Improving Resource Allocation in MOEA/D with Decision-Space Diversity Metrics

Yuri Lavinas[1(✉)], Claus Aranha[1], and Marcelo Ladeira[2]

[1] University of Tsukuba, Tsukuba, Japan
lavinas.yuri.xp@alumni.tsukuba.ac.jp, caranha@cs.tsukuba.ac.jp
[2] University of Brasilia, Brasília, Brazil
mladeira@unb.br

Abstract. One of the main algorithms for solving Multi-Objective Optimization Problems is the Multi-Objective Evolutionary Algorithm Based on Decomposition (MOEA/D). It is characterized by decomposing the multiple objectives into a large number of single-objective subproblems, and then solving these subproblems in parallel. Usually, these subproblems are considered equivalent, but there are works that indicate that some subproblems can be more difficult than others, and that spending more computational resources in these subproblems can improve the performance of MOEA/D. One open question about this strategy of "Resource Allocation" is: what should be the criteria for allocating more computational effort on one problem or another? In this work we investigate this question. We study four different ways to prioritize subproblems: Randomly, Relative Improvement, Diversity in Decision Space (proposed in this work), and inverted Diversity in Decision Space (also proposed in this work). We compare the performance of MOEA/D using these four different "priority functions" on the DTLZ and UF benchmarks. We evaluate the resulting IGD, proportion of non-dominated solutions, and visually analyse the resulting resource allocation and Pareto Front. The result of our experiments is that the priority function using diversity in decision space improved the MOEA/D, achieving better IGD values and higher proportion of non-dominated solutions.

Keywords: Multi-Objective Optimization · Resource Allocation · Priority Functions

1 Introduction

Multi-objective Optimization Problems (MOP) are minimization[1] 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

[1] or maximization

C. Martín-Vide et al. (Eds.): TPNC 2019, LNCS 11934, pp. 134–146, 2019.
https://doi.org/10.1007/978-3-030-34500-6_9

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 [10,12,17].

The Multi-Objective Evolutionary Algorithm Based on Decomposition, (MOEA/D) [13] is an effective algorithm for solving MOPs. The key idea of the MOEA/D is to decompose the multi-objective optimization problem into a set of single objective subproblems. All subproblems are then solved in parallel.

Standard MOEA/D handles all subproblems uniformly. However, it has been observed that some subproblems are harder than others, and take more effort to converge to an optimal solution [16]. Therefore, MOEA/D may sometimes waste computational effort by trying to improve solutions that are not very promising [1]. This can be a critical issue in large-scale MOP problems that require costly simulations on supercomputers to evaluate each candidate solution [8].

To address this issue, *Resource Allocation* (RA) techniques have been proposed. They allocate different amounts of computational effort to different subproblems, based on an estimation of the relative importance of each subproblem [7,14,16]. The estimation of subproblem importance is done by *Priority Functions*, and one of the most popular approaches for this is the *Relative Improvement* (R.I.), which calculates a subproblem's priority by how much its incumbent solution has improved in recent iterations.

In this work, we study new Priority Functions to improve the quality of Resource Allocation. In particular, we propose a new Priority Function based on the diversity of subproblems and their solutions on decision space. In other words, how much the incumbent solution is different from its parent solution indicates whether the algorithm should focus on it or not. We propose two Priority Functions: Decision-Space Diversity (DS), which gives higher priority to solutions that are dissimilar from their parents, and Inverted Decision-Space Diversity (iDS), which gives higher priority to solutions that are similar to their parents. This work extends our previous short paper [9] by including the iDS and investigating the experimental results in much more depth.

In our experiments, we analyze how much each priority function managed to improve the MOEA/D. The results 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. On the other hand, the iDS did not perform so well. A surprising result is that assigning random priority also improves MOEA/D performance, although not to the degree of DS. Based on these results we suggest an improvement of the MOEA/D-DE base line, using DS for Resource Allocation (MOEA/D-SRA - Simple Resource Allocation).

