Resource Allocation in MOEA/D: What is important?

Yuri Lavinas[†], Claus Aranha[††], Marcelo Ladeira[††], Tetsuya Sakurai [‡]

筑波大学 [†], 筑波大学 [††], University of Brasilia[‡], 筑波大学 [‡‡]

1 Introduction

Multi-objective Optimization Problems (MOP) are minimization[†] problems characterized by multiple, conflicting objective functions. It arises in real world applications that require a compromise among multiple objectives. The set of optimal trade off solutions in the decision space is the *Pareto Set (PS)*, and the image of this set in the objective space is the *Pareto Front (PF)*. Finding a good approximation of the Pareto Front is a hard problem for which multiple Evolutionary Algorithms have been proposed, [1)].

The Multi-Objective Evolutionary Algorithm Based on Decomposition, (MOEA/D) [2)] is an effective algorithm for solving MOPs. The main characteristic of MOEA/D is to decompose the multi-objective optimization problem into a set of single objective subproblems. It has been observed that some subproblems require more attention than others, and take more effort to converge to an optimal solution [3)] wasting computational effort by trying to improve solutions that are not very promising [4)] . This can be a critical issue in large-scale MOP problems that require costly simulations [5)] .

To address this issue, *Resource Allocation* (RA) techniques have been proposed. They distribute the computational effort to each subproblems differently, based on an estimation of the relative importance of each subproblem [3, 6, 7)] . The estimation of subproblem importance is done by *Priority Functions.*

In this work, we study Priority Functions to improve the quality of Resource Allocation. Here, we first report results of Priority Function based on critical issues: Decision-Space Diversity (DS) [8, 9)] and Inverted Decision-Space Diversity (iDS) [9)] that are based on the diversity of subproblems and their solutions on decision space, and the Relative Improvement (R.I.) [6, 3)] . that is based on how much a solution objectives values improved over iterations.

The results of our first experiment show that DS largely improved the performance of MOEA/D, leading to better Inverted Generational Distance (IGD) and higher percentage of non-dominated solutions on benchmark functions. To our surprise, result is that assigning random priority also improves MOEA/D performance, although not to the degree of DS.

The Random Priority Function changes MOEA/D, by making it possible to hold around 50% of the population, changing only the other half of the population. Based on the surprising results of this Priority Function we extend the studies by making it possible to (indirectly) control the percentage of the population that will be hold We conduct a small experiment on this more controlable Random Priority Function. The results show that limiting the number of solutions has a great positive impact on the performance of MOEA/D in terms of IGD values.

2 Background

2.1 Priority functions

Priority functions are used in Resource Allocation (RA) to determine the preferences between subproblems [10)] . These functions take information about the progress of the algorithm's search, and

Resource Allocation in MOEA/D: What is important?

[†] Yuri Lavinas(lavinas.yuri.xp@alumni.tsukuba.ac.jp)
[††] Claus Aranha(caranha@cs.tsukuba.ac.jp)
[‡] Marcelo Ladeira(mladeira@unb.br)
[‡‡] Tetsuya Sakurai(sakurai@cs.tsukuba.ac.jp)
Department of Computer Science, University of Tsukuba (†)
Department of Computer Science, University of Brasilia (††)
Department of Computer Science, University of Tsukuba (‡)
Department of Computer Science, University of Tsukuba (‡‡)

[†]or maximization

Algorithm 1 MOEA/D-SRA (Simple Resource Allocation)

```
1: Initialize the weight vectors $\lambda_i$, the neighborhood $B_i$, the priority value $u_i$ every subproblem $i = 1, ..., N$.
2: while Termination criteria do
3:     for i = 1 to N do                                   ▷ Number of subproblems
4:         if rand() $< u_i$ then
5:             Generate an offspring $y$ for subproblem $i$.
6:             Update the population by $y$.
7:     if Number of subproblems updated $\leq 3$ then
8:         (Update all subproblems, reset all $u_i$ to 1)
9:     Evaluate the population, and update all $u_i$ using a Priority Function.
```

Algorithm 2 Priority Function: Decision Space Diversity (DS)

```
1: Input: $X^t$ decision vectors of solutions; $X^{t-1}$, decision vectors from the previous solutions; N, the population size.
2: for i=1 to N do
3:     u[i] = $||X_i^t$ - $X_i^{t-1}||$
4: $u$ = scale ($u$)                                    ▷ between 0 and 1
5: Return $u$
```

guide the distribution of computational resources among subproblems over iterations [11] .

