Data Design Nature-Inspired Computing: Project Report

Francesco Pelosin

September 2, 2017

1 Introduction

In cellular biology, a vesicle is a small structure within a cell generated by amphiphilic molecules. Vesicles are a basic tool of the cell for organizing metabolism, transport, enzyme storage, as well as being chemical reaction chambers. They are relevant in many technological applications such as drug delivery in medicine. Experimentation in this field is difficult because of the high dimensionality of the search space and the high cost of each experiment. A particular set of molecule, under particular experimental condition and variables, may self-assemble and originate vesicles. In this project we are interest in the vesicle formation. In order to investigate this process, a set of real experiments have been conducted in laboratory. A stochastic optimisation technique is then applied to these data in order to extract meaningful features inside the high dimensionality of the search space and select the proper levels. The design of these experiments had the goal to select the best combination of factors to identify the best molecular compositions that may generate vesicles through an evolutionary algorithm.

To address this problem we implemented a nature inspired approach, a method that takes inspiration from the nature. This new set of algorithms are based on the combination of heuristic approaches in high level frameworks aimed at efficiently and effectively exploring the search space. In particular we will use the "Ant Colony Optimisation" algorithm (ACO). The ant colony optimisation is a probabilistic technique for solving optimisation problems inspired by the behaviour of real ants, in particular when they harvest food.

2 Problem statement

As mentioned in the introduction, to understand the problem of vesicle formation, a set of experiments have been conducted.

For each experiment we have to consider:

- Response: Turbidity (Y)
- Composition variables and process variables (X): Reactants, PH, ion strength, temperature...
- The experimental space (the search space): all possible combinations of all the factors, their levels and their interactions.

Turbidity level is related to the vesicle volume and has been regarded in this study as the system response to be optimised. The response, named Turbidity (T) is a measure of both the number of evolving microstructures and the size of these microstructures ($T = KNV2$).

Composition variables and process variables are the results of a set of experiments conducted with different concentrations and particular laboratory

protocols and they are known as "mixture experiments". Each experiment was described by the mixture $(x_1, ..., x_{16})$ where x_i represents the number of volume units of the X. Each x_i can assume a value between 0 and 1 in particular with step of 0.2. Let's call a mixture $(x_1, ..., x_{16})$ a wheel.

Composition Constraint Each wheel has to satisfy a property for which the sum of the levels of the factors must equal to one, more formally:

$$\sum_{i=1}^{16} x_i = 1$$

Where x_i is the level of factor i. As an example, an experiment wheel looks like:

0	0	0	0	0	0	0	0	0	0	0	0	0	0.2	0.4	0.4	**0.12937**

where the first 16 variables are the factors of the compositions and the 17th value is the response for the specific combination of the levels.

Each factor can take 6 different levels between 0 (absence of factor) 0.2, 0.4, 0.6, 0.8 and 1 (single factor). The goal is a combinatorial problem and consists to find the best composition between the factor levels.

The total number of points in the search space corresponds to 15.504 possible mixtures which is a subset of all the combinations 6^{16}. We now will discuss our solution to reach the composition that gives the best vescicle formation.

3 Solution proposed

In order to find the best mixture we decided to use a stochastic optimization technique inspired by nature, the "Ant Colony Optimisation" (ACO) algorithm.

ACO is a probabilistic technique for solving optimisation problems inspired by the behaviour of real ants, when they harvest food. In particular when ants look for food they start by exploring in a random way the environment. When they retrieved some resources ants return to the colony laying down pheromones on the trail to the food. Pheromones adds probability to redirects other ants to take the same path toward the objective. This substance is also subject to evaporation, that is after a while old traces of pheromone starts evaporating in the air. This phenomenon naturally rewards short paths to the food, in fact this procedure eventually makes all the ants choose the shortest path to the resource.

So the steps performed by the algorithm are:

1. Ants explore the environment randomly

2. When an ant discovers a source of food, it returns back to the nest laying down pheromone on the trail

3. Other colony ants start following short paths with more probability thanks to pheromone trails

4. Shortest path becomes more and more attractive while longest disappears thanks to evaporation

5. At the end, the whole colony choose the shortest path

We now show how this biological inspiration can be transferred into an algorithm for discrete optimization.

