Fast Evolutionary Multi-objective Optimization Algorithm (FastEMO)

ABSTRACT

This paper proposes a new fast and qualitative algorithm for computationally intensive Multi-Objective Optimization Problems (MOP), called FastEMO (Fast Evolutionary Multi-Objective Optimization algorithm). The aim of this method is to efficiently approximate the set of Pareto optimal solutions with a very low complexity algorithm, making it possible to use very large population sizes without sacrificing quality. FastEMO is based on the structure of ASREA (Archived-Based Stochastic Ranking Evolutionary Algorithm) and possesses four novel modifications: a replacement strategy with a decreasing parent population size down to the archive size, a specific parents selection strategy, a good combination of genetic operators and an increasing archive size on the last generation. FastEMO outperforms five other mainstream multi-objective algorithms (NSGA2, NSGA3, ASREA, PAES and ESPEA) both in quality (using different metrics on 2 and 3 objective ZDT and DTLZ functions) and in runtime, due to its low $O(man)$ complexity.

CCS CONCEPTS

• **Theory of computation → Automated reasoning**;

KEYWORDS

Evolutionary Algorithms, Multi-Objective Optimization Problems, Runtime

ACM Reference Format:

. 2019. Fast Evolutionary Multi-objective Optimization Algorithm (FastEMO) . In *Proceedings of the Genetic and Evolutionary Computation Conference 2019 (GECCO '19).* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Multi-objective optimization is very important in engineering, because most real-world problems involve multiple conflicting objectives, where the improvement in one objective function results in the degradation of another. Many very efficient multi-objective optimization algorithms have been proposed since David Goldberg suggested to use the distance to the Pareto front (PF) as a fitness value for an individual in 1989 [9].

However, computers have changed since this time : they now all use multi-core CPUs or even many-core GPU cards. The latest RTX 2081 ti NVIDIA card has 4352 cores, each running several threads in order to be used efficiently. This means that for embarrassingly parallel algorithm, such as Multi-Objectives Evolutionary Algorithms (MOEAs), one can use population sizes of the magnitude of 10000 individuals, in order to make the best use of the card. Unfortunately, even recent MOEAs, such as NSGA3 [5], are not designed for this kind of population size: as our tests show, NSGA3 requires in average 27s/generation for 10000 individuals and to 270s/generation for 100000 individuals, which is the order of magnitude that future cards will enable, when Moore's law is expected to stop about 10 years from now.

We propose a very fast evolutionary algorithm, named FastEMO, to solve the aforementioned issue. In order to make it easy to use and cross-platform, we integrated FastEMO in the EASEA (EAsy Specification of Evolutionary Algorithms) platform[1]. EASEA is a publicly available platform, designed to simplify the development of evolutionary algorithms. The main concept of EASEA is the use of a unified approach for describing all the evolutionary algorithms by a single general architecture with a flexible parameter configuration [2], [3]. FastEMO is now available in the EASEA platform.

The paper is organised as follows: Section 2 defines the proposed algorithm. Section 3 specifies the experiments, performed to evaluate this proposal. Section 4 reports the results of these experiments. Section 5 concludes this work.

2 PROPOSED ALGORITHM

FastEMO (Fast Evolutionary Multi-Objective Algorithm) has been designed to support very large population sizes while still offering the best quality Pareto front (PF) in 2 or 3 objective problems.

FastEMO relies on the original structure of the ASREA [16] algorithm. ASREA ensures good convergence by creating a stochastic PF thanks to a small limited archive set of non-dominated solutions.

Like ASREA, this approach uses the following features:

- three populations sets: the parent population set, the archive population set and the offspring population set,
- a small sized limited archive of non-dominated solutions,
- a crowding distance mechanism for updating the archive,
- genetic variation operators (selection, crossover, mutation) for creating new solutions.