2 Background

Priority functions are used in Resource Allocation (RA) to determine the preferences between subproblems [4]. These functions take information about the progress of the algorithm's search, and guide the distribution of computational resources among subproblems over iterations [2].

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

```
1: Initialize the weight vectors λ_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 ≤ 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.
```

We divide the works concerned with RA in two groups. The first group uses Relative Improvement (R.I.) as priority function and includes MOEA/D-GRA [16], MOEA/D-DRA [14] and the Two-Level Stable Matching-Based Selection in MOEA/D [11]. The second group is based on the use of an external archive and includes EAG-MOEA/D [2] and MOEA/D-CRA [7].

Zhou and Zhang claim that MOEA/D-GRA is an extension of MOEA/D-DRA and MOEA/D-AMS [5,16]. They reason that MOEA/D-GRA uses a similar Priority Function as MOEA/D-DRA and MOEA/D-AMS and that MOEA/D-GRA simulates MOEA/D-DRA or MOEA/D-AMS by changing a single parameter. This Priority Function is the R.I. Priority Function. This function defines the priority of each subproblem of the current iteration t based on how much the incumbent solution of that subproblem has improved over the last ΔT iterations. The idea behind R.I. is that a subproblem that has improved recently should have higher priority for RA.

These studies show that using Priority Functions for RA can help MOEA/D performance. However, each of these works focuses on improvements on the algorithm rather than on the Resource Allocation method itself. For example, Zhang et al.'s work [14] improves MOEA/D-DRA by using a 10-tournament selection. Therefore, the influence of the choice of Priority Functions on RA is still unclear.

In this context, our goal is to improve the performance of MOEA/D based on the choice of Priority Function. To achieve this, we isolate the Priority Function as our only change to the standard algorithm and we propose two new Priority Functions based on decision space diversity (DS and iDS). Our idea is that MOEA/D will benefit from a focus on diversity, since it is a critical issue to the search process.

3 MOEA/D-SRA - MOEA/D with Priority Functions

We use the basic framework described in Algorithm 1. This algorithm is based on MOEA/D-GRA [16], however, unlike that work, we only add Resource Allocation to MOEA/D-DE, with no other changes. Our proposed algorithm has a

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

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 ΔT 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) + 1.0×10^{-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

simple code structure and represents well the class of variants of MOEA/D with Resource Allocation without a population archive. In consequence, we can study any proposed Priority Function and integrate them to the MOEA/D framework.

Algorithm 1 differentiates from MOEA/D-DE [14] only on lines 4 and 7–9. Line 9 assigns a priority value u_i between 0 and 1 to each subproblem i, based on a priority function. Line 4 performs the Resource Allocation, by selecting whether each subproblem (and its incumbent solution) is to be updated on that iteraction. Each subproblem has probability equal to u_i of being selected for update on line 4, so subproblems with higher priority values are updated more often over the run of the algorithm.

At the start of the algorithm, all subproblems have the same priority $u_i = 1$. Also, if the number of subproblems selected for update at line 4 is less than 3 at any iteration, we reset all the priority values u_i to 1 and update all subproblems in that iteration on lines 7–8. Finally, note that even though some of the incumbent solutions are not updated under this algorithm, they may still be replaced by neighboring solutions that are updated, as in the standard MOEA/D-DE.

The choice of Priority Function allow us to easily design MOEA/D variants that allocate more resources on any desired characteristics of the population and the search, such as diversity or convergence. In this work, we consider three Priority Functions: R.I., DS and iDS. There are two main differences between R.I. and the proposed Priority Functions. First, R.I. uses information from the objective space of solutions while DS and iDS use information from the decision space of solutions. Second, R.I. has to maintain information about the objective values of all solutions in all past ΔT iterations, while DS and iDS only use information from the current iteration and the one before it. For R.I., we set the value of ΔT to 20, as in the work of Zhou and Zhang [16].

