Bacterial Foraging Optimization (BFO)

Mahesh Dananjaya

Bacterial Foraging Algorithm   Cellular Automata?

- ❑ In 2002, Passiono et al. proposed optimization technique based on bacterial foraging strategy.
- ❑ E.g. Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), **Bacterial Foraging Optimization** (BFO) etc.
- ❑ Based on the group behaviour of animals and **foraging** (search) of nutrient of microbial creatures.
- ❑ Natural selection gradually diminish animals with poor foraging.
- ❑ Successful animals are viable to reproductive success.
- ❑ BFOA algorithm is a distributed optimization process.

Foraging Strategy

- ❑ Bacteria or foraging organism takes necessary actions to optimize the energy (nutrient) intake per unit time.
- ❑ Affected by environmental constraints (nutrient gradient) predators etc.), individual capabilities (sensing and cognitive capabilities)
- ❑ E.g. E-Coli bacteria is searching for foods by using flagella (actuator) and chemo effector (sensor).
- ❑ Sometimes, bacteria demonstrate **swarm** behavior by organizing as single entity in foraging
- ❑ This simple idea can be applied for real world optimization problems.

Related Work

- ❑ **Passino** et al., Biomimicry of Bacterial Foraging for Distributed Optimization and Control (Original)
- ❑ Liu, Y. and **Passino**, Biomimicry of Social Foraging Bacteria for Distributed Optimization: Models, Principles, and Emergent Behaviors (Swarm)
- ❑ Ajith **Abraham** et al., Adaptive Computational Chemotaxis in Bacterial Foraging Optimization: An Analysis (Adaptive)
- ❑ Ajith **Abraham** et al., Bacterial foraging optimization algorithm: theoretical foundations, analysis, and applications (Adaptive).
- ❑ Kim, D. H., **Abraham**, A., Cho, J. H.: A hybrid genetic algorithm and bacterial foraging approach for global optimization (Hybrid)
- ❑ Biswas, A., Dasgupta, S., Das, S., and **Abraham**, A.: Synergy of PSO and Bacterial Foraging Optimization: A Comparative Study on Numerical Benchmarks (Hybrid)

Related Work

- ❑ A. **Stevens**, “Simulations of the gliding behavior and aggregation of myxobacteria,”
- ❑ A. **Stevens**, “A stochastic cellular automaton, modeling gliding and aggregation (Cellular Automata)
- ❑ **S. Halgamuge** et al. proposed Simplifying the Bacteria Foraging Optimization Algorithm in 2010.

BFOA-Classical

Why E-COli? Simple, comprehensive foraging strategy, E. coli spends less than 1% of its energy budget for motility.

- ❑ Bacteria, e.g., **E-COLI**, tries to maximize their energy intake per unit time.
- ❑ Therefore, they are travelling along nutrient gradient. Objective is to achieve optimal nutrients. In optimization, reduce the objective function value.
- ❑ 4 Main paradigms: Chemotactics, Swarming, Reproduction, Elimination and Dispersal.
- ❑ So, the objective of the optimization to reduce,

$$J\,(i,\,j,\,k,l)$$

$\boldsymbol{\theta^i(\,j,\,k,\,l\,)}$ **then** $\boldsymbol{J(\theta)}$ **- Objective**

Ith	: Bacteria
Jth	: Chemotaxis step
Kth	: Reproduction Step
Lth	: Elimination-Dispersal Step

BFOA - Parameters

- p: Dimension of the search space,
- S: Total number of bacteria in the population,
- N_c : The number of chemotactic steps,
- N_s : The swimming length.
- N_{re} : The number of reproduction steps,
- N_{ed} : The number of elimination-dispersal events,
- P_{ed} : Elimination-dispersal probability,
- C (i): The size of the step taken in the random direction specified by the tumble.
- $D_{attractant}$: depth of the attractant released by the cell(amount)
- $W_{attractant}$: width of the attractant released by the cell (diffusion rate)
- $H_{repellent}$: height of the repellent effect(amount), In original algo., its equal to $D_{attractant}$
- $W_{repellent}$: Width or rate of repellent

Chemotaxis (Foraging) Cellular Automaton?

- One major step in BFOA is the simulated **chemotactic** movement
- This process simulates the movement of an E.coli cell through **swimming** and **tumbling** via flagella.
- $\theta^i(j, k, l)$: position (assignment) of ith bacteria on jth chemotaxis, kth reproduction and lth elimination-dispersal.
- **Tumble**: generate a random vector $\Delta(i) \in \Re^p$ with each element $\Delta_m(i)$,m = 1, 2, ..., p, a random number on [-1, 1].

In Jth chemotaxis loop,

$$\theta^i(j+1, k, l) = \theta^i(j, k, l) + C(i) * \left(\Delta(i) / \mathrm{sqrt} \left(\Delta^T(i) * \Delta(i)\right) \right)$$

In original Passion's algorithm C(i) is fixed and =0.1.It controls the optimal convergence

Adaptive Chemotaxis

:) Better Convergence?

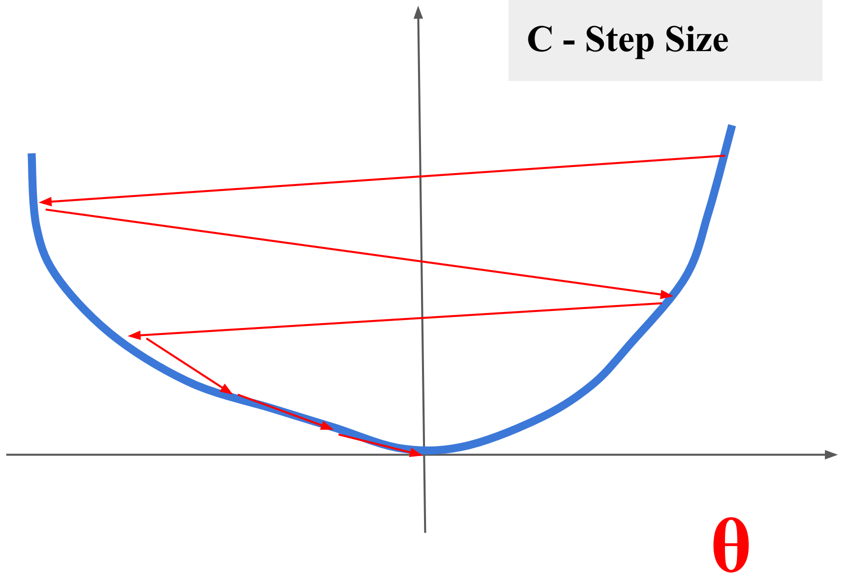
- **Abraham** et al., In "Adaptive Computational Chemotaxis in Bacterial Foraging Optimization: An Analysis" presented this concept
- convergence rate matters, other one is bacterium does not stop taking chemotactic steps and oscillates about the optima. (from paper)
- So, C can be made adaptive,

