B. Ross
October 2019
COSC 5P71: Notes on Multi-objective optimization

All of the GA's we have studied up until now have assumed that the fitness function worked on a single objective. For some problems (eg. find maximum of $f(X) = X*X$), it is easy to translate the problem in to a single objective: use $f(X)$ as the fitness function itself.

However, many problems in the real world are defined by multiple characteristics. The Sudoku problem is one example. A correct solution for a puzzle means that each row, column, and grid has the integers between 1 and 9 without repeats. In other words, there are 3 objectives to solving a puzzle : row score, column score, and grid score. A common way to handle such problems is to combine these separate measurements into one overall score:

$$Score = row\_score + col\_score + grid\_score$$

This formula is also known as a *weighted sum*. A more general form for this formula is:

$$Score = (W1*row\_score) + (W2*col\_score) + (W3*grid\_score)$$

The first formula shows that the weights Wi are all equal (to 1 in fact). It is possible to play with the weights, thereby changing the shape of the fitness landscape. For example, if columns happened to be much harder to solve, their weight could be increased relative to the other scores, so that the overall score is much more sensitive to column scores.

Although weighted sums are commonly used, they can also be difficult to use for many kinds of problems. For example, some problems have multiple objectives that are not naturally merged together (unlike Sudoku). Combining the scores is akin to mixing apples and oranges... the scores refer to entirely different concepts that are not directly related to each other. However, they all do affect the overall search space, often in complex and unpredictable ways.

An example of the complexity of using weights is as follows. When different scores are merged, one score or *dimension* may overwhelm the others. This can happen because the scales of score measurements can differ drastically with one another. Perhaps one score commonly ranges between 500 and 1000 in value, while another score is between 0.1 and 0.2. This second score will barely register on the radar because of the large first score. A weighting scheme will need to reduce the first score and increase the second, in order to balance them. Finding effective weights for such problems is very challenging. Even changing the problem just a little, perhaps with different training data, can mean that the weights need to be adjusted even further. Moreover, some problems have many dimensions, and finding effective weights can be haphazard and arbitrary, because changing one weight may mean a chain reaction requiring adjustment of all the other weights. When the search space is so sensitive to changes in weights, it indicates that there can be a high degree of *bias* introduced by the user when deciding weights. The solutions you get will be highly dependent on the weights given by the user.

An active area of research is multi-objective optimization. The idea is quite simple. The weighted sum idea is discarded entirely, and fitness measurements happen in 2 stages. In stage 1, each objective is measured with its natural fitness measurement, as is seen in the weighted sum formulae above. However, these scores are not merged at all, but are kept separate for each population member within a *vector* of scores. A GA would therefore evaluate each individual according to all the multi-objective evaluation tests as are necessary for the problem.

Stage 2 involves finding overall rankings for the population. Recall that ranked fitness measurements discard absolute fitness scores, and instead replace them with integer numbers (1, 2, 3,..., with 1 being the most fit, 2 being 2nd fittest, etc.). The ranking done here uses the Pareto ranking strategy. The idea

behind Pareto ranking is that it will never try to compare apples with oranges: each dimension of the problem is always kept independent of the other dimensions, and an individual is better than, or *dominates*, another individual if it is shown to be at least as good in all dimensions, and better in at least one dimension. For a minimization problem (one in which we are trying to minimize scores), then for 2 individuals $U(u(1), u(2), ..., u(k))$ and $V(v(1), v(2), ..., v(k))$, with k objectives, we say that:

U dominates V iff:

for all objectives i: $u(i) \leq v(i)$
and there exists at least one objective i where: $u(i) < v(i)$

The first expression with "for all" says that there is U is at least as good as V is in all objectives. And the 2nd expression ("there exists") say that there is at least 1 objective in which U is definitely better than V. Hence it is clear that U is superior to V, because it is better in at least 1 objective, and not worse in any objective.

The Pareto ranking algorithm relies on the idea of domination. It first goes through the entire population to find the non-dominated individuals. These are the individuals in which nothing dominates them. These will be assigned rank 1, the fittest individuals in the population. The ranking algorithms takes an individual A, and then looks through the rest of the population to see if any individual B dominates A. If so, then A cannot be in rank 1, and it is skipped. If however, it is found that there is no B that dominates A, then A is assigned rank 1. Once the entire population is evaluated for the rank 1 members, these rank 1 individuals are marked as "processed", and the whole procedure is repeated on the remaining population to find the rank 2 individuals... those that are undominated by any yet unranked individuals. This repeats until the entire population is assigned a rank.

The end result of the Pareto ranking is that each member of the population has a single Pareto rank value assigned to it. The lower the rank, the better the individual. These ranks can then be converted to a Roulette wheel or used within a tournament selection to create the next generation.

There will usually be sets of individuals in each rank as well. The individuals in a rank dominate all the individuals with higher rank numbers, and are in turn dominated by the sets with lower ranks. However, individuals in the same rank set are incomparable, in the sense that none of them is clearly better or worse than any other member of that set. Each individual will be better in some dimensions of the problem, but worse in others.

