A New Approach to Distribute MOEA Pareto Front Computation

Federica Sarro, Alessio Petrozziello, Dan-Qi He
{f.sarro,danqi.he.17,a.petrozziello}@ucl.ac.uk
University College London

Shin Yoo
shin.yoo@kaist.ac.kr
Korea Advanced Institute of Science and Technology

ABSTRACT

Multi-Objective Evolutionary Algorithms (MOEAs) offer compelling solutions to many real world problems, including software engineering ones. However, their efficiency decreases with the growing size of the problems at hand, hindering their applicability in practice.

In this paper we propose a novel master-worker approach to distribute the computation of the Pareto Front (PF) for MOEAs (dubbed MOEA-DPF) and empirically evaluate it on a real-world software project management problem. With respect to previous work, our proposal can be used with any MOEA to tackle multi-objective problems regardless of their formulation/representation.

Our results show that MOEA-DPF runs significantly faster (up to 3.1x speed-up using two workers) than its sequential counterpart while maintaining (and even improving) the quality of the PF. We conclude that MOEA-DPF provides an effective and simple solution to speed-up the execution of MOEAs by distributing the PF computation, making them effective for real-world problems.

CCS CONCEPTS

• **Software and its engineering → Search-based software engineering**; • **Computing methodologies → Distributed algorithms**;

KEYWORDS

Distributed Pareto Front, Multi-Objective Evolutionary Algorithms, Search Based Software Engineering, Software Project Management.

ACM Reference Format:

Federica Sarro, Alessio Petrozziello, Dan-Qi He and Shin Yoo. 2020. A New Approach to Distribute MOEA Pareto Front Computation. In *Genetic and Evolutionary Computation Conference Companion (GECCO '20 Companion), July 8–12, 2020, Cancún, Mexico.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3377929.3390024

1 INTRODUCTION

Multi-Objective Evolutionary Algorithms (MOEAs) have been successfully applied to the optimisation of real-world problems [15] and not least to Search-Based Software Engineering (SBSE).

The success of MOEAs for SBSE is mainly due to their ability to provide high quality solutions to the problems at hand, at the same time providing the engineers with interesting trade-offs among competing conflicting goals. MOEAs' attractiveness to software engineers is testified by the fact that they have recently started gaining industrial uptake based on very compelling and human-competitive solutions compared to more traditional Software Engineering approaches [1, 2, 11, 14]. In order to be practicable, these solutions need to be deployable and work at scale in continuos integration environments. However, strong limitations to these approaches rise with the growing size of the problem at hand and the time needed towards the evaluation of the fitness function, which could make the optimisation too slow and not adoptable in practice [9].

In the last few years, emerging frameworks for distributed computing have been used to mitigate this problem. For example, the Map-Reduce paradigm adopted by Hadoop has been used for the parallelisation of single objective Genetic Algorithms [5, 7, 8, 12], where the main gain is in the distributed evaluation of a single fitness function. When moving towards the need to optimize two or more objectives, the fitness evaluation itself is no longer the only expensive component [3]: The construction of the Pareto Front (PF) becomes computationally demanding [4].

This paper aims to advance the state-of-the-art by proposing a simple yet effective technique to distribute the computation of their PF and by empirically assessing its effectiveness for a real-world SBSE problem [6, 13] using Apache Spark. Our proposal, named MOEA-DPF, has the main advantage of being independent from specific multi-objective problem formulations as well as the MOEA used. Moreover, MOEA-DPF does not require any modification to the original multi-objective optimisation problem and can be applied regardless of data types of the genotype representation, as it instead was required by previous proposals [4, 10, 16]. The results of our study reveal that MOEA-DPF (using two workers) is up to 3.1x times faster than its sequential version and maintains (and in some cases significantly improves) the quality of the PF.

2 OUR PROPOSED MOEA-DPF

Our proposal, MOEA-DPF, is a master-worker approach that uses worker nodes (a.k.a. islands) to run independent MOEAs on a subset of the original population, and a master node to periodically compute a global PF from the local PFs sent by the workers, and to communicate back to each worker their own solutions which are globally dominated. Thus, the key aspect and novelty of MOEA-DPF is the interaction between the master and the workers: Exchanging PFs, rather than individual chromosomes, allows to periodically replace solutions that are optimal on a certain worker but globally dominated by solutions residing on other workers. We explain the approach in more detail below.