We highlight the MOEA/D-GRA [3] , MOEA/D-DRA [6] and the Two-Level Stable Matching-Based Selection in MOEA/D [12] that uses Relative Improvement (R.I.) as priority function. These studies show that using Priority Functions for RA can help MOEA/D performance.

3 MOEA/D-SRA

Algorithm 1 describes the MOEA/D with Simple Resource Allocation (MOEA/D-SRA) [8] . For more information, see the work of Lavinas et. al [8] .

Algorithm 3 Priority Function: Inverted Decision Space Diversity (iDS)

```
1: Input: $X^t$ decision vectors of solutions; $X^{t-1}$, decision vectors from the previous solutions; N, the population size.
2: for i=1 to N do
3:     $u[i]$ = $||X_i^t$ - $X_i^{t-1}||$
4: $u$ = 1$-$scale ($u$)                                ▷ between 0 and 1
5: Return $u$
```

Algorithm 4 Priority Function: Relative Improvement

```
1: Input: $Y^t$, objective function values from the incumbent solutions; $Y^{t-\Delta T}$, objective function values from incumbent solution of iteration $t - \Delta T$, $u$ from the previous $\Delta T$ iteration;
2: for i=1 to N do
3:     $u[i] = \frac{Y^t[i] - Y^{t-1}[i]}{Y^t[i]}$
4: if $max(u) = 0$ then
5:     $\forall u[i]; u[i] = 1$
6: else
7:     $u = u$ / (max($u$) + $1.0 \times 10^{-50}$)
8: Return $u$
```

Algorithm 5 Priority Function: Random

```
1: Input: N, the population size.
2: for i=1 to N do
3:     u[i] = value sampled uniformly from the [0,1) interval
4: Return $u$
```

3.1 Decision Space Distance: DS and iDS Priority Functions

The idea behind *Decision Space Distance* (DS) is that by considering diversity as the Priority Function more resources are given to solutions that are different to their parents, forcing MOEA/D to focus on less explored areas, and leading to higher exploration of the decision space. Algorithm 2 details the calculation.

In the opposite fashion, the *inverted Decision Space Distance* (iDS) gives more resources to solutions that are similar to their parents, forcing them to improve more. iDS is the inverse of DS, prioritizing solutions that DS does not prioritize.

Algorithm 3 details the implementation.

3.2 Relative Improvement Priority Function

The Relative Improvement (R.I.) Priority Function allocates resources to subproblem based on an estimation of problem difficulty. Subproblems where the fitness of the incumbent solution has improved further over the last ΔT iterations receive higher priority under this function. Algorithm 4 details the implementation. For more information about this priority function, see the works of Zhou, Zhang and Nasir [3, 6, 12] .

3.3 Random Priority Functions

As a baseline for comparison we also define a Random Priority Function. This function samples the priority value u_i from an uniform distribution. This should, on average, allocate the same amount of resources for all subproblems over the optimization process. Algorithm 5 gives the details on its implementation.

4 Experimental Results and Discussion

We compare MOEA/D-SRA, Algorithm 1, with five different RA strategies: four RA using DS, i-DS, Random and Relative Improvement Priority Functions (R.I.); and MOEA/D-DE with no RA.

For comparison we use two function sets: the DTLZ function set, with 100 dimensions and k = dimensions - number of objectives +1, where the number of objectives is 2; and the UF function set, with 100 dimensions.

4.1 Experimental Parameters and Evaluation

We use MOEA/D-DE standard parameters [13] for each strategy: update size $nr = 2$, neighborhood size $T = 20$, and the neighborhood search probability $\delta_p = 0.9$. The DE mutation operator value is $F = 0.5$. The Polynomial mutation operator values are $\eta_m = 20$, $p_m = 0.03333333$ and the lower and upper bounds are respectively $(-2, 2)$. The decomposition function is Simple-Lattice Design (SLD), the scalar aggregation function is Weighted Sum (WS), the update strategy is the Restricted Update Strategy and we performed a simple linear scaling of the objectives to [0, 1]. For every strategy/function pair we perform 21 repetitions with 30000 function evaluations and population size $N = 350$.

