Revisiting the NSGA-II Crowding-Distance Computation

Félix-Antoine Fortin
felix-antoine.fortin.1@ulaval.ca

Marc Parizeau
marc.parizeau@gel.ulaval.ca

Laboratoire de vision et systèmes numériques
Département de génie électrique et de génie informatique
Université Laval, Québec (Québec), Canada G1V 0A6

ABSTRACT

This paper improves upon the reference NSGA-II procedure by removing an instability in its crowding distance operator. This instability stems from the cases where two or more individuals on a Pareto front share identical fitnesses. In those cases, the instability causes their crowding distance to either become null, or to depend on the individual's position within the Pareto front sequence. Experiments conducted on nine different benchmark problems show that, by computing the crowding distance on unique fitnesses instead of individuals, both the convergence and diversity of NSGA-II can be significantly improved.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*heuristic methods*

Keywords

Multi-objective evolutionary algorithms; NSGA-II; crowding distance

1. INTRODUCTION

The last decade was a fertile breeding ground for multi-objective evolutionary algorithms (MOEAs) [1]. They have led to several well-known second generation algorithms such as SPEA2 [16], PAES [10], PESA [2] and NSGA-II [5], that have emerged over time as reference platforms for solving real world multi-objective problems. But Zhou et al. have recently observed [14] that most MOEAs used in today's research and application areas in fact share a common framework with Deb's NSGA-II [5] algorithm.

The popularity of NSGA-II can be attributed to its relatively low computing and storage complexities, which can even be further reduced using an improved algorithm for non-dominated sorting [8], its usage of elitism and its often praised parameterless approach. Contrary to its predecessor NSGA [13], it does not rely on a fitness sharing parameter to preserve diversity, but on a simpler mechanism called the crowding distance [5].

This metric provides NSGA-II with a density estimation of solutions for any specific Pareto front. It is defined as the distance of two neighboring solutions on either side of a solution along each objective axis. However, this formulation of the crowding distance-computation is unstable when two or more solutions share the same fitness. The density estimate assigned to a solution can actually vary between multiple calculations for the same solutions and between implementations of the same procedure. This problem combined with the choice of different selection strategies cannot only impact the convergence and diversity preservation capabilities of NSGA-II, but also of most other MOEAs based on the NSGA-II framework. For example, the recent work of Sindhya et al. [12] who use the crowding distance to preserve diversity in a hybrid MOEA framework.

In this paper, we identify the cause of this instability, propose a remedial, and show that it can improve both convergence and diversity for several bi-objective benchmark functions, without affecting time complexity. Moreover, attention is focused on keeping constant the original algorithm's behavior when every fitness is distinct, and on keeping the proposed fixes in line with the essential logic of the crowding metric. Even though this study is limited to vanilla NSGA-II, results are expected to be transposable to other MOEAs that employ crowding-distance computation.

The rest of this paper is structured as follows. Section 2 presents the problem identified in NSGA-II's crowding-distance method of computation. Then, Section 3 details the proposed approach for solving this problem. Experimental results comparing the original procedure to the new approach follow in Section 4. Finally, conclusions are drawn in Section 5.

2. PROBLEM OVERVIEW

NSGA-II is typically used to solve multi-objective optimization problems (MOPs). We define a MOP as the minimization of :

$$\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_M(\mathbf{x})]$$

where M is the number of objective functions $f_i : \mathbb{R}^n \mapsto \mathbb{R}$. Solutions are composed of n decision variables $\mathbf{x} = [x_1, x_2, \ldots, x_n]$. A solution $\mathbf{p}$ is said to dominate another solution $\mathbf{q}$ if $f_i(\mathbf{p}) \leq f_i(\mathbf{q})$ for all $i = 1, \ldots, M$ and $f_j(\mathbf{p}) < f_j(\mathbf{q})$ for a least one j. This dominance relationship can be used to attribute rank: a rank of 1 corresponds to a

GECCO'13, July 6–10, 2013, Amsterdam, The Netherlands.

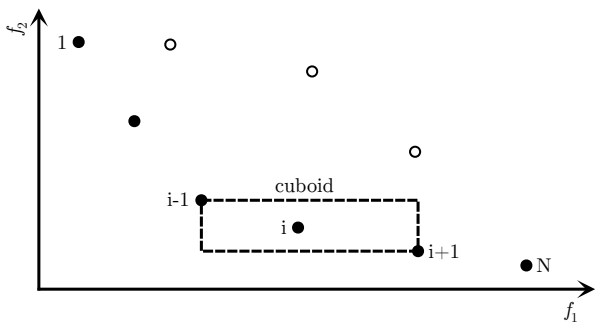


Figure 1: Crowding-distance cuboid example. Points marked in filled circles are solutions of the same nondominated front.

non-dominated individual; in general the rank indicates the index of the Pareto front to which belongs the solution.

To guide the selection process along the evolution, NSGA-II defines a crowded-comparison operator ($\prec_c$). Every individual $\mathbf{i}$ in a population has two attributes: a nondomination rank ($\mathbf{i}_{\text{rank}}$) and a crowding distance to the closest neighbors ($\mathbf{i}_{\text{dist}}$). The partial order $\prec_c$ is based on these two attributes and defined as :

$$\mathbf{i} \prec_c \mathbf{j} := \mathbf{i}_{\text{rank}} < \mathbf{j}_{\text{rank}} \vee (\mathbf{i}_{\text{rank}} = \mathbf{j}_{\text{rank}} \wedge \mathbf{i}_{\text{dist}} > \mathbf{j}_{\text{dist}})$$

The operator favors solutions with lesser domination rank when two solutions lie on different fronts, but when both belong to the same front, the one in a less dense region, corresponding to a higher crowding distance, is preferred.