An example: Pareto ranking a population (Goldberg)

Consider a population for a minimization problem (low scores preferable) with the following fitness vectors:

1: (2, 4)
2: (2, 10)
3: (3, 4)
4: (4, 3)
5: (5, 10)

We must first determine the rank 1 set. Remember: an individual is in rank 1 if no other individual dominates it. Be careful to **not** use the wrong idea, that an individual is in rank 1 if it dominates everything else! We then find the following for the above 5 individuals:

1: (2, 4) – nothing dominates it, so it is rank 1.
2: (2, 10) – it is dominated by individual 1
3: (3, 4) – it is dominated by individual 1
4: (4, 3) – nothing dominates it, so it is rank 1
5: (5, 10) – it is dominated by individual 1

So the rank 1 set consists of individuals 1 and 4. Notice how (2, 4) and (4, 3) contribute unique aspects to the problem. Individual 1 has a low 1_{st} 1, while individual 4 has a lower 2_{nd} dimension. These individuals are removed (marked "processed"), and the procedure is repeated on the remaining population:

2: (2, 10) – dominated by nothing, so it is rank 2
3: (3, 4) – ditto
5: (5, 10) – dominated by individual 2

So rank 2 has individuals 2 and 3. The final pass will mark individual 5 as rank 3.

An alternate variation: dominance ranking (Fonseca and Fleming)

An alternative to Pareto ranks is to compute dominance ranks. Here, the rank of every individual is the number of individuals in the population that dominate it, plus one. Hence, the rank 1 individuals will be the same as those having rank 1 in the Pareto ranking above. But the rest of the ranks will usually be more highly distributed, to a higher maximum rank value. This might tend to reduce the amount of convergence that might arise when fewer ranks are being used with the first Pareto ranking scheme.

In the above population, the dominance ranks are: 1 (indiv 1, 4); 2 (indiv 2, 3); 5 (indiv 5).

Comments

Pareto ranking is advantageous over weighted sums for the following reasons:

- User bias is less likely, unlike the weighted sum.
- It is easier to use than weighted sums, since the problem is kept as natural as possible, with little user intervention.
- At the end of a run, the user is given a set of rank 1 individuals as potential solutions. The user is then free to decide which of the set, if not all of it, is a preferred solution.
- Pareto ranking has been shown to be remarkably effective for solving complex multi-objective problems.
- It is fairly efficient to compute.

However, there are also deficiencies with Pareto ranking:

- Convergence due to duplication of individuals in each rank is common. This happens largely because of the loss of fitness information that occurs with ranks. With a weighted sum, small fractions in the sum may help distinguish individuals, which act as a natural force for diversity. In other words, there could be 1000's of different fitness values in the population. But with a Pareto ranked population, there may be perhaps only 10 to 20 discrete ranks in the population. This makes it easier for one individual to be selected repeatedly over others in weaker ranks, resulting in its excessive duplication in the population. To get around this, Pareto search often uses diversity strategies to help promote genetic diversity within the ranks, and population as a whole. The dominance ranking approach might tend to help diversity.
- Pareto ranking begins to lose its effectiveness with too many dimensions are being considered. Practical limit is perhaps 5 or 6. Because the population is now evaluated along so many dimensions, it becomes more difficult to find clear cases of domination between population members. In other words, the population becomes diluted in the Pareto search space, and everything will be rank 1, with few in lower ranks. GA then does a random search.
- It is more difficult to analyze population fitness during a run. This is because there is no absolute fitness score that everything uses. Rather, the population in each generation always has a rank 1 set, rank 2, etc.

One strategy often used with Pareto and GA is to do multiple runs as usual, and then take the rank 1 sets found in all the runs, merge them together, and re-rank them. The new rank 1's obtained would be designated as being the solutions for all the runs.

One final comment: if you are given 2 individuals and their fitness vectors, and then are asked to determine which is in rank 1, this cannot be done. Although you might be able to say whether one individual dominates the other, you cannot say for certain that an individual is of rank 1 (or 2, etc) without looking at the entire population. All it takes is one individual elsewhere to dominate it, and that individual will no longer be in rank 1.

Further information

1. *Evolutionary Algorithms for Solving Multi-Objective Problems* (2e)
C.A. Coello Coello, D.A. van Veldhuizen, G.B. Lamont, Kluwer, 2007.

Great resource specializing in multi-objective evolutionary algorithms.

2. *Genetic algorithms in search, optimization, and machine learning* by David E. Goldberg.
Imprint Reading, Mass. : Addison-Wesley Pub. Co., 1988.
QA 402.5 G635 1988 (5th floor)

Read the short discussion of Pareto ranking from Goldberg's book. He uses a simple example of a problem that analyzes different factories based on 2 dimensions: number of accidents, and cost of production for an item. The cheaper an item is to make, the more accidents occur. Likewise, accidents can be greatly reduced, but at the cost of increasing the production cost per item. It is not easy to reconcile these 2 different measurements with a weighted sum. But the Pareto approach gives an interesting solution to it.

3. "Procedural Texture Evolution Using Multiobjective Optimization" by B.J. Ross and H. Zhu. *New Generation Computing*, vol. 22, n. 3, 2004, pp. 271-293.
http://www.cosc.brocku.ca/~bross/research/gentropy2.pdf

This paper uses multi-objective optimization to evolve procedural textures. Pay particular attention to the pseudo-code in Figure 2, as well as the discussion in Section 3.1: