Symbiotic Organisms Search

Winter 2021

Partial Courtesy of Dr. Doddy Prayogo

Basics of Symbiotic organisms search

- Introduced by Cheng and Prayogo in 2014
- Optimize continuous or real value problems
- Stochastic, population-based search
- Requires no algorithmic parameters

Cheng, M.-Y and Prayogo, D. (2014), “Symbiotic Organisms Search: A new metaheuristic optimization”, Computers & Structures, 139, pp. 98–112.

http://140.118.5.112:85/SOS/

Dr. Prayogo Prof. Cheng

Concepts (1)

- SOS iteratively a population of candidate solutions to promising areas in the search space in the process of seeking the optimal global solution.
- New solution generation is governed by imitating the biological interaction between two organisms in the ecosystem (symbiosis).
- Three phases that resemble the real-world biological interaction model are introduced: mutualism phase, commensalism phase, and parasitism phase.

Concepts (2)

- **Symbiosis**

Natural organisms rarely live in isolation. Many of them rely on other species. Some may depend on another for its survival. This interaction is known as symbiosis.

	-	0	+
-	Competition		
0	Amensalism	Neutralism	
+	**Parasitism**	**Commensalism**	**Mutualism**

- = is being harmed, 0 = neutral, + = get benefit

Concepts (3)

- **Mutualism**

Symbiotic relationship between organisms of different species when both benefit from the interaction.

- Example:

- Bee and Flower

- Rhinocero and oxpeckers

Concepts (4)

- **Commensalism**

Symbiotic relationship between organisms of different species when one benefits but the other is neutral (unaffected).

- Example

- Clownfish and Sea Anemone

- Remora and Shark

Concepts (5)

- **Parasitism**

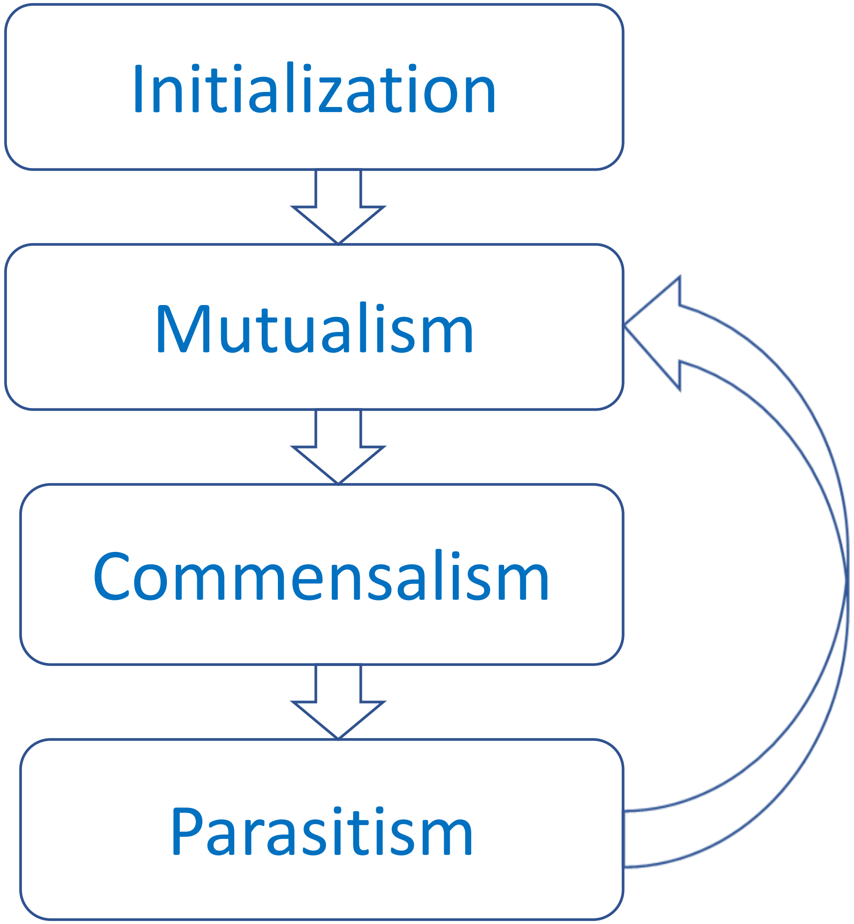
Symbiotic relationship between organisms of different species when one benefits but the other is harmed.

