**Test Problems**

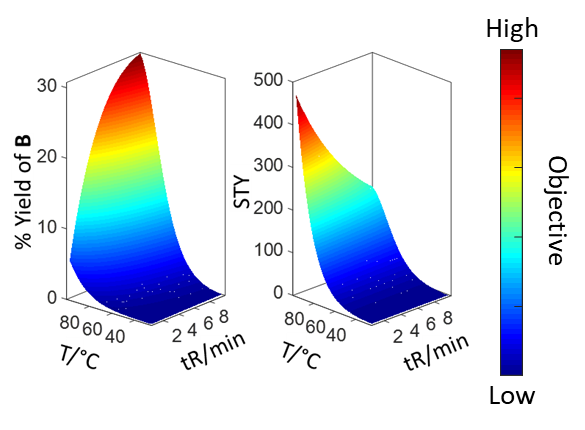
To compare the performance of multi-objective optimisation algorithms (MOOAs) for optimising chemical systems, a kinetic-based reaction simulator has been created. Four reactions with known kinetic parameters (pre-exponential factors and activation energies) were identified in the literature: (i) Van de Vase reaction; (ii) nucleophilic aromatic substitution between 2,4-difluoronitrobenzene and morpholine; (iii) isomerisation of lactose to lactulose; (iv) Paal-Knorr reaction between 2,5-hexanedione and ethanolamine. These examples provide a good representation of non-competitive (iv) and competitive reactions, including competing parallel (i, ii & iii) and consecutive pathways (ii & iii). Although reactions (iii) and (iv) contain reversible reactions, the *k*-1 rate constants are negligible and are therefore omitted. Six test problems have been formulated using reaction variable limits as boundaries and different process metrics as objectives. The boundaries and objectives for each problem have been selected to ensure a diverse range of Pareto fronts were generated in terms of morphology, uniformity and continuity.

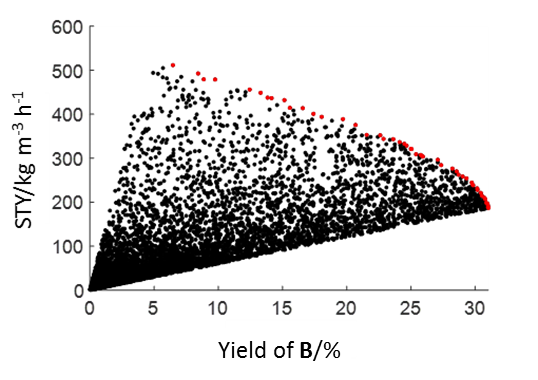
|  |  |  |  |
| --- | --- | --- | --- |
| **Test Problem** | **Variables** | **Objectives** | **Description of Pareto Front** |
| VdV1 | 2 | 2 | Density of solutions fall away near to the Pareto front, which is  non-uniformly distributed between linear and convex regions |
| SNAr1 | 2 | 3 | Optimal solutions follow a convoluted path through objective space with concave regions |
| SNAr2 | 4 | 3 | Convex, non-uniformly distributed Pareto front |
| Lactose1 | 2 | 2 | Pareto front is a convex curve with many solutions |
| PK1 | 2 | 2 | Pareto front is a convex curve with relatively few solutions |
| PK2 | 3 | 2 | Pareto front consists of three discontinuous linear and concave regions |

**VdV1**

|  |  |
| --- | --- |
| (i) | *A1* = 2.1 × 1010 min-1, *A2* = 2.1 × 1010 min-1, *A3* = 1.5 × 108 min-1,  *Ea,1*= 81.1 kJ mol-1, *Ea,2*= 81.1 kJ mol-1, *Ea,3* = 71.2 kJ mol-1 |

|  |  |
| --- | --- |
|  | |
| subject to: | Residence time/min ∈ [0.5, 10]  Temperature/°C ∈ [25, 100]  [**A**]/M = 1 |