We compare the results of the different strategies based on their Inverted Generational Distance (IGD) metrics (Lower values of IGD are better). The Pairwise Wilcoxon Rank Sum Tests is used to analyze differences in IGD values with confidence $\alpha = 0.05$ and with the Hommel adjustment method for multiple comparisons.

4.2 Analysis of IGD Results

Table 1 shows that the methods with RA always show better IGD values. The Pairwise Wilcoxon test results in Table 2 indicate significant differences in IGD favoring the use of Resource Allocation over MOEA/D-DE without RA, no matter the Priority Function.

Among the Priority functions, the DS achieves best IGD values in about 75% of all the functions (see Table 1). R.I. achieved better results on some functions, however our statistical analysis indicated that the DS was superior to each of the other methods (See Table 2). Our baseline, Random Priority Function, had very good results, in some cases even better than DS or R.I.

5 A Deeper Analysis of Random Priority Function

In Section 4.2, we showed that there is no statistically difference between the R.I. priority function and the Random priority function (Table 2). This suggests that updating only part of the population every iteration under MOEA/D might have a positive effect.

Right now, the Random Priority Function, subsection 3.3, selects around 50% of the population at each iteration. The same overall behavior can be achieve by setting a fix value of 0.5 as the Priority Function value. To study how much of the population should be updated, we propose a new experiment where we fix the priority values, u_i, to: 0.10, 0.20, 0.40, 0.60, 0.80 and 1.00. We highlight

Table 1: IGD medians in parenthesis for MOEA/D-DE without Priority Functions, DS, iDS, Random and R.I. The number in parenthesis is the standard deviation. The best value for each function is indicated in Bold.

IGD	MOEA/D-DE	DS	i-DS	Random	R.I.
UF1	0.425 (0.042)	**0.146 (0.013)**	0.326 (0.022)	0.250 (0.032)	0.223 (0.030)
UF2	0.130 (0.009)	0.104 (0.012)	0.104 (0.009)	0.103 (0.010)	**0.093 (0.008)**
UF3	0.301 (0.006)	**0.274 (0.010)**	0.283 (0.006)	0.278 (0.009)	0.281 (0.008)
UF4	0.112 (0.003)	0.108 (0.002)	0.111 (0.003)	0.109 (0.003)	**0.107 (0.003)**
UF5	2.163 (0.060)	**1.333 (0.106)**	1.923 (0.093)	1.591 (0.086)	1.551 (0.103)
UF6	0.408(0.054)	**0.177 (0.042)**	0.332 (0.049)	0.250 (0.039)	0.236 (0.033)
UF7	0.392 (0.067)	**0.141 (0.015)**	0.332 (0.034)	0.231 (0.039)	0.208 (0.034)
UF8	0.380 (0.028)	**0.261 (0.010)**	0.303 (0.014)	0.278 (0.013)	0.282 (0.013)
UF9	0.521 (0.014)	**0.440 (0.0154)**	0.476 (0.015)	0.475 (0.013)	0.474 (0.015)
UF10	4.105 (0.153)	**2.97 (0.216)**	3.788 (0.179)	3.24 (0.189)	3.094 (0.273)
DTLZ1	411.5 (157.1)	387.8 (95.08)	564.4 (112.1)	301.7 (123.0)	**257.8 (99.73)**
DTLZ2	0.309 (0.027)	**0.140 (0.019)**	0.218 (0.035)	0.187 (0.022)	0.179 (0.017)
DTLZ3	1409 (300.2)	1006 (318.0)	1440 (346.1)	801.6 (225.3)	**245.4 (380.5)**
DTLZ4	0.404 (0.071)	**0.131 (0.034)**	0.263 (0.050)	0.206 (0.029)	0.230 (0.102)
DTLZ5	0.326 (0.027)	**0.148 (0.020)**	0.224 (0.034)	0.186 (0.020)	0.179 (0.019)
DTLZ6	29.23 (2.814)	**0.096 (0.777)**	23.29 (1.267)	16.68 (2.663)	16.32 (2.586)
DTLZ7	3.213 (0.353)	**0.287 (0.222)**	2.229 (0.244)	1.719 (0.213)	0.783 (0.208)

Table 2: Statistical analysis of the IGD difference between the methods: MOEA/D-DE without Priority Functions, DS, iDS, Random and R.I; paired on the benchmark functions, using the Pairwise Wilcoxon Rank Sum test. "$\leftarrow$" indicates superiority of the row method, while "$\uparrow$" indicates superiority of the column method. "$\simeq$" indicates no statistical difference.