C - Step Size

$$C(i) = |J(\theta)| / (|J(\theta)| + \lambda)$$

λ : constant parameter

λ also provides the flexibility



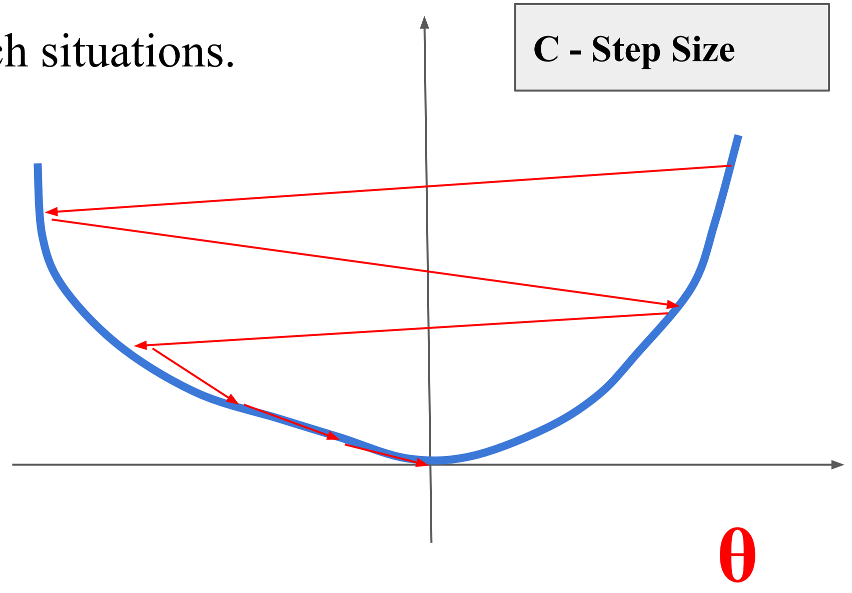
Adaptive Chemotaxis to avoid lock-in states

- ❑ Sometimes lock-in can occurs, a special case of stagnation
- ❑ The lock in condition arises when a bacterium has reached somewhat **near to the optima** of a function and then its further movements are not possible due to comparatively large step size
- ❑ Therefore adaptive chemotaxis can avoid such situations.

$$C(i) = |J(\theta)| / (|J(\theta)| + \lambda)$$

λ : constant parameter

- **Large Step Sizes -> Large Variants**



Special Cases

- ❑ If the optimum value of the objective function is not exactly zero,
- ❑ So, step-size adapted according to original adaptation may not vanish near optima
- ❑ Step-size would shrink if the bacterium come closer to the optima, but it may not approach zero always. (**tiny oscillations**)
- ❑ To get faster convergence for such functions it becomes necessary to modify the adaptation scheme.
- ❑ Use of gradient information to adapt step size is not practical in BFOA*.

$$C(i) = |J(\theta) - J_{Best}| / (|J(\theta) - J_{Best}| + \lambda)$$

***BLACK-BOX : *classical BFOA uses only an approximation of the gradient and not the analytical gradient information (explicit gradient information). C = 1 when way from global optima**

Swimming and Tumbling

Single Chemotaxis stage contains multiple swim steps

New $\Delta(i)$

$\theta^i(j+1, k, l) = \theta^i(j, k, l) + C(i) * (\Delta(i) / \mathrm{sqrt}(\Delta^T(i) * \Delta(i)))$, $\Delta(i) \in \mathfrak{R}^P$

And use this $\theta^i(j + 1, j, k)$ to compute the new $J(i, j+1, k, l)$

While m< N_S (The swimming length)

Let $J(i, j, k, l) = J_{Last}$ and

If $J(i, j+1, k, l) < J_{last}$ (if doing better), let $J_{last} = J(i, j+1, k, l$ and let

$\theta^i(j+1, k, l) = \theta^i(j, k, l) + C(i) * (\Delta(i) / \mathrm{sqrt}(\Delta^T(i) * \Delta(i)))$

And use this $\theta^i(j + 1, j, k)$ to compute the new $J(i, j+1, k, l)$

Else “This is the end of the while statement” - So generate new **$\Delta(i)$**

Tumble

Swim

Swarming

- Grouping behaviour observed on bacterial motility, e.g., E-Coli, Typhimurium
- stable spatio-temporal patterns (swarms) are formed in semi solid nutrient medium.
- A group of E.coli cells arrange themselves in a traveling ring by moving up the **nutrient gradient** when placed amidst a semisolid matrix with a single nutrient chemo-effecter.
- When a group of E. coli cells is placed in the center of a semisolid agar with a single nutrient chemo-effector (sensor), they move out from the center in a traveling ring of cells by moving up the nutrient gradient created by consumption of the nutrient by the group.
- cells when stimulated by a high level of *succinate*, release an **attractant** *aspartate*, which helps them to aggregate into groups and thus move as concentric patterns of swarms with **high bacterial density** (Chemically Stimulated)

Swarming - Algorithmic Perspective

- **Passino** et al, in their original paper presented a way to implement this on the algo.

The cell-to-cell signaling effect , $J_{CC}(\theta)$,

$\Theta = [\theta_1, \theta_2, \theta_3 \; .. \; \theta_P]^T$

$J_{CC}(\theta) = J_{CC}(\theta, P(j,k,l)) = \sum_{i} J_{CC}(\theta, \theta^i(j,k,l))$ for $i = 1,2,3,.....,S$

$= \sum [-d_{attractant} \exp(-w_{attractant} \sum (\theta_m - \theta^i_m)^2)] + \sum [h_{repellent} \exp(-w_{repellent} \sum (\theta_m - \theta^i_m)^2)]$

m= 1,2,3,......P dimension of the search space.

$J_{CC}(\theta, P(j,k,l))$ is added to the actual objective function $J(\theta)$ result in time-varying OF.

$d_{atractant}$, $a_{ttractant}$, $h_{repellant}$, $w_{repellant}$ are different coefficients that should be chosen properly.

Swarming - Algorithmic Perspective

- **Passino** et al, in their original paper presented a way to implement this on the algo.

$\Theta = [\theta_1, \theta_2, \theta_3\ .. \theta_P]^T$

The cell-to-cell signaling effect , $J_{CC}(\theta)$,

$J_{CC}(\theta) = J_{CC}(\theta, P\,(j,k,l)) = \sum_{i} J_{CC}(\theta, \theta^i\,(j,k,l))$ for i = 1,2,3,.....,S

$= \sum [\, -d_{attractant} \exp(-w_{attractant} \sum (\theta_m - \theta^i_m)^2)\,] + \sum [\, h_{repellent} \exp(-w_{repellent} \sum (\theta_m - \theta^i_m)^2)\,]$

Sometimes we use, $d_{attractant} = h_{repellent} = m_{ar}$

$\sum\ m_{ar} [- e^{-w \sum (\theta m - \theta i,m)2} + e^{-w \sum (\theta m - \theta i,m)2)}\,]$

S - Number of Bacteria

P - Dimension

Outer ∑ is from 1 - S , Inner ∑ is from 1 - P

Objective Function with Swarming

Oops!! Where is Environment

- **Passino** et al, in their original paper presented a way to implement this on the algo.