To estimate the density of solutions surrounding a particular solution on a front, Deb et al. suggest to consider each of the M objectives independently, and sum the distances between the nearest solutions that have respectively smaller and greater value for the objective. This quantity $\mathbf{i}_{\text{dist}}$, called the crowding distance, is illustrated in Figure 1 (taken from Deb et al. [5]) for the case $M = 2$, where crowding is simply measured as an L_1 distance between neighboring solutions $i - 1$ and $i + 1$. For $M \geq 3$, however, the analogy does not hold, because objectives are processed independently and nearest neighbor solutions may change for each objective.

The crowding distance computation algorithm is presented in Figure 2. $\mathcal{F}$ represents a Pareto front composed of $N = |\mathcal{F}|$ individuals. $\mathcal{F}[i].m$ refers to the mth objective of the ith individual in front $\mathcal{F}$. Parameters $f_m^{\min}$ and $f_m^{\max}$ are the minimum and maximum values for objective m. Every individual has its distance initialized to 0 (line 4). For each objective m, the individuals are first sorted in ascending order based on their value for this objective (line 7). The boundary solutions, those with smallest and largest objective value, are assigned an infinite crowding distance (line 8). Then, for each intermediary solution, the procedure computes the normalized difference between the following and preceding individuals for the current objective m, and sums it to the individual crowding distance.

If two solutions i and j share the same fitness, that is

$$\sum_{m=1}^{M} |\mathcal{F}[i].m - \mathcal{F}[j].m| = 0,$$

each is attributed a different crowding distance. This situation is illustrated in Figure 3, the solutions i and $i + 1$

```
1: procedure CrowdingDistance(F)
2:     N = |F|
3:     for i = 1 ... N do
4:         F[i]_dist = 0
5:     end for
6:     for m = 1, ..., M do
7:         sort(F, m)
8:         F[1]_dist = F[N]_dist = ∞
9:         for i = 2 ... N − 1 do
10:            F[i]_dist = F[i]_dist + (F[i+1].m − F[i−1].m) / (f_m^max − f_m^min)
11:        end for
12:    end for
13: end procedure
```

Figure 2: Crowding-distance computation algorithm.

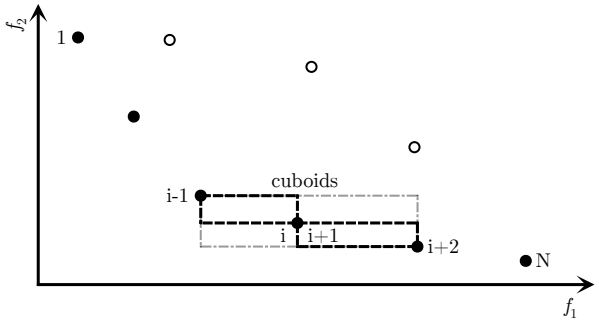


Figure 3: Crowding-distance graphical computation example where two solutions i and $i + 1$ share the same fitness.

have the same fitness. This fitness is normally associated to a crowding distance represented by the cuboid of Figure 1, also drawn in grey in Figure 3. However, since both i and $i + 1$ have the same fitness, their respective cuboids might be quite different. If three or more solutions share the same fitness, every fitness except the two boundaries along each objective will be assigned a crowding distance of 0.

This formulation of the crowding distance presents an instability in the produced results when the front $\mathcal{F}$ includes some individuals sharing the same fitness. This instability stems from the sort at line 7 of Figure 2. Assuming an unstable sort, the crowding distance assigned to individuals sharing the same fitness can vary between calculations. Most implementations, including Deb's reference implementation, use an unstable quicksort to sort the individuals based on the value of the mth objective, resulting in an unstable crowding-distance computation. Given a front of 7 individuals, examples of possible results are presented in Table 1. As can be seen, solutions with identical fitnesses can be assigned different crowding distance and, more dramatically, boundary solutions can even be swapped, resulting in multiple individuals receiving an infinite crowding distance. Moreover, two solutions sharing the same fitness do not necessarily share the same genotype. The fitness diversity preservation mechanism might therefore impact negatively the parameter-space diversity.

Using a stable sort algorithm could solve the problem of the crowding-distance instability, but individuals sharing the same fitnesses will still be attributed different crowding distance, depending on their position in the front sequence. We

index	fitness	distances 1	distances 2	distances 3
1	$(0, 5)$	∞	∞	5
2	$(0, 5)$	∞	5	∞
3	$(2, 2)$	7	7	7
4	$(3, 1)$	0	2	1
5	$(3, 1)$	3	1	3
6	$(3, 1)$	2	2	1
7	$(5, 0)$	∞	∞	∞

Table 1: Three examples of crowding-distance computation using an unstable quicksort on a front of 7 individuals. The distances are not normalized.

allege that these problems are attributable to an unformulated hypothesis regarding uniqueness of fitnesses. That is, for the reference crowding distance to work as intended, individuals should have distinct fitnesses. This hypothesis does not stand in practice as we will see in the results section with benchmark functions. The key to what we are proposing for guiding the NSGA-II selection during evolution is to strictly limit ourselves to unique fitnesses instead of individuals.