Aiming to improve on diversity preservation and convergence rate towards the PF, we have made the following modifications compared with ASREA:

- parent selection strategy for mating pool (for better convergence),
- parent population replacement strategy (for faster convergence rate towards the PF),
- specific genetic operations strategy (for better convergence and diversity),
- parent population size equal to archive size (for better convergence and speedup),

[1] http://easea.unistra.fr/

GECCO '19, July 13–17, 2019, Prague, Czech Republic

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

- variable sized archive which increases on the last generation (to obtain a larger number of solutions),
- removal of a separate ranking assignment procedure by including domination checking into the archive update procedure.

We use the following terminology:

- N: initial population size,
- A: archive population size for all generations, except for the last one,
- O_{min}: minimal current offspring population size for taking part in the parent selection procedure (proposed min value is 4), to improve diversity.
- A_{max}: archive population size for the last generation (proposed max value is 10000),
- *i-pop*: initial population set with size = N,
- *p-pop*: parent population set with size = N for first generation. Then, parent population size is adjusted to archive size = A,
- *o-pop*: offspring population set with size = N,
- *a-pop*: archived population set with size = A,
- *r-pop*: result population set with size = A_{max},
- o_i: iterator for individuals in *o-pop*,
- a_i: iterator for individuals in *a-pop*.

The algorithm is described below :

The algorithm begins with a random initialization of the initial population (*i-pop*) in the required search space. Then this population is copied into the first parent population set (*p-pop*) with size N and every individual is evaluated. The archive that saves non-dominated solutions (*a-pop*) is empty. In the next step, the main loop of the algorithm starts, by checking for a stopping criteria (a number of generations). If met, the algorithm stops. Otherwise, a new offspring population (*o-pop*) is created as described in the next subsection. After the first generation, *p-pop* size is reduced to archive size. When the number of generations has reached its maximum value, the archive size is set to A_{max}.

Result: r-pop
Define N, A, A_{max}
Generate *i-pop*, empty *a-pop*, *r-pop* and *p-pop*
Calculate fitness values for every individual in *i-pop*
Copy *i-pop* to *p-pop* (size N)
while *stopping criterion is not met* **do**
  while *o-pop is not full* **do**
    Select parent 1 from *p-pop* by binary tournament selection
    if *if o-pop size* $< O_{min}$ **then**
      Select parent 2 from *p-pop* by binary tournament selection
    else
      Select parent 2 as best individual from current *o-pop*
    end
    Apply BLX-α Crossover on selected parents
    Apply Gaussian Mutation on offspring
    Evaluate offspring
    Add offspring to *o-pop*
  end
  if *last generation* **then**
    Resize *a-pop* from A to A_{max}
  end
  Update *a-pop* by checking dominance and using crowding distance diversity mechanism (see Algorithm 2)
  if *first generation* **then**
    Resize *p-pop* from N to A
  end
  Best Random Replacement *p-pop* (size A) by individuals from *a-pop* (see Algorithm 3)
end
Copy solutions from *a-pop* to *r-pop*

Algorithm 1: FastEMO pseudo code

2.1 Creation of the offspring population

2.1.1 Parent selection strategy. In ASREA [16] parents are randomly selected from *p-pop*, which consists of *a-pop* and the previous *o-pop*. Instead, for efficiently guiding the search process towards the PF, FastEMO uses a higher selection pressure than ASREA : only non-dominated solutions from *p-pop* and the best solutions from current *o-pop* (which is going to be created) are introduced in the offspring producing procedure. In this approach, the mating pool is generated by two different selection operators for parent 1 and parent 2. Parent 1 is selected by a binary tournament from p-pop. For parent 2, if the current *o-pop* size is less than O_{min} (proposed value = 4), parent 2 is also selected from *p-pop* by a binary tournament selection. Otherwise, it is selected as the best individual from the current *o-pop*. This strategy helps overcoming two important challenges:

- Avoiding premature convergence, by selecting parents from different solutions,
- Achieving a selection bias towards the most promising solutions; it is specially useful in some of the MOPs, which have a small number of non-dominated solutions in earlier generations (like ZDT2[11]).

