# Bifunctional Catalyst

Sima Shafaei and Jim LeFevre

CSE 620 Combinatorial Optimization and Modern Heuristics

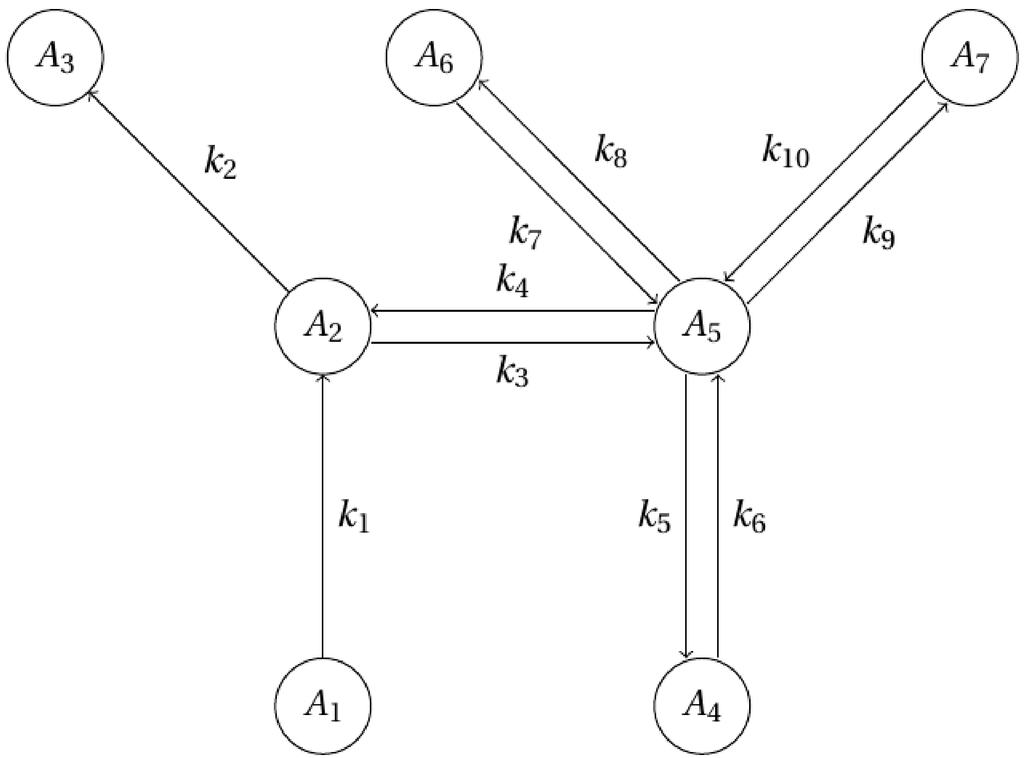
Fall 2020

Slides Version 2.0

# Introduction

- A process for converting methylcyclopentane to benzene makes use of a bifunctional catalyst blend to speed up the reaction.
- The catalyst blend contains a hydrogenation component and an isomerization component.
- The hydrogenation proportion $u$ can be adjusted between 0.60 and 0.90.
- The reaction also involves 2 intermediates and 3 byproducts.
- The process can be subdivided into 10 stages, with 10 corresponding $u$ values.
- What 10 values of $u$ will yield the greatest amount of benzene?

# Optimization Problem




$$\frac{dx_1}{dt} = -k_1 x_1$$
$$\frac{dx_2}{dt} = k_1 x_1 - (k_2 + k_3) x_2 + k_4 x_5$$
$$\frac{dx_3}{dt} = k_2 x_2$$
$$\frac{dx_4}{dt} = -k_6 x_4 + k_5 x_5$$
$$\frac{dx_5}{dt} = k_3 x_2 + k_6 x_4 - (k_4 + k_5 + k_8 + k_9) x_5 + k_7 x_6 + k_{10} x_7$$
$$\frac{dx_6}{dt} = k_8 x_5 - k_7 x_6$$
$$\frac{dx_7}{dt} = k_9 x_5 - k_{10} x_7$$

Reference [1]

# Optimization Problem

## Problem details

- Reaction constants *k* are calculated from empirical data
- The initial chemical quantities:
  - 1.0 for methylcyclopentane
  - 0.0 for the other 6 chemicals

| Attribute | Specification |
|---|---|
| Cost function | $x_7(2000)$ |
| Variable | $\mathbf{u} = [u_1,\ldots,u_{10}]$ |
| Constraints | $u_i >= 0.60$, $i = 1,\ldots,10$ |
| | $u_i <= 0.90$, $i = 1,\ldots,10$ |

## Search Space

- The search space has 10 identical dimensions.
- Each axis is a real number with a range [0.60, 0.90].
- Real numbers are expressed with 8 significant digits.