3. PROCESSING UNIQUE FITNESSES

In this section, we present the modifications needed to fix the instability in the crowding distance of NSGA-II and to limit the bias induced by individuals sharing the same fitness during selection. Section 3.1 covers the crowding-distance computation with unique fitnesses. Section 3.2 presents an operator to select fitness and associated individuals, using the crowded-comparison operator $\prec_c$. Section 3.3 describes how individuals on the last Pareto front can be selected using the crowding distance, even when individuals share the same fitness. Finally, section 3.4 summarizes the general approach and presents the different operators in the context of the revised NSGA-IIr main loop.

3.1 Crowding-Distance Computation

In our approach, the crowding-distance computation presented in Figure 2 remains unchanged. However, the provided input $\mathcal{F}$ is replaced with unique fitnesses instead of individuals. A set of unique fitnesses associated with a front $\mathcal{F}$ is defined as :

$$\{\mathbf{f}(\mathbf{i}) \mid \mathbf{i} \in \mathcal{F}\}$$

Once the crowding distance is computed for the unique fitnesses, the distances are assigned to individuals with the corresponding fitness. With this approach, every individual sharing the same fitness inherits the same crowding distance and therefore the minimum crowding distance is always greater than 0. The crowding distance calculated this way represents a correct density estimation in the objective space when individuals share the same fitness. The advantage of the previous formulation was that some individuals with equal fitnesses could be assigned a crowding distance of 0, limiting their chances of reproduction and therefore stimulating diversity preservation. However, we argue that the diversity preservation goal should not be accomplished with a biased metric, but a correct interpretation of the metric through the selection process. The next section describes how unique fitnesses can be used to select a new population.

```
1: procedure UFTournSelection(I)
2:     N = |I|
3:     F = {f(i) | i ∈ I}
4:     if |F| = 1 then
5:         return I
6:     end if
7:     S = {}
8:     while |S| ≠ N do
9:         k = min{2(N − |S|), |F|}
10:        G = sample(F, k)
11:        for j = 1, 3, 5, ..., |G| − 1 do
12:            if G[j] ≺c G[j + 1] then
13:                p = G[j]
14:            else if G[j + 1] ≺c G[j] then
15:                p = G[j + 1]
16:            else
17:                p = sample({G[j], G[j + 1]}, 1)
18:            end if
19:            S = S ∪ {sample({i ∈ I | f(i) = p}, 1)}
20:        end for
21:    end while
22:    return S
23: end procedure
```

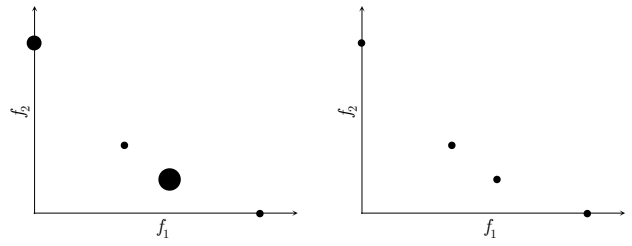
Figure 4: Unique fitness based tournament selection.

3.2 Unique Fitness Selection

In NSGA-II, the parent solutions are selected by a binary tournament operator. Furthermore, in Deb's reference implementation, the tournament is organized in such a way that an individual may be present at most two times amongst nominees. Our new operator is based on the same tournament selection logic so that if every individual has a different fitness, the process is identical.

The unique fitness binary tournament selection is presented in Figure 4. The operator takes a population $\mathcal{I}$ of N individuals where every fitness has previously been assigned a rank and a crowding distance. The crowded-comparison operator $\prec_c$ can therefore be used to compare them. The operator starts by building a set F of unique fitnesses associated with the individuals (line 3). In the extreme event where every solution in the population shares the same fitness, no selection occurs and the population is simply returned as is (lines 4 and 5). Otherwise, a set of solutions S is initialized empty, and while the number of selected solutions is not equal to the size of the population, the procedure samples k fitnesses that are stored in a list G. The procedure $\text{SAMPLE}(L, k)$ takes a list L and returns a list of k randomly chosen elements without replacement. The number k of sampled fitnesses is the minimum value between the number of unique fitnesses, corresponding to the cardinality of F, and two times the number of solutions left to select, $N - |S|$. The fitnesses are paired and compared using the crowded-comparison operator ($\prec_c$). A solution associated with the winning fitness is then randomly selected amongst the individuals sharing this fitness and added to set S. After the selection of N solutions, set S is returned.

The selection of unique fitnesses lifts the selection bias towards over-represented fitnesses by reducing multiple solutions sharing the same fitness to a single point in the objective space. It is therefore no longer required to assign a crowding distance of zero to individual of equal fitness as the



(a) Selection on individuals (b) Selection on fitnesses

Figure 5: Comparison of the effect on the selection of solutions when choosing amongst individuals vs fitnesses. The fitness points are from Table 1. The point radii are proportional to their number of representatives during selection.

selection operator now correctly enforces diversity preservation by picking unique points in the objective space.

