1.525	1.243	1.496
C_2	2.429	1.881	2.222	1.496
w	0.636	0.443	0.351	0.730
Pop. Size	37	34	35	60
Topology	gbest	gbest	gbest	gbest

choose between ring, star and "global-best" topology. This means that three values for each particle in the PSO/DE tuner are reserved to this choice and the largest one in the best particle is chosen as winner (see Figure 1). The range of the PSO parameters allowed by the tuning algorithm are $C_1, C_2 \in [0, 4]$, $w \in [0, 1.5]$, Pop. size $\in [32, 150]$, and topology $\in Gbest, Ring, Star$.

3. RESULTS AND CONCLUSIONS

We performed parameter tuning on a training set of 8 functions (numbers 0, 5, 6, 8, 9, 10, 12, 13 in [5]) and a validation set of 20. Our goal was to check (i) whether our method was able to get results competitive with other more complex and/or computationally expensive state-of-the-art tuners (irace), (ii) whether we could find repeatable patterns in the parameter sets obtained by different techniques, and (iii) the generalization ability of every tuner.

The comparison was performed by running 10 times our tuners and irace, and searching the best population size and topology with the parameter values for C_1, C_2, w suggested in [1]. We then selected the best parameter configuration found by each method, based on the Friedman test and the visual examination of the results using boxplots. The resulting configurations, shown in Table 2, are similar: all of them use global best topology, a small population size, an inertia value between 0.35 and 0.65, while extreme values are rejected. Moreover, all of them have $C_2 > C_1$, pointing out that, given a global topology, a good configuration relies on social information to reach good results. The standard parameters seem to need a larger population to balance the smaller importance of the "collaboration between particles" factor: one could say that there is less cooperation between particles, but this fact is attenuated by the larger swarm size.

Table 1 reports the results obtained by the best configuration found by each tuner (best set of SPSO parameters) for the functions where there are statistically significant differences among the optimizers. The first column is the function under consideration. The following ones are divided into four blocks, each of which corresponds to either a tuner or SPSO. Within each block, the average best fitness and the standard deviation over all runs are reported. Results reported on a gray background highlight those cases in which the median over 30 runs obtained by one method is significantly better than the one yielded by the other methods, according to the Friedman test, with a level of significance of 0.01. As post-hoc procedure, the Dunn-Sidak correction was applied to study the existence of pairwise statistical differences.

As expected, the results are very similar, since the resulting classifiers are similar to each other: they all use global topology, a similar population size while the setting of parameters C_1, C_2 and w lie within very small ranges. It should be noticed that the three tuned optimizers reach good results also in functions that were not included in the training phase, showing good generalization.

In conclusion, we can say that simple EA-based parameter tuning is able to get comparable results to a well-established state-of-the art method in parameter tuning, like irace. Moreover, in our case, the parallel implementation makes the tuning process computationally efficient, which allows our tuners to perform their task in a much more reasonable time with respect to sequential implementations.

Acknowledgments

Roberto Ugolotti is funded by Compagnia di S.Paolo and Fondazione Cariparma. Youssef S. G. Nashed, and Pablo Mesejo are funded by the European Commission (Marie Curie ITN MIBISOC, FP7 PEOPLE-ITN-2008, GA n. 238819). The authors want to thank Marco Grossi and Gian Maria Marconi for their precious help.

4. REFERENCES

[1] J. Kennedy and M. Clerc, 2006. Online: http://www.particleswarm.info/Standard_PSO_2006.c.

[2] J. Kennedy and R. Eberhart. Particle Swarm Optimization. In *Proc. IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, 1995.

[3] M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, and M. Birattari. The irace package, iterated race for automatic algorithm configuration. Technical Report TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium, 2011.

[4] R. Storn and K. Price. Differential Evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report, International Computer Science Institute, 1995.

[5] Y. S. G. Nashed, P. Mesejo, R. Ugolotti, J. Dubois-Lacoste, and S. Cagnoni. A comparative study of three GPU-based metaheuristics. In *Parallel Problem Solving from Nature - PPSN XII*, volume 7492 of *Lecture Notes in Computer Science*, pages 398–407. Springer, 2012.

[6] Y. S. G. Nashed, R. Ugolotti, P. Mesejo, and S. Cagnoni. libCudaOptimize: an open source library of GPU-based metaheuristics. In *Proceedings of the 14th international conference on Genetic and evolutionary computation conference (GECCO) companion*, pages 117–124, 2012.