Fast Pareto Front Approximation for Cloud Instance Pool Optimization

Ana-Maria Oprescu
Dept. of Computer Science
Vrije Universiteit, Amsterdam,
The Netherlands
A.M.Oprescu@vu.nl

Alexandra (Vintila) Filip
Dept. of Computer Science
Vrije Universiteit, Amsterdam,
The Netherlands
A.A.Vintila@vu.nl

Thilo Kielmann
Dept. of Computer Science
Vrije Universiteit, Amsterdam,
The Netherlands
Thilo.Kielmann@vu.nl

ABSTRACT

Computing the Pareto Set (PS) of optimal cloud schedules in terms of cost and makespan for a given application and set of cloud instance types is NP-complete. Moreover, cloud instances' volatility requires fast PS recomputations. While genetic algorithms (GA) are a promising approach, little knowledge of an approximated PS's quality leads to GAs running for *overly* many generations, contradicting the goal of *quickly* computing an approximate solution. We address this with MOO-GA, our GA enhanced with a *domain-tailored termination criteria delivering fast, well-approximated Pareto sets.* We compare to NSGAIII using PS convergence and diversity, and computational effort metrics. Results show MOO-GA consistently computing better quality Pareto sets within *one second* on average (df=98, p-value$<10^{-3}$).

CCS Concepts

•Networks → Cloud computing; •Computing methodologies → Genetic algorithms; •Mathematics of computing → Evolutionary algorithms;

1. INTRODUCTION

Dominant in high-throughput computing, bag-of-tasks applications are computationally demanding and may seem an ideal match for commercial cloud offerings [1]. However, allocating *the right number* of instances, of *the right type*, for *the right time*, strongly depends on the application, and is left to the user. State-of-the-art cloud scheduler BaTS [2] and its fast GA approximate within seconds Pareto sets of makespan-cost options [3]. However, this GA lacks *an adaptive termination criterion enforcing solution quality.*

The well-known GA termination problem is difficult. State-of-the-art algorithms [4, 5], including NSGAIII [6], still use either a maximum number of generations or a maximum number of objective function evaluations. Some [7, 8, 9] studied meta performance indicators, statistically detecting their convergence. In contrast, we satisfy the need for quality assessment reflecting the problem domain [10] with a new termination criteria, based on our problem domain: cloud scheduling. We develop MOO-GA by enhancing BaTS' GA with our new termination criteria, to achieve *fast, quality-controlled cloud instance pool optimization.*

GECCO '15 July 11-15, 2015, Madrid, Spain

ACM ISBN 978-1-4503-3488-4/15/07.
DOI: http://dx.doi.org/10.1145/2739482.2764720

We compare MOO-GA to NSGAIII and exact Pareto sets in terms of both quality and computational effort. MOO-GA generally outperforms NSGAIII for all metrics.

2. MOO-GA'S TERMINATION CRITERIA TAILORED FOR CLOUD SCHEDULING

GA termination represents a trade-off between *the quality of the Pareto set*(PS) - deteriorated by premature termination, and *the wasted computation* - caused by excessive iterations not improving the PS quality. As GAs are stochastic, we need robust metrics. Usually GAs have too little information to easily compute the utopic point, used by some quality metrics. Using BaTS' sampling phase results, we compute the *cheapest* (CP) and the *fastest* (FP) points (schedules) [2], valid for both the estimated and the real PSs. The utopic point (UP) has the *fastest* makespan and the *cheapest* price.

Hyperarea Difference (HD) measures the distance between a PS and the (unknown) real one [11]. A lower HD is better. In Eq. 1a the Hypervolume (HV) metric [12] is strictly monotonic [9]. To compute the HV, we first scale the objective space using the CP and FP points and then we use Eq. 1b. As MOO-GA computes the HD at each iteration, we derived a computationally fast HV expression(Eq. 1b).

$$\mathrm{HD} = 1 - \mathrm{HV}. \tag{1a}$$

$$\mathrm{HV} = \sum_{r=2}^{n} \sum_{i=1}^{n-r+1} \sum_{j=i+r-1}^{n} (-1)^{r+1} \binom{j-i-1}{r-2} (1-t_i)(1-c_j) + \sum_{i=1}^{n} (1-t_i)(1-c_i) \tag{1b}$$

