B. Ross
July 2020

Notes on Many-Objective Ranking

A multi-objective problem is one in which there are multiple criteria for measuring the quality of an individual. Pareto ranking is the most common way of assigning fitness scores to individuals. (See the notes document, "More on MOP"). One problem with Pareto ranking is that it collapses when 4 or more objectives are considered. This is because, using Pareto dominance, the more objectives are used, the more likely individuals in the population will be undominated, and will have the same rank value (1). Many real-world problems require the consideration of a multitude of objectives, however. We call problems that have 4 more more objectives *many-objective problems* (MaOPs).

There are alternative strategies for dealing with many-objective optimization problems. One effectives strategy was proposed by {Bentley&Wakefield.], and furher studied by [Corne & Knowles 2007]. Among different strategies tested, the average rank (AR) approach was found to be the most useful one.

Consider the following table. The population has 6 individuals. There are 4 objectives (A, B, C, D), and lower fitness values are preferred. Pareto ranks shows that 4 individuals are undominated, and have rank 1. The other 2 have rank 2. If the population were larger, there would be far more individuals with rank 1 and 2, and the Pareto rank would not be effective because there would be no selective pressure to find strong individuals.

MOP Ranking example **Minimization: low scores preferable**

	Raw fitness				Pareto		Ranks			Sum Ranks	Re-ranked	Sum	Re-ranked
Indiv	A	B	C	D	rank	A	B	C	D			(normalized)	
1	1	9	5	4	1	2	1	2	2	7	1	1.47	1
2	2	100	4	8	1	3	2	1	3	9	2	2.03	2
3	10	9	9	10	2	4	1	4	4	13	4	2.60	5
4	16	100	8	4	2	5	2	3	2	12	3	2.57	4
5	16	9	500	0	1	5	1	5	1	12	3	2.37	3
6	0	1000	1000	1000	1	1	3	6	5	15	5	3.20	6
					max rank:	5	3	6	5				

The ranking portion of the table works as follows. Since lower fitness values are preferred, we will give them priority, and hence lower-value ranks. If you look at objective A, individual #6 has the lowest value of 0, and we assign that rank 1 (the best). The next best value is individual 1 with 1, so we assign it rank 2 (2nd best). The rest of the column is ranked accordingly. There are 2 values with the same rank of 5. The remaining objectives (B, C, D) are similarly ranked separately. At this point, we sum these ranks for each individual, and use the "sum ranks" as the new fitness score, where low values are preferred. Note that this would be identical to the aforementioned "average rank" if we divided the sum by the number of objectives (4). Since scaling all the sums by 4 does not alter the overall ordering of ranks, and has no effect when using tournament selection, the division can be skipped, and we use the sum of ranks (SR) instead.

One problem that can arise with sum of ranks is when different objectives use different ranges of ranks. For example, in the above table, note the "max rank" row below the Ranks. Objective B has just 3 ranks used, while C has 6. In real-life problems, sometimes this happens to a much greater extent with large-sized populations (say, 1000). In these cases, simply summing the ranks can result in biases, in which the objectives that have lots of ranks (high max rank value) take over the overall sum at the expense of those with fewer ranks. For many problems, this results in unusual scores, and odd answers.

A simple solution to this is to "normalize" the ranks by dividing each rank by the maximum rank found for that

objective. In the above, we divide the ranks in column A by 5, those in B by 3, and so on. Then we sum these normalized ranks, resulting in a normalized sum of ranks. All individuals will have normalized SR scores between 0 and the # ranks (4.0 in the above). This approach tends to evenly balance the contributions of each objective to the overall SR score, and usually creates more sensible solutions to problems.

Sometimes a normalized SR is not enough. There are problems in which some objectives are either more critical to solve than others, or are more difficult to optimize. In these cases, one can introduce a weight for a particular objective. In the above table, if objective A is more important to optimize, we might multiply A by a weight of (say) 3, before adding it to the other ranks. Although it is preferrable avoid using additional user parameters as weights, sometimes it cannot be avoided.

References

[Bentley & Wakefield 1998] P.J. Bentley, J.P. Wakefield, "Finding Acceptable Solutions in the Pareto-Optimal Range using Multiobjective Genetic Algorithms". In: Chawdhry P.K., Roy R., Pant R.K. (eds) *Soft Computing in Engineering Design and Manufacturing*. Springer, London. 1998.

[Corne & Knowles 2007] D. Corne, J. Knowles, "Techniques for Highly Multiobjective Optimisation: Some Nondominated Points are Better than Others", *GECCO 2007*.