Implementation The real world problem refers to find shortest paths in a environment, so the translation in our domain can be viewed as a problem of finding shortest (rewarding) paths into a graph.

To implement the algorithm, we initially created a matrix P consisting of 16 columns and 6 rows which represent our graph (environment). Each column is one of the factor entering in the composition, and each row is the level that the variable can assume (0, 0.2, 0.4, 0.6, 0.8, 1) expressed in indices. Most important, each cell correspond to the amount of pheromone that a particular vertex of the graph has accumulated by the ants. For example: $p_{ij} = 1$ corresponds to the fact that j^{th} factor at i^{th} level has 1 unit of pheromone.

At the beginning all the cells of the matrix P are equal to 1. This initialization process ensures that all the paths of the graph are equally probable. In order to consistently explore the search space, k ants randomly choose a valid path (which sum to 1) and release some pheromones. This roughly correspond to select a completely random combination of factors and use it as explorative solutions for the next generations of ants. For example a solution can be :

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0 (0)	2	2	2	2	2	2	2	2	2	2	2	2	2			
1 (0.2)														2		
2 (0.4)															2	2
3 (0.6)																
4 (0.8)																
5 (1)																

Response: 0.12937

Table 1: In this case 2 units of pheromone has been added to the selected path

In order to evaluate the response we search inside the dataset that we have and that correspond to the whole search space. The main objective should be to find the optimal mixture with the minimum amount of experiments (since

each time we look inside the dataset it's like we were test the composition in laboratory).

After the first step of random exploration, we let the k ants to explore the graph by looking the pheromone matrix for a certain number of generations. In particular each ant will:

1. Evaluate the pheromone trails in the matrix
2. Compute the probability of a given movement from the pheromone quantities
3. Evaluate the path proposed
4. Release pheromone for the next generation by updating the matrix according to the goodness of the path found

The ants will eventually converge to a common path based on the probability of choosing good old trails.

For the second step, the probability to move from one vertex to another is computed by summing all the column pheromone quantity of the feasible levels for the next value and dividing each entry by this sum, this allows the ant to assign a probability by looking to the past trails. Table 2 may clarify the concept.

	j	$j+1$	
0 (0)		1 / 5	
1 (0.2)	... **Ant**	1 / 5	...
2 (0.4)		2 / 5	
3 (0.6)		1 / 5	
4 (0.8)		0	
5 (1)		0	

Table 2: In this case the ant is moving from vertex $p_{1,j}$ to $p_{1,j+1}$ the feasible levels are 0, 1, 2 and 3. This means that the sum of the current ant's path is 0.4.

In order to accomplish the fourth step we compared the k ant's solution found at each generation, the best one will get a bigger amount of pheromone to be released. This helps the convergence to good solutions. In addition we kept track of the best solution for each generation and at each iteration we also rewarded that solution.

The last point was to model the pheromone volatility, the approach was to reduce the total amount of pheromone quantity for each cell at each generation iteration. This permits to get rid of bad solutions and achieve the convergence of the algorithm.

4 Evaluation and Results

Tuning After tuning some of the parameters of the algorithm, we were able to achieve very good results. In particular the main one are:

- The number of generations
  - Turns out that the best number of generation is 10, with an higher value the algorithm seems to do nothing interesting. It means that the algorithm require a small number of iterations to converge.
- The number of ants
  - Turns out that the best number of ants is 100, with an smaller value the searching process is very poor and usually the algorithm get stuck to local maxima. It is in fact intuitive to see, since more ants imply more exploration of the search space therefore more probability to discover the global optimum.
- The evaporation factor
  - The evaporation factor plays a central role to the convergence of the algorithm. Our tuned value is 0.5 so at the end of every generation the old paths are decreased by half of their weight. Thanks to evaporation we are able to reward good paths and discard bad paths
- The pheromone quantity for the best solution and for other solutions
  - These two parameter was crucial to tune, the best way to achieve the global maximum was to reward the best solution ten times more than other trails, so we added 20 pheromone units at each generation both to the best path seen so far and the current best path of the generation. We added 2 pheromone units to other solution found good or bad.