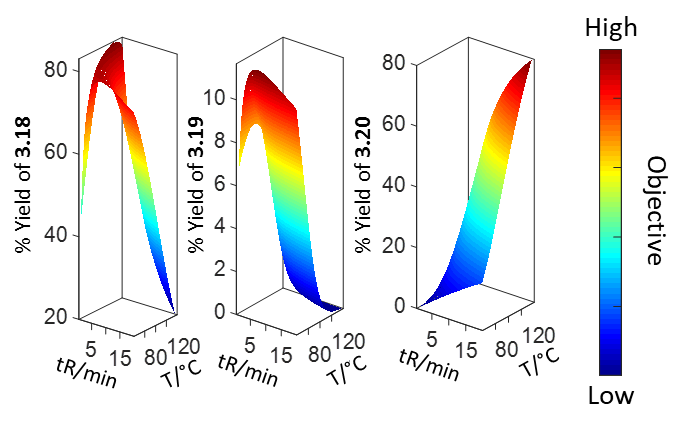


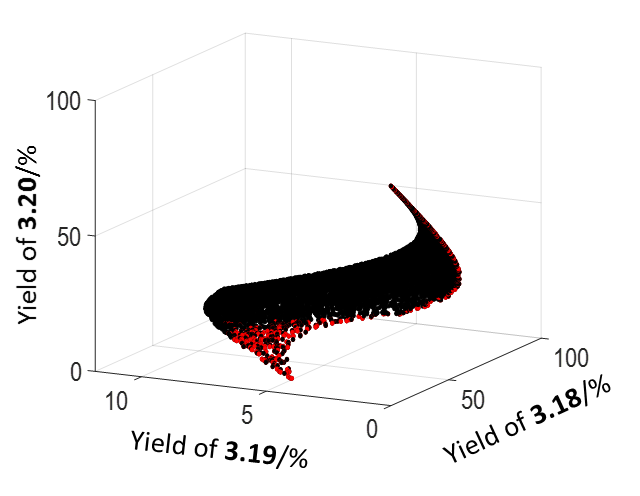


**SnAr1**

|  |  |
| --- | --- |
| (ii) | *A1* = 1.6 × 106 M-1 min-1, *A2* = 1.4 × 104 M-1 min-1, *A3* = 1.0 × 104 M-1 min-1,  *A4* = 3.7 × 108 M-1 min-1, *Ea,1*= 43.2 kJ mol-1, *Ea,2*= 35.3 kJ mol-1,  *Ea,3* = 40.8 kJ mol-1, *Ea,4* = 68.9 kJ mol-1 |

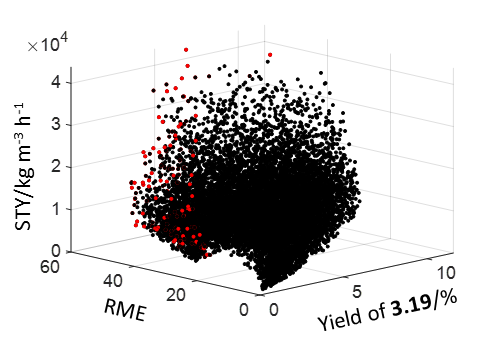
|  |  |
| --- | --- |
|  | |
| subject to: | Residence time/min ∈ [0.5, 20]  Temperature/°C ∈ [60, 140]  [**3.16**]/M = 1  [**3.17**]/M = 3 |





**SnAr2**

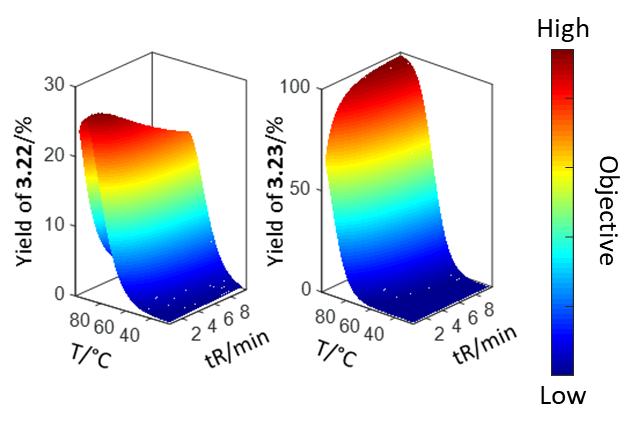
|  |  |
| --- | --- |
|  | |
| subject to: | Residence time/min ∈ [0.5, 2]  Temperature/°C ∈ [60, 140]  [**3.16**]/M ∈ [0.1, 2.0]  [**3.17**]/M ∈ [2, 5] |