A small scale illustrative example of the difference in bias between the original binary tournament and the one we propose is presented in Figure 5. Represented are the fitnesses from Table 1. Basically, using the original tournament selection, we observe that some of the points have more chances of being picked for a tournament based solely on their number of representatives and therefore have an artificial higher chance of survival. However, if the selection is based on unique fitnesses, we see that every point in the objective space has an equal chance of participating in a tournament, thus the chance of becoming a parent is strictly proportional to the rank and the crowding distance.

3.3 Last Front Selection

NSGA-II elitist $(\mu + \mu)$ strategy consists in keeping from the combination of $2N$ parents and offspring, the N best solutions situated on the first Pareto fronts $\mathcal{F}_1$ to $\mathcal{F}_l$. When the total number of solutions contained in these sets exceeds the population size, the crowding distance is used to select the individuals on the *last* front $\mathcal{F}_l$ to reduce the count to N. Solutions are sorted in descending order of crowding distance, and the first ones are kept to complete the selection. However, in our approach, individuals sharing the same fitness also share the same crowding distance, it therefore cannot be directly used as a diversity preservation mechanism. We propose to replace the sort by an operator that can help promote diversity while preserving the same behavior as before, when each individual in $\mathcal{F}_l$ has a distinct fitness.

The last front selection procedure is presented in Figure 6. It takes two arguments: a front $\mathcal{F}$ and the number of solutions to select k. The algorithm starts by sorting the unique fitnesses associated with the members of $\mathcal{F}$ in descending order of crowding distance ($>_{\text{dist}}$) and stores the result in a list F (line 2). It then proceeds to fill the initially empty set of solutions S by cycling over the sorted unique fitnesses F. For each fitness, the algorithm first selects each individual sharing this fitness and puts the result in set T (line 6). If the resulting set is not empty, the procedure randomly picks one solution from T, adds it to set S and finally removes it from the front $\mathcal{F}$ so that individuals can be picked only once (lines 8 to 10). When all k solutions have been selected, the loop stops and set S is returned (line 14).

By using a round-robin selection based on the crowding distance, we make sure that diversity is preserved in the objective space by preferring solutions with higher crowding distance. The diversity is also enforced by requiring individuals with different fitnesses to be selected before selecting higher crowding-distance points again.

```
1: procedure LastFrontSelection(F, k)
2:     F = sort({f(i) | i ∈ F}, >dist)
3:     S = {}
4:     j = 1
5:     while |S| ≠ k do
6:         T = {i ∈ F | f(i) = F[j]}
7:         if T ≠ {} then
8:             s = sample(T, 1)
9:             S = S ∪ {s}
10:            F = F \ {s}
11:        end if
12:        j = j (mod |F|) + 1
13:    end while
14:    return S
15: end procedure
```

Figure 6: Procedure that selects k individuals on the last front $\mathcal{F}_l$ based on the crowding distance of unique fitnesses using a round-robin algorithm.

3.4 Main Loop

The main logic of the original NSGA-II is mostly unchanged by our approach. Therefore most variants should be able to take advantage of the modifications we are proposing and possibly improve their performance when solutions share identical fitnesses. The global approach and the application of the operators presented in the previous sections are summarized by the NSGA-IIr procedure of Figure 7.

The procedure starts by initializing two populations of N individuals each, parents P_1 and offspring Q_1. At each generation t, both population are combined in R_t and then sorted in a list of Pareto fronts $\mathcal{F}$, where $\mathcal{F}_1$ is the first non-dominating front. Each front is added in order to the next parent population P_{t+1} which is initially empty. A crowding distance is assigned to every unique fitness of the front $\mathcal{F}_j$ before adding it to the rest of the population. The loop stops when the number of individuals in a front exceeds the number of spots left in P_{t+1} in order to respect the constraint of N solutions. The k remaining individuals are then selected from the last jth front using the LastFrontSelection procedure described previously and added to complete the parent population P_{t+1}.

The actual parents P'_{t+1} of the next generation offspring are selected using the unique fitnesses tournament selection introduced earlier. In the original NSGA-II, the selection is part of a make-new population procedure. However, it appears in our results that this second round of selection is in part responsible for the performance delivered by NSGA-II and should therefore be explicitly stated in the algorithm. Finally, a new generation of offspring Q_{t+1} is created by recombination and mutation of the individuals in P'_{t+1}.

Regarding time complexity, establishing the set of unique fitnesses can be done in $O(N)$, the last front selection is dominated by the sort in $O(N \log N)$, and the unique fitness tournament selection in $O(2N) = O(N)$. The adopted modifications therefore do not change the overall time complexity which is essentially bounded by the nondominated sort.

```
1: procedure NSGA-IIr
2:     Initialize P_1 and Q_1
3:     for t = 1, ..., NGEN do
4:         R_t = P_t ∪ Q_t
5:         F = NONDOMINATEDSORT(R_t)
6:         P_{t+1} = {}
7:         j = 1
8:         while |P_{t+1}| + |F_j| ≤ N do
9:             F = {f(i) | i ∈ F_j}
10:            CROWDINGDISTANCE(F)
11:            P_{t+1} = P_{t+1} ∪ F_j
12:            j = j + 1
13:        end while
14:        k = N − |P_{t+1}|
15:        P_{t+1} = P_{t+1} ∪ LASTFRONTSELECTION(F_j, k)
16:        P'_{t+1} = UFTOURNSELECTION(P_{t+1}, N)
17:        Q_{t+1} = GENERATEOFFSPRING(P'_{t+1})
18:    end for
19: end procedure
```

Figure 7: NSGA-IIr main loop.