This way, the performance is maintained during all generations.

2.1.2 Genetic operator strategy. Once the mating parents are selected, genetic operators are applied to create a new offspring for *o-pop*.

Crossover. In this approach, for the crossover operation, we use the Blend Crossover Operator (BLX-α) [10]. Blend-α crossover creates a new offspring by selecting a random value from the interval between vector's elements of two parents.

Formally, this crossover operator generates an offspring as a random linear recombination of two parents s^1 and s^2.

If $s_i^1 < s_i^2$, a uniformly distributed random value is generated from the interval:

$$[s_i^1 - (s_i^2 - s_i^1) \cdot \alpha,\ \ s_i^2 + (s_i^2 - s_i^1) \cdot \alpha] \quad (2)$$

Otherwise,

$$[s_i^2 - (s_i^1 - s_i^2) \cdot \alpha,\ \ s_i^1 + (s_i^1 - s_i^2) \cdot \alpha] \quad (3)$$

As shown in equations (2) and (3) above, the BLX-α operator has two important features : the location of the offspring depends on the difference in parent solutions and the interval of possible locations is manageable (increased in direction of the parent with better fitness

by α value). We use these features of the operator for organizing an adaptive search, suitable for many MOPs. The α factor of the crossover was chosen by searching the optimum results obtained with different α values on multiple test problems. The best results for all test problems were obtained with value of $\alpha = .75$, which was fixed for all experiments.

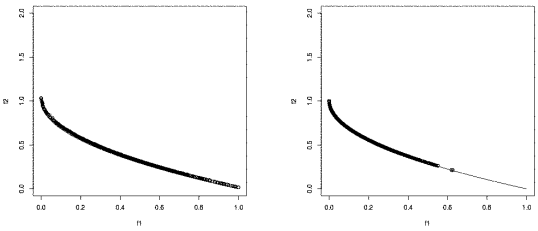
Mutation. To improve the diversity preservation technique, we propose to apply a Gaussian mutation mechanism [4]. The Gaussian distribution is the basis of the Gaussian mutation operator. It adds noise to each element of a solution's vector to create a new offspring:

$$x' = x + \sigma \cdot N(0,1), \quad (4)$$

where σ is a mutation rate.

Parametric tests revealed that the Gaussian mutation ensures the best results with a value of σ equals to 0.5. The Gaussian mutation operator is used because it satisfies the three main requirements for mutation operators:

- it is covering the whole solution space (ergodicity),
- it does not show drift and does not favour a specific direction because of the symmetry of the Gaussian distribution,
- it can scale the randomly drawn samples in the whole solution space with σ.



a) Gaussian mutation b) Polynomial mutation

Figure 1: FastEMO with Gaussian and Polynomial mutation operator on ZDT4 function (population size = 10000, number of generations = 11)

Compared to Polynomial mutation, the Gaussian mutation provides better diversity by better covering the solution space, as shown in Figure 1.

2.2 Archive update strategy

As in ASREA, FastEMO uses a limited archive [14] to retain only non-dominated solutions of a current generation in the next generation. But for FastEMO, the strategy is different, as illustrated in the pseudo-code of Algorithm 2.

In order to avoid unnecessary ranking assignment storage, two operations are included in the archive update procedure: dominance check and copy of the new best solutions (genome and objective function values) into *a-pop*.