A master program generates a population of size p and assigns a fixed set of p/n solutions to each of the n islands (i.e. workers). Each worker locally executes an MOEA (independently from the other nodes) on its p/n solutions and it sends its local Pareto Front to the master at every m iterations (i.e. migration rate). Once the master has received the local PFs from all workers, it computes a global PF from their union and determines all those solutions which

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
GECCO '20 Companion, July 8–12, 2020, Cancún, Mexico
© 2020 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-7127-8/20/07.
https://doi.org/10.1145/3377929.3390024

are locally optimal but globally dominated by solutions produced by other workers. This allows us to compute a globally dominated solution set for each worker node, which is then used along with a replacement policy to remove globally dominated solutions from each local MOEA population (i.e. running on a worker). An example of a replacement policy is replacing globally dominated solutions with random ones. Once the replacement policy is applied, each worker continues the execution of its own MOEA until the next migration occurs. The evolutionary process is terminated by the master using a given termination criterion, for example, when a maximum number of fitness evaluations is reached.

MOEA-DPF is highly customisable by choosing (i) the MOEA executed by each worker; (ii) the number of workers; (ii) the replacement policy, which determines how many dominated solutions should be replaced and how; (iv) the migration rate; (v) the framework used for the distributed computation.

3 EMPIRICAL EVALUATION

We have empirically investigated the effectiveness of our proposed MOEA-DPF on a real-world multi-objective software engineering problem: the overtime planning problem [6, 13]. This problem seeks for optimal overtime assignments, which are able to minimise project duration, average overrun risk and total amount of overtime deployed. In particular, we ran MOEA-DPF on three different software projects (i.e. `Web`, `Price` and `Broker` [13]) and assessed whether the quality of its PFs is at least as good as the quality of those produced by its sequential counterpart (i.e. MOEA-SPF), and whether MOEA-DPF executes faster than MOEA-SPF. We measure the PF's quality with Hypervolume and Generational Distance, and execution time as wall clock time. Due to the stochastic nature of MOEAs, we run 20 independent runs per algorithm and check for statistically significant differences using the Wilcoxon signed-rank test ($\alpha < 0.05$ with Bonferroni correction). To conduct the computational search, we used NSGAII$_a$ (as originally done by Sarro et al. [13]) with 1,000 individuals and 250,000 fitness evaluations for both MOEA-DPF and MOEA-SPF. For MOEA-DPF, we used the Best Fitted Reproduction Replacement policy (which replaces the globally dominated solutions with an equal number of solutions generated via crossover between globally non-dominated solutions randomly chosen from different workers), a migration rate equal to 100 generations, and two workers. NSGAII$_a$-SPF was implemented using jMetal v. 5 [13], while its parallel version (NSGAII$_a$-DPF) was implemented by extending the jMetal framework with Apache Spark. We ran all the experiments on the Amazon AWS c5.2xlarge instance (8vCPU and 16GB RAM). The results of the comparison (see Table 1) show that NSGAII$_a$-DPF produces similar or even significantly better PFs than NSGAII$_a$ with a significantly lower execution time (p-values < 0.001). The average speed-up achieved by NSGAII$_a$-DPF using two workers is 2.3x for the `Broker` project, 2.4x for the `Web` project and 3.1x for the `Price` project.

4 CONCLUSION AND FUTURE WORK

In this paper, we have proposed MOEA-DPF, a simple yet effective approach to distribute the computation of MOEA Pareto Front. Our proposal is applicable to any MOEA and multi-objective optimisation problem, regardless of its formulation and representation.

Table 1: Hypervolume, Generational Distance and Execution Time of NSGAII$_a$-SPF and NSGAII$_a$-DPF for each of the three benchmark projects. *Entry: mean/st. dev (p-value).*

Project	Algorithm	Hypervolume	Generational Distance	Time (seconds)
`Web`	NSGAII$_a$-SPF	0.51/0.07	0.01/0.01	1761.10/8.26
	NSGAII$_a$-DPF	0.51/0.08 (0.46)	0.01/0.00 (0.17)	728.24 /4.44 (< 0.001)
`Price`	NSGAII$_a$-SPF	0.52/0.07	0.01/0.00	748.86 /6.17
	NSGAII$_a$-DPF	0.56/0.10 (0.07)	0.01/0.01 (0.10)	243.27 /1.10 (< 0.001)
`Broker`	NSGAII$_a$-SPF	0.34/0.08	0.03/0.01	3518.37/17.48
	NSGAII$_a$-DPF	0.43/0.08 (0.00)	0.01/0.01 (0.00)	1552.46/2.99 (< 0.001)

We empirically evaluated the effectiveness of MOEA-DPF by using NSGAII*a* to tackle a multi-objective problem from the software engineering domain, namely the software project overtime problem [6, 13]. Results show that MOEA-DPF runs significantly faster (up to 3.1x speed-up using only two workers) than its sequential counterpart and produces similar (or even better) Pareto Fronts. Future work will investigate MOEA-DPF with different replacement policies, migration rates, number of workers, computing frameworks, as well as its use for other MOEAs and optimisation problems.