$\Theta = [\theta_1, \theta_2, \theta_3 .. \theta_P]^T$

The cell-to-cell signaling effect , $J_{CC}(\theta)$,

$J_{CC}(\theta) = J_{CC}(\theta, P\,(j,k,l)) = \sum_{i} J_{CC}(\theta, \theta^i\,(j, k, l))$ for $i = 1,2,3,\ldots\ldots,S$

$= \sum [\, -d_{attractant} \exp(-w_{attractant} \sum (\theta_m - \theta^i_m)^2)\,] + \sum [\, h_{repellent} \exp(-w_{repellent} \sum (\theta_m - \theta^i_m)^2)\,]$

$$J(\theta) = J(\theta) + J_{CC}(\theta)$$

Problem?

$$J(i,j,k,l) = J(i,j,k,l) + J_{CC}(\theta^i, P(j,k,l))$$

add on the cell-to cell attractant–repellent profile to simulate the swarming behaviors

Environment Dependent Swarming

- **Passino and Liu** et al, proposed an environment dependent cell-to-cell signalling mechanism in their next paper.

$\Theta = [\theta_1, \theta_2, \theta_3 .. \theta_P]^T$

The env. dependent cell-to-cell signaling effect , $J_{ar}(\theta)$,

$J_{CC}(\theta) = J_{CC}(\theta, P(j,k,l)) = \sum \mathbf{J_{CC}(\theta, \theta^i(j, k, l))}$ for $i = 1,2,3,.....,S$

$$J_{ar}(\theta) = \exp(M - J(\theta)) * \mathbf{J_{CC}(\theta)}$$

Problem?

$$J(\theta) = J(\theta) + \mathbf{J_{ar}(\theta)} \quad , \quad J(i,j,k,l) = J(i,j,k,l) + \mathbf{J_{CC}(\theta^i)}$$

Wondering what is M? **M is a tunable parameter!!!**

Reproduction

N_C : Chemotaxis steps

- For every chemotaxis steps, we find healthy bacteria to perform natural selection.
- $j < N_c$ perform chemotaxis step. If

$$J^{i}_{\text{Helath}} = \sum_{j}^{N_C+1} \mathbf{J(i, j, k, l)}$$

Lower the cost
higher the **Health**

Within a single lifetime, bacteria perform N_C chemotaxis steps, within a each chemotaxis step, it performs N_m (In best case) number of swim steps before tumbling.

The Sr bacteria with the highest J health values die and the remaining Sr bacteria with the best values split.S_r is usually taken as S/2,

$$S_r = S/2$$

Elimination-Dispersal

- ❑ After N_{re} reproduction steps, bacteria perform **elimination-dispersal**
- ❑ Some of the bacteria may get killed or displaced into different environment of the search space.
- ❑ For each bacteria i = 1, 2, 3, .. S , we find $\mathbf{P_{ed}}$ probability of dispersal.
- ❑ If a bacteria is killed, algorithm place a virtual bacteria on a random location of the search space.

BFOA

```
Initialize parameters;
Initialize the location of the population;
for $l = 1$ to $N_{ed}$ do
    for $k = 1$ to $N_{re}$ do
        for $j = 1$ to $N_c$ do
            for $i = 1$ to $S$ do
            Calculate
            $J = J(i, j,k,l) + Jcc(\theta,P(j,k,l))$;
            Generate a random direction $\Delta(i)$;
            $J_{last} = J$;
            Move using (1);
            $\theta_i(j+1,k,l) = \theta_i(j,k,l)+C(i)\varphi(i)$ and $m = 0$;
            while $m < N_s$ do
                $m = m+1$;
                Calculate a new
                $J = J(i, j,k,l) + Jcc(\theta,P(j,k,l))$;
                if $J_{last} > J$ then
                Move using (1);
                $\theta_i(j+1,k,l) = \theta_i(j,k,l)+C(i)\varphi(i)$
                Else **$m = Ns$**
        Split the best bacteria, eliminate the worst;
    Disperse at random some bacteria
```

Simplifying BFOA

Please Sir, I want some more?

- **Halgamuge** et al. proposed Simplifying the Bacteria Foraging Optimization Algorithm in 2010.
- The algorithm is structurally complex due to its "**nested loop architecture**"
- Also, algorithm, includes several parameters whose selection deeply influences the result.
- So can we simplify the algorithm structure?
- modifications for the BFOA to improve its computation speed and convergence,

Can we simplify the classical BFOA?

Simplifying BFOA

- **Step-Size C(i),** has the most significant impact over stability and convergence of the algorithm.
- The second issue is to develop an adaptation scheme for C(i) which could improve the search and convergence.
- Algorithm structure was changed
- Also, As adaptation technique, we use a modification the **1/5-th rule** extracted from the Evolution Strategies (ES).
- More like BFOA+ES+PSO
- Oops!

Analysis of Swarming

$J_{CC}(\theta) = J_{CC}(\theta, P(j,k,l)) = \sum J_{CC}(\theta, \theta^i (j,k,l))$ for $i = 1,2,3,\ldots..,S$

$= \sum [-d_{attractant} \exp(-w_{attractant} \sum (\theta_m - \theta^i_m)^2)] + \sum [h_{repellent} \exp(-w_{repellent} \sum (\theta_m - \theta^i_m)^2)]$

$= \sum [-d_{attractant} \exp(-w_{attractant} ||\theta - \theta^i||_2{}^2)] + \sum [h_{repellent} \exp(-w_{repellent} ||\theta - \theta^i||_2{}^2)]$

Also we take, $d_{attractant} = h_{repellent} = d$

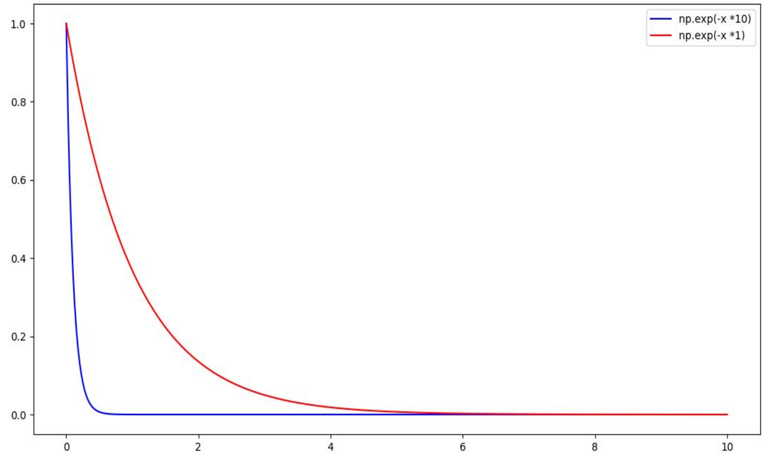
$= d_{attractant} \sum \Big[-\exp(-w_{attractant} ||\theta - \theta^i||_2{}^2) + \exp(-w_{repellent} ||\theta - \theta^i||_2{}^2) \Big]$, **Let** $y = ||\theta - \theta^i||_2{}^2$