This operation can be accomplished in either $O(MN^2)$ using Deb's fast nondominated sorted [5] or in $O(N \log^{M-1} N)$ using a more efficient divide and conquer approach [8].

4. EXPERIMENTS

As previously stated, the reference NSGA-II algorithm includes two distinct phases: a Pareto and crowding distance elitist selection for downsizing the $2N$ parents + offspring population, on the one hand, and a binary tournament selection to generate the next generation offspring, on the other hand. The first question that we want to raise is how important is this second phase in the overall performance of NSGA-II? This question is important, because it appears to have not been addressed before in the literature. Moreover, it is not clear whether or not every implementation of NSGA-II really implements this step as both its description and its role is not made very explicit in the original paper. We therefore start by comparing the performances of NSGA-II with and without tournament selection for offspring generation, before carrying on with a comparison of the reference NSGA-II with the proposed NSGA-IIr.

To conduct our experiments, we selected nine benchmark problems from the literature. We limit this study to 2-objective problems, as MOEAs for many objectives are currently shifting their focus from evaluation of diversity measures, like the crowding distance, to the usage of predefined multiple targeted searches [4]. The first five benchmarks are ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6, which were initially designed by Zitzler et al. [15], and later named and used by Deb et al. [5]. The remaining four are DTLZ1, DTLZ2, DTLZ3 and DTLZ4, designed by Deb et al. [6] for an arbitrary number of objectives and decision variables. They are respectively set to 2 objectives and 30 variables.

For each algorithm and each problem, populations of 100 individuals are evolved over 250 generations. The decision variables are encoded into the solutions with real numbers defined on the range specified for each problem. Offsprings are created by pairing selected parents. When no tournament selection is applied, pairs are created at random. The recombination and mutation operators are respectively the simulated binary crossover (SBX) and the polynomial mutation [3]. The parameters of the operators and their probability of application are based on values commonly found in the literature:

- crossover probability (p_c): 0.9;
- mutation probability (p_m): $1/n$, where n is the number of decision variables;
- crossover distribution index (η_c): 20;
- mutation distribution index (η_m): 20.

Performances are measured in terms of convergence towards the optimal Pareto front, on the one hand, and diversity of solutions along that front, on the other hand, using the two metrics introduced by Deb et al. [5]. Convergence is measured as the average minimum Euclidean distance between last generation individuals and uniformly sampled points on the optimal Pareto front. As reference points for the Pareto front, a set of 1000 points is used for each problem, provided online by the System Optimization group of the Swiss Federal Institute of Technology Zurich[1]. The smaller the value for this metric, the better the convergence towards the optimal Pareto front. Diversity is measured as the non-uniformity of the average distance between adjacent solutions of the optimal Pareto front. The closer the metric is to 0, the better the diversity of solutions. For each metric, results are reported as averages over 100 independent runs.

To determine whether observed differences between mean results are significant, we use the Mann-Whitney U test and report resulting p-values. We set the threshold $\alpha \leq 0.01$ for strongly rejecting the null hypothesis and highlight these results in bold. For $0.01 < \alpha \leq 0.05$ we also consider the results significant, but with a weaker confidence.

Algorithms are implemented in Python using the DEAP framework [7], and executed under CPython 2.7.3. The Mann-Whitney U tests are executed using SciPy 0.12.0-dev.

4.1 Results

Table 2 presents the average convergence and diversity metrics for NSGA-II, with and without binary tournament selection for generating offspring. As can be seen, the convergence is significantly better when using tournament selection for five out of nine benchmarks (ZDT1, ZDT2, ZDT3, ZDT6, and DTLZ3), while significantly worse for only one (DTLZ4). On the other hand, the diversity appears mostly unaffected by the usage of the new operator on most benchmarks except for ZDT4, DTLZ2, DTLZ3 and DTLZ4. Diversity is increased for ZDT4, DTLZ3 and DTLZ4, and decreased for DTLZ2. In light of theses results, the usage of a second round of non-elitist selection for generating offspring appears justified in order to achieve better convergence towards the optimal Pareto front. We therefore expect the proposed tournament selection using unique fitnesses to provide at least similar performances since it is based on the same logic. However, the impact of the tournament on final solutions diversity appears negligible on most test cases.

Table 3 presents results for the comparison of NSGA-II and NSGA-IIr. The columns for NSGA-II are the same as those presented in Table 2 for the algorithm with tournament selection. For every problem except ZDT3, a strongly significant convergence advantage is observed for the proposed NSGA-IIr algorithm. This result can be surprising

[1] `http://www.tik.ee.ethz.ch/sop/download/supplementary/testproblems/`

	Convergence				Diversity			
Problem	without tourn. sel.	with tourn. sel.	gain	p-value	without tourn. sel.	with tourn. sel.	gain	p-value
ZDT1	0.00143	0.00137	4.10%	**0.007**	0.35585	0.35794	-0.59%	0.488
ZDT2	0.00140	**0.00126**	**9.57%**	**0.000**	0.36451	0.36726	-0.75%	0.226
ZDT3	0.00150	**0.00138**	**7.38%**	**0.000**	0.54967	0.55060	-0.17%	0.313
ZDT4	0.00508	0.00434	14.4%	0.166	0.44572	**0.36698**	**17.6%**	**0.000**
ZDT6	0.01156	**0.00678**	**41.3%**	**0.000**	0.34431	0.34961	-1.54%	0.072
DTLZ1	25.756	24.654	4.28%	0.136	0.84028	0.84591	-0.67%	0.264
DTLZ2	0.00180	0.00180	-0.41%	0.255	0.35899	0.36717	-2.28%	$0.029^{\dagger}$
DTLZ3	68.492	**62.140**	**9.27%**	**0.009**	0.94664	**0.91473**	**3.37%**	**0.004**
DTLZ4	0.00109	0.00123	-12.8%	$0.034^{\dagger}$	0.55657	0.52243	6.13%	$0.022^{\dagger}$