- Example

- Cuckoo and other birds

- Malaria parasite and human

Flowchart



Initialization

- The initial population (ecosystem) is chosen randomly if nothing is known about the system.
- Assume a uniform probability distribution for all random decisions unless otherwise stated.
- It is usually required to have lower and upper bounds for every decision variables (organisms)

Mutualism

The mutualism symbiosis between organism *i* and organism *j* ($j \neq i$) is modeled as follows

$$\mathbf{X_{i,\,new} = X_{i,\,old} + rand[0,1] * (X_{best} - BF1 * MutualVector)}$$

$$\mathbf{X_{j,\,new} = X_{j,\,old} + rand[0,1] * (X_{best} - BF2 * MutualVector)}$$

where

MutualVector = mean ($\mathbf{X_{i,\,old}} + \mathbf{X_{j,\,old}}$)

BF1 (Benefit Factor of organism i)= random value either 1 or 2

BF2 (Benefit Factor of organism j)= random value either 1 or 2

$\mathbf{X_{i,\,old}}$ = Old solution of Organism i

$\mathbf{X_{j,\,old}}$ = Old solution of Organism j

$\mathbf{X_{i,\,new}}$ = New solution of Organism i after mutualism interaction

$\mathbf{X_{j,\,new}}$ = New solution of Organism j after mutualism interaction

Commensalism

The commensalism symbiosis between organism *i* and organism *j* ($j \neq i$) as follows

$$\mathbf{X_{i,\,new} = X_{i,\,old} + rand[-1,1] * (\,X_{best} - X_{j,\,old}\,)}$$

Where:

$\mathbf{X_{i,\,old}}$ = Old solution of Organism *i*

$\mathbf{X_{j,\,old}}$ = Old solution of Organism *j*

$\mathbf{X_{i,\,new}}$ = New solution of Organism *i* after mutualism interaction

Greedy rule

- For both Mutualism and Commensalism, organisms are updated only if their new fitness is better than their pre-interaction fitness.

Parasitism

- Organism $\mathbf{X_i}$ is selected randomly from the ecosystem and serves as a host to the **Parasite_Vector**.
- **Parasite_Vector** is created in the search space by duplicating organism $\mathbf{X_i}$, then modifying the randomly selected dimensions using a random number.
- **Parasite_Vector** tries to replace $\mathbf{X_j}$ in the ecosystem.
- Both organisms are then evaluated to measure their fitness.
- If **Parasite_Vector** has a better fitness value, it will kill organism $\mathbf{X_j}$ and assume its position in the ecosystem.
- If the fitness value of $\mathbf{X_j}$ is better, $\mathbf{X_j}$ will have immunity from the parasite and the **Parasite_Vector** will no longer live in that ecosystem.

Loop

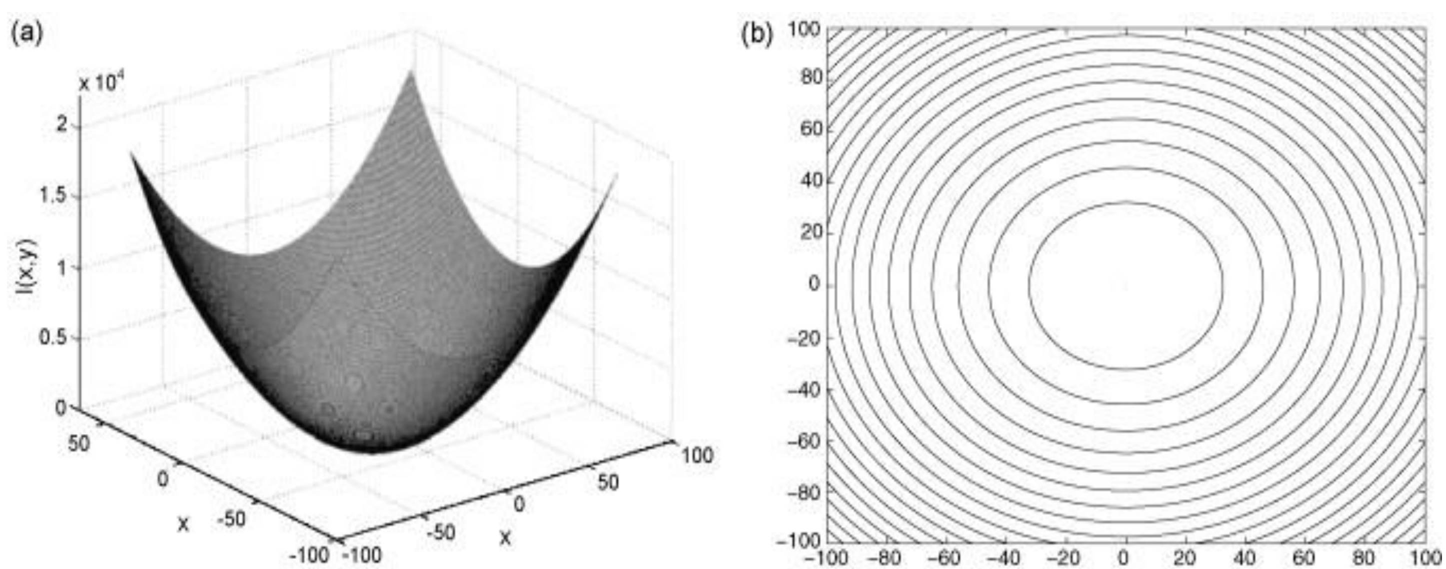
- Mutualism, Commensalism, and Parasitism continue until a stopping criterion is reached
- Stopping criterion
  - Number of iterations
  - Convergence
  - Computation time