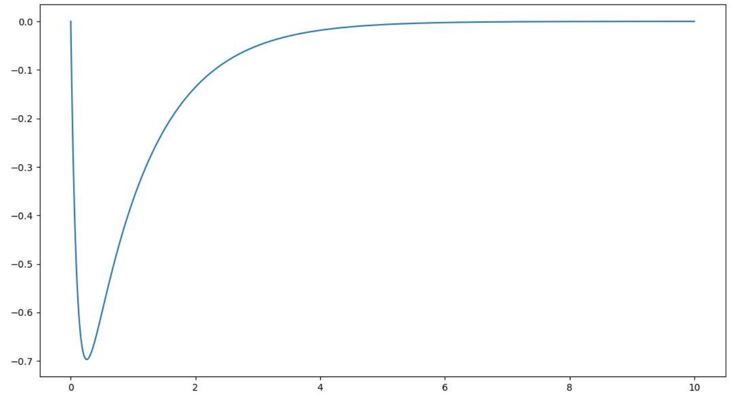
$= d_{attractant} \sum \Big[-\exp(-w_{attractant} y) + \exp(-w_{repellent}\, y) \Big]$, **AND** $w_{repellent} > w_{attractant}$

Analysis of Swarming

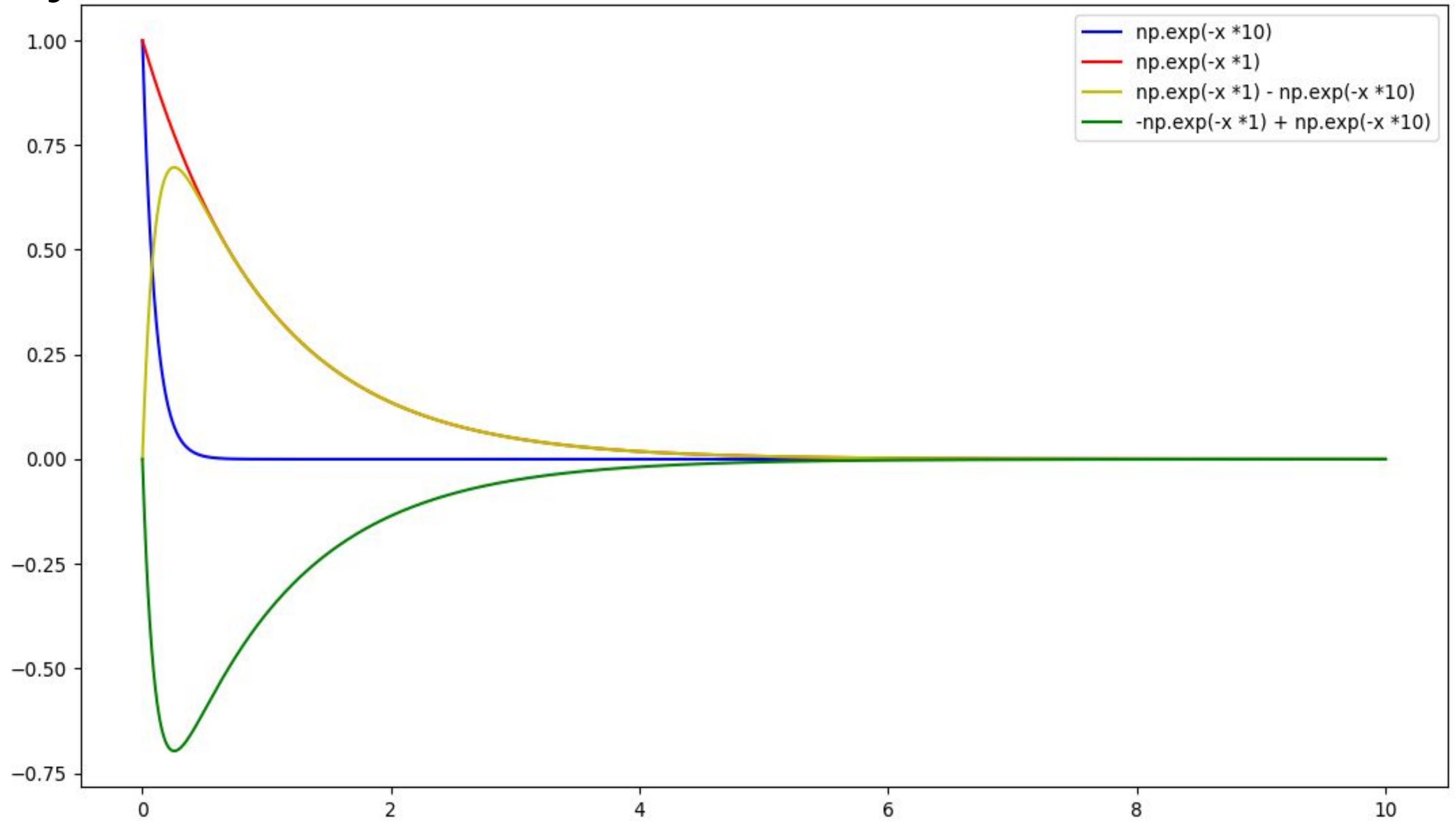
$$= d_{attractant} \sum \left[-\exp(-w_{attractant} \|\theta - \theta^i\|_2^2) + \exp(-w_{repellent} \|\theta - \theta^i\|_2^2) \right], \text{ Let } y = \|\theta - \theta^i\|_2^2$$

$$= -d_{attractant} \sum \left[\exp(-w_{attractant} y) - \exp(-w_{repellent} y) \right], \text{ AND } w_{repellent} > w_{attractant}$$





Analysis



np.exp(-x *10)
np.exp(-x *1)
np.exp(-x *1) - np.exp(-x *10)
-np.exp(-x *1) + np.exp(-x *10)
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
0
2
4
6
8
10

Analysis of Environmental Swarming

The env. dependent cell-to-cell signaling effect , $J_{ar}(\theta)$,

$J_{CC}(\theta) = J_{CC}(\theta, P\,(j,k,l)) = \sum \mathbf{J_{CC}(\theta, \theta^{i}\,(\,j,\,k,\,l\,))}$ for $i = 1,2,3,\ldots..,S$

