# Nature Inspired Optimization – Assignment 2

## Is the problem Non-Trivial?

We can tell from the two objective functions that this problem is a non-trivial optimization problem because, both objective functions are looking the minimal solution (smallest frequency & cost). If we look at the functions they both have a common element, L. the *f1(x)* objective function (frequency) is reduced by L increasing in size (because L affects the denominator which will decrease the fractions overall size when L increases). In contrast, the *f2(x)* objective function (cost) reduces in size when L gets smaller (as there are less materials needed to be purchased).

This means that there is not a single solution which simultaneously optimizes each objective as the objective functions are ‘conflicting’. Instead we need to produce a range of solutions with trade-offs between the two objectives to choose from.

## Implementation

I decided to use JAVA as my main programming language for two reasons, firstly I have greater experience programming in JAVA than any other language. Secondly because it allowed me to easily integrate with the MOEA framework to calculate the solutions. I choose to use the MOEA framework because it has many implementations of multiobjective evolutionary algorithms which can be applied to your problem and I had access to example solutions provided by Dr Shan He. It is also open sourced and provides fully documented code upon download.

I based my implementation the example solutions provided, using the same code structure and changing where necessary to fit with the assessments problem. In the ‘solver’ class I created a RunTest() method which accepts an algorithm name as input to use in the computation, this allows me to easily run the same test with different algorithms. I also used the ‘Analyzer’ class to analyse the 30 runs results and calculate the hypervolume values for each run. However, I could not find any way of exporting the results from the analyser to a variable so in order to gain access to these results the program outputs them to a file and then read & interoperates the data (extracting the hypervolume values). For this I used a yaml library called ‘snakeyaml’ in which I extract all of the hypervalues for the 30 runs and compare them to find the best one which allows me to identify the best run out of the 30 to plot to a graph. The program also outputs the raw values for each run of each of the variables (d1, d2, d3, b, L) and the objective function results. The best solution from each algorithm (chosen from the best hypervolume) are plotted on the same graph based on a modified version from the example solution.

The values of the variables are set in the ‘newSolution()’ method in which I set the solutions variables of d1, d2, d3, b & L. Here I create a new ‘RealVariable’ for each of the variables which allows me to set upper & lower bounds for the values inside of them. Most of the variables are specified in the assignment documentation however d2 needs to be calculated. I set the upper limit of d2 to be 0.59, this is because the maximum width of material 3 (the top layer) is 0.57 & the minimum width of d3 is 0.6. As d2 + material3 = d3, if material3 has a max width of 0.57, d2 has to be at least 0.03 in order to keep within d3's bounds of 0.3. The minimum width of d2 was set to 0.02, because d2 is made up of the materials 1 & 2. Both material 1 & 2 have a minimum width of 0.01 so the smallest that d2 can be is 0.02 (because material1 + material2 = d2).

The computation of the algorithm is done in the ‘evaluate()’ method, this is called by the MOEA framework when computation starts. Here I first calculate the values of EI & uL (which are used in the objective functions) and then calculate the objective functions themselves which are added to the solution.

## Constraints

The constraints the program enforces are the values of U, material 2 & material 3. I have these three values as constraints because out of the 7 conditionals specified in the task documentation, 5 of them are already constrained within the bounds of the RealVariables. This means that I only have to check that the remaining 3 are within bounds for the iteration to be valid.

I check the constraints of U, material 2 & material 3 widths by calculating the degrees of difference from the upper or lower bounds each of these values are and then adding them as constraints to the solution. If they do not exceed the bounds of the variables then the constraint values are set to 0.

After this the MOEA framework does the rest and recalculates the variables if they are out of bounds.