Reference [1]

# Difficulties

- The fitness landscape is known to exhibit over 300 local maxima [2]. Therefore, many algorithms are at risk of converging to a local maximum.
- High dimensionality → high computational complexity.
- The fitness function has a high computational complexity.
  - Requires the solution of a system of differential equations

# Related Problems

- The problem can be formulated as an NLP:

maximize $f(\mathbf{u})$

subject to $g_i(\mathbf{u}) \geq 0, i = 1, \ldots, 20$

where

$$f(\mathbf{u}) = \mathbf{x}(2000)^t * \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}^t$$

$$\mathbf{x}(t) = \mathbf{x}_0 + \int_0^t \frac{d\mathbf{x}}{dt} dt$$

$$\mathbf{x}_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\frac{d\mathbf{x}}{dt} = \begin{bmatrix} \frac{dx_1}{dt} & \ldots & \frac{dx_7}{dt} \end{bmatrix}^t$$

$$g_1(\mathbf{u}) = \mathbf{u}^t * \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^t - 0.60$$

$$g_2(\mathbf{u}) = \mathbf{u}^t * \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^t - 0.60$$

$$\ldots$$

$$g_{11}(\mathbf{u}) = -(\mathbf{u}^t * \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^t - 0.90)$$

$$\ldots$$

$$g_{20}(\mathbf{u}) = -(\mathbf{u}^t * \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}^t - 0.90)$$

# Optimization Methods

- Classical:
  - Dynamic programming has been used successfully on this problem in the past [1][3], so we chose this method.
  - Specifically, iterative dynamic programming (IDP): each iteration 'zooms in' on a search space around the best result from the prior iteration
- Meta-heuristic:
  - We evaluated a basic genetic algorithm (GA), a GA with sharing, and a GA with deterministic crowding (DC).
  - DC exhibited the best performance and was selected for further analysis.

# Pseudocode – DC

```
for i in 1:g
  for j in 1:n/2
    Randomly select (without replacement) p1 and p2
    Apply variation operator on p1 and p2 to generate c1 and c2
    Evaluate fitness f of c1 and c2
    if dist(p1, c1) + dist(p2, c2) <= dist(p1, c2) + dist(p2, c1)
      if f(c1) > f(p1), then select c1 and remove p1
      if f(c2) > f(p2), then select c2 and remove p2
    else
      if f(c2) > f(p1), then select c2 and remove p1
      if f(c1) > f(p2), then select c1 and remove p2
```

Reference [4]

# DC Implementation

- The initial phenotypes were assigned by random assignment from a uniform distribution across the search space
- No genotype encoding was used
    - Phenotypes were directly altered for crossover and mutation
- Fitness function:  benzene produced ($x_7$ at $t$ = 2000)

# DC Implementation

- Variation operator:
  - Crossover:

$$c_1 = \alpha p_1 + (1 - \alpha) p_2$$

$$c_2 = \alpha p_2 + (1 - \alpha) p_1$$

    - $\gamma$= 0.1 and α is a random number from a uniform distribution in [-$\gamma$, 1 + $\gamma$]
  - Mutation:

$$c_1' = c_1 + \sigma\alpha$$

    - $\sigma$ = 0.1 and $\alpha$ is a random number from a normal distribution with mean 0 and standard deviation 1

Reference [5][6]

# Pseudocode – IDP

```
Choose a number of x-grid points N, a number of allowable u values M,
initial control policy u_p, and initial region r(k)
Generate the [N, 10] x-grid, where each element is a starting point for
integration from stage k to k + 1
for i in iterations:
 Go to stage 10; for each x-grid point:
  Integrate stage 10 with each of M allowable values of u(9); store
  the u with best fitness
 Go to stage 9; for each x-grid point:
  for each of M allowable values of u(8):
   Integrate stage 9; find the x-grid point closest to result; get
   corresponding u value u(9)'
   Continue integration through stage 10, using u(9)'; store the value
   of u(8) that yields best fitness
 Repeat until stage 1. Find the u that yields the best fitness, and
 store this as u_max.
 Update r(k) = r(k)*γ and u_p = u_max
```

Reference [1]

# IDP Implementation

- The policy $\mathbf{u}_p$ and region $r$ combine to establish a range of allowable $u$ values to explore
  - $\mathbf{u}_p$ is a vector of size 10
    - i.e., each stage has its own range
  - The allowable values fall within $[u_p(k) - r/2, u_p(k) + r/2]$
  - After each iteration, $r$ decreases and the range is re-centered on an updated $\mathbf{u}_p$
    - However, the range may never extend outside of [0.60, 0.90]
  - $M$ different values are randomly selected from this range