If the archive has reached its maximum size, the crowding distance control is applied to handle the overflow of *a-pop*, aiming to maintain the diversity of solutions. Actually, a crowding distance density estimator is used to provide well diversified solutions across the PF in NSGA2 [6] and ASREA. But it has been shown in [13], that

Result: insert/reject new individual in/from *a-pop*
o_i = index of current offspring individual
$a_i = 0$
while $a_i < A$ **do**
  if *a new o-pop[o_i] is dominated by a new a-pop[a_i]* **then**
    reject new individual and go to next one from *o-pop*
  else
    if *a-pop[a_i] is dominated by o-pop[o_i]* **then**
      remove dominated individual from *a-pop*
    else
      increment a_i : go to next individual in *a-pop*
    end
  end
  Add the new individual into archive;
  if *a-pop size = A+1* **then**
    if *last generation = true* **then**
      return
    end
    Find the worst individual in *a-pop* by crowding distance
    Delete the worst individual from *a-pop*
  end
end

Algorithm 2: Update *a-pop*

the crowding distance operator used in NSGA2, does not work well with more than two objectives. As well, during the 3-objectives experiments, ASREA shows a global decrease in the size of the objective space covered by an approximation set (see HV value for DTLZ1 and DTLZ3 benchmarks in Table 4 and Figure 3 column c)).

In FastEMO we successfully promote solutions diversity by jointly using a Gaussian distribution for the mutation operator and crowding distance control for *a-pop*. As was mentioned above, to obtain a larger amount of solutions, we change the size of the archive to A_{max} on the last generation in Algorithm 1. When this size is reached, Algorithm 2 stops.

The optimal *a-pop* size is fixed as $15m$ (where m is the number of objectives), which is a little bit larger than in ASREA. This archive size does not increase the computation complexity and it is enough to maintain a good population diversity (see results of experiments in Table 4).

2.3 Best Random Replacement strategy

As explained in subsection 2.1.1, a strong selection pressure is used in FastEMO to push the next population towards the PF. For the same reason, the replacement of the parent population is based on a random selection from the best solutions of the archive set. The pseudo-code of the replacement strategy is given in Algorithm 3.

Result: new *p-pop*
Clear *p-pop*
while *While p-pop is not full* **do**
  Randomly select one individual from *a-pop*
  And copy one it to *p-pop*
end

Algorithm 3: Best Random Replacement *p-pop*

3 EXPERIMENTS

In order to evaluate the potential of the proposed algorithm for solving different multi-objective problems and to tune its parameters. It was first tested on several generic MOPs benchmark. The setting parameters of FastEMO for all test functions are provided in Table 1 below.

Table 1: Parameters of FastEMO

Parameter	Value
Size of archive	15·m, m is the num. of objectives
Crossover probability	.9
Crossover alpha factor	.75
Mutation probability	1/n, n is the num. of variables
Mutation rate	.5

To assess the relative performance of FastEMO, compared to other algorithms, it is pitted against five representative MOEAs : NSGA2 [6], NSGA3 [5], ASREA [16], PAES [12] and ESPEA [1]. The comparative details of these algorithms are presented in Table 3. We use the Java jMetal [8] framework for running NSGA2, NSGA3, PAES and ESPEA. Algorithms ASREA and FastEMO are executed in the EASEA platform.

In order to compare the quality of the MOEAs, 20 runs of each algorithms with different initial populations were performed for each problem to accurately assess the mean of the performance. The number of generations is defined as the termination criterion. All the experiments have been conducted on the same Intel(R) Pentium(R) CPU 4405U @ 2.10GHz 4 processors laptop.

3.1 Benchmark Suite

All chosen algorithms were tested and compared on five 2-objective ZDT [11] and on five 3-objective DTLZ [7] benchmarks. The characteristics of the tests are represented on Table 2.

Table 2: Characteristics of the benchmark problems

Problem	Pop. size	Nb. of dim.	Nb. of obj.	Nb of gen.
ZDT1	10000	30	2	10
ZDT2	10000	30	2	10
ZDT3	10000	30	2	10
ZDT4	10000	10	2	13
ZDT6	10000	30	2	10
DTLZ1	10000	20	3	50
DTLZ2	10000	30	3	50
DTLZ3	10000	30	3	50
DTLZ4	10000	30	3	50
DTLZ7	10000	30	3	50

3.2 Performance Metrics

In order to assess the performance for each algorithm the runtime (RT) is measured as well as two of the most used quality indicators from[15]:

(1) Hypervolume (HV) maximisation[19]: it provides the volume of the objective space that is dominated by a PF, therefore, it shows the convergence quality towards the PF and the diversity in the obtained solutions set,

(2) Inverted Generational Distance (IGD) minimization [17]: it is an inverted variation of Generational Distance that: i) calculates the minimum Euclidean distance between an obtained solution and the real PF and ii) measures both the diversity and the convergence towards the PF of the obtained set (if enough members of PF are known [18]).