Smallest Euclidean Distance to Utopic Point ($\mathrm{D_{UP}}$) is a companion metric to the HD, using problem space (cloud scheduling) knowledge to allow earlier termination without considerable loss of quality. We compute $\mathrm{D_{UP}}$ as the minimum Euclidean distance to UP from any PS solution. Intuitively, once MOO-GA finds two consecutive PSs with the same HD (*stagnation phase*), it may be close to a "good-enough" PS. Here, the $\mathrm{D_{UP}}$ enables fine-tuning the PS quality: MOO-GA terminates once two consecutive PSs have the same $\mathrm{D_{UP}}$. As HD decreases monotonically, newer PSs' quality cannot degrade compared to the *stagnation phase* PS.

$$\mathrm{D_{UP}} = \min_{i=1}^{n} \sqrt{(c_i - c_{cheapest})^2 + (t_i - t_{fastest})^2} \tag{2}$$

3. EVALUATION AND DISCUSSION

We compare the MOO-GA *fast, well-approximated* Pareto sets (PS) of cloud schedules to the exact PS [3] and the NSGAIII [13] approximations. Chosen PS metrics show domain-specific desired qualities and computational metrics, timeliness. Test problems are domain-specific workloads [3].

Workloads: An on-demand instance type (OD) has an hourly price and execution speed; its related spot type (S)

Table 1: IGD, HD, DSC and DSF average and stdev values

Cfg	IGD				HD					DSC				DSF			
	MOO-GA		NSGAIII		MOO-GA		NSGAIII		Real	MOO-GA		NSGAIII		MOO-GA		NSGAIII	
	avg	stdev	avg	stdev	avg	stdev	avg	stdev		avg	stdev	avg	stdev	avg	stdev	avg	stdev
20	0.0235	0.0031	0.0755	0.0191	0.0829	0.0019	0.2591	0.0633	0.0622	0.0094	0.0202	0.6262	0.0446	0.1191	0.0677	0.1061	0.0411
100	0.0450	0.0151	0.1103	0.0262	0.1281	0.0341	0.2037	0.0958	0.0685	0.0702	0.0983	0.4731	0.0495	0.0342	0.0204	0.4711	0.0661
40	0.0183	0.0043	0.0674	0.0106	0.0231	0.0100	0.0605	0.0458	0.0157	0.0195	0.0512	0.2281	0.0235	0.1499	0.0510	0.5320	0.0364

Table 2: Spread, NDC and Cluster average and stdev values

Cfg	Spread					NDC					Cluster				
	MOO-GA		NSGAIII		Real	MOO-GA		NSGAIII		Real	MOO-GA		NSGAIII		Real
	avg	stdev	avg	stdev		avg	stdev	avg	stdev		avg	stdev	avg	stdev	
20	0.6801	0.1054	0.0637	0.0165	0.6742	29.1000	2.4432	17.7200	1.6167	35	1.6814	0.1524	4.1138	0.1626	3.5714
100	0.6456	0.1909	0.0452	0.0171	0.6228	21.5600	3.2461	12.5000	1.9614	29	1.5618	0.1829	2.6143	0.3697	4.2414
40	0.4148	0.1975	0.0128	0.0024	0.2619	21.4800	2.6821	11.5000	0.9742	26	2.0510	0.2785	5.3712	0.7074	5.3462

Table 3: Runtime and NFE average and stdev values

Cfg	Runtime (milisec)					NFE			
	MOO-GA		NSGAIII		Real	MOO-GA		NSGAIII	
	avg	stdev	avg	stdev		avg	stdev	avg	stdev
20	387	190	2765	859	55882	44317	13096	44372	13097
100	222	131	2616	1314	1532563	44387	22032	44440	22037
40	352	156	2476	698	970438	45009	11835	45054	11830

Table 4: T-score values

Cfg	Cluster	NDC	Spread	IGD	HD	DSC	DSF
20	-76.40	-27.19	-40.43	-18.79	-19.46	-88.24	1.14
100	-17.86	-16.72	-21.93	-15.10	-5.20	-25.61	-44.22
40	-30.57	-24.48	-14.24	-30.14	-5.57	-25.94	-42.66