Reference [1]

# Relation to Existing Methods

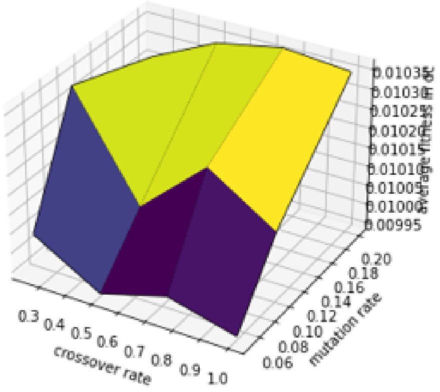
- DC
  - Evaluates complete solutions
  - Solutions are approximate:  may not be exact maxima; may be local but not global maxima
- IDP
  - Determines solutions to reduced problems, then builds complete solutions
  - Solutions are approximate, as above

# Complexity

- DC: $O$(population*iterations)
  - No steps in algorithm are > linear complexity
- IDP: $O(N*M*\text{iterations})$
  - In each iteration, $M$ candidates are evaluated at $N$ grid points for each stage
  - Number of stages is relevant but was considered fixed for this project
- In both cases, fitness evaluation is expensive, but the cost per evaluation is linear
- The choice of parameters determines which method is more expensive

# Experimental Methodology - DC

- Algorithms were implemented in Python, using the NumPy and SciPy libraries.
- Preliminary experiments tested $F_{max}$ at various values of $P_c$ and $P_m$.
- Parameters for final experiments:
  - $P_c = 0.75$ and $P_m = 0.2$
  - Population = 100
  - Iterations = 50



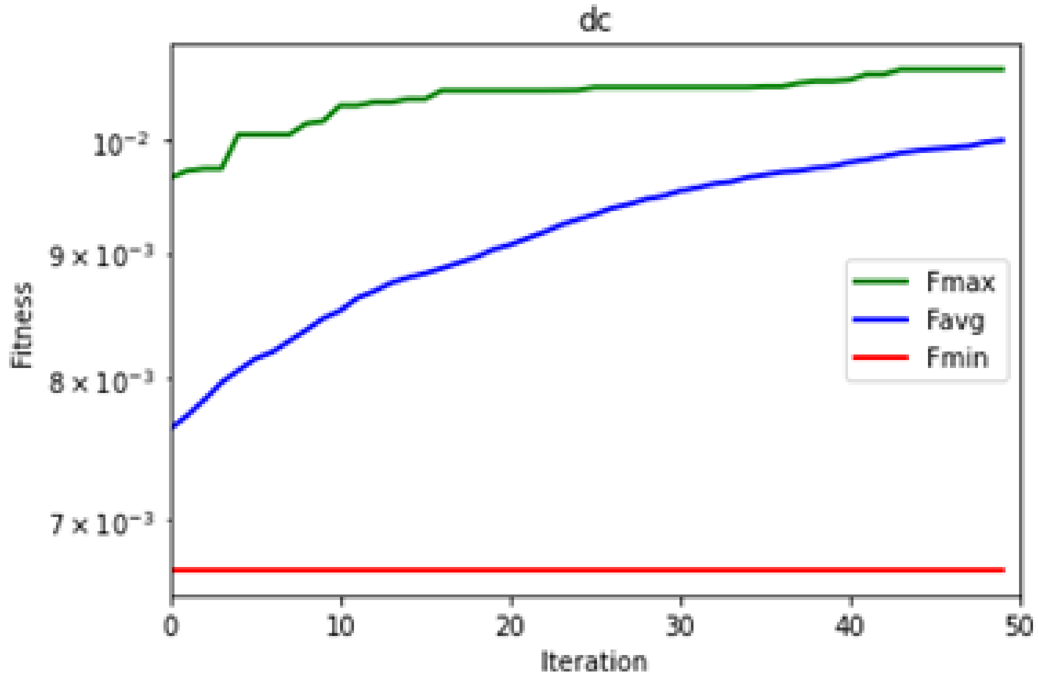

| | 0.05 | 0.1 | 0.2 |
|---|---|---|---|
| 1 | 0.00994484 | 0.0101158 | 0.0103505 |
| 0.75 | 0.00999093 | 0.0102279 | $\boxed{0.0103668}$ |
| 0.5 | 0.00994237 | 0.010076 | 0.0103357 |
| 0.25 | 0.0100427 | 0.0103368 | 0.0102566 |