4 RESULTS AND DISCUSSION

Table 4, 5 and Figure 3, 2 show the results:

On 2-objectives functions: FastEMO and ESPEA provide the best convergence towards the PF and PF width, with similar values of HV and IGD compared to NSGA2, NSGA3, ASREA and PAES. However, FastEMO is 14 to 50 times faster than ESPEA.

On 3-objectives functions: FastEMO is the best algorithm, except for DTLZ3, where NSGA3 is the winner. However, it must be considered that NSGA3 is more than 100 times slower than FastEMO, as it uses 1305s compared to 11s for FastEMO. If we increase the number of generation to 150, FastEMO converges towards the PF with the best value of HV = .453 in 25s, which is still 52 times faster than NSGA3 on 50 generations and 162 times faster than NSGA3 in 150 generations. In the other 3-objectives test functions, the values of HV and IGD for NSGA3 and FastEMO are similar, but FastEMO is in average 120 times faster than NSGA3. The obtained fronts for DTLZ1, DTLZ3, DTLZ6 functions are shown in Figure 3. FastEMO provides more solutions and ensures better diversity than the other algorithms. The results of ESPEA on 3-objectives tests are not provided in the Table 4, because no result was obtained in less than 1h on the jMetal framework.

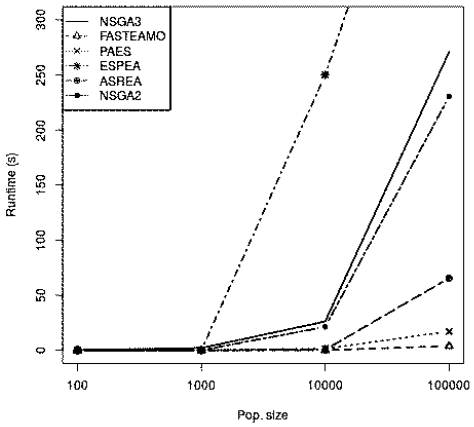


Figure 2: One generation runtime comparison for 6 algorithms on DTLZ2 test function

Table 5 and Figure 2 shows the runtime value for one generation *versus* population size for each algorithm on the very fast DTLZ2

Table 3: Comparison of MOEAs

MOEA	Fitness assignment	Elitism	Strategy of diversity	Archive	Strengths	Weakness
NSGA2 [6]	Deterministic ranking procedure	Yes	Crowding distance	No	Good convergence towards PF	High computation complexity
NSGA3 [5]	Deterministic ranking procedure	Yes	Vector based niching	No	Good convergence towards PF for > 2 obj.	High computation complexity,slow by niching selection
ASREA [16]	Stochastic ranking procedure	Yes	Crowding distance	Yes	Cut down computation complexity	Bad convergence for > 2 obj.
PAES [12]	Pareto dominance ranking procedure	Yes	Cell-based density	Yes	Cut down computation complexity	Depends on cell sizes and number of obj.
ESPEA [2]	Preference based procedure	Yes	Crowding distance	Yes	Good quality	High computation complexity for > 3 obj.