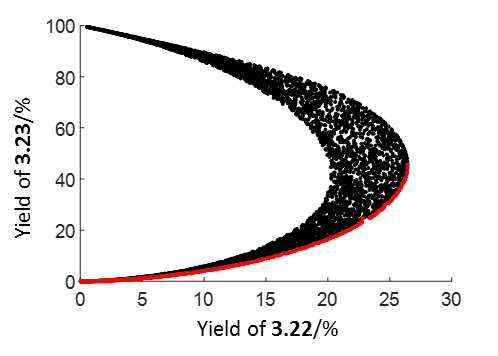


**Lactose1**

|  |  |
| --- | --- |
| (iii) | pH = 11, *A1* = 9.5 × 1014 min-1, *A2* = 7.0 × 1024 min-1, *A3* = 4.0 × 107 min-1,  *Ea,1*= 105.1 kJ mol-1, *Ea,2*= 174.0 kJ mol-1, *Ea,3* = 54.9 kJ mol-1 |

|  |  |
| --- | --- |
|  | |
| subject to: | Residence time/min ∈ [0.5, 10]  Temperature/°C ∈ [25, 100]  [**3.21**]/M = 1 |

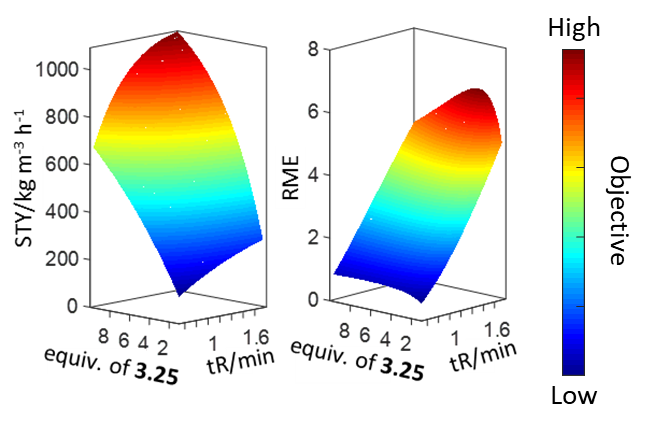


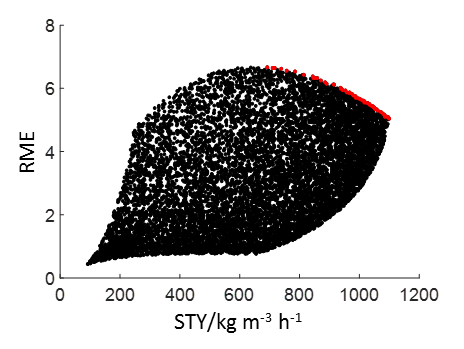


**PK1**

|  |  |
| --- | --- |
| (iv) | *A1* = 15.4 M-1 min-1, *A2* = 405.2 min-1, *Ea,1*= 12.2 kJ mol-1, *Ea,2*= 20.0 kJ mol-1 |

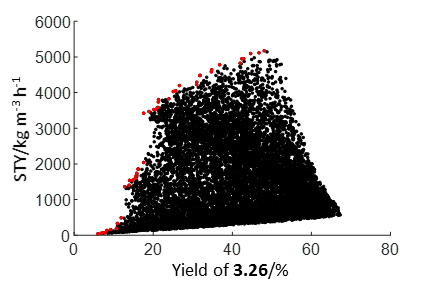
|  |  |
| --- | --- |
|  | |
| subject to: | Residence time/min ∈ [0.5, 2]  Equivalents of **3.25** ∈ [1, 10]  Temperature/°C = 50  [**3.24**]/M = 1 |