has similar speed, but different price. The ODs run tasks according to 1) sampled execution speed [2] at \$0.020; 2) 2x faster at \$0.065; 3) 3x faster at \$0.130; the related S types cost \$0.003, \$0.007 and \$0.013. A **OD-S** setup has at most 100 instances with at most OD={20,40,100}, S={20,60,100}.
Parameters: NSGAIII's *pop_size*=100 and p=4 [6]. MOO-GA's *pop_size*=2000, elitism percentage p_e=30%, number of pairs extracted for crossover=30% of the *pop_size* and mutation probability of a gene=1/15000. BLX-a crossover uses a =0.3, NDC and Cluster use μ =0.05.
Metrics: a)**convergence**: *Inverse generational distance*(IGD) [6], b)**diversity and significance** trade-offs: *Number of Distinct Choices*(NDC), *Cluster*(CL) and *Pareto Spread* - higher NDC [11], lower CL and wider Spread [11] are preferred and c)**computational effort**: *Number of function evaluations* (NFE) and *Runtime* - MOO-GA's adaptive termination criteria varies the NFE across different runs and for fairness we run NSGAIII with `max`NFE=$_{\text{MOO-GA}}$NFE. However, NSGAIII will actually run [13] until $_{\text{NSGAIII}}$NFE>`max`NFE.
Evaluation results: We run each setup (**Cfg**) fifty times on DAS4 [14] standard compute nodes and report the average and standard deviation. Tables 1, 2 and 3 show all results. Table 4 shows the statistical significance scores. MOO-GA *generally outperforms* NSGAIII *for all metrics.* Low MOO-GA HD variability mean the metric is problem size independent and robust to stochasticity, confirming its use in MOO-GA's termination criteria. MOO-GA Spread larger than the real PS Spread means MOO-GA PS contains solutions beyond the real PS and we study the distance between the second most extreme real solution and its closest solution from the approximated PS. MOO-GA counterparts are closer to the real schedules than the NSGAIII ones, except the second fastest solution in the **20** setup. The MOO-GA NDC is lower than real PS NDC, but the MOO-GA CL outperforms the real PS CL, thus providing a *representative approximated PS*.

Notably, MOO-GA reduced the execution time for any considered setup to less than *1 second* on average. The NSGAIII is at least *5 times* slower than MOO-GA, while the exhaustive search takes as long as 25 minutes.

4. CONCLUSIONS

Cloud infrastructure volatility (e.g., fluctuating spot market prices) requires quick recomputation of Pareto sets of makespan-cost pairs corresponding to various cloud instance pools. In this work, we introduced heuristics derived from cloud scheduling to help MOO-GA, our extended genetic algorithm [3] used by BaTS [2], deliver fast, well-approximated Pareto sets. Results show that exact Pareto set computation is unfeasible for online instance pool re-configuration, while our heuristics-enhanced MOO-GA computes controllable-quality approximations in less than *one second time* and of better quality than state-of-the-art NSGAIII. We also show how domain knowledge greatly improves the quality and performance of a genetic algorithm. We plan to focus on application-specific objectives and domain-specific metrics, further increasing the diversity of Pareto set approximations.

Acknowledgments This research has been partially funded by the Dutch public-private partnership COMMIT/.

5. REFERENCES

[1] Scientific Computing Using Spot Instances. aws.amazon.com/ec2/spot-and-science

[2] Oprescu A, Kielmann T, Leahu H, Budget Estimation and Control for Bag-of-Tasks Scheduling in Clouds. *PPL*, 2011, 21(2):219–243

[3] Vintila A, Oprescu AM, Kielmann T, Fast (Re-)Configuration of Mixed On-demand and Spot Instance Pools for High-throughput Computing. In *ACM ORMaCloud*, 2013

[4] Sato H, Inverted PBI in MOEA/D and Its Impact on the Search Performance on Multi and Many-objective Optimization. In *GECCO*, 2014

[5] Bleuler S, et al., Multiobjective genetic programming: reducing bloat using SPEA2. In *IEEE CEC*, 2001

[6] Deb K, Jain H, An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I. *IEEE TEC*, 2014, 18(4):577–601

[7] Trautmann H, et al., A Convergence Criterion for Multiobjective Evolutionary Algorithms Based on Systematic Statistical Testing. In *PPSN X*, 2008

[8] Guerrero JL, et al., A Stopping Criterion Based on Kalman Estimation Techniques with Several Progress Indicators. In *GECCO*, 2009

[9] Zitzler E, Knowles J, Thiele L, Quality Assessment of Pareto Set Approximations. Multiobjective Optimization, LNCS, 2008

[10] Safe M, et al., On Stopping Criteria for Genetic Algorithms. In *Advances in Artificial Intelligence*, 2004

[11] Azarm S, Wu J, Metrics for Quality Assessment of a Multiobjective Design Optimization Solution Set. *Journal of Mechanical Design*, 2001, 123(1):18–25

[12] Veldhuizen DAV, *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations.* Ph.D. thesis, Air Force Inst. of Tech. Wright Patterson AFB, OH, USA, 1999

[13] A Java Framework for Multiobjective Optimization. http://www.moeaframework.org

[14] The Distributed ASCI Supercomputer. www.cs.vu.nl/das4