Parameter setup

- The ecosystem size may be set between 20 to 50, according to the original paper.
- No algorithmic parameters (such as CR and MR for GA; w, c_1, c_2 for PSO)

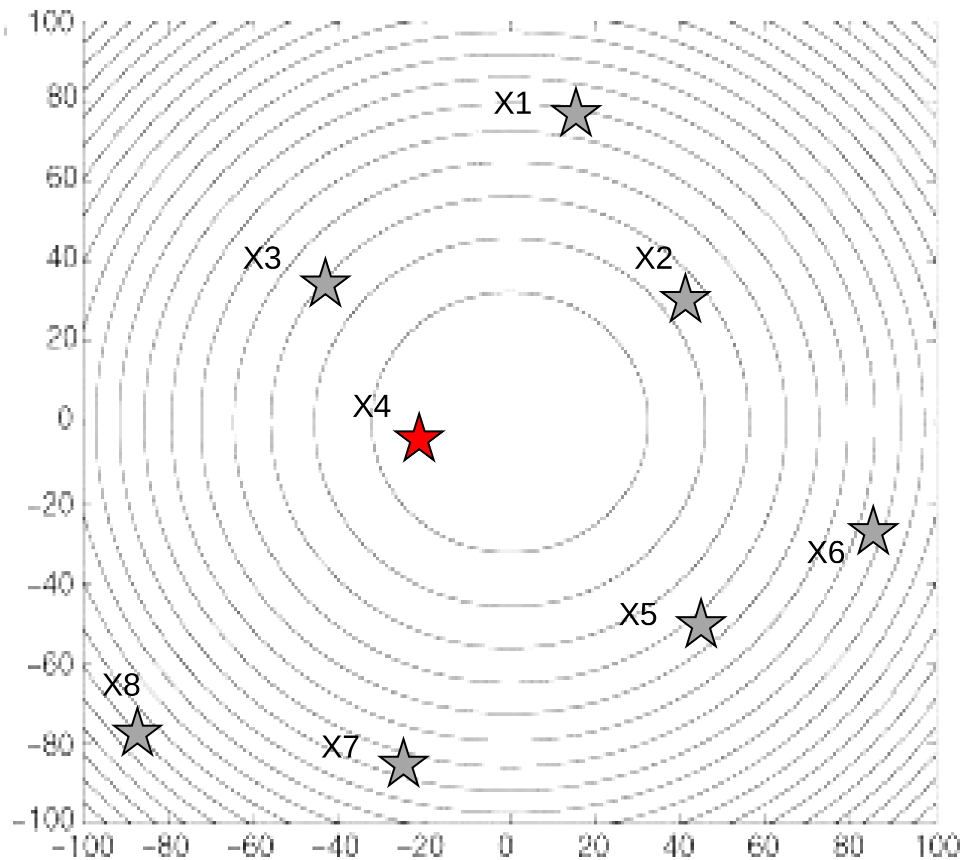
Example (1)

- 2D Sphere function
- Minimize Σx^2



Example (2)