**PK2**

|  |  |
| --- | --- |
|  | |
| subject to: | Residence time/min ∈ [0.5, 2]  Temperature/°C ∈ [25, 150]  Equivalents of **3.25** ∈ [1, 10]  [**3.24**]/M = 1 |



**Simulator – How it Works**

The simulation procedure for each test problem is outlined below. Firstly, the pre-exponential factors, *A*, and activation energies, *Ea*, are used to calculate the rate constants, *k*, for each step in the reaction using the Arrhenius equation, where *T* = temperature and *R* = gas constant (8.314 J mol-1 K-1):

The differential rate equations for each step are then solved using an ordinary differential equation (ODE) solver. For example, the differential rate equations for the VdV1 test problem are:

|  |
| --- |
|  |
|  |
|  |
|  |

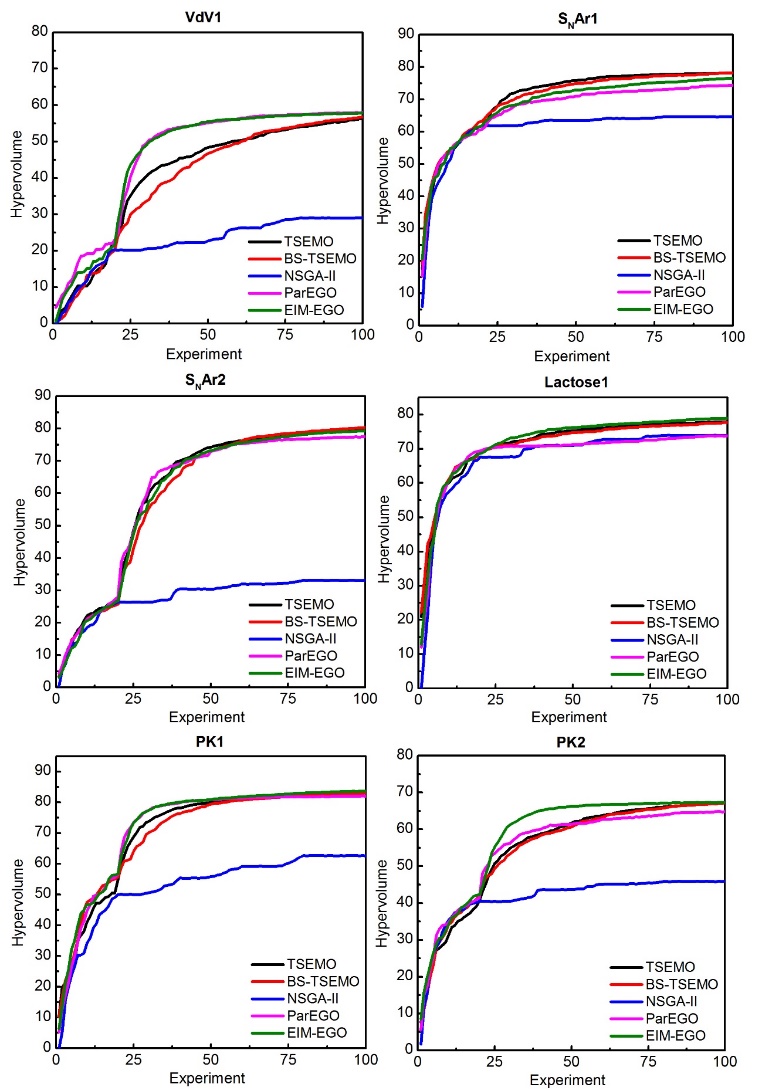
In this case, the reactor is modelled as four CSTRs-in-series by solving simultaneously the coupled ODE equations, which are terminated after four reactor volumes to ensure steady-state is simulated. This provides the percentage of each species in the reaction mixture under different sets of conditions, which are subsequently used to calculate the objectives for the given test problem. Random noise inherent with experimental systems is also included by applying a maximum absolute error of 0.25% and maximum relative error of 0.5% to the outputs, where *Y* = yield and *rand* = random number between 0 and 1. If the adjusted yield, *Yadj*, is less than 0 or greater than 100, then it is forced onto the nearest boundary.

