Supplementary Information - The Development of a Semtex-H Simulant for Terahertz Spectroscopy.

N. Greenall,[1] A. Valavanis,[1] H.J. Desai,[2] D.O. Acheampong,[2] L.H. Li,[1] J.E. Cunningham,[1] A.G. Davies,[1] E.H.Linfield[1] and A.D. Burnett.[1,3*]

[1]School of Electronic and Electrical Engineering, University of Leeds, LS2 9JT, UK.

[2]Centre for Applied Science and Technology, St Albans, Hertfordshire, AL4 9HQ, UK.

[3]School of Chemistry, University of Leeds, Leeds, LS2 9JT, UK.

* A.D.Burnett@Leeds.ac.uk

Genetic Algorithms

Genetic algorithms are in a large part based on the principles of natural selection. This broad class of algorithms work iteratively on a population of solutions until an exit condition is met; a flowchart of the process used for both stage 1 and 2 discussed in the main article is shown in Figure S1. An initial population is generated randomly with a uniform distribution.[1] During each iteration of the population, the population is evaluated and ranked by a fitness function. The top two ranked solutions are selected as 'elites', and pass directly to the next iteration of the population. 'Parents' are then selected from the previous iteration of the population to form a new population through either 'crossover' or 'mutation'.[1] 80 % of the new non-elite population is formed by crossover, a randomised mixture of two parents, from the previous iteration. The remaining 20 % are formed by mutation, i.e. random change to a single parent. The process continues to iterate over the population, until the fitness function score of the top mixture has changed by less than 1×10^{-6} over 50 iterations.

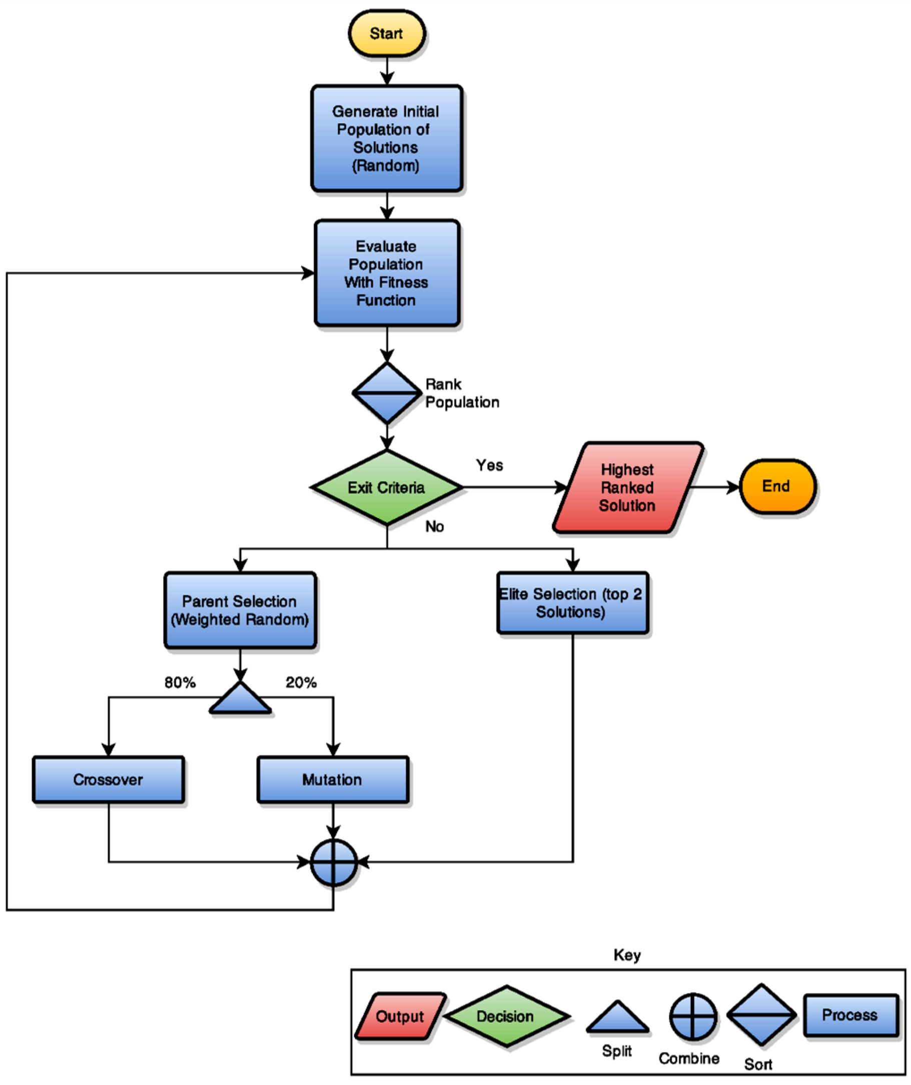
Documentation and a detailed explanation of the implementation used can be found at [1]. In the stage 1 of the selection process where chemical components of the mixture are selected, a variation of the genetic algorithm which optimises integer problems was used, during genetic algorithm operations such as 'crossover' or 'mutation' solutions are forced to take integer values,[2] using a penalty function.[3] In the stage 2 of the process where the concentration of the fixed components are optimised, a non-linear constraint[4] was used to enforce concentrations summing to total value of one.