- Initialization

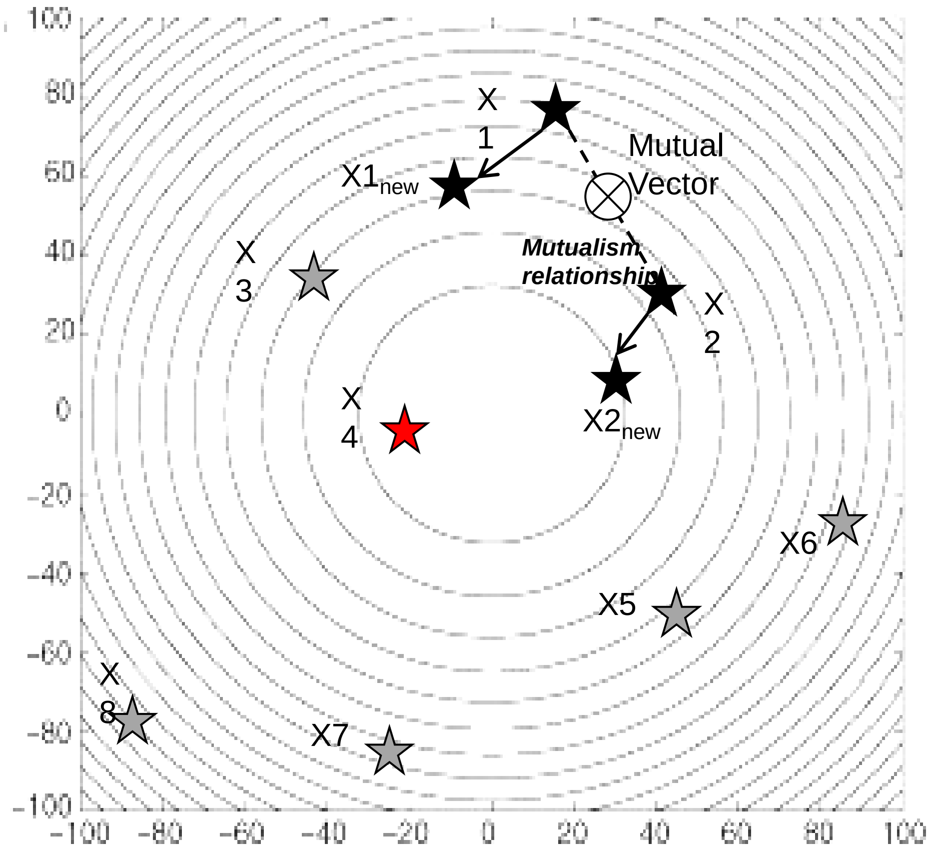


Random location of organisms

X1 = [19, 75] ~ F(X1) (Fitness) = 5896
X2 = [41, 22] ~ F(X2) (Fitness) = 2165
X3 = [-40,30] ~ F(X3) (Fitness) = 2500
X4 = [-10,-4] ~ F(X4) (Fitness) = 116
X5 = [44, -51] ~ F(X5) (Fitness) = 4537
X6 = [83, -30] ~ F(X6) (Fitness) = 7789
X7 = [-22, -84] ~ F(X7) (Fitness) = 7540
X8 = [-87,-79] ~ F(X8) (Fitness) = 13810

Example (3)