3.1 Decision Space Distance: DS and iDS Priority Functions

To consider diversity on the decision space, we measure the difference of a current solution and its parent using the (2-)Norm. Given this difference we define the metrics: *Decision Space Distance* (DS) and the *inverted Decision Space Distance* (iDS).

The DS Priority Function directly uses this distance. Algorithm 2 details the calculation. First we calculate the distance between an incumbent solution and its parent solution, and then we scale the distance to the (0,1] interval, based on the entire solution set.

The idea behind 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.

In the opposite fashion, the 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: like DS, the distance between incumbent solution and parent solution is calculated, and then scaled. However, the priority is based on the complement of this scaled value.

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 [11,14,16].

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

To examine the effects of Resource Allocation (RA) under different Priority Functions on MOEA/D, we perform a comparative experiment on benchmark functions. In this experiment, we use MOEA/D-DE implemented by the MOEADr package [3], modified to include RA as described in the previous section.

We compare MOEA/D-SRA, Algorithm 1, with five different RA strategies: four RA using DS, i-DS, Random and Relative Improvement Priority Functions; and MOEA/D-DE with no RA. In the following figures and tables, these strategies are referred, respectively, as: DS, i-DS, Random, R.I. and MOEA/D-DE.

For comparison we use two function sets: the DTLZ function set [6], with 100 dimensions and $k =$ dimensions $-$ number of objectives $+1$, where the number of objectives is 2; and the UF function set [15], with 100 dimensions.

4.1 Experimental Parameters and Evaluation

We use the conventional MOEA/D-DE parameters [10] for each RA 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 the IGD indicate better approximations. We also evaluate the proportion of non-dominated solutions.

The difference in IGD among the different techniques is analyzed using the Pairwise Wilcoxon Rank Sum Tests (paired on all functions examined) with confidence $\alpha = 0.05$ and with the Hommel adjustment method for multiple comparisons. For reproducibility purposes, all the code, data and experimental scripts are available online[1].

[1] https://github.com/yclavinas/MOEADr/tree/tpnc_2019.

Table 1. IGD medians ans standard deviation (in parenthesis) for every function and technique. 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)

4.2 Analysis of IGD Results

Tables 1 and 2 show that the methods using RA, specially the DS Priority Function, perform better than not using Resource Allocation. Methods with RA always show better IGD values, and higher proportion of non-dominated solutions. The Pairwise Wilcoxon test results in Table 3 indicate significant differences in IGD favoring the use of Resource Allocation over MOEA/D-DE without RA, no matter the Priority Function. We can see examples of the achieved Pareto Fronts on Fig. 1a and b.

Among the Priority functions, the DS achieves best IGD values in 13 out of 17 functions (see Table 1). While R.I. achieved better results on some functions, overall the DS showed itself more appropriate, as our statistical analysis (paired over all tested function), showed that the DS was superior to each of the other methods (See Table 3).

One surprising result is that the statistical test did not show a difference between the R.I. priority function and the Random priority function. While the Random priority function was initially designed as an "easy" baseline, this result indicates that holding part of the population every iteration under MOEA/D might have a positive effect, regardless of Priority Function. It also indicates that there is still space for finding better Priority Functions than those currently listed in the literature. Another interesting result is that iDS did not perform nearly

Table 2. Proportion of non-dominated: median values and standard deviation (in parenthesis) of non-dominated solutions on UF and DTLZ benchmarks.