Table 4: Results comparison on the performance metrics

Problem	Metrics	FastEMO	ASREA	NSGA2	NSGA3	PAES	ESPEA
ZDT1	HV	**0.667**	0.375	0.007	0.623	0.586	**0.665**
	IGD	**2.2e-05**	0.006	0.021	0.001	0.005	**1.0e-05**
	Runtime (s)	**1.898**	5.434	8.552	12.015	19.250	26.528
ZDT2	HV	**0.332**	0.0	0.0	0.295	0.199	**0.332**
	IGD	**1.5e-05**	0.021	0.049	0.001	0.008	**1.3e-05**
	Runtime (s)	**1.837**	5.200	8.615	11.892	19.210	28.581
ZDT3	HV	**0.517**	0.364	0.053	0.469	0.486	**0.516**
	IGD	3.0e-05	0.003	0.014	0.001	0.002	**8.7e-06**
	Runtime (s)	**0.980**	5.139	8.917	15.895	19.009	25.601
ZDT4	HV	**0.633**	0.090	0.0	**0.623**	0.364	**0.656**
	IGD	**6.3e-04**	0.021	0.112	0.001	0.011	**2.2e-04**
	Runtime (s)	**0.507**	6.512	8.811	16.035	1.542	26.209
ZDT6	HV	**0.405**	0.0	0.0	0.455	0.309	**0.400**
	IGD	**3.1e-05**	0.012	0.141	0.001	0.009	**1.1e-05**
	Runtime (s)	**1.598**	5.029	8.804	12.260	18.607	25.758
DTLZ1	HV	**0.772**	0.0	0.0	**0.786**	0.454	
	IGD	**4.7e-04**	0.053	0.615	**4.4e-04**	0.003	
	Runtime (s)	**5.638**	28.900	890.851	1138.203	46.522	>3600
DTLZ2	HV	**0.450**	0.425	**0.447**	0.413	0.242	
	IGD	**2.2e-04**	5.8e-04	**1.8e-04**	5.9e-04	0.002	
	Runtime (s)	**13.171**	29.290	928.466	1369.675	95.214	>3600
DTLZ3	HV	0.0	0.0	0.0	**0.413**	0.121	
50 gen.	IGD	0.286	0.7661	4.881	**9.6e-04**	0.010	
	Runtime (s)	**11.064**	30.402	916.747	1305.272	27.244	>3600
DTLZ3	HV	**0.453**	0.297	0.0	0.413	0.227	
150 gen.	IGD	**4.2e-04**	0.002	0.886	9.6e-04	0.006	
	Runtime (s)	**25.951**	91.670	2927.695	4050.123	216.507	>3600
DTLZ4	HV	**0.434**	0.415	0.0	0.406	0.0	
	IGD	**5.4e-04**	5.5e-04	0.017	6.8e-04	0.009	
	Runtime (s)	**11.286**	30.975	546.619	1412.159	28.241	>3600
DTLZ7	HV	**0.324**	0.294	0.262	0.291	0.083	
	IGD	**7.8e-04**	0.001	0.002	0.002	0.020	
	Runtime (s)	**12.957**	31.720	942.500	1032.572	176.639	>3600