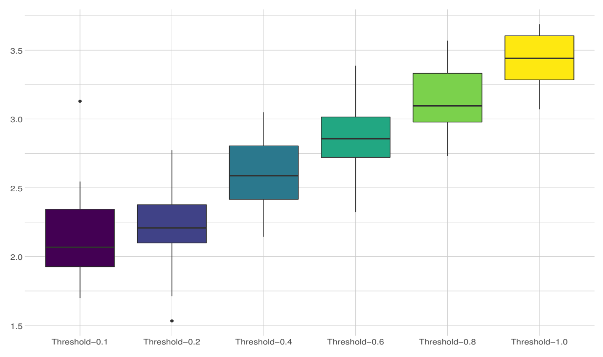
	DS	iDS	R.I.	Random	MOEA/D-DE
DS	-	1.3e-12 $\leftarrow$	1.1e-06 $\leftarrow$	5.9e-08 $\leftarrow$	$<$ 2e-16 $\leftarrow$
iDS	1.3e-12 $\uparrow$	-	0.00078 $\uparrow$	0.00172 $\uparrow$	9.0e-05 $\leftarrow$
R.I.	1.1e-06 $\uparrow$	0.00078 $\leftarrow$	-	0.41140 $\simeq$	1.0e-08 $\leftarrow$
Random	5.9e-08 $\uparrow$	0.00172 $\leftarrow$	0.41140 $\simeq$	-	1.0e-08 $\leftarrow$
MOEA/D-DE	$<$ 2e-16 $\uparrow$	9.0e-05 $\uparrow$	1.0e-08 $\uparrow$	1.0e-08 $\uparrow$	-

that with lower values, such as 0.10 and 0.20, the algorithm displays a more conservative behavior, updating around 10% and 20% of the population, respectively. On the other hand, the algorithm displays a more aggressive behavior with higher values, such as 0.60 and 0.80 (updating 60% and 80% of the population, respectively) Note that when the fix value is equal to 1.00, all subproblems are selected to be updated, therefore, this case simply reproduces the standard MOEA/D-DE.

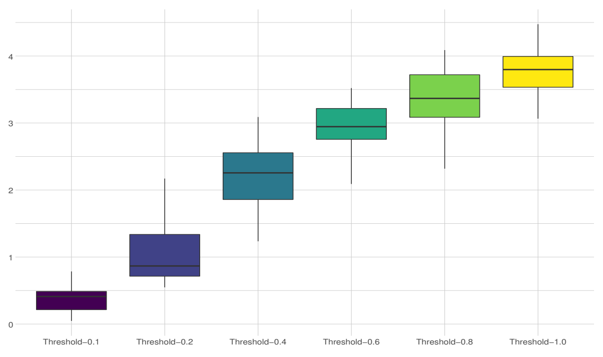
We use the same experiments parameters and evaluation methods as in Section 4.1. We consider only UF10 and DTLZ7 functions.

5.1 IGD Results

Table 3 and Figure 1 show that the by updating only a small fraction of the solutions improves the performance of MOEA/D in terms of IGD values. That is, if we only update around 10% of the population leaving around 90% of the population



(a) UF10.



(b) DTLZ7.

Fig. 1: Box-plot of IGD values of all threshold methods on two functions.