Simulant Generation Implementation

To select the components, and the concentrations of these components, of a potential simulant, we use two stages based on a genetic algorithm optimiser. The first stage selects the components of a simulant from a larger library of compounds, the second stage optimises the concentrations of the mixture to create the optimal solution.

In the stage 1 of the selection of a suitable mixture (see Figure S2) a population of 50 random combinations of compounds is created from the library of initial 'pure' spectra. Each combination consists of five compounds, which can be repeated to give greater weight to a particular component. The fitness function used by the genetic algorithm first converts repetition into concentration; a component which occurs once in the combination equates to a 20% concentration, a component which occurs twice equates to a 40% concentration, all the way to a component which occurs five times to produce a 100% concentration. This is then used in conjunction with the ‘pure’ spectra taken from the library of spectra to model the mixture spectra using Equation 1 in the main text. This mixture spectra is then compared against the target spectra, Semtex-H, using Equation 2, and normalised using Equation 4 to produce a fitness score. This process is repeated as described above until the fitness function score of the top mixture in the population has changed by less than 1×10^{-6} over 50 iterations.

Stage 1 is then repeated 30 times to produce different viable mixtures from different initial populations that have been generated randomly with a uniform distribution. The resulting top mixture from each run are then collected, duplicates removed and ranked using the fitness score. The top two mixtures are then used in the second stage of the process (see Figure S2). The second stage is similar to the first, but the compounds used in the mixture are now fixed. Here the genetic algorithm is instead used to optimise the concentrations of each component, and the fitness function instead uses the concentrations of each component to model the spectra of the different mixtures, which is then compared against the target spectra as in the previous section. A constraint function is also applied to the solution population to ensure that the concentrations sum to 100%.



Start
Generate Initial Population of Solutions (Random)
Evaluate Population With Fitness Function
Rank Population
Exit Criteria
Yes
No
Highest Ranked Solution
End
Parent Selection (Weighted Random)
Elite Selection (top 2 Solutions)
80%
20%
Crossover
Mutation
Key
Output
Decision
Split
Combine
Sort
Process

Figure S1: A flow diagram presenting the methodology behind the genetic algorithm.