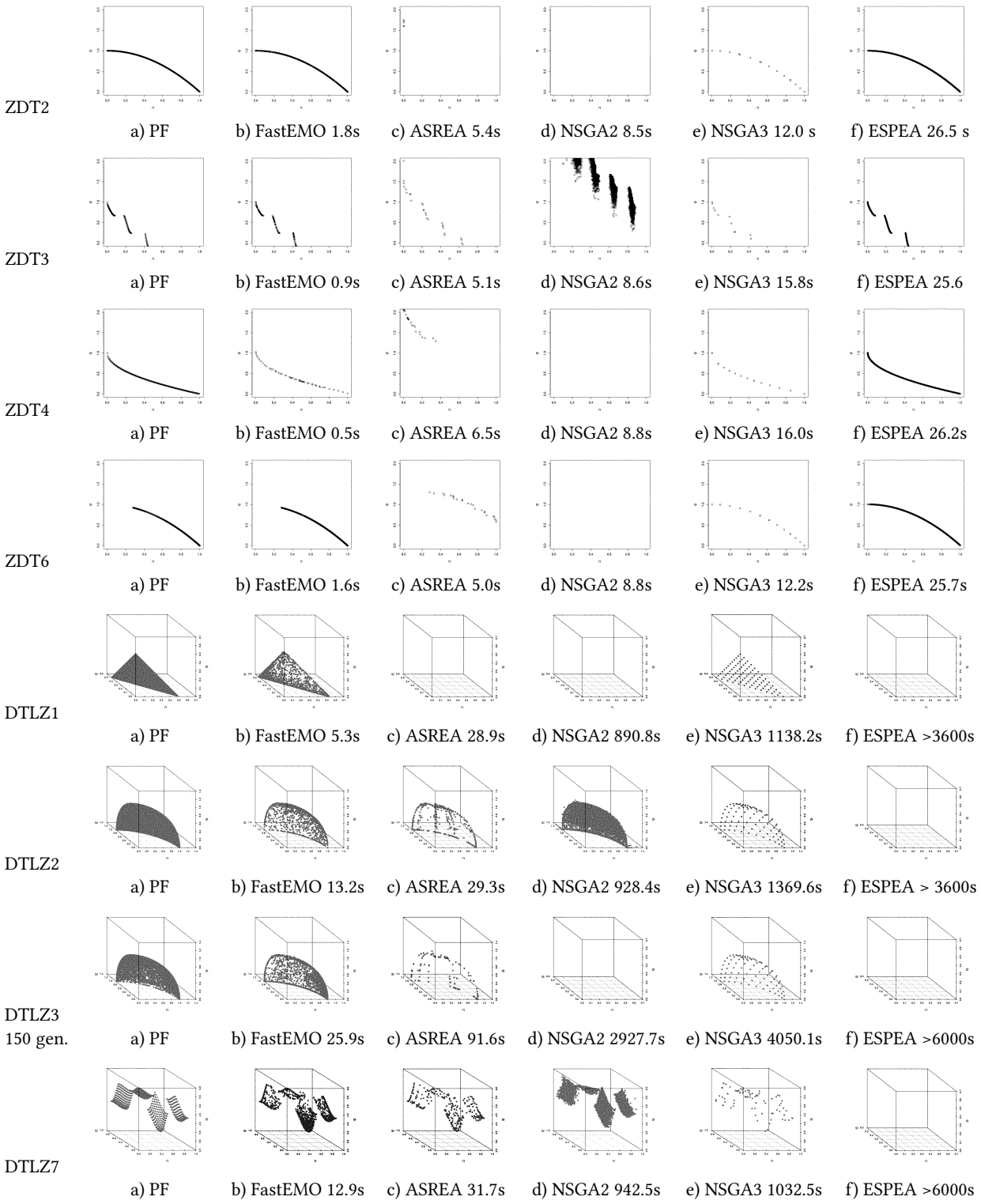


Figure 3: Approximation sets of the five algorithms on different benchmarks

test function (only 4.10^{-3}s evaluation time). FastEMO is significantly faster than all other algorithms.

Table 5: One generation runtime on DTLZ2

Algorithm	100	1000	10000	100000
FastEMO	**0.0010 s**	**0.013 s**	**0.16 s**	**3.9 s**
ASREA	0.0012 s	0.024 s	0.62 s	65.52
NSGA2	0.023 s	0.200 s	21.25 s	230.42
NSGA3	0.233 s	2.100 s	26.28 s	270.76
PAES	0.009 s	0.051 s	1.50 s	17.10 s
ESPEA	0.012 s	0.055 s	250.10 s	580.41 s

5 CONCLUSION

This paper describes a new fast multi-objective optimization algorithm called FastEMO. It is suitable for the large population sizes that can be evaluated on modern massively parallel machines as well as for smaller populations. Based on the structure of ASREA, this approach proposes a much simpler architecture, which only computes a small Pareto Front limited to the size of the archive. During the computation:

(1) the size of the parent population is reduced to the size of archive,
(2) parents are selected from both the archive and from the growing offspring population,
(3) on the last generation, the archive is increased to obtain more non-dominated solutions.

