Enable Diversity within MOSES

Nil Geisweiller

November 12, 2012

(stripped of non-disclosable content, resulting in blank spaces or pages)

Abstract

Report on the current state of the diversity implementation, experiments with toy problems . What can be concluded at the moment and what should be done.

Chapter 1

Methodology

I explain here how exactly diversity has been implemented in MOSES.

1.1 General Idea

At that point diversity only affects the metapopulation. It works by altering the ranking of the candidates so that non-diverse candidates are pushed down and have a lower probability of getting chosen as exemplar to spawn new demes.

Technically speaking it does so by computing for each candidate of the metapopulation a diversity penalty, subtracted from their scores.

Naturally, it has the side effect of increasing the overall diversity of the fraction of the metapopulation that is output at the end of learning (the best N candidates, accounting for diversity). It also seems that it may speed-up the search but it's not very clear.

1.2 What isn't being done

While exploring the deme, diversity doesn't count at all. This might not be good because there is no guaranty that the search isn't gonna drift towards non diverse candidates.

To my experience candidates found in a demes tends to be semantically close to the exemplar, in other words, not accounting for diversity while searching the deme might be OK, but there is no precise measurements about it at the moment.

The main reason for not doing that is the computational cost (and of course the implementation cost as well, though to a lower extend), the question is "is it worth it?". I don't have the answer.

1.3 Measuring diversity

In that section I give the exact methodology used to calculate the diversity and ranking the candidates according to that.

1.3.1 Ranking while accounting for diversity

Let M be the current metapopulation and P an empty metapopulation. We're gonna iteratively move every candidate from M to P, the new metapopulation, with their scores updated to account for diversity.

1. Take the best candidate of M (not accounting for diversity), move it to P.
2. For all candidates m_i in M compute the distance between m_i and each candidate p_j in P, called $d(m_i, p_j)$.
3. For each distance $d(m_i, p_j)$ compute its diversity penalty, $dp(m_i, p_j)$.
4. Calculate the overall penalty between m_i and P, that is aggregate all $dp(m_i, p_j)$ with $j = 1, \ldots, |P|$, called adp_i.
5. For each m_i in M subtract adp_i from its score.
6. Choose the best candidate of M (accounting for diversity, this time), move it to P.
7. If M is empty swap it with P and stop, otherwise go to step 2.

Might sound costly, but in practice it is done incrementally, only new candidates go through the costly parts of the algorithm and it usually takes less than a second for each generation.

I explain in the next subsections how exactly the distance, the diversity penalty and the aggregation are calculated.

1.3.2 Distance

To calculate $d(x, y)$ we use a p-norm over the absolute difference between the behavioral score of x and y. Formally

$$d(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}$$

When p tends to infinity it tends to the max norm

$$d(x, y) = \max_{i=1}^{n} |x_i - y_i|$$

As implemented in MOSES, if p is negative then the p-norm becomes the max norm.

Behavioral score for toy problems

The behavioral score is a vector of floats where each element corresponds to an input vector, so it indicates the behavior of the candidate for each input with respect to the target output.

In the case of the toy problem an element of the behavior score is 0 if the candidate output matches the target output, -1 if it doesn't.

Behavioral score for our real work problem

In the case of our financial series there are 2 ways to define the behavioral score

1. Count only positive contribution to precision, in that case each element has the value
$$\frac{tp_I}{tp + fp}$$
where tp_I is the number of true positives for input vector I, tp and fp are the total number of true and false positives. Of course
$$\sum_I \frac{tp_I}{tp + fp} = \frac{tp}{tp + fp} = precision$$

2. Count both positive and negative contribution, in that case each element has the value
$$\frac{tp_I - tp_I}{2 \times (tp + fp)}$$
where tp_I and fp_I are the number of true and false positives for input vector I, tp and fp are the total number of true and false positives. It turns out that
$$\sum_I \frac{tp_I - fp_I}{2 \times (tp + fp)} = \frac{tp - fp}{2 \times (tp + fp)} = precision - \frac{1}{2}$$
let's call it *centered precision*.

1.3.3 Diversity penalty

To turn distance into diversity penalty, we take the inverse of it, offset with 1 so that it doesn't go to infinity when the distance is null.

Here we have a additional parameter called the *diversity pressure*, it tells how much should we amplify the diversity penalty, formally

$$dp(x, y) = \frac{diversity\_pressure}{1 + d(x, y)}$$

You may note that the diversity penalty equals to diversity pressure when the distance is null.