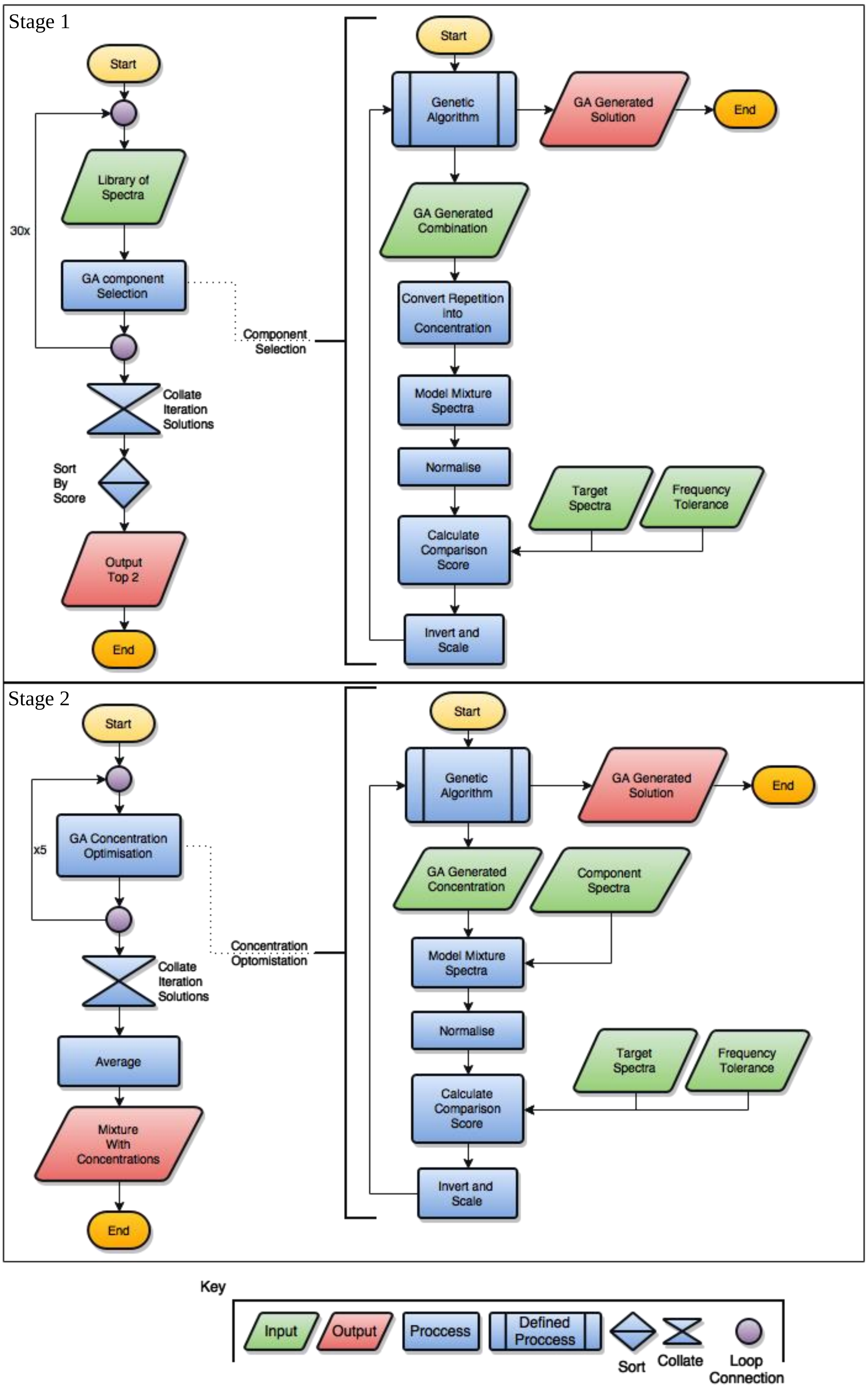


Figure S2: Flow diagrams showing the processes used to select the mixture components (top), and to optimise the concentrations of each component (bottom). The 'genetic algorithm' mentioned is described schematically in Figure S1.

References

1. Matlab. *How the genetic algorithm works*. [Online]. 2015. [Accessed 17 Dec 2015]. Available from: http://uk.mathworks.com/help/gads/how-the-genetic-algorithm-works.html
2. Deep, Kusum, Krishna Pratap Singh, M.L. Kansal, and C. Mohan. *A real coded genetic algorithm for solving integer and mixed integer optimization problems*. Applied Mathematics and Computation, **212**, 2, 505–518, 2009.
3. Deb, Kalyanmoy. *An efficient constraint handling method for genetic algorithms*. Computer Methods in Applied Mechanics and Engineering, **186**, 2–4, 311–338, 2000.
4. Conn, A. R., N. I. M. Gould, and Ph. L. Toint. "A Globally Convergent Augmented Lagrangian Barrier Algorithm for Optimization with General Inequality Constraints and Simple Bounds," *Mathematics of Computation*, **66**, 217, 261–288, 1997.