This algorithm outperforms well-known MOEAs, such as NSGA2, NSGA3, PAES, ESPEA and ASREA not only in runtime (with an $O(man)$ complexity) but also in quality, for both 2 and 3 objective problems.

Future work will focus on parallelizing FastEMO on GPGPU cards as well as exploring its efficiency on more than 3 objectives problems.

Due to the limited number of pages of this paper, more benchmark tests are available on:
http://easea.unistra.fr/index.php/EASEA_papers

REFERENCES

[1] Marlon Alexander Braun, Pradyumn Kumar Shukla, and Hartmut Schmeck. 2015. Obtaining optimal pareto front approximations using scalarized preference information. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. ACM, 631–638.

[2] Pierre Collet, Evelyne Lutton, Marc Schoenauer, and Jean Louchet. 2000. Take it EASEA. In *International Conference on Parallel Problem Solving from Nature*. Springer, 891–901.

[3] Pierre Collet and Marc Schoenauer. 2003. GUIDE: Unifying evolutionary engines through a graphical user interface. In *International Conference on Artificial Evolution (Evolution Artificielle)*. Springer, 203–215.

[4] Kalyanmoy Deb and Debayan Deb. 2014. Analysing mutation schemes for real-parameter genetic algorithms. *International Journal of Artificial Intelligence and Soft Computing* 4, 1 (2014), 1–28.

[5] Kalyanmoy Deb and Himanshu Jain. 2014. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints. *IEEE Trans. Evolutionary Computation* 18, 4 (2014), 577–601.

[6] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation* 6, 2 (2002), 182–197.

[7] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. 2002. Scalable multi-objective optimization test problems. In *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, Vol. 1. IEEE, 825–830.

[8] Juan J Durillo and Antonio J Nebro. 2011. jMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software* 42, 10 (2011), 760–771.

[9] Goldberg D. E. 1989. *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley.

[10] Larry J Eshelman and J David Schaffer. 1993. Real-coded genetic algorithms and interval-schemata. In *Foundations of genetic algorithms*. Vol. 2. Elsevier, 187–202.

[11] Deb Kalyanmoy et al. 2001. *Multi objective optimization using evolutionary algorithms*. John Wiley and Sons.

[12] Joshua D Knowles and David W Corne. 2000. Approximating the nondominated front using the Pareto archived evolution strategy. *Evolutionary computation* 8, 2 (2000), 149–172.

[13] Saku Kukkonen and Kalyanmoy Deb. 2006. Improved pruning of non-dominated solutions based on crowding distance for bi-objective optimization problems. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*. IEEE, 1179–1186.

[14] Mahesh B Patil. 2018. Using External Archive for Improved Performance in Multi-Objective Optimization. *arXiv preprint arXiv:1811.09196* (2018).

[15] Nery Riquelme, Christian Von Lücken, and Benjamin Baran. 2015. Performance metrics in multi-objective optimization. In *Computing Conference (CLEI), 2015 Latin American*. IEEE, 1–11.

[16] Deepak Sharma and Pierre Collet. 2010. An archived-based stochastic ranking evolutionary algorithm (ASREA) for multi-objective optimization. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*. ACM, 479–486.

[17] David A Van Veldhuizen and Gary B Lamont. 2000. Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. *Evolutionary computation* 8, 2 (2000), 125–147.

[18] Qingfu Zhang, Aimin Zhou, and Yaochu Jin. 2008. RM-MEDA: A regularity model-based multiobjective estimation of distribution algorithm. *IEEE Transactions on Evolutionary Computation* 12, 1 (2008), 41–63.

[19] Eckart Zitzler and Lothar Thiele. 1999. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE transactions on Evolutionary Computation* 3, 4 (1999), 257–271.