- Mutualism



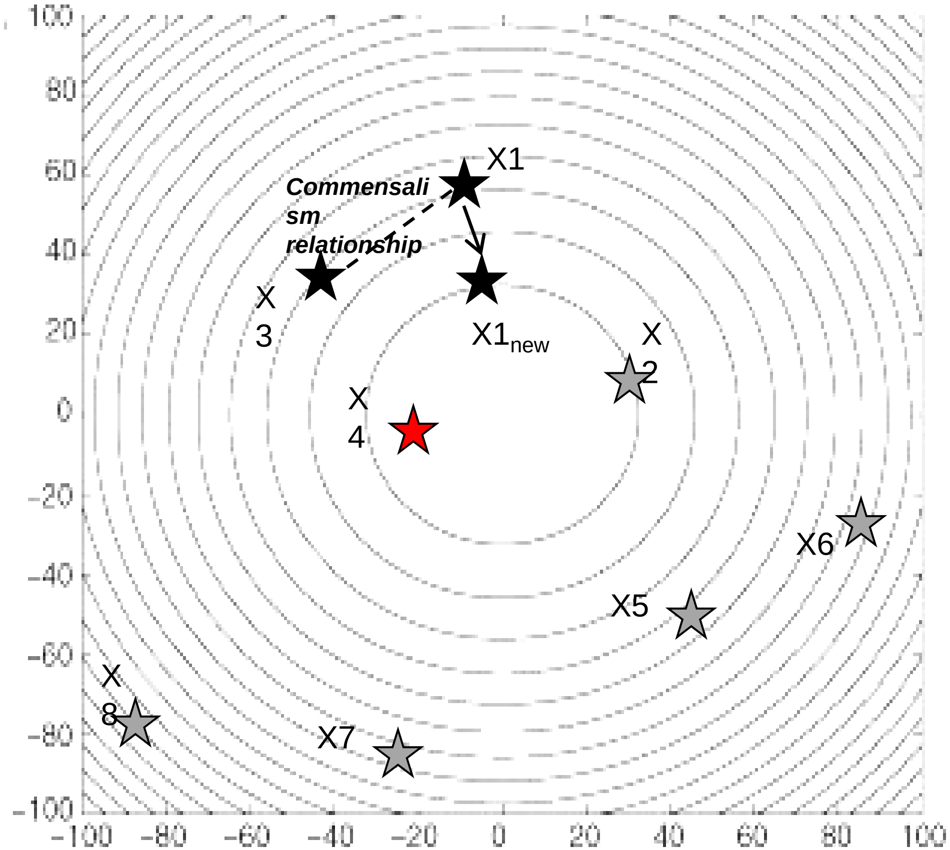
Organism i ~ X1 = [19, 75],
F(X1) = 5896
Organism j ~ X2 = [41, 22],
F(X2) = 2165
MutualVector = Average (X1old + X2old)
= [30, 48.5]

Xinew= Xiold + rand[0,1] * (Xbest – BF1 * MutualVector)
X1new= [19, 75] + [0.83, 0.37]*([-10,-4] – 1*[30, 48.5]
= [19, 75] + [-33.2, -19.4]
= [-14.2, 55.6]
F(X1new) =3290 better than F(X1old)

Xjnew= Xjold + rand[0,1] * (Xbest – BF2 * MutualVector)
X2new= [41, 22] + [0.07 0.21]*([-10,-4] – 2*[30, 48.5])
= [41, 22] + [-4.9, -21.2]
= [36.1, 0.8]
F(X2new) =1304 better than F(X2old)

Example (4)

- Commensalism



X1= [-14.2, 55.6]   F(X1) = 3290
X3= [-40,30]   F(X3) = 2500

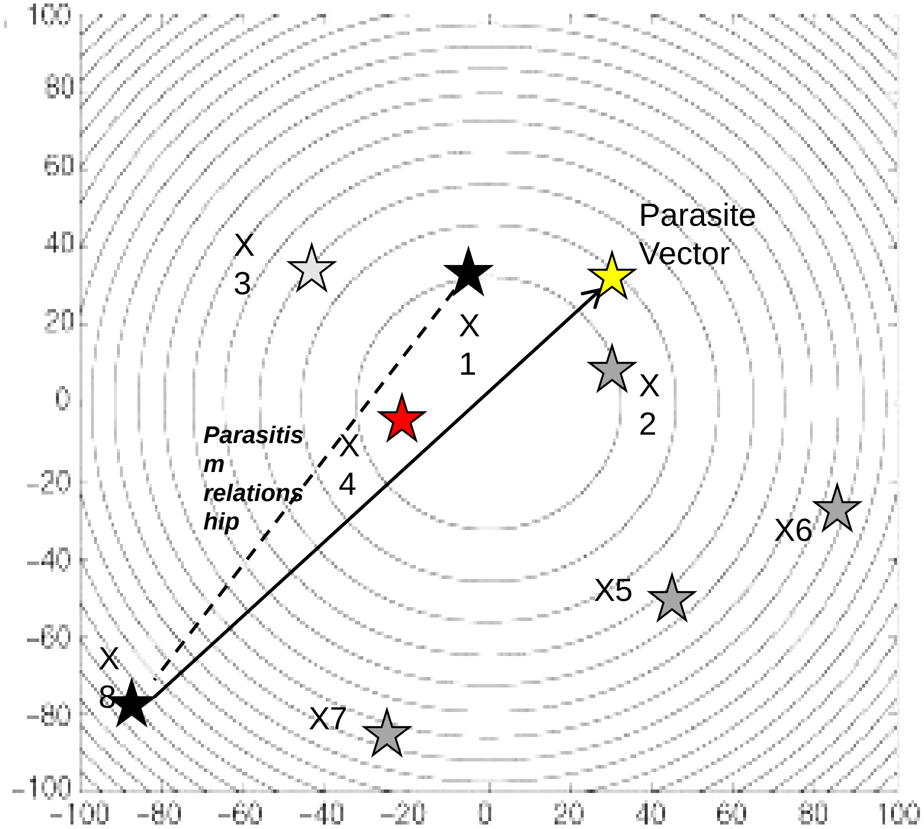
Xinew= Xiold + rand[-1,1] * (Xbest –Xj)