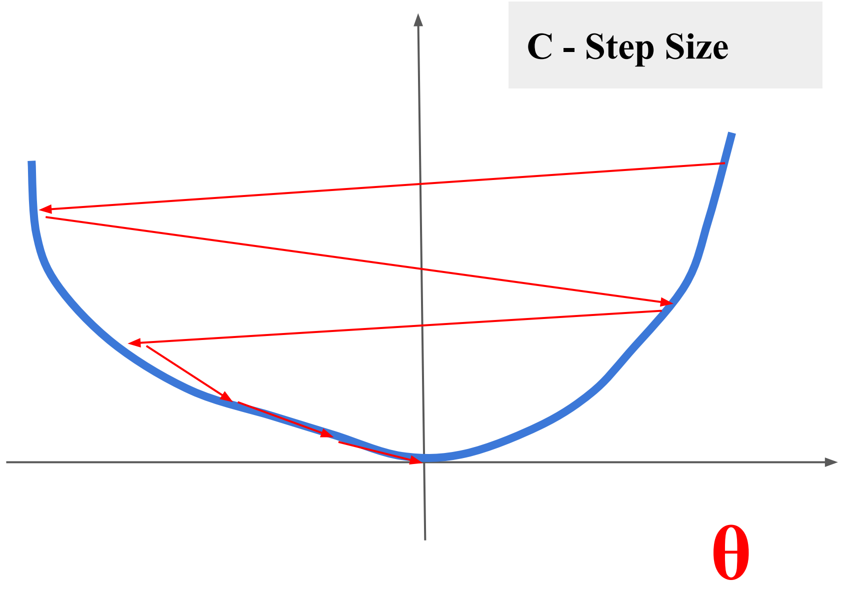
$$J_{ar}(\theta) = \mathbf{exp\,(M\text{-}J(\theta))\, *\, J_{CC}(\theta)}$$

Analysis of Chemotaxis

$$C(i) = |J(\theta)| / (|J(\theta)| + \lambda)$$

λ : constant parameter

$$C(i) = \frac{1}{\left(1 + \frac{\lambda}{|J(\theta)|}\right)}$$



Convergence of C does not reach 0 and stuck in some unknown point. May be depends on the λ VALUE

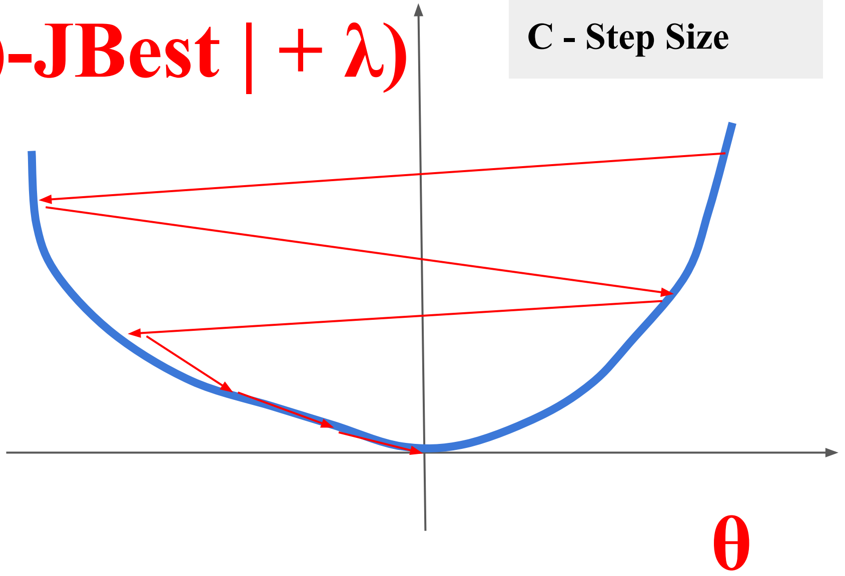
Analysis of Chemotaxis

$$C(i) = |J(\theta)-JBest| / (|J(\theta)-JBest| + \lambda)$$

C - Step Size

λ : constant parameter

$$C(i) = \frac{1}{\left(1 + \dfrac{\lambda}{|J(\theta)-JBest|}\right)}$$



In large wide spaces, this can be observed clearly

Mathematical Model-Adaptive Chemotaxis

- Abraham derive it for a 2D and 3D space with step functions

Improvements

- Scale invariant adaptive mechanism/ Adaptive Techniques
- Advancements in Reproduction (especially when multi-swarming)
- Swarming Revised to Scale-Fitness invariant
- Add Progressive measurements to individual bacteria (especially when unstructured reproduction)
- Multi-Swarming and clustering
- Multi-populations

Improved BFOA

- **By Jun Li**, Analysis and Improvement of the Bacterial Foraging Optimization Algorithm
- Used,C. Ying, M. Hua, J. Zhen, and W. Qinghua, “Fast bacterial swarming algorithm based on particle swarm optimization,” Journal of Data Acquisition and Processing, no. 4, pp. 442-448, 2010.

Q = 1 – (2ged*L) where ged = generation DIV 20 then Sed = S*Q

Q : Percentage of elimination and dispersal of bacteria,

L: the percentage of the initial elimination-dispersal

Also, **g= generation DIV 10** and **STEP=STEP/2g**

Multi-BFOA for dynamic environment

- **Daas** et al., Multi-Bacterial Foraging Optimization for Dynamic Environments

After each generation/reproduction, Let i,j be populations

Distance measurements: $D_{i,j} = \mathrm{sqrt}\left(\sum \| \mathbf{GBest_i - Gbest_j} \|_2^2 \right)$

If $(D_{i,j} < R)$: Reevaluate

Also, When ENV_CHANGED : Reevaluate

Multi-Swarming, Clustering and Multi-Populations

- ❏ **Branke** et al., A Multi-Population Approach to Dynamic Optimization Problems
- ❏ **Branke** et al. proposed, Multiswarms, Exclusion, and Anti-Convergence in Dynamic Environments
- ❏ **Li** et al., A Clustering Particle Swarm Optimizer for Dynamic Optimization.
- ❏ **Li** et al., Maintaining Diversity by Clustering in Dynamic Environments
- ❏ **Iman** et al. Particle Swarm Optimization Algorithm in Dynamic Environments: Adapting Inertia Weight and Clustering Particles.