ACKNOWLEDGMENTS

Work supported by Amazon AWS grant and ERC grant no. 741278.

REFERENCES

[1] N. Alshahwan, A. Ciancone, M. Harman, Y. Jia, k. Mao, A. Marginean, A. Mols, H. Peleg, F. Sarro, and I. Zorin. 2019. Some challenges for software testing research (invited talk paper). In *Procs. of ISSTA*. 1–3.

[2] N. Alshahwan, X. Gao, M. Harman, Y. Jia, K. Mao, A. Mols, T. Tei, and I. Zorin. 2018. Deploying Search Based Software Engineering with Sapienz at Facebook. In *Procs. of SSBSE*.

[3] C. A. Coello Coello. 2015. *Multi-objective Evolutionary Algorithms in Real-World Applications: Some Recent Results and Current Challenges.* 3–18.

[4] K. Deb, P. Zope, and A. Jain. 2003. Distributed Computing of Pareto-Optimal Solutions with Evolutionary Algorithms. In *In Procs. of EMO*. 534–549.

[5] L. Di Geronimo, F. Ferrucci, A. Murolo, and F. Sarro. 2012. A Parallel Genetic Algorithm Based on Hadoop MapReduce for the Automatic Generation of JUnit Test Suites. In *Procs. of ICST*. 785–793.

[6] F. Ferrucci, M. Harman, J. Ren, and F. Sarro. 2013. Not Going to Take This Anymore: Multi-objective Overtime Planning for Software Engineering Projects. In *Procs. of ICSE*. 462–471.

[7] F. Ferrucci, P. Salza, MT. Kechadi, and F. Sarro. 2015. A Parallel Genetic Algorithms Framework Based on Hadoop MapReduce. In *Procs. of ACM SAC*. 1664–1667.

[8] F. Ferrucci, P. Salza, and F. Sarro. 2017. Using Hadoop MapReduce for Parallel Genetic Algorithms: A Comparison of the Global, Grid and Island Models. *Evolutionary Computation* 25, 1 (2017), 1–54.

[9] M. Harman. 2007. The Current State and Future of Search Based Software Engineering. In *Proc. of FOSE 2007*. 342–357.

[10] A L. Jaimes and CA Coello Coello. 2005. MRMOGA: Parallel evolutionary multiobjective optimization using multiple resolutions. In *Procs. of CEC*. IEEE, 2294–2301.

[11] B. Marculescu, R. Feldt, R. Torkar, and S. Poulding. 2018. Transferring interactive search-based software testing to industry. *JSS* 142 (2018), 156 – 170.

[12] P. Salza, F. Ferrucci, and F. Sarro. 2016. Elephant56: Design and Implementation of a Parallel Genetic Algorithms Framework on Hadoop MapReduce. In *Procs. of GECCO Companion*. 1315–1322.

[13] F. Sarro, F. Ferrucci, M. Harman, A. Manna, and J. Ren. 2017. Adaptive Multi-Objective Evolutionary Algorithms for Overtime Planning in Software Projects. *IEEE TSE* 43, 10 (2017), 898–917.

[14] F. Sarro, A. Petrozziello, and M. Harman. 2016. Multi-objective software effort estimation. In *Procs. of ICSE*. 619–630.

[15] T. Stewart, O. Bandte, H Braun, N. Chakraborti, M. Ehrgott, M. Göbelt, Y. Jin, H. Nakayama, S. Poles, and D. Di Stefano. 2008. Real-world applications of multiobjective optimization. In *Procs. of EMO*. 285–327.

[16] S. Yoo, M. Harman, and S. Ur. 2013. GPGPU test suite minimisation: Search Based Software Engineering Performance Improvement using Graphics Cards. *EMSE* 18, 3 (2013), 550–593.