1.3.4 Aggregation

To aggregate diversity penalties we have 3 ways

1. the max
$$adp = \max_{j=1}^{|P|} dp(x, y_j)$$
2. the generalized mean
$$adp = 1/|P| \left(\sum_{j=1}^{|P|} dp(x, y_j)^e \right)^{1/e}$$
3. the generalized sum
$$adp = \left(\sum_{j=1}^{|P|} dp(x, y_j)^e \right)^{1/e}$$

As implemented we actually have only 2 parameters, the *diversity exponent*, e in the formulas, and the *normalization.* The max is a special case of the generalized mean (when e tends to infinity, actually implemented when e is negative), and the mean is obtained when the normalization is set to true (that is it is *normalized* by the size of P).

When a generalized sum is used instead of the mean or the max, the diversity penalty can become pretty aggressive, as the upper bound of adp isn't anymore $diversity\_pressure$ but $|P| * diversity\_pressure$.

1.3.5 Summarize parameters

So to sum up we have 4 (actually 5) parameters to tune

- p, controlling the exponent of the p-norm of the distance, with a special case when p is negative, the max norm.
- *diversity_pressure*, controlling the scale of the diversity penalty
- *diversity_exponent*, controlling the exponent of the mean or sum of the aggregation of diversity penalties, with a special case when it is negative, the max.
- *diversity_normalization*, to switch between generalized mean and sum when $diversity\_exponent > 0$.
- There is an additional parameter that was already implemented, *discard_dominated*, which allows to discard dominated candidates, formally x dominates y if
$$\forall i \ in \ 1, \ldots, n, x_i \geq y_i$$

$$\wedge$$

$$\exists i \textit{ such that } x_i > y_i$$

If *discard_dominated* is set to true then dominated candidates are simply removed from the metapopulation. This may contribute to its diversity as well[1].

---

[1] As I'm writing this, I'm wondering if I could have an option that relaxes the definition of dominance, from strict dominance to loose dominance, where the second condition is removed, another way to enforce diversity aggressively.

Chapter 2

Experiments

Now we present a set of experiments with 3 toy problems

2.1 Toy problems

2.1.1 Settings

The toy problems are:

1. 5-parity
2. 11-multiplexer
3. 7-majority

We're not trying to solve these problems here, MOSES can already do it with great ease. The goal is to run enough evaluations so that MOSES does find good but not best solutions (just like in our real world problem), and we analysis the impact of the diversity pressure on the scores and the diversity of the output.

The number of evaluations is fixed to 10K. All diversity parameters are being varied, except p, fixed to 2, corresponding to the Euclidean distance. For the other parameters

1. $diversity\_pressure$ ranges
    (a) from 0 to 64 for 5-parity,
    (b) from 0 to 4096 for 11-multiplexer
    (c) and from 0 to 256 for 7-majority

    You may have noticed that the range corresponds to twice the number of classification errors of its respective problem (which is how MOSES defines their fitness functions). That way the diversity penalties will be within the same scales of the problem's fitness scores.

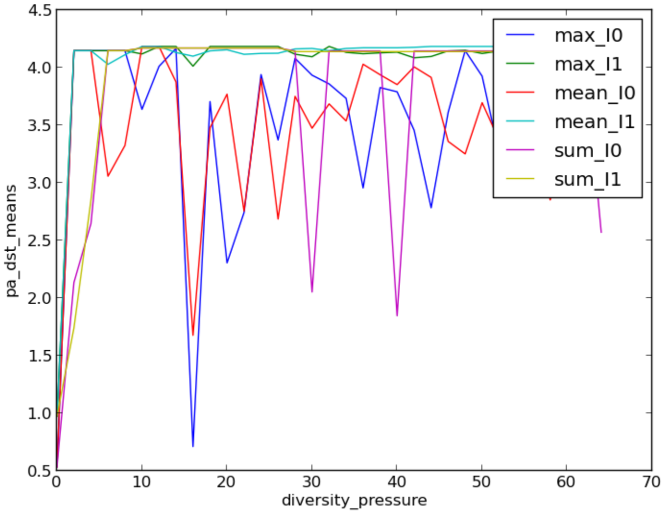

Figure 2.1: Average distance between each pair of the 10 best candidates for 5-parity

2. *diversity_exponent* and *diversity_normalize* take for all problems
   (a) (-1, undefined), corresponding to the max
   (b) (1, true), corresponding to the mean
   (c) (1, false), corresponding to the sum
3. *discard_dominated* takes for all problems true or false.

2.1.2 Results

For each parameter setting we're gonna look at

1. the average distance between the 10 best candidates (accounting for diversity) output by MOSES.
2. the score of the top candidate output by MOSES.

5-parity

Diversity Figure 2.1 shows the average distance between each pair of candidates of the 10 best candidates output by MOSES.

Let's first explain to which setting corresponds each line

- max_l0 corresponds to aggregating with the max function and discarding dominated candidates (that is *discard_candidate* is set to true).