Proportion	MOEA/D-DE	DS	i-DS	Random	R.I.
UF1	24% (03%)	**90% (08%)**	28% (04%)	52% (06%)	41% (13%)
UF2	28% 03%)	**92% (06%)**	44% (05%)	77% (06%)	63% (15%)
UF3	16% (03%)	**76% (09%)**	34% (05%)	52% (08%)	29% (09%)
UF4	59% (04%)	**97% (11%)**	63% (06%)	86% (03%)	76% (08%)
UF5	15% (02%)	**97% (04%)**	30% (04%)	64% (07%)	57% (13%)
UF6	22% (04%)	**90% (08%)**	28% (04%)	50% (05%)	39% (16%)
UF7	27% (04%)	**91% (07%)**	38% (05%)	58% (07%)	54% (10%)
UF8	48% (05%)	**97% (03%)**	69% (06%)	87% (05%)	95% (10%)
UF9	38% (04%)	**98% (04%)**	61% (05%)	89% (05%)	89% (06%)
UF10	38% (05%)	**98% (03%)**	61% (08%)	88% (06%)	90% (07%)
DTLZ1	09% (02%)	**98% (07%)**	13% (04%)	50% (16%)	41% (18%)
DTLZ2	15% (02%)	**92% (08%)**	31% (05%)	68% (08%)	70% (19%)
DTLZ3	02% (0.9%)	**95% (11%)**	05% (02%)	29% (13%)	15% (15%)
DTLZ4	06% (02%)	**94% (17%)**	18% (05%)	43% (10%)	58% (21%)
DTLZ5	15% (02%)	**92% (08%)**	31% (05%)	68% (08%)	70% (19%)
DTLZ6	05% (03%)	**99% (27%)**	07% (02%)	15% (11%)	28% (13%)
DTLZ7	16% (05%)	**85% (11%)**	21% (06%)	32% (10%)	45% (22%)

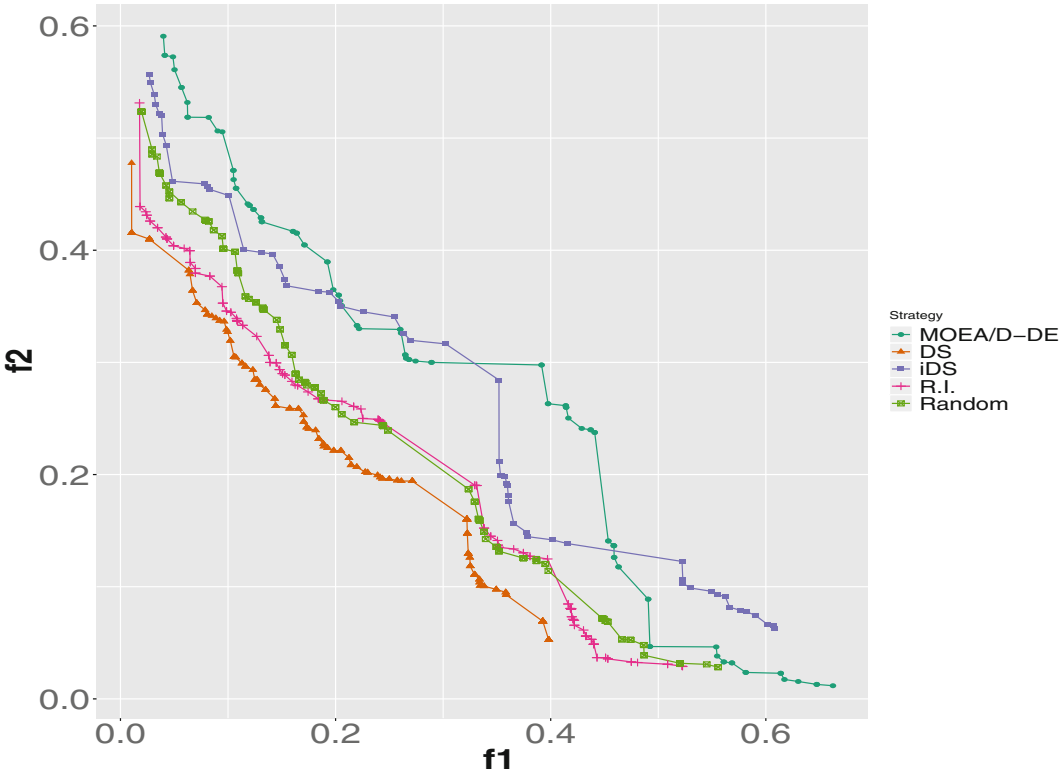
Table 3. Statistical analysis of the IGD difference between methods, paired on the benchmark functions, using the Pairwise Wilcoxon Rank Sum test. "←" indicates superiority of the row method, while "↑" indicates superiority of the column method. "≃" indicates no statistical difference.

	DS	iDS	R.I.	Random	MOEA/D-DE
DS	–	1.3e−12 ←	1.1e−06 ←	5.9e−08 ←	< 2e−16 ←
iDS	1.3e−12 ↑	–	0.00078 ↑	0.00172 ↑	9.0e−05 ←
R.I	1.1e−06 ↑	0.00078 ←	–	0.41140 ≃	1.0e−08 ←
Random	5.9e−08 ↑	0.00172 ←	0.41140 ≃	–	1.0e−08 ←
MOEA/D-DE	< 2e−16 ↑	9.0e−05 ↑	1.0e−08 ↑	1.0e−08 ↑	–

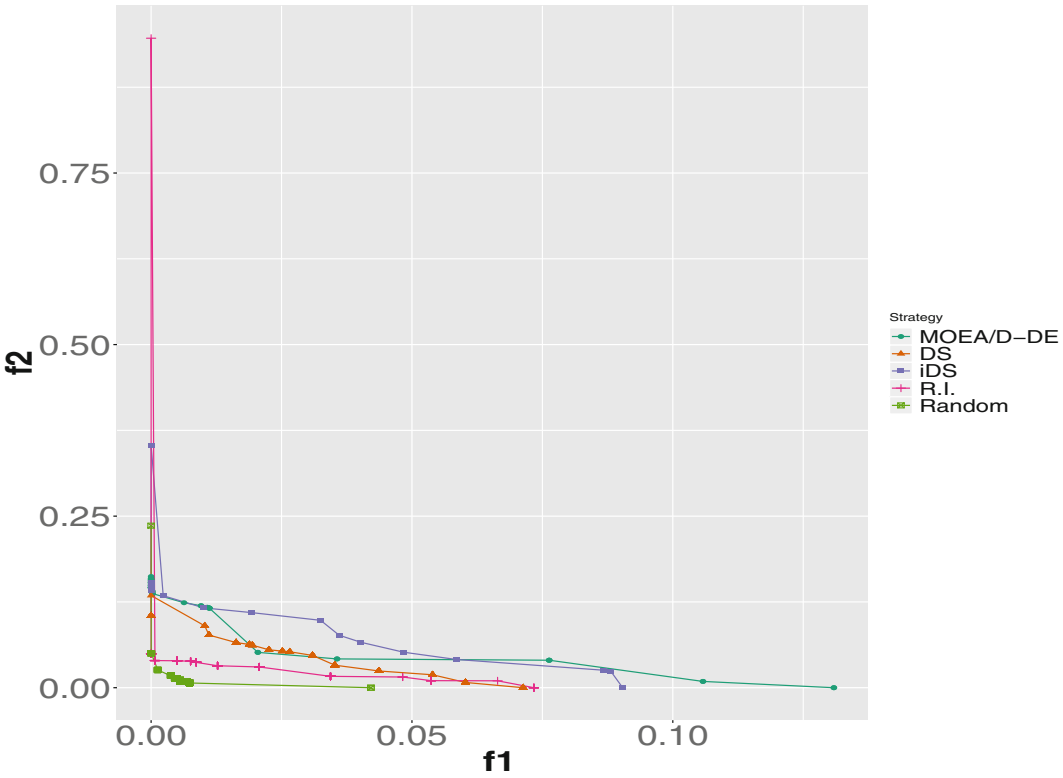
as well as DS, R.I. or Random, although it still performed better than not using Resource Allocation at all.

4.3 Analysis of the Proportion of Non-dominated Solutions

Looking at the proportion of non-dominated solutions (Table 2), we see that DS obtains the highest value on all functions for both benchmark sets, often with a very clear lead.



(a) UF7.



(b) DTLZ1.

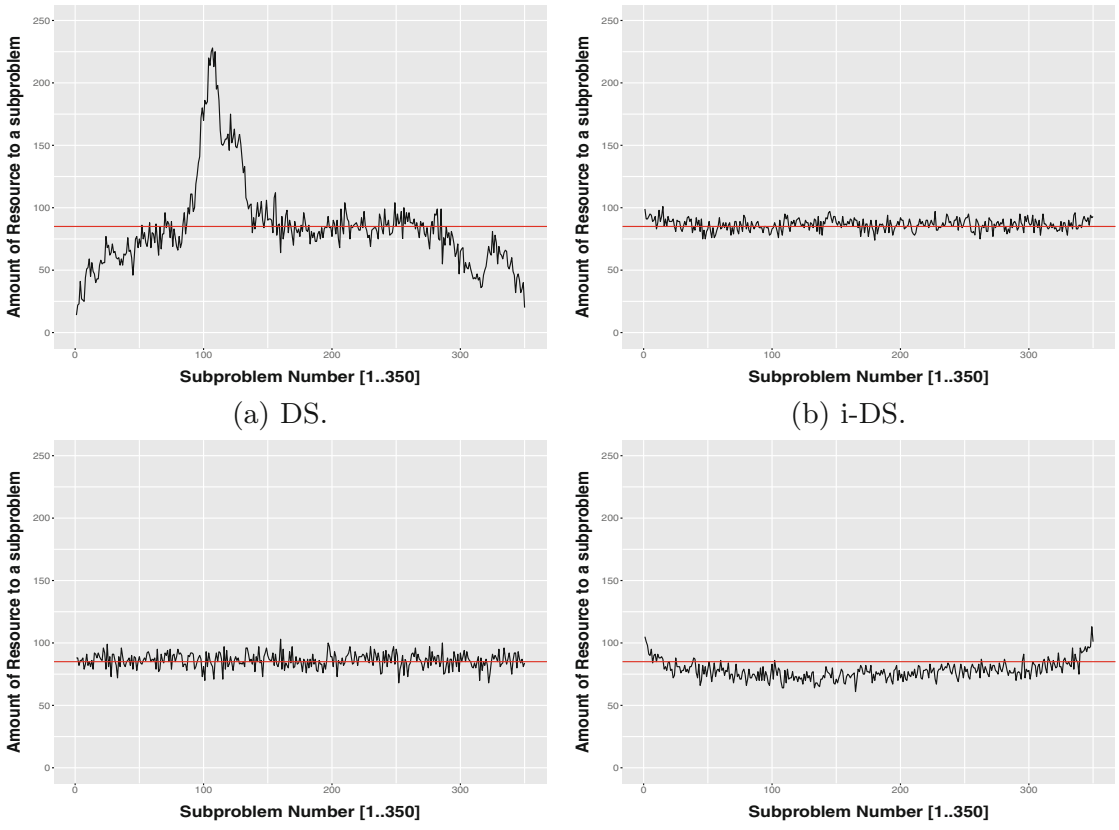
Fig. 1. Median Pareto Front of all methods on two functions.

This reflects the idea behind the creation of the DS Priority Function: by focusing more resources on solutions that are different from its parents, DS is able to achieve a more diverse set of solutions. On the other hand, R.I. focus on improving the fitness values of individual solutions means it is not surprising that it did not perform so well in terms of non-dominated solutions.

4.4 Discussion on Resource Allocation

Figures 2a, b, c, d illustrates the differences in Resource Allocation for the four Priority Functions. They show how many times each subproblem (horizontal axis) was updated (vertical axis) over the course of the optimization for each priority function on the UF7 function. We can see that DS allocated a higher amount of resources to subproblems 95 to 120, while iDS and R.I. slightly favored the first and last few subproblems. Random allocated the resources uniformly.