X1new= [-14.2, 55.6] + [0.13, 0.87] *([-10,-4] –[-40, 30])
= [-14.2, 55.6] + [3.9, -29.58]
= [-10.3, 26.0]

F(X1new) =783 better than F(X1old)

Example (5)

- Parasitism



X1= [-10.3, 26]   F(X1) = 783

X8= [-87, -79]   F(X8) = 13810

Parasite Vector = Xi mutated in a random dimension (1st dimension here)

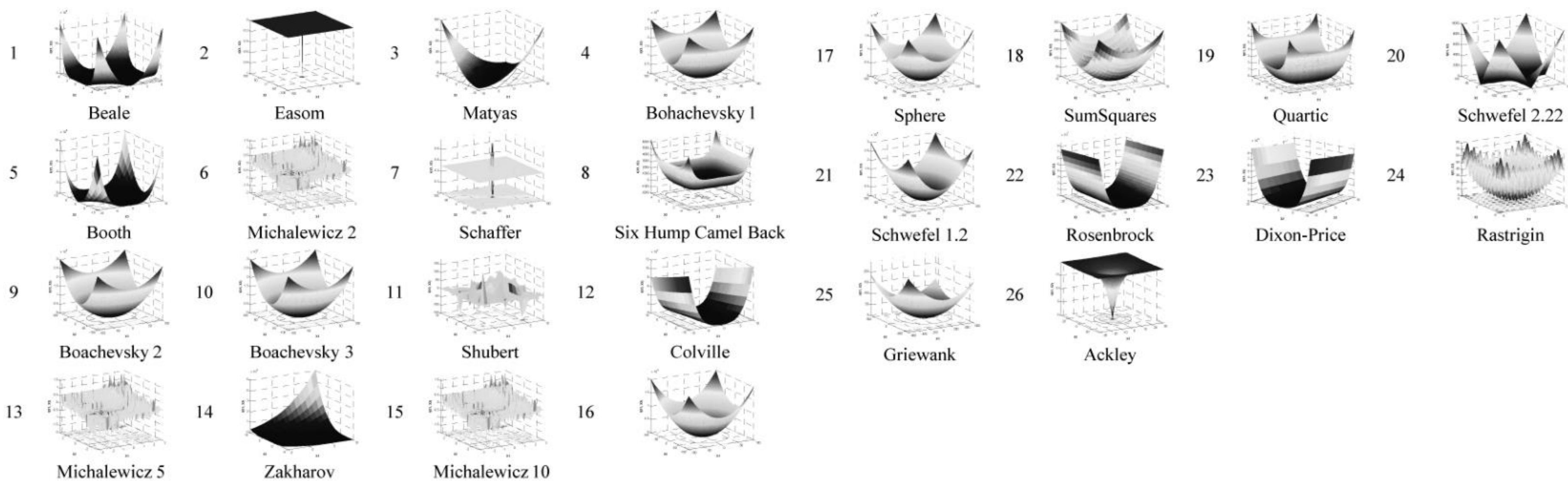
ParasiteVector (PV) = [30, 26]

F(PV) =1576 better than F(X8)

X8 will be vanished, PV will replace it as the new X8.