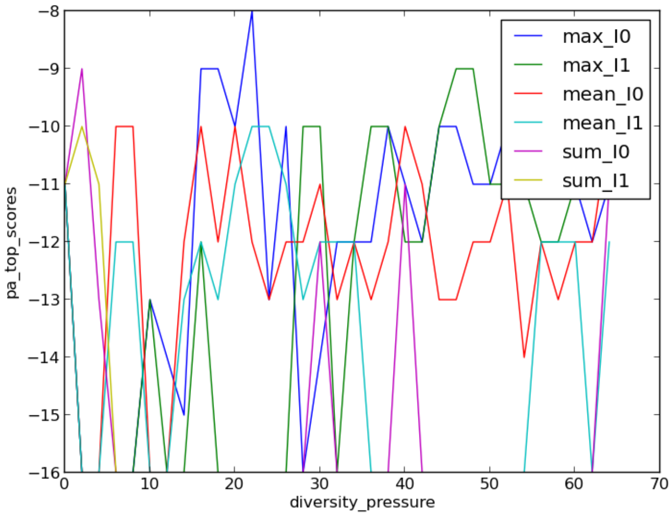

Figure 2.2: Best score output by MOSES for 5-parity

- max_I1, like max_I0 but *discard_candidate* is set to false.
- mean_I0 and mean_I1 corresponds to aggregating with the mean, resp. discarding and not discarding dominated respectively.
- sum_I0 and sum_I1 corresponds to aggregating with the sum, resp. discarding and not discarding dominated respectively.

In that set of experiments, the best settings are max_I1, mean_I1 and sum_I1 which seem to converge much more regularly to the supposedly max diversity. That is all settings that do not discard dominated candidates. I'm not sure why but I think it is because when all non dominated candidates approaches a good score then they tends to differ by fewer elements in the behavioral score and the ones differing by more that are usually dominated and therefore eliminated.

Also the max diversity (in the case of max_I1, mean_I1 and sum_I1) is reached rather rapidly, by setting the diversity pressure at less than 1/6 of the maximum score.

Scores Figure 2.2 shows for each setting the score of the best candidate output by MOSES.

There doesn't seem to be some clear pattern here, so I'd conclude that diversity doesn't help the search itself, not here at least.

11-mux

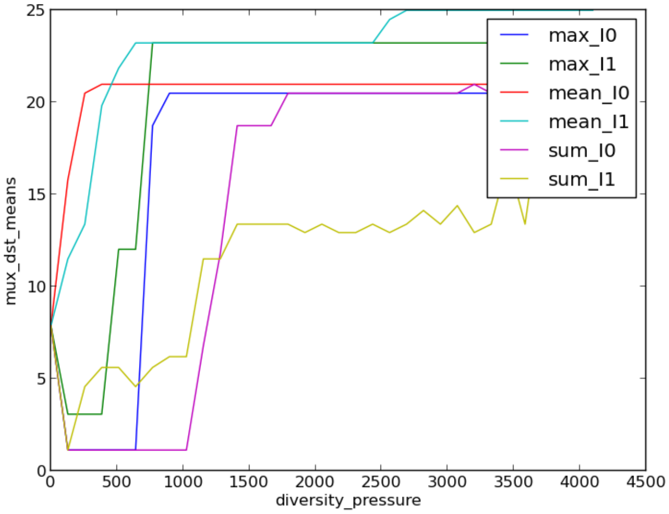

Figure 2.3: Average distance between each pair of the 10 best candidates for 11-multiplexer

Diversity Here the best setting seems to be mean_I1, although mean_I0 converges faster to a lower mean, likely removing the dominated candidates permits to get more diversity rapidly but then prevents from having too distant candidates. Again maximum diversity is seemed to be reached when *diversity_pressure* is set to less than a 1/6 of the maximum score amplitude. Although surprisingly with *diversity_pressure* set above the max score amplitude it increases again with mean_I1. I do not know why.

Scores As shown in Figure 2.4, and as opposed to the 5-parity experiment, here introducing diversity increases substantially the score. It could be the breadth of the distribution to select the next candidate wasn't high enough and introducing diversity has rectified that.

7-maj

Diversity As figure 2.5 shows we get a similar pattern as with the other problems, mean_I1 is the winner, although again mean_I0 converges faster to a lower value.

Scores Here, no difference whatsoever, I'd need to dig into to the log to understand why.