Multi-Bacterial Foraging Optimization for Dynamic Environments by Daas

BFOA Vs SGD

- It never ends searching nutrients (No **Saddle** points)
- Even though the bacterium has some food, it persistently searches for more.
- Specially, In E-Coli behaviour it never stop **swimming** and **tumbling**.
- When encountering an nutrient gradient, bacterium will spend more time swimming and less time tumbling.
- It moves fast toward increasing nutrient concentrations
- Baseline Behaviour : move with same proportion of **swimming** and **tumbling**

MuNichBFor

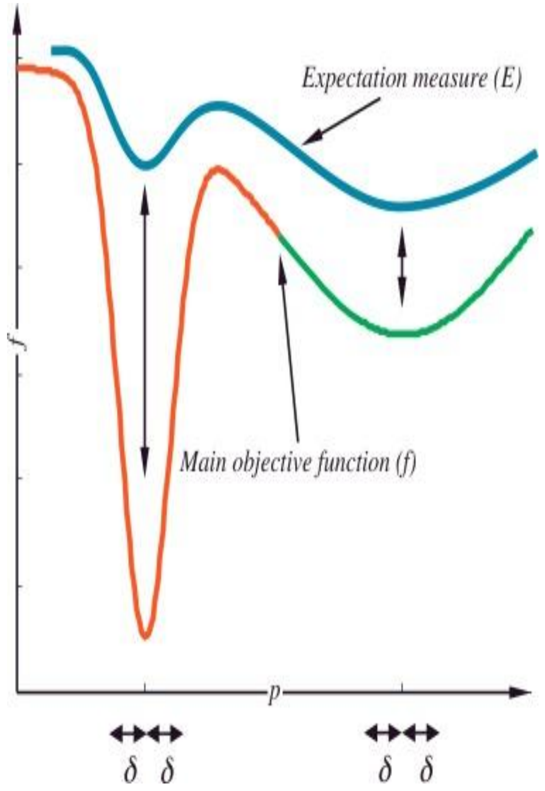
Multi **Nich**e **B**acterial **For**aging Optimization Algorithm for Multi-modal Functions

Mahesh Dananjaya

(locate and maintain multiple optima in a multimodal landscape)

Multi-Niching Algorithms and Multi-Modal Functions

- **S.W. Mahfoud**, Niching methods for genetic algorithms IlliGAL Rep. 95001, Urbana, University of Illinois, Illinois Genetic Algorithm Lab, 1995.
- **D.E. Goldberg, K. Deb, J. Horn**, “Massive Multimodality, Deception and Genetic algorithms”, in R. Manner, B.Manderick (Eds.) Parallel Problem Solving from Nature 2, pp. 37-46, North Holland, 1992.
- **K. Deb and D. E. Goldberg**, “An investigation of niche and species formation in genetic function optimization,” in Proc. 3rd Int. Conf. Genetic Algorithms, J. D. Schaffer, Ed. San Mateo, CA: Morgan Kaufmann, 1989, pp. 42–50.



Multi-Niching Algorithms and Multi-Modal Functions

- ❑ Metaheuristic optimization algorithms are originally designed to find the global maxima of the objective function, e.g. Genetic Algorithms, ES, PSO.
- ❑ As an example, simple GA cannot maintain the controlled competition among the competing schemata corresponding to different peaks.
- ❑ Also, the stochastic error associated with the genetic operators, specially selection, causes the population to converge to one alternative or other,
- ❑ In genetic algorithms, this problem with finite populations is known as the “”genetic drift”.
- ❑ All the original evolutionary algorithms converge to the best peak.
- ❑ Sometimes, we might be interested not only in the best solutions (global optimum), but also about other optima.
- ❑ **In BFOA, due to reproduction, there is a information loss**

Preselection and Crowding Algorithms

- ❑ **Cavicchio D . J** , Adaptive search using simulated evolution Unpublished doctoral dissertation University of Michigan Ann Arbor
- ❑ **K.A. de Jong**, An Analysis of the Behavior of a Class of Genetic Adaptive Systems, PhD thesis, Dept. of Computer and Communication Sciences, University of Michigan, 1975. (Original Crowding)
- ❑ **S.W.Mahfoud**, 'Crowding and preselection revisited', Proc. 2nd Conf. Parallel Problem Solving from Nature, pp.27-36, 1992. (***Deterministic Crowding***)
- ❑ **W. Cedeño**, “The multi-niche crowding genetic algorithm: analysis and applications,” PhD Thesis, University of California Davis, 1995.
- ❑ **K. Deb and D. E. Goldberg**, “An investigation of niche and species formation in genetic function optimization,” in Proc. 3rd Int. Conf. Genetic Algorithms, J. D. Schaffer, Ed. San Mateo, CA: Morgan Kaufmann, 1989, pp. 42–50.

Crowding Algorithms

- ❑ Niches are created by replacing the existing individuals according to their similarity with other individuals of the populations.
- ❑ Generation Gap (G) and Crowding Factor (CF) is used for the algorithm.
- ❑ **Generation Gap (G)** : only proportion G of the population is permitted to reproduce n the each generation.
- ❑ When selecting an individual to die, **CF (Crowding Factor)** individuals are picked at random from the population.
- ❑ Then, the one which is most similar to the new individual is chosen to be replaced.
- ❑ Here, the similarity is defined in terms of number of matching alleles.
- ❑ Then this new individual then replaces the this chosen individual.

Deterministic Crowding

- ❑ For replacement,instead of using De Jong's crowding factor (CF), deterministic crowding compares the new offspring directly to their parents, and replaces the parents only if the children have higher fitness.
- ❑ This algorithm does not use De Jong’s **Crowding Factor (CF)**.
- ❑ Thus, similarity between new one and old ones are used, particularly, the most similar child-parent pair is used to determine the replacement comparisons.
- ❑ This creates a selection pressure for the algorithms.
- ❑ Therefore, maintain multiple niches.
- ❑ But, Crowding techniques are vulnerable to massive deceptions in the objective function landscapes.

Sharing Algorithms

- **David E.Goldberg and John Richardson**, 'Genetic algorithms with sharing for multimodal function optimization', Proc. 2nd Inter. Conf. Genetic Algorithms, pp.41-49., 1987
- **J.E. Baker**, “Reducing Bias and Inefficiency in the Selection Algorithm”, in Genetic Algorithms and their Applications: Proc. of the 2nd Int. Conf on Genetic Algorithms, pp. 14-21, 1987
- Lies between fitness evaluation and selection -> effecting classical selection procedure
- **J. H. Holland**, Adaptation in Natural and Artificial Systems. Ann Arbor, MI: Univ. of Michigan Press, 1975.