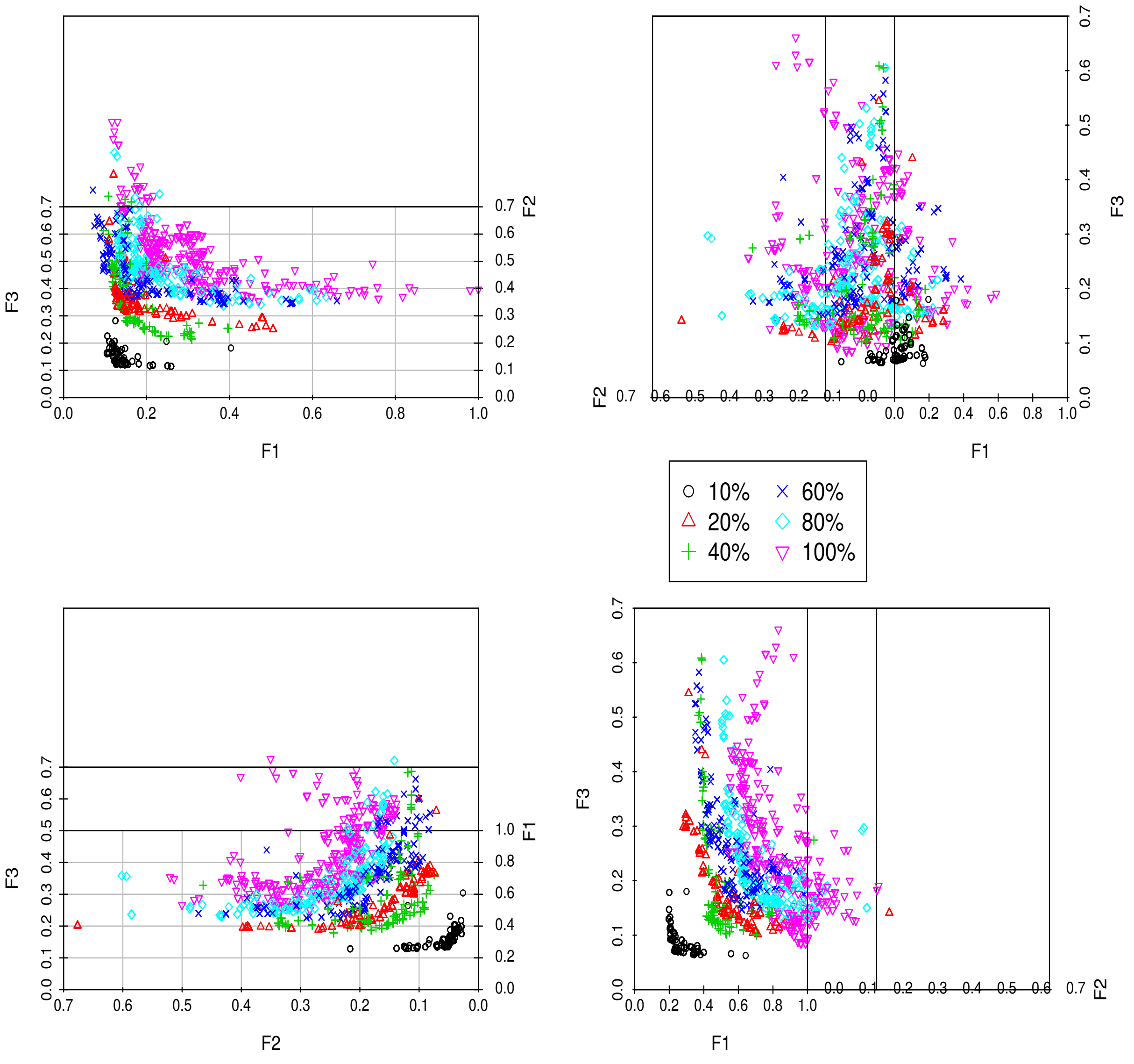


Fig. 2: Pareto Front of all threshold Priority Functions on UF10.