To compare the performance of the algorithms, the hypervolume is calculated after each iteration, where the hypervolume is defined as the volume between the current Pareto front and a reference point (i.e. larger hypervolume = better Pareto front). The hypervolume is calculated using a Monte-Carlo approximation, which determines the percentage of 100,000 random points in the objective space which are dominated by the current Pareto front. The utopian and anti-utopian point for the objective space of each test problem were selected by creating a superset of the   
non-dominated solutions from all runs across all algorithms. The reference point for the objective space was then defined as the anti-utopian point shifted by 0.01 of the difference between the utopian and anti-utopian point.

The performance of Thompson sampling efficient multi-objective optimisation (TSEMO), Pareto efficient global optimisation (ParEGO), NSGA-II and expected improvement matrix efficient global optimisation (EIM-EGO) were compared using this approach. Implementations of ParEGO, NSGA-II and EIM-EGO were all available in the platform for evolutionary multi-objective optimisation (PlatEMO) toolbox in MATLAB. An implementation of TSEMO was available on GitHub, and was compared with both one and four points (batch sequential, BS-TSEMO) per iteration. TSEMO, ParEGO and EIM-EGO were all chosen as they represent surrogate model-based multi-objective optimisation algorithms, whereas NSGA-II is a commonly used genetic algorithm. The TSEMO, BS-TSEMO, ParEGO and EIM-EGO were initialised using a LHC design of size 20. Each algorithm had a function evaluation budget of 100, and was ran 20 times for each test problem to compare average performance. To account for the function evaluation budget, the NSGA-II population size and total number of generations were changed to 20 and 5 respectively.

Plots showing the average change in hypervolume throughout the optimisations, and boxplots of the optimisation results after 60 function evaluations are displayed below. These results can be used as benchmarks to compare against other/new multi-objective algorithms.

F:\test.tif



**Simulator – How to Use**

* Software requirements: MATLAB, optimization toolbox, statistics toolbox.
* To test an algorithm, it must be written in the following format as a .m file:

Inputs:

* data.x = conditions
* data.y = responses
* lowerbounds = lowerbounds of conditions
* upperbounds = upperbounds of conditions
* Opt = algorithm options

Outputs:

* conditions = conditions for next iteration (can contain as many rows as desired)

Example:

Opt = TSEMO\_options();

conditions = TSEMO(data.x, data.y, lowerbounds, upperbounds, Opt);

* Each test problem can be used by running the respective script:
* VdV1 = VdV1\_Test\_Problem
* SnAr1 = SnAr1\_Test\_Problem
* SnAr2 = SnAr2\_Test\_Problem
* Lactose1 = Lactose1\_Test\_Problem
* PK1 = PK1\_Test\_Problem
* PK2 = PK2\_Test\_Problem
* When the script is run, the user is prompted to select the .m file containing the algorithm they want to test.
* The TSEMO algorithm has been included as an example, and can be run by selecting the “TSEMO\_example.m” file when prompted.
* The hypervolume is calculated after each experiment, and will be plotted in real-time.
* By default, each test problem will initiate with 20 LHC experiments, and will terminate after an additional 80 experiments have been run by the algorithm (total experiments = 100). These can be adjusted within the test problem scripts by changing the size of the “initial\_sample\_size” variable and “total\_expts” while loop respectively.
* The following optimisation data is stored in the “data” structure:

Data structure:

* data.x = all conditions
* data.y = all natural log transformed responses
* data.z = all untransformed responses
* data.idxs = index of the conditions which have Pareto-optimal solutions
* data.optconds = conditions which have Pareto-optimal solutions
* data.opt = number of Pareto-optimal solutions per iteration
* data.front = Pareto-optimal solutions
* data.hv = hypervolume per iteration