Sharing Algorithms (Goldberg & Richardson)

- ❑ Another mechanism for maintaining population diversity is sharing, rst introduced by Holland (1975) and improved by Goldberg & Richardson (1987),
- ❑ SHARING is applied in between individual fitness evaluation and selection, thus influence the selection for divergence.
- ❑ **The premise behind sharing is to reduce the fitness of individuals that have highly similar members within the population.**
- ❑ This **rewards** individuals that uniquely exploit areas of the domain, while discouraging redundant (highly similar) individuals in a domain.
- ❑ This provided “population diversity pressure” which helps to maintain multi-niches of the population.
- ❑ This method calculate “similarity matrix” of the individuals in the population.
- ❑ This reduce the “**Selection Pressure**”

Sharing Algorithms (Goldberg & Richardson)

- Sharing Fitness Value

$$f'_i = f_i/m_i$$, f_i is the fitness value of ith individual

- m_i is the niche count of ith individual

$$m_i = \sum^{N} sh(d_{i,j})$$, N is population size

- Sharing Function sh(x) is given by,

$$sh(d_{i,j}) = \begin{cases} 1 - (d_{i,j}/\sigma_{sh})^{\alpha_{sh}} & \text{if } d_{i,j} < \sigma_{sh} \\ 0 & \text{otherwise} \end{cases}$$

$d_{i,j}$: distance between i and j, usually Euclidean Distance

$$d_{i,j} = ||\theta_i - \theta_j||_2$$

So Complexity is $O(N^2)$

Limitations

- Setting the dissimilarity *threshold* **requires a priori knowledge** of how far apart the optima are. For real optimization problems, however, no information about the search space and the distance between the optima is generally available.
- On the other hand, is the same for all individuals. This supposes that all peaks must be nearly equidistant in the domain. For these reasons, sharing can fail to maintain all desired peaks if they are not equidistant or if the estimated distance between two peaks is incorrect.
- Various empirical formulas have been proposed to set the dissimilarity threshold but this problem remains the major flaw of the method.The sharing scheme is very expensive as a result of the computation of niche counts of complexity $O(N^2)$ per generation
- Also, Algorithm is vulnerable to to massive ***deceptions*** in the multimodal doamin

Variants of Sharing

- **Miller, B.L. & Shaw**, M.J.: Genetic Algorithms with Dynamic Niche Sharing for Multimodal Function Optimization, IEEE International Conference on Evolutionary Computation, pp 786-791, Piscataway, NJ: IEEE Press 1995
- Adaptive schemes have been used.

Selection Algorithms

- **J.E. Baker**, “Reducing Bias and Inefficiency in the Selection Algorithm”, in Genetic Algorithms and their Applications: Proc. of the 2nd Int. Conf on Genetic Algorithms, pp. 14-21, 1987 (SUS).
- Multiple selection have been used for the multi niching criteria.
  - Roulette Wheel Selection (RWS)
  - **Stochastic Universal Selection (SUS)**
  - Tournament Selection
  - Restricted Tournament Selection (Crowding Method)

Stochastic Universal Selection (SUS) is the most widely used selection algorithm.

Clearing Algorithms

- ❑ **A. Petrowski**, “A clearing procedure as a niching method for genetic algorithms,” in Proc. 1996 IEEE Int. Conf. Evolutionary Computation, Nagoya, Japan, 1996, pp. 798–803.
- ❑ Sometimes elitist version is used
- ❑ Parameters,
  - ❑ K (Kappa) : Capacity of the niche
  - ❑ σ_{sh} : Clearing Radius (to find dissimilarity)

Clearing Algorithms

Complexity i O(qN) where q is the number of niche maintained

- ❑ The clearing method is very similar to fitness sharing but is based on the concept of limited resources of the environment (proposed by **Petrowski**)
- ❑ Instead of sharing the resources between all individuals of a single subpopulation as in fitness sharing, clearing attributes them only to the best members of the subpopulation
- ❑ In practice, the **k (kappa)** capacity of a niche specifies the maximum number of elements that this niche can accept. (**Mostly K=1**)
- ❑ hus, clearing preserves the fitness of the best individuals (**dominant individuals**) of the niche and resets the fitness of the others that belong to the same subpopulation (dominated individuals)
- ❑ Clearing can be coupled with elitism strategies to preserve the best elements of the niches during the generations.

Clustering Algorithms

- Clustering analysis and dynamic niching have been developed to reduce computational complexity and increase sharing effectiveness
- **Yin, X. & Germay**, N.: A Fast Genetic Algorithm with Sharing Scheme Using Cluster Analysis Methods in Multimodal Function Optimization, Proc. Inter. Conf. Artificial Neural Nets and Genetic Algorithms, pp450-457, Innsbruck, Austria 1993.
- **A. Pétrowski**, An Efficient Hierarchical Clustering Technique for Speciation.
- **Justin Gan and Kevin Warwick**, A Genetic Algorithm with Dynamic Niche Clustering for Multimodal Function Optimisation.
- There are two types; Hierarchical and Non-Hierarchical

Clustering Algorithms

- **K** is the number of clusters which is either a priori demand or adaptive parameters.
- There are two main steps.
  - Process of representation: a cluster C c is represented by its centroid G(CJ) :

$$\mathbf{G(C_c) = (\bar{x}_{c1}, \ \bar{x}_{c2},..., \bar{x}_{cj},.... \bar{x}_{cp}) \text{ where } c = 1,2,...'k \ (1)}$$

$$\mathbf{\text{With } \bar{x}_{cj} = \sum x_{ij} / n_c}$$

p : number of variables in a data unit, k : number of clusters, n_c : number of data units in cth cluster, c_j : mean on the jth variable in the c^{th} cluster, X_{ij} : value on the jth variable for the ith data unit, X_i : vector for the ith data unit

Clustering Algorithms

- Second Step is.
  - Process of assignment : each data unit is assigned to the cluster with the nearest centroid

$$X_i \in C_C \text{ if } d(x_i, G(C_C)) = \min d(x_i, G(C_l)) \text{ for } l=1,2,3,\ldots k$$

The algorithm minimizes implicitly the partition error E, i.e. the total within group error sum of square

$$\mathbf{E} = \sum_{C=1}^{k} \sum_{i=1}^{n_c} \boxed{\sum_{j=1}^{p} (x_{i,j} - \bar{x}_{c,j})^2}$$

Euclidean Distance between cluster centroid and individual

K : Number of Clusters
Nc : Number of individuals in the cth cluster
P : Number of variables (dimension) of the problem

Adaptive Clustering Algorithms (No of Niches)

- Clustering analysis and dynamic niching have been developed to reduce com
- algorithm allows the number of clusters to vary during the initial assignment of the data units.
- On the other hand, it needs two supplementary parameters: minimal distance (coarsening parameter) d_{min} and **maximal distance** (refining parameter) d_{max} • The algorithm proceeds in the following steps
- initial value of k, take the first k data units as initial clusters of one member each. Compute all pairwise distance among these first k data units. If the smallest distance is less than d_{min} , then merge the two associated clusters and compute the distance between the centroid of the new cluster and all remaining clusters.
- Continue merging nearest clusters as necessary until all centroids are separated by a distance at least as large as d min

Adaptive Clustering Algorithms (No of Niches)

- Assign each of the remaining N-k data units to the cluster with the nearest centroid. After each assignment, update the centroid of the gaining cluster and compute the distance to the centroids of the other clusters. Merge the new cluster with the cluster having the nearest centroid if the distance between centroids is less than d_{min} and continue merging as necessary until all centroids are at least d_{min} distance apart. If the distance to the nearest centroid is greater than d max , then take the data unit as a new cluster of one member
- After data units k+ 1 through N have been assigned, take the existing cluster centroids as fixed seed points and reallocate each data unit to its nearest seed points.