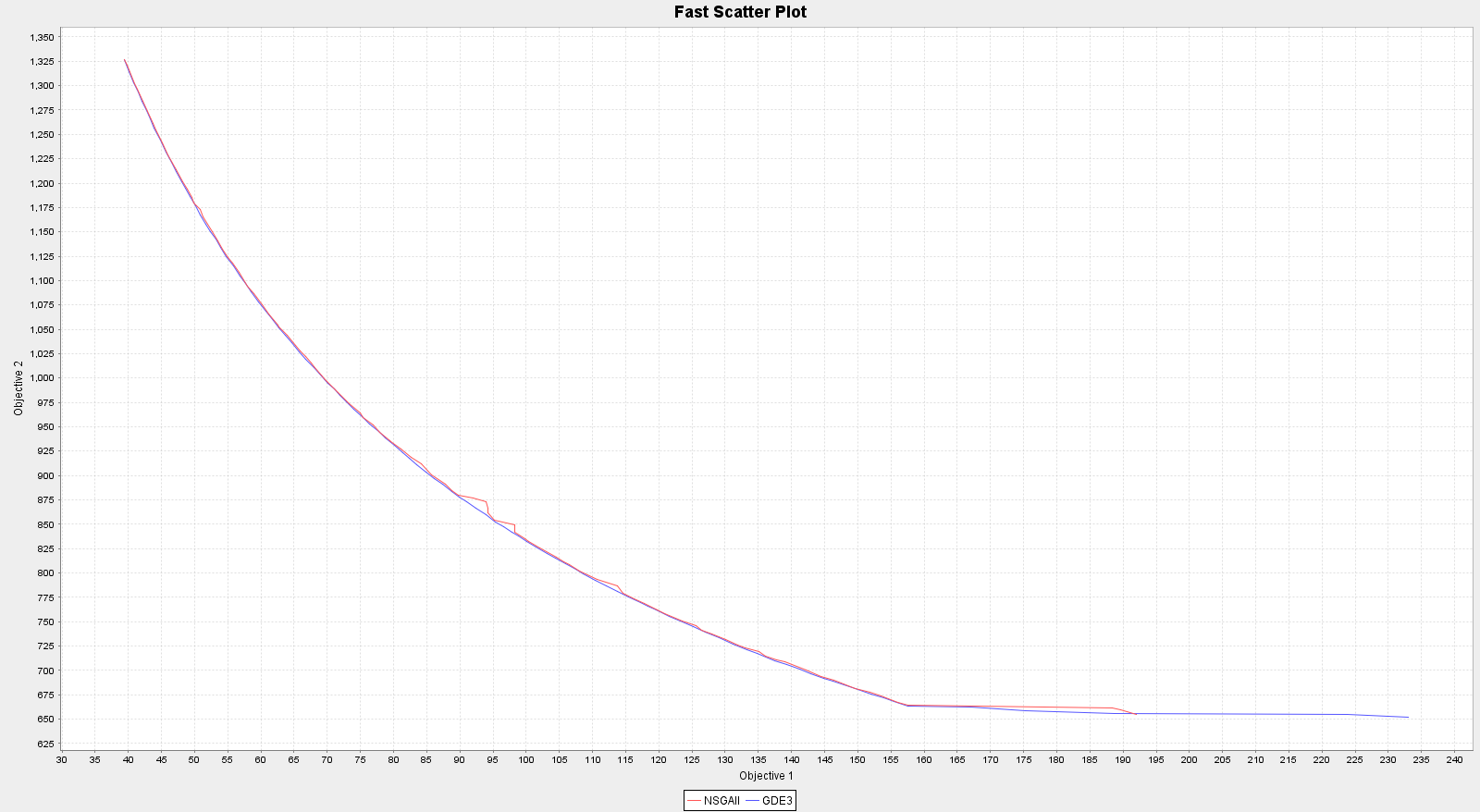
## Hypervolume indicator

My implementation uses the “hypervolume” metric as a method of choosing the best run and as a comparison between the two algorithms. The hypervolume metric describes “how much of the objective space is dominated by a set of solutions” so in our problem of minimization of both objectives, a larger hypervolume value indicates a solution closer to the minimum.

Another performance metric is “spread”, this calculates how uniformly spread across the objective space the solutions are within their set. This might not be the best metric to use for our problem as it would not allow us to determine if a run is closer to the minimal values than another run is.

## Results

I decided to use the NSGAII and GDE3 algorithms for this problem.



In this run they produce a very similar Pareto front and the final hypervolumes were:

NSGAII - 0.759423080349427  
GDE3 - 0.7708372363172727