SOS Applications (1)

- Benchmark functions

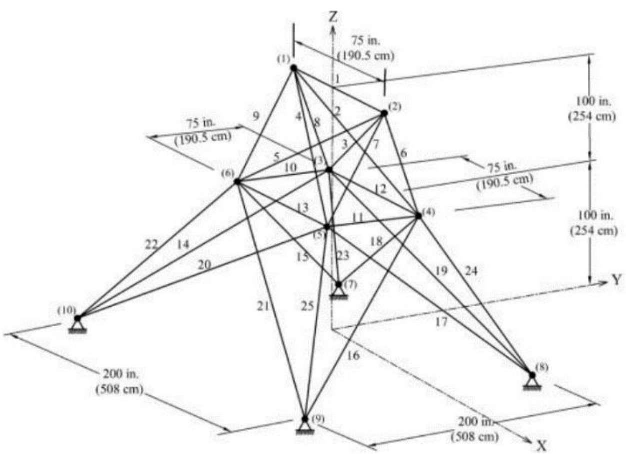


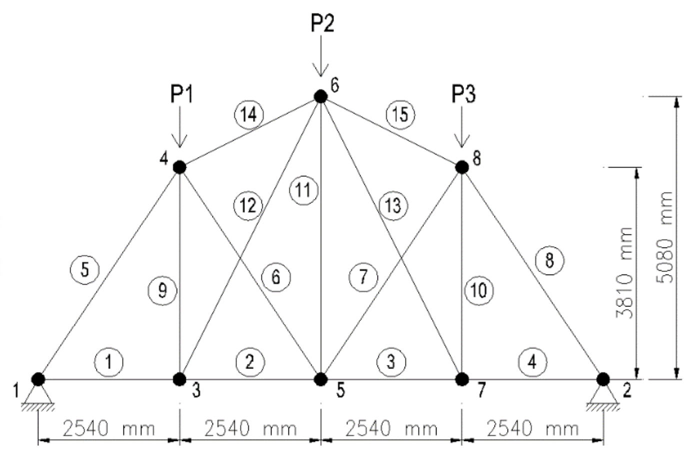
Benchmark functions

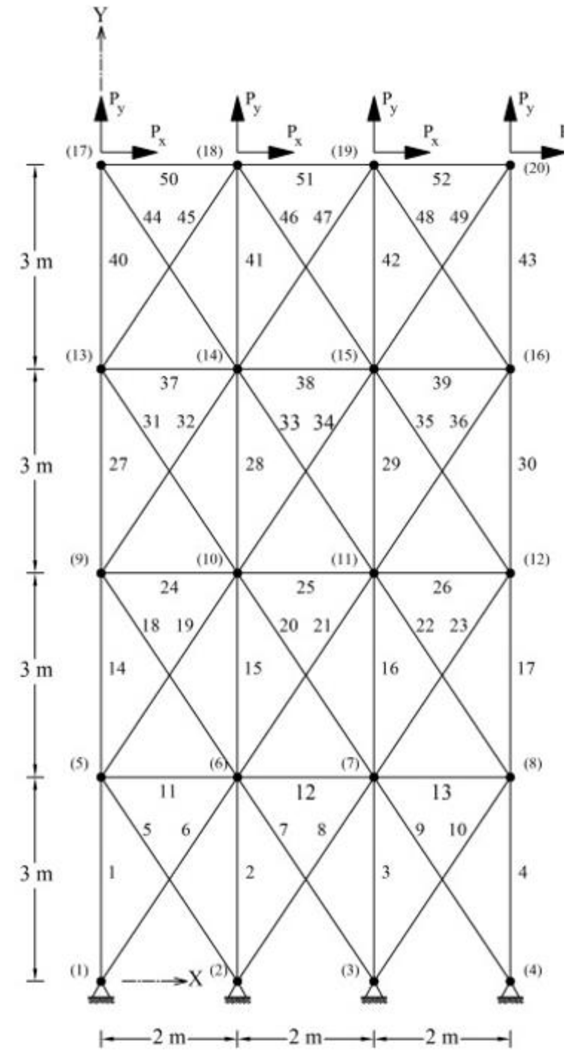
- SOS was compared with other algorithms using the result of 30 runs for every problem
- SOS found the global optimum value for 22 of the 26 functions and outperformed all the other algorithms (GA, PSO, DE).
- SOS was the only algorithm able to solve Dixon-Price and produced the best result of all on the exceptionally difficult Rosenbrock

SOS Applications (2)

- Structural engineering design optimizations







Conclusions

- SOS is simple and easy to program
- Suitable for continuous/real-valued domains
- No algorithmic parameters to adjust
- May be disadvantageous when the evaluation of objective function is time-consuming: it takes more than one evaluations per organism at each iteration
- Need some adjustments for solving discrete problems