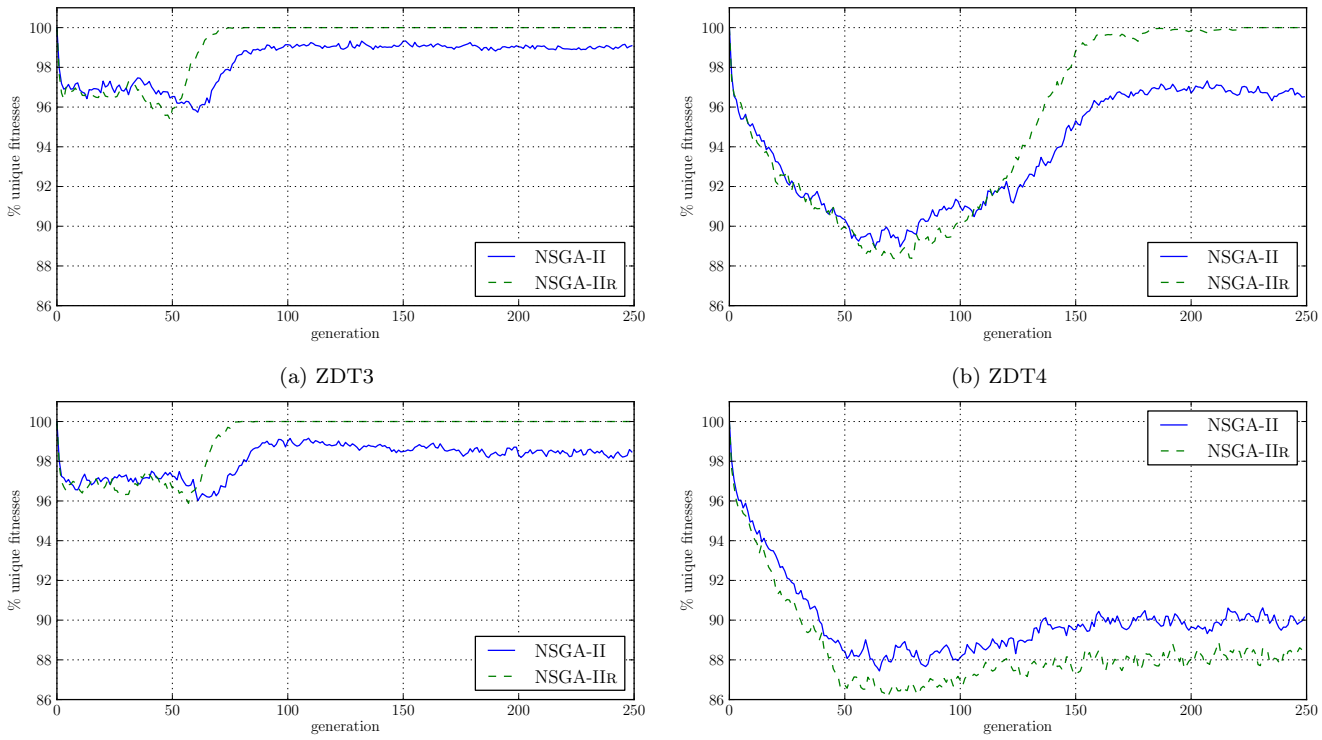
Table 2: Comparison of the average convergence and diversity metrics for the baseline NSGA-II algorithm, with and without binary tournament selection. Results in bold are strongly significant, while the ones marked with a dagger† are moderately significant.

	Convergence					Diversity			
Problem	NSGA-II	NSGA-IIr	gain	p-value	1% gen.	NSGA-II	NSGA-IIr	gain	p-value
ZDT1	0.00137	**0.00125**	**8.78%**	**0.000**	241	0.35794	**0.34240**	**4.34%**	**0.000**
ZDT2	0.00126	**0.00113**	**10.4%**	**0.000**	238	0.36726	**0.34676**	**5.58%**	**0.000**
ZDT3	0.00139	0.00136	1.73%	0.071	239	0.55060	0.54490	1.04%	$0.025^{\dagger}$
ZDT4	0.00434	**0.00345**	**20.5%**	**0.001**	225	0.36698	**0.34715**	**5.40%**	**0.000**
ZDT6	0.00678	**0.00564**	**16.8%**	**0.000**	237	0.34961	**0.33188**	**5.07%**	**0.000**
DTLZ1	24.654	**20.331**	**17.5%**	**0.000**	219	**0.84591**	0.86284	**-2.00%**	**0.009**
DTLZ2	0.00180	**0.00173**	**3.65%**	**0.001**	231	0.36717	**0.34708**	**5.47%**	**0.000**
DTLZ3	62.140	**50.765**	**18.3%**	**0.000**	220	**0.91474**	0.94670	**-3.49%**	**0.000**
DTLZ4	0.00123	**0.000924**	**24.8%**	**0.000**	181	0.52243	0.62706	-20.0%	0.493