Results analysis Finally we achieved some pretty good results, we are able to find the global optimum with not too much effort. The following plots represent some statistics where we compare the best case (when we reach the global optimum) and the worst case (which is still a very good result since it is part of the top 25% of data) and some statistics of the average performance of the algorithm in 40 runs which can be summarized in:

- Average best solution proposed gives a response of 1.05
  - As we can see in Figure **??** there are two outliers which represent the top two results which correspond to a response of 1.31 and 1.29 the third best solution is around 0.90 so our algorithm on average performs very well.
- The standard deviation of the solution proposed is 0.25

  - Since the top two solutions are very distant from the rest and given that our average response is very close to them, this standard deviation reveals that we are almost always getting the top two solutions and sometimes the algorithm gets stuck on local optima near the top solutions.

- Average number of unique experiment to perform before convergence is 457
  - This a very important indicator since in real life the number of experiments to perform could be a critical factor due to the cost of experimentation and time to invest. Our algorithm converges with a mean number of 457 unique experiments which means that our algorithm converges almost always to the global optimum by exploring the 3% of the search space.

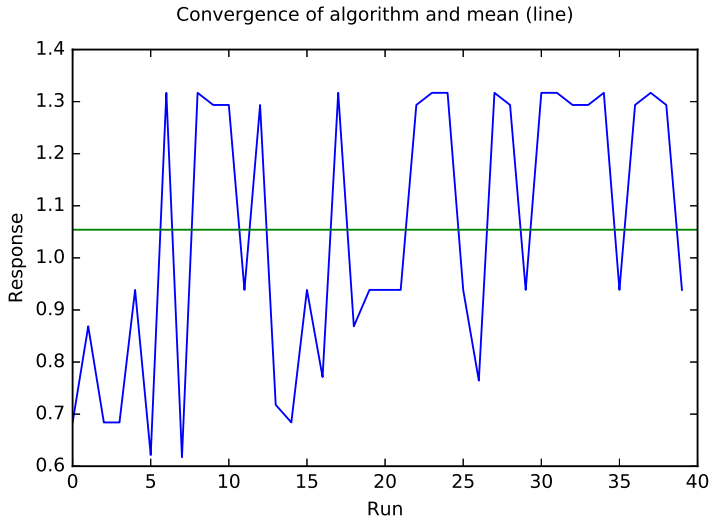

Figure 1: This plot shows the best results for each one of the 40 run. The green line is the average. As we can see the algorithms sometimes get stuck on local maxima, but still with very good results.

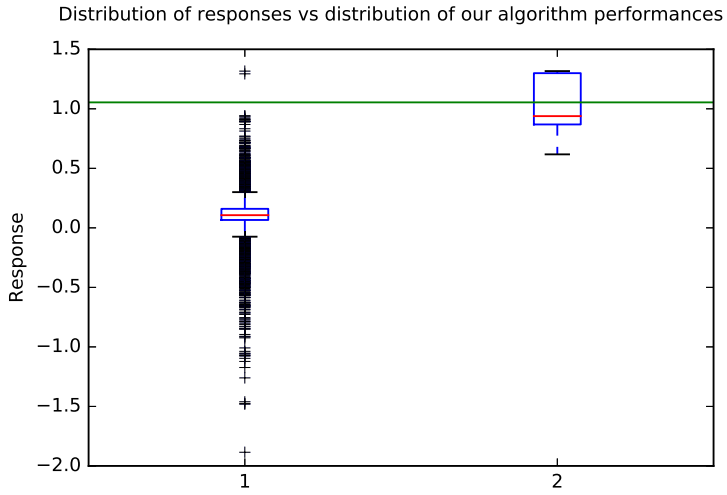

Figure 2: This plot compares the distribution of the overall solutions and the distribution of the solutions examined by our algorithm. As we can the algorithm examine very good solutions. The green line represent the mean of 40 run of the algorithm