# Experimental Methodology - IDP

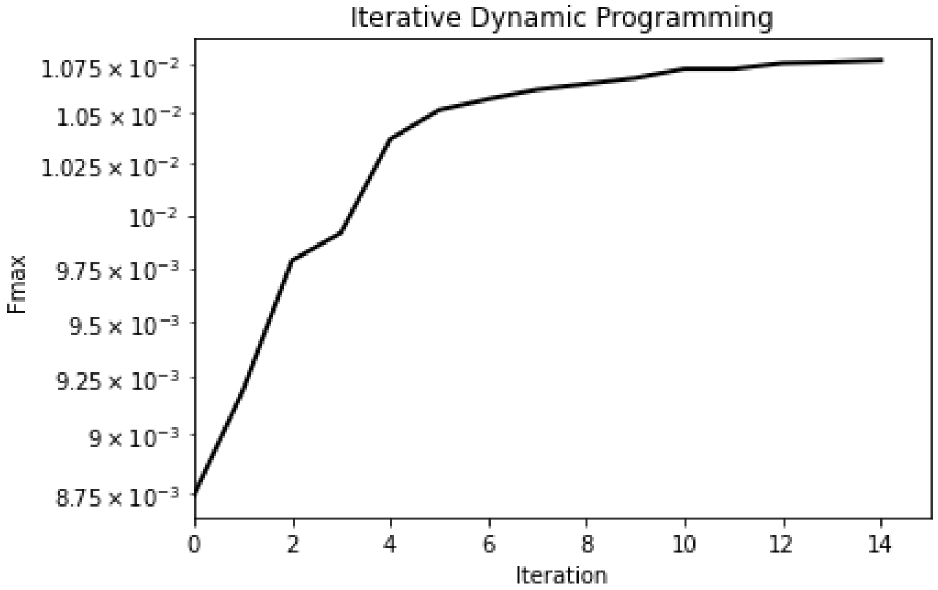
- Parameters:
  - $N = 30$
  - $M = 7$
  - $r = 0.3$
  - $\gamma = 0.7$
  - Initial $\mathbf{u}_p = 0.75$
  - Number of iterations = 20

# Experimental Analysis – DC

- Monotonic convergence of both $F_{max}$ and $F_{avg}$
- Final best value of fitness: 0.0108
- Elapsed time: 4,149 s



# Experimental Analysis – DP




- Rapid convergence to peak with fitness 0.0108
- Nearly monotonic increase in fitness
- Elapsed time:  5,178 s

# Experimental Analysis

- Both methods found the same peak, the one identified by [1] as the global peak.

| Phenotype | [1] | DC | IDP |
|---|---|---|---|
| $u_1$ | 0.6661 | 0.67650516 | 0.66365801 |
| $u_2$ | 0.6734 | 0.68028349 | 0.68009955 |
| $u_3$ | 0.6764 | 0.67512455 | 0.68362714 |
| $u_4$ | 0.9000 | 0.90000000 | 0.89549726 |
| $u_5$ | 0.9000 | 0.90000000 | 0.89985661 |
| $u_6$ | 0.9000 | 0.90000000 | 0.89991729 |
| $u_7$ | 0.9000 | 0.899953528 | 0.89927994 |
| $u_8$ | 0.9000 | 0.90000000 | 0.89980769 |
| $u_9$ | 0.9000 | 0.90000000 | 0.90000847 |
| $u_{10}$ | 0.9000 | 0.90000000 | 0.89992845 |

# Conclusions

- Previously established global maximum is 0.0101 [1]
- DC and IDP appear to have located this peak
  - Our fitness result at this peak is slightly higher than [1] and [2]
    - Possibly due to differences in the differential equation solver algorithms used or use of more significant digits (in *u* values and in solver implementation)
- Outcome demonstrates advantage of a GA to optimize a problem with many local optima
  - Niching aspect of DC also helps to identify other local optima with relatively high fitness

# References


[1] Luus, R., Dittrich, J., & Keil, F. J. (1992). Multiplicity of solutions in the optimization of a bifunctional catalyst blend in a tubular reactor. *The Canadian Journal of Chemical Engineering*, *70*(4), 780-785.

[2] Esposito, W. R., & Floudas, C. A. (2000). Deterministic Global Optimization in Nonlinear Optimal Control Problems. *Journal of Global Optimization*, *17*, 97-126.

[3] Luus, R., & Bojkov, B. (1994). Global optimization of the bifunctional catalyst problem. *The Canadian Journal of Chemical Engineering*, *72*(1), 160-163.

[4] Mahfoud, S. W. (1995). A Comparison of Parallel and Sequential Niching Methods. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, 136-143.

[5] Heris, M. K. Genetic Algorithm in Python – Part A – Practical Genetic Algorithms Series. In *YouTube*.

[6] M. K. Heris. Genetic Algorithm in Python – Part B – Practical Genetic Algorithms Series. In Y*ouTube*.