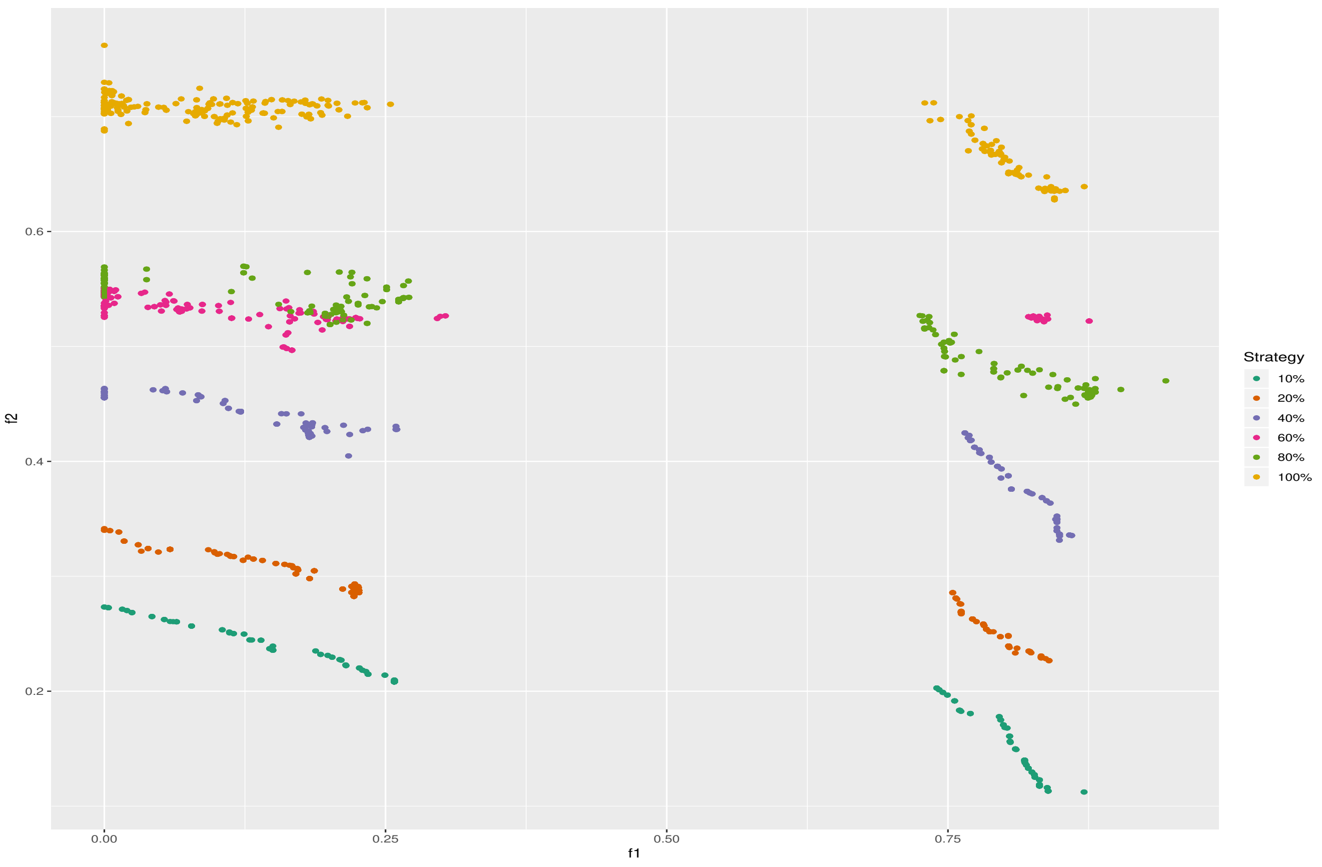


Fig. 3: Pareto Front of all threshold Priority Functions on DTLZ7.

Table 3: IGD medians in parenthesis for every Threshold Priority Function. The number in parenthesis is the standard deviation. The best value for each function is indicated in Bold. Then, it shows the proportion of Non-dominated: Median values and standard deviation (in parenthesis) of non-dominated solutions on UF10 and DTLZ7 benchmarks.

IGD	0.10	0.20	0.40	0.60	0.80	1.00
UF10	**2.068 (0.338)**	2.208 (0.330)	2.588 (0.234)	2.856 (0.236)	3.096 (0.228)	3.441 (0.181)
DTLZ7	**0.411 (0.209)**	0.867 (0.472)	2.256 (0.501)	2.946 (0.387)	3.367 (0.491)	3.798 (0.347)

Proportion	0.10	0.20	0.40	0.60	0.80	1.00
UF10	**96% (3%)**	93% (9%)	88% (6%)	80% (6%)	71% (7%)	44% (7%)
DTLZ7	**90% (10%)**	64% (8%)	38% (10%)	28% (7%)	23% (10%)	19% (6%)

unchanged, the algorithm performs the best. The same trend happens with the proportion of non-dominated solutions.

The Pairwise Wilcoxon test results indicate significant differences in IGD favoring the idea of updating only small subgroups of the whole population. In a more concrete way, the results indicate that updating 10% or 20% of the population is better than updating 90% or 100% of the population. Figures 2 and 3 show example of the achieved Pareto Fronts in UF10 and DTLZ7.

6 Conclusion

In this work, we studied different ideas of Priority Functions, those based on metrics (DS, R.I.) and those that have give the same priority to all subproblems. Both ideas have their merits, since for some problems, using metrics had better IGD results while for other problems using metrics had worse IGD results.Although both groups are based on divergent strategies in terms of inspiration, all of them limit the number of solutions of the population that are updated from a iteration to the next. We recall the title of this work "Resource Allocation in MOEA/D: What is important?" and we understand that holding part of the population unchanged and only updating a small subpopulation seems to be of great importance. This goes some way towards enhancing our understanding of Resource Allocation in MOEA/D.