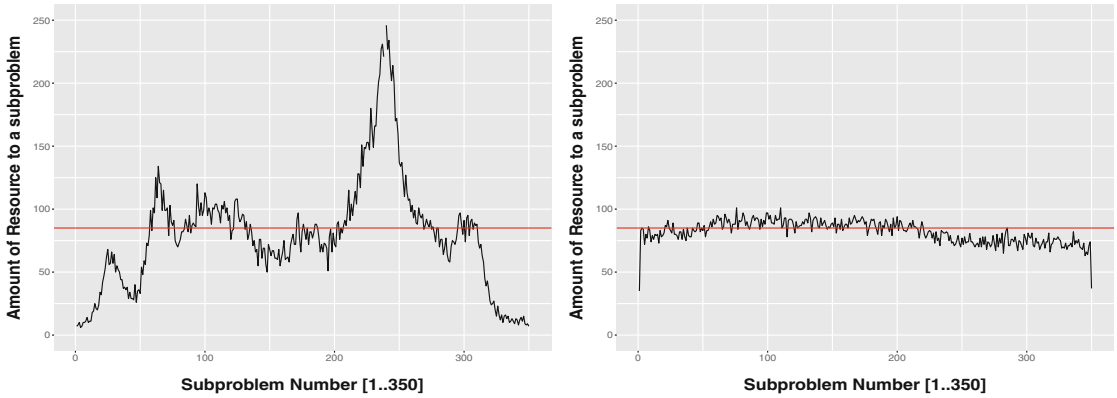
From these figures, we can clearly see that the DS is able to choose and focus on certain regions of the problem, while other priority functions are much more



(a) DS. (b) i-DS.

(c) Random. (d) R.I.

Fig. 2. Resource allocation on the UF7 function. The horizontal axis indicates the subproblems, while the vertical axis indicate the number of updates accumulated over the optimization. Higher values on the vertical axis indicate higher priority for that subproblem. The horizontal red line indicates the default number of updates (with no Resource Allocation).



(a) DS. (b) R.I.

Fig. 3. Resource Allocation on the DTLZ1 function. The horizontal axis indicates the subproblems, while the vertical axis indicate the number of updates accumulated over the optimization. Higher values on the vertical axis indicate higher priority for that subproblem. The horizontal red line indicates the default number of updates (with no Resource Allocation).

shy in their allocation of resources. Figure 3 show the same thing on function DTLZ1, focusing on the DS and R.I. priority functions. This may be the reason why DS showed better IGD values across most of the functions in this experiment. The same pattern can be seen across all functions in the benchmark, please check the code repository for more information.

5 Conclusion

In this paper, we study the choice of Priority Function on Resource Allocation Techniques in MOEA/D. We propose a simple algorithm (MOEA/D-SRA) which adds Resource Allocation in a straightforward way to MOEA/D-DE, and we compare four different Priority Functions in this algorithm.

Our results on the DTLZ and UF benchmarks show that the DS Priority Function obtains better IGD in most of the functions, and a much higher proportion of non-dominated solutions in all of the studied functions. An analysis of the actual resource allocation shows that this is because DS identifies more clearly the subproblems that require resource allocation, while the other priority functions, including the R.I., which is often used in the literature, are much more timid in their prioritization of subproblems. From these results, we can clearly recommend the use of DS instead of R.I. in Resource Allocation, and the use of Resource Allocation in general when applying MOEA/D based methods, with the MOEA/D-SRA as a simple improvement over MOEA/D-DE.