Table 3: Comparison of the average convergence and diversity metrics between the baseline NSGA-II and the proposed NSGA-IIr algorithm. Results in bold are strongly significant, while the ones marked with a dagger† are moderately significant.

given that benchmark objectives are defined in a continuous space and that identical fitnesses should therefore happen infrequently. However, we have observed that the number of unique fitnesses actually evolves over generations, and that it can even fall under 90% in some circumstances. Figure 8a and 8b shows the evolution of the average percentage of unique fitnesses for ZDT3 and ZDT4. We observe that ZDT3 curves for NSGA-II and NSGA-IIr have similar shapes, although NSGA-IIr is able to converge faster to a population composed entirely of unique fitnesses. We can see that for this problem, the number of non-unique fitnesses is always low, and therefore NSGA-IIr offers little convergence gain, as reported in Table 3. However, for ZDT4 the initial number of non-unique fitnesses is up to three times higher, and NSGA-IIr has more room to make a difference. Around generation 70, more than 10% of individuals shared their fitness with a least another individual, which tends to show that the new crowding distance can play a more significant impact, especially during the exploration phase of the evolution. Another example of this effect is visible for problems ZDT1 and DTLZ3, as shown in Figures 8c and 8d, where the convergence gain is much higher for DTLZ3 than for ZDT1. Another interesting observation is that for ZDT1, ZDT3, and ZDT4, but not for DTLZ3, the number of unique fitnesses is always lower for NSGA-IIr at first, but later becomes higher, eventually reaching 100%. We postulate that the exception stems from the fact that DTLZ3 has not been able to converge, given the uniform parametrization that we have set for all problems. Indeed, even though the convergence metric for NSGA-IIr has reached a significantly better level, neither of the two algorithms were able to come close to the problem's optimal Pareto front.

Figure 9 presents the evolution of the convergence gain of NSGA-IIr over NSGA-II. The gain is always strictly positive for every problem during the entire evolution, except for ZDT4. In every case the convergence metric is strictly decreasing, however for ZDT4 we observe a brief increases of the convergence metric during the evolution for both algorithms. This explains the drastic variations of the gain for this problem. The graphic y-axis was cut to -20%, but for generation 200, the gain is actually around -60% before returning to over 50% a few generation later. We also observe that for most functions, the convergence gain is at its maximum when the percentage of unique fitnesses produced by NSGA-II is at its minimum. ZDT3 is an example of this, represented by the red curve in Figure 9 and the blue one in Figure 8a. We see that for ZDT3, the average percentage of unique fitnesses with NSGA-II is at its minimum, under 96%, around generation 60, while the gain reaches over 37% at the same time. The gain then rapidly decreases as NSGA-II percentage of unique fitnesses increases and the algorithm converges towards the optimal front. The gain offered by NSGA-IIr is higher during the first 150 generations for each problem and generally decreases to the value presented in



(a) ZDT3

(b) ZDT4

(c) ZDT1

(d) DTLZ3

Figure 8: Evolution of the average number of unique fitnesses.

Table 3. These results are therefore fairly conservative since the number of generations is fixed and both algorithms are allocated enough time to converge on every problem except DTLZ1 and DTLZ3. We computed the generation at which NSGA-IIR outperformed by 1% the best result of NSGA-II. The generation indices are given in column *1% gen.* of Table 3. We note that the proposed approach converges faster and maintains a superior convergence rate for most problems.

As mentioned previously, both algorithms are unable to converge to the optimal Pareto front on problems DTLZ1 and DTLZ3. Nevertheless, NSGA-IIR was able in both cases to significantly improve the convergence metric, even though it constantly maintained during the whole 250 generations a higher number of individuals with non-unique fitnesses. By maintaining the parameters identical for every problem, we actually revealed an interesting fact about NSGA-IIR, that it has the potential to reduce the sensitivity of NSGA-II to evolution parameters. We observe that for these problems the number of non-unique individuals is higher. This results from the parametrization, as operators should normally produce diverse solutions. However, since NSGA-IIR strictly processes unique fitnesses, this problem is diminished and its impact on the convergence reduced, therefore explaining the better performances on these problems.

Regarding diversity in the final Pareto front, we typically observe that NSGA-IIR allows for a significant gain of around 5% on five out of nine benchmarks. Out of the remaining four, only the DTLZ1 and DTLZ3 diversities were significantly decreased, but since their convergence metric was much improved, albeit not enough to reach their optimal Pareto-set, we argue that this result may not be representative. The general performance of NSGA-IIR in maintaining diversity can also be attributed to unique fitnesses. Since, for most cases, NSGA-IIR was able to converge to an optimal Pareto-set composed of strictly different fitnesses, the diversity is favored as no points on the Pareto front are the same, which is the first requirement in maximizing inter-point distance.