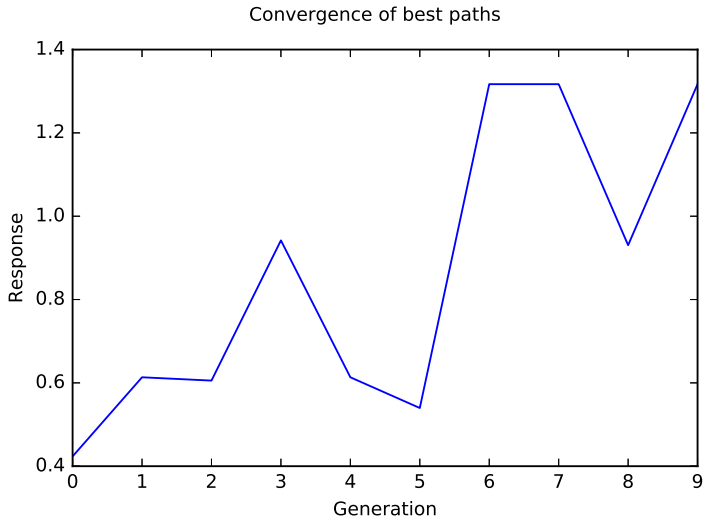

Figure 3: Best solution case: This plot shows the best paths responses for each generation, as we can see the ants increase the goodness of the solutions at each iteration and eventually reach the global optimum.

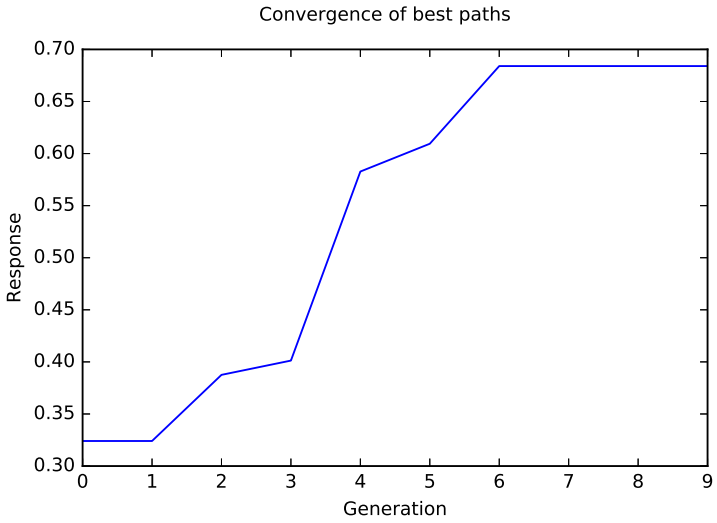

Figure 4: Worst solution case: This plot shows the best paths responses for each generation, as we can see the ants increase the goodness of the solutions at each iteration and eventually get stuck on a local maxima.

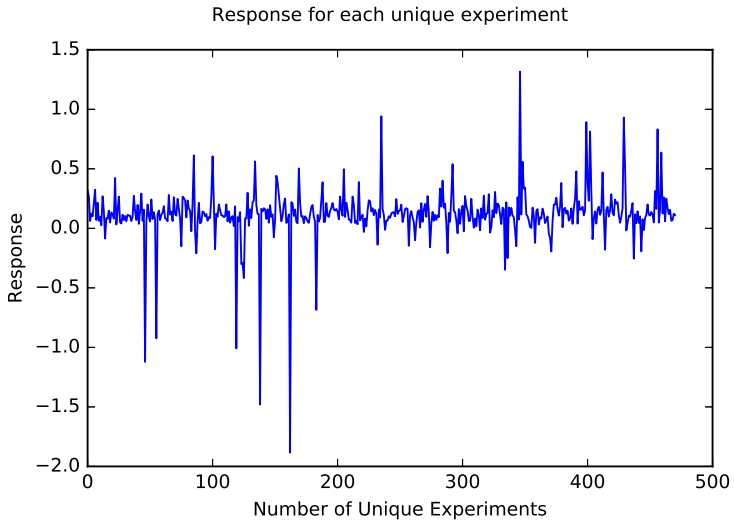

Figure 5: Best solution case: This plot shows each experiment which has to be conducted toward the convergence of the algorithm, we can see that with slightly more than 500 experiments the algorithm retrieve several good solutions. In fact around the 400th experiment ants have already found the global optimum.