We understand that the behavior demonstrated by in this study MOEA/D with Priority Functions is similar to the behavior of MOEA/D with the external archive population. That is because by using the external archive, MOEA/D is able to maintain the non-dominated solutions found during the search while by using Priority Functions MOEA/D is also able to maintain non-dominated solutions, , since there is a limit of solutions that can be replaced from a solution to the next. In contrast, MOEA/D benefits better from Resource Allocation techniques since the non-dominated solutions are still able to influence the search while having no extra computational cost (there is no need to store two populations - the main one and the one from the external archive).

Also, MOEA/D with Priority Functions might be interpreted as a steady-state MOEA/D. There is because MOEA/D with Priority Function limits the number of solutions that are changed between iteration while in any steady state evolutionary algorithms, only a few (mostly one or two) solutions of the a population are replaced [14)]. This relationship becomes even clearer when we consider the analysis of Section 5, where we can see that there is a well-defined trend between holding smaller fractions of the population and the improvement of MOEA/D.

The small number of functions investigated does not allow to a more general and definitive conclusion, however, an extensive analyses would be a fruitful area for further work.

参考文献

1) Anupam Trivedi, Dipti Srinivasan, Krishnendu Sanyal, and Abhiroop Ghosh. A survey of multiobjective evolutionary algorithms based on decomposition. *IEEE Transactions on Evolutionary Computation*, Vol. 21, No. 3, pp. 440–462, 2017.

2) Qingfu Zhang and Hui Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on evolutionary computation*, Vol. 11, No. 6, pp. 712–731, 2007.

3) Aimin Zhou and Qingfu Zhang. Are all the subproblems equally important? resource allocation in decomposition-based multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, Vol. 20, No. 1, pp. 52–64, 2016.

4) Leonardo CT Bezerra, Manuel López-Ibáñez, and Thomas Stützle. Comparing decomposition-based and automatically component-wise designed multi-objective evolutionary algorithms. In *International Conference on Evolutionary Multi-Criterion Optimization*, pp. 396–410. Springer, 2015.

5) Takehisa Kohira, Hiromasa Kemmotsu, Oyama Akira, and Tomoaki Tatsukawa. Proposal of benchmark problem based on real-world car structure design optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 183–184. ACM, 2018.

6) Qingfu Zhang, Wudong Liu, and Hui Li. The performance of a new version of moea/d on cec09 unconstrained mop test instances. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pp. 203–208. IEEE, 2009.

7) Qi Kang, Xinyao Song, MengChu Zhou, and Li Li. A collaborative resource allocation strategy for decomposition-based multiobjective evolutionary algorithms. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2018.

8) Yuri Lavinas, Claus Aranha, and Testuya Sakurai. Using diversity as a priority function

for resource allocation on moea/d. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 215–216. ACM, 2019.

9) Yuri Lavinas, Claus Aranha, and Marcelo Ladeira. Improving resource allocation in moea/d with decision-space diversity metrics. p. to appear, 2019.

10) V Chankong and YY Haimes. Multiobjective decision making: Theory and methodology north holland. *New York*, 1983.

11) Xinye Cai, Yexing Li, Zhun Fan, and Qingfu Zhang. An external archive guided multiobjective evolutionary algorithm based on decomposition for combinatorial optimization. *IEEE Transactions on Evolutionary Computation*, Vol. 19, No. 4, pp. 508–523, 2015.

12) MD Nasir, Arnab Kumar Mondal, Soumyadip Sengupta, Swagatam Das, and Ajith Abraham. An improved multiobjective evolutionary algorithm based on decomposition with fuzzy dominance. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pp. 765–772. IEEE, 2011.

13) Hui Li and Qingfu Zhang. Multiobjective optimization problems with complicated pareto sets, moea/d and nsga-ii. *IEEE Transactions on evolutionary computation*, Vol. 13, No. 2, pp. 284–302, 2009.

14) Kaustuv Nag, Tandra Pal, and Nikhil R Pal. Asmiga: An archive-based steady-state micro genetic algorithm. *IEEE transactions on cybernetics*, Vol. 45, No. 1, pp. 40–52, 2014.