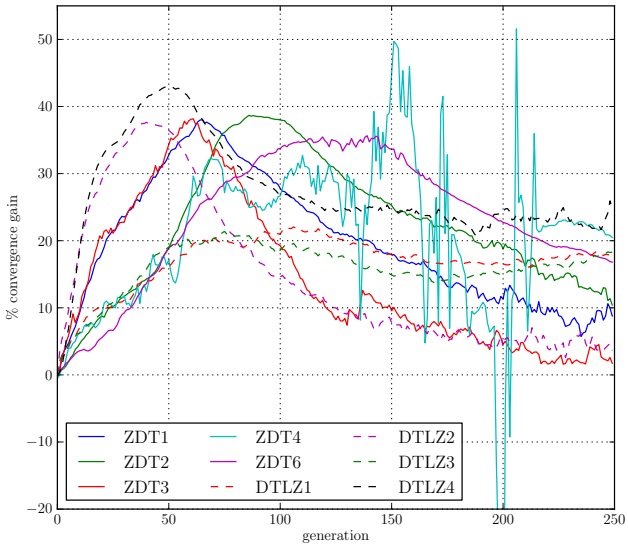


Figure 9: Evolution of the average convergence gain.

5. CONCLUSION

In this paper, we have addressed the issue of instability in the crowding-distance computation when solutions can share identical fitnesses. We have presented the cause of the problem and illustrated its effects on the crowding distance.

To lift this instability, we have modified Deb's algorithm so that every individual sharing a common fitness is assigned the same distance. We have also adapted the binary tournament selection and the last front selection to correctly exploit the proposed crowding distance definition and enforce diversity of solutions. Moreover, we have shown that the original behavior of these operators remains unchanged when the implicit assumption that every fitness is different holds. We have analyzed the importance of the binary tournament selection on the performance of the baseline algorithm. We came to the conclusion that the convergence benefits from the additional selective pressure but the diversity is generally unaffected. Experiments using NSGA-II as a baseline have demonstrated that the proposed NSGA-IIr can perform significantly better in terms of convergence on a majority of two-objective benchmarks commonly used in the literature. The diversity of the final Pareto front produced by NSGA-IIr is also significantly better in some of the cases and a slight to zero impact is observed on the others. Furthermore, for two problems where the parametrization appeared to be inadequate, our approach was nevertheless able to converge better. A careful choice of parameters for the operators and the algorithm should normally reduce in itself the number of non-unique fitnesses present in the population. However, it can be difficult to select the right set of parameters, even more when the problem is black-box which is regularly encountered in real world applications. The proposed approach can therefore be interpreted as a fail-safe measure that tends to mitigate the importance of parameter selection and reduces the need for crafting effort [9, 11].

Future work should study the impact of the proposed approach on problems with more than two objectives and compare the performances to recent algorithms designed to tackle many-objective problems [4, 12]. Moreover, application of NSGA-IIr to discrete and combinatorial MOPs, where a high number of solutions sharing identical fitnesses is common, could be studied to better seize its application spectrum. Finally, it would also be interesting to determine wether other MOEAs could benefit from the concept of unique fitness put forward in this paper.

Acknowledgment

This work has been supported by the Fonds de recherche du Québec - Nature et technologies (FRQNT) and by the Natural Sciences and Engineering Research Council of Canada (NSERC). Computations were made on the Colosse supercomputer at Université Laval, under the administration of Calcul Québec and Compute Canada.

6. REFERENCES

[1] C. Coello Coello. Recent trends in evolutionary multiobjective optimization. In A. Abraham, L. Jain, and R. Goldberg, editors, *Evolutionary Multiobjective Optimization*, Advanced Information and Knowledge Processing, pages 7–32. Springer London, 2005.

[2] D. Corne, J. Knowles, and M. Oates. The pareto envelope-based selection algorithm for multiobjective optimization. In *Parallel Problem Solving from Nature PPSN VI*, pages 839–848. Springer, 2000.

[3] K. Deb and R. Agrawal. Simulated binary crossover for continuous search space. *Complex Systems*, 9:1–34, 1994.

[4] K. Deb and H. Jain. Handling many-objective problems using an improved NSGA-II procedure. In *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pages 1–8, June 2012.

[5] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, Apr. 2002.

[6] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable multi-objective optimization test problems. In *Congress on Evolutionary Computation (CEC 2002)*, pages 825–830. IEEE Press, 2002.

[7] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, and C. Gagné. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 2171–2175(13), Jul. 2012.

[8] F.-A. Fortin, S. Grenier, and M. Parizeau. Generalizing the improved run-time complexity algorithm for non-dominated sorting. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO 2013)*, July 2013. to appear.

[9] N. Hansen, A. Auger, R. Ros, S. Finck, and P. Posik. Comparing results of 31 algorithms from the black-box optimization benchmarking bbob-2009. In ACM, editor, *Workshop Proceedings of the GECCO Genetic and Evolutionary Computation Conference*, pages 1689–1696, 2010.

[10] J. Knowles and D. Corne. Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary computation*, 8(2):149–172, 2000.

[11] K. Price. Differential evolution vs. the functions of the 2nd iceo. In *Evolutionary Computation, 1997., IEEE International Conference on*, pages 153–157, Apr. 1997.

[12] K. Sindhya, K. Miettinen, and K. Deb. A hybrid framework for evolutionary multi-objective optimization. *IEEE Transactions on Evolutionary Computation*, 2012. in press.

[13] N. Srinivas and K. Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.

[14] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang. Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, 1(1):32–49, 2011.

[15] E. Zitzler, K. Deb, and L. Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, 2000.

[16] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength Pareto evolutionary algorithm. Technical Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland, May 2001.