One surprising observation was that the Algorithm which assigned random priority values to subproblems performed as well as R.I., and better than not using Resource Allocation at all. This suggests that holding part of the population inactive at every iteration of MOEA/D might have a positive effect unrelated to resource allocation. As future work, we are further investigating this effect,

aiming to find out what would be the optimal fraction of the population that should be kept inactive to improve the performance of standard MOEA/D.

References

1. Bezerra, L.C.T., López-Ibáñez, M., Stützle, T.: Comparing decomposition-based and automatically component-wise designed multi-objective evolutionary algorithms. In: Gaspar-Cunha, A., Henggeler Antunes, C., Coello, C.C. (eds.) EMO 2015. LNCS, vol. 9018, pp. 396–410. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-15934-8_27
2. Cai, X., Li, Y., Fan, Z., Zhang, Q.: An external archive guided multiobjective evolutionary algorithm based on decomposition for combinatorial optimization. IEEE Trans. Evol. Comput. **19**(4), 508–523 (2015)
3. Campelo, F., Aranha, C.: MOEADr: Component-wise MOEA/D implementation, R package version 1.2.0 (2018). https://cran.R-project.org/package=MOEADr
4. Chankong, V., Haimes, Y.: Multiobjective Decision Making: Theory and Methodology. North Holland, New York (1983)
5. Chiang, T.C., Lai, Y.P.: MOEA/D-AMS: improving MOEA/D by an adaptive mating selection mechanism. In: 2011 IEEE Congress on Evolutionary Computation (CEC), pp. 1473–1480. IEEE (2011)
6. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable test problems for evolutionary multiobjective optimization. In: Abraham, A., Jain, L., Goldberg, R. (eds.) Evolutionary Multiobjective Optimization, pp. 105–145. Springer, London (2005). https://doi.org/10.1007/1-84628-137-7_6
7. Kang, Q., Song, X., Zhou, M., Li, L.: A collaborative resource allocation strategy for decomposition-based multiobjective evolutionary algorithms. IEEE Trans. Syst. Man Cybern. Syst. (2018)
8. Kohira, T., Kemmotsu, H., Akira, O., Tatsukawa, T.: 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)
9. Lavinas, Y., Aranha, C., Sakurai, T.: Using diversity as a priority function for resource allocation on MOEA/D. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. ACM (2019)
10. Li, H., Zhang, Q.: Multiobjective optimization problems with complicated pareto sets, MOEA/D and NSGA-II. IEEE Trans. Evol. Comput. **13**(2), 284–302 (2009)
11. Nasir, M., Mondal, A.K., Sengupta, S., Das, S., Abraham, A.: An improved multiobjective evolutionary algorithm based on decomposition with fuzzy dominance. In: 2011 IEEE Congress on Evolutionary Computation (CEC), pp. 765–772. IEEE (2011)
12. Trivedi, A., Srinivasan, D., Sanyal, K., Ghosh, A.: A survey of multiobjective evolutionary algorithms based on decomposition. IEEE Trans. Evol. Comput. **21**(3), 440–462 (2017)
13. Zhang, Q., Li, H.: MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans. Evol. Comput. **11**(6), 712–731 (2007)
14. Zhang, Q., Liu, W., Li, H.: The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances. In: 2009 IEEE Congress on Evolutionary Computation, CEC 2009, pp. 203–208. IEEE (2009)

15. Zhang, Q., Zhou, A., Zhao, S., Suganthan, P.N., Liu, W., Tiwari, S.: Multiobjective optimization test instances for the CEC 2009 special session and competition. University of Essex, Colchester, UK and Nanyang Technological University, Singapore, special session on performance assessment of multi-objective optimization algorithms, Technical report 264 (2008)
16. Zhou, A., Zhang, Q.: Are all the subproblems equally important? Resource allocation in decomposition-based multiobjective evolutionary algorithms. IEEE Trans. Evol. Comput. **20**(1), 52–64 (2016)
17. Zitzler, E., Künzli, S.: Indicator-based selection in multiobjective search. In: Yao, X., et al. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 832–842. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30217-9_84