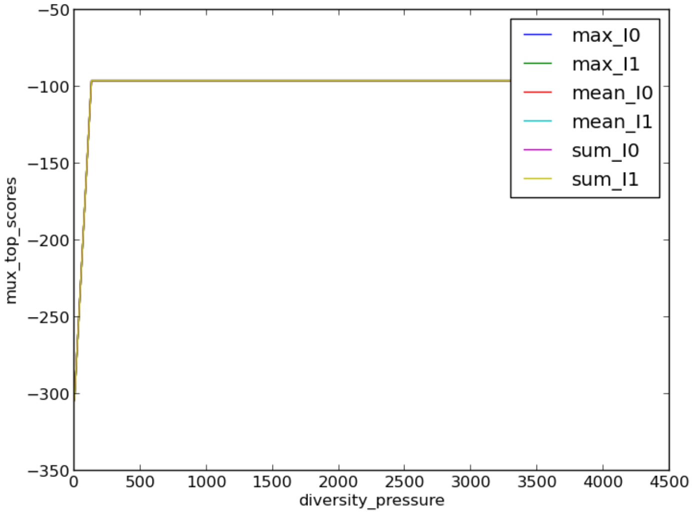

Figure 2.4: Best score output by MOSES for 11-multiplexer

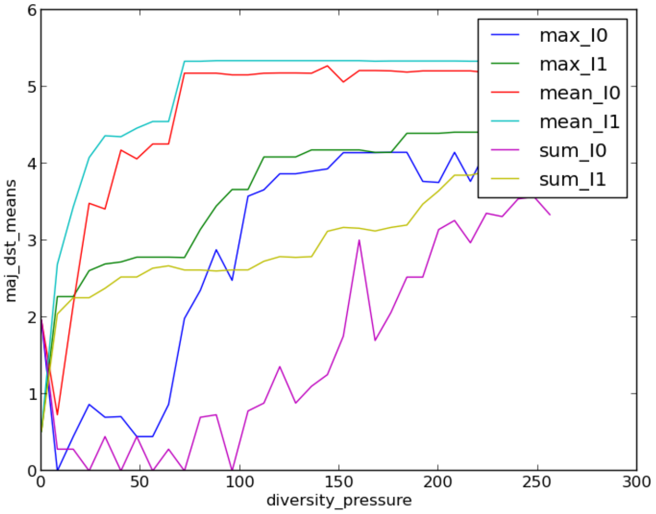

Figure 2.5: Average distance between each pair of the 10 best candidates for 7-majority

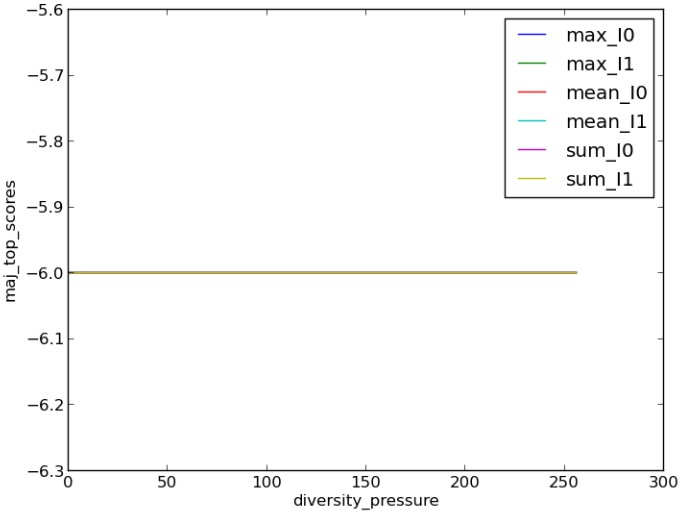

Figure 2.6: Best score output by MOSES for 7-majority

Chapter 3

Conclusion

3.0.3 Toy Experiments

The toy experiments display rather clear patterns which leads us to conclude that using mean as aggregating function is probably the best.

Also, not discarding dominated candidates may bring more diversity at first but seems not to help or even hamper diversity when the diversity pressure is set well.

3.1 What remains to be done

- Try with the fitness function centered precision.
- Try the weighting approach to avoid clusters.
- Possibly try with hundreds of features instead of 30. There's been earlier trial that have failed so no real comparison was possible.
- Possible try to compare random seeds and diversity pressure for toy problems.
- Possibly try to vary p, as well as other values of $diversity\_exponent$ rather than just -1 and 1.
- Experiment with other distances, such as Tanimoto or angular distances.
- Possibly experiment using uncompressed bscores to see if compression is detrimental.
- Possibly implement loose dominance and experiment with it.
- There is one parameter that although doesn't affect diversity directly is very important to play with and we haven't done so. That is the breadth of the distribution over the metapopulation to pick up the next exemplar. If the distribution is too narrow then only best candidates will be selected to create new demes leading to early convergences and local optimum. If the distribution is too broad, then possibly too bad candidate (possibly not diverse enough) will be selected to create worthless demes.