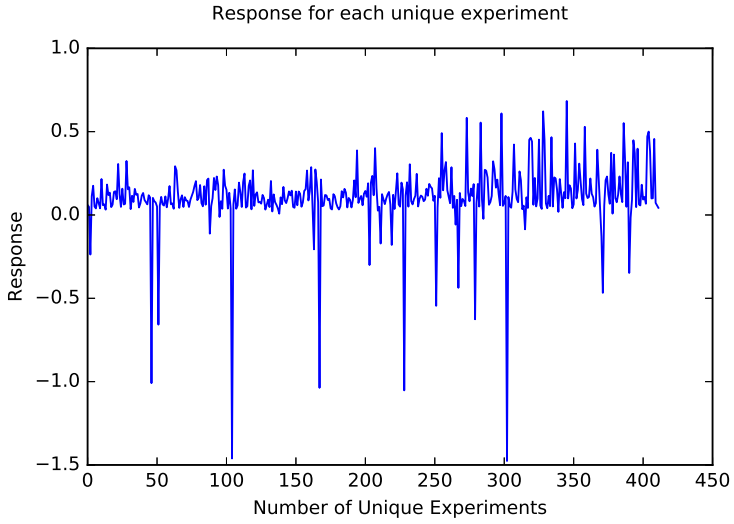

Figure 6: Worst solution case: This plot shows each experiment which has to be conducted toward the convergence of the algorithm, we can see that with slightly more than 500 experiments the algorithm retrieve several good solutions. In this case the search process wasn't so fruitful in fact the algorithm got stuck to a local maxima (which is still a very good result).

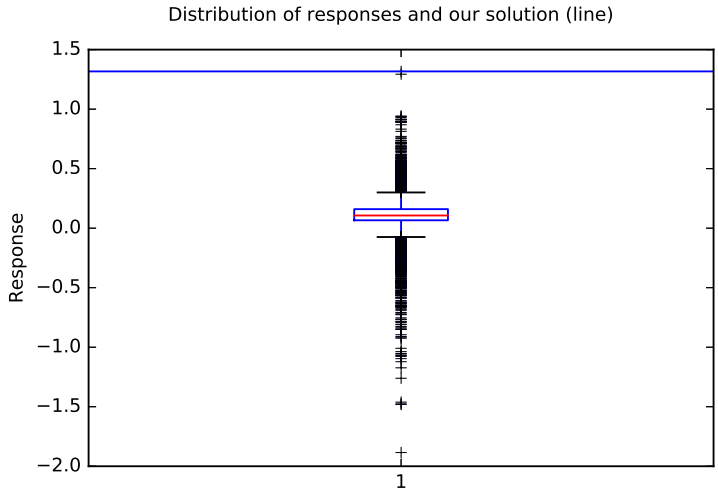

Figure 7: Best solution case: This plot shows the distribution of the experiments responses and the algorithm best solution proposed which is represented with a blue line.

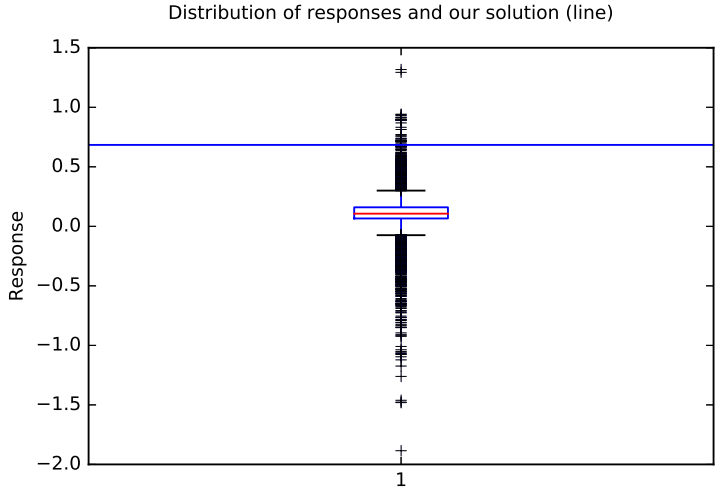

Figure 8: Worst solution case: in this case we can see that the algorithm got stuck in a local maxima.