Sharing with Clustering

- Sharing Fitness Value

$$f'_i = f_i/m_i$$, f_i is the fitness value of ith individual

- m_i is the niche count of ith individual

$$m_i = n_c - n_c*(d_{ic}/2*d_{max})^{\alpha}$$

when $X_i \in C_C$

$d_{i,j}$: distance between i and j, usually Euclidean Distance

$$d_{i,c} = ||\theta_i - C_c||_2$$

So Complexity is O(2kN)

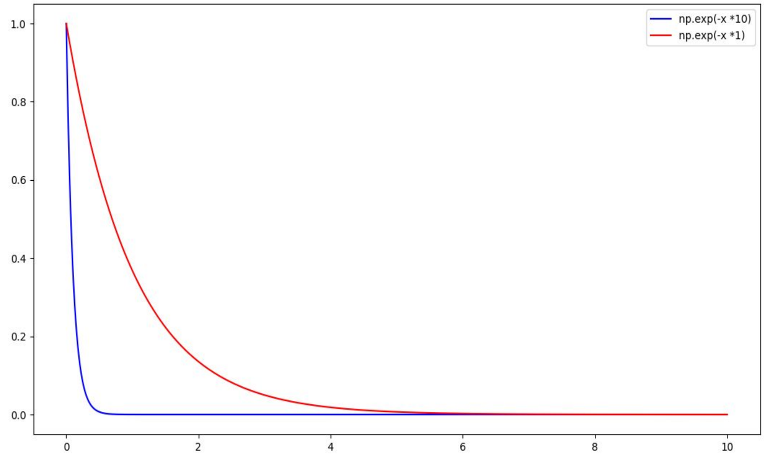
Variants

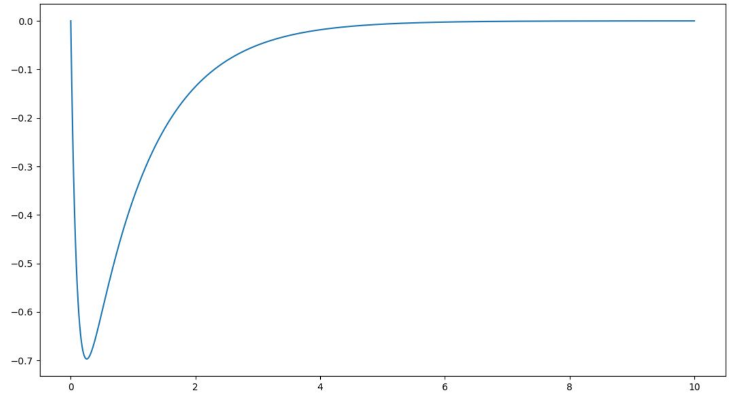
- Rather calculating from beginning, tag each individual and make niches.

Analysis of Swarming

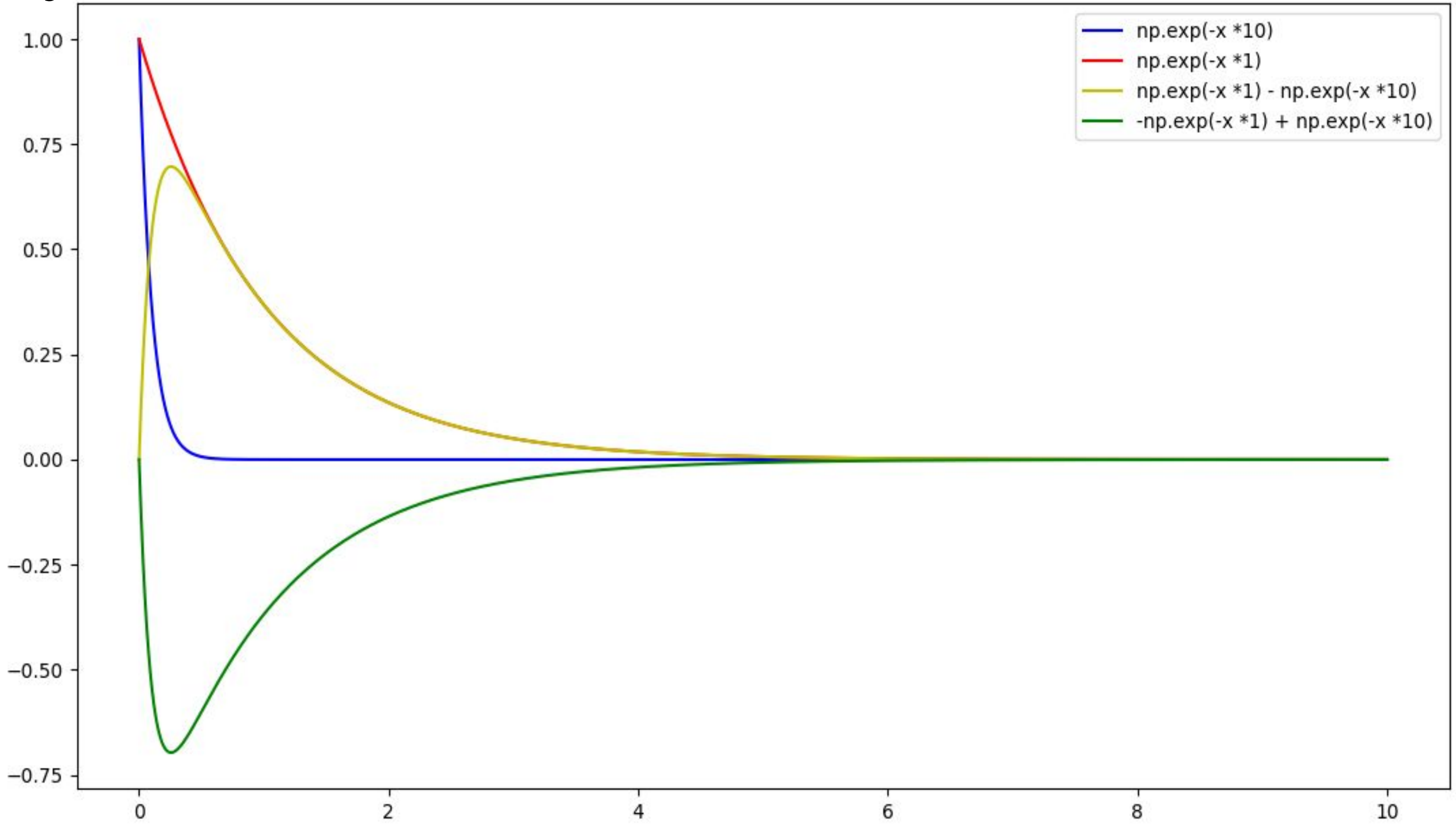
$$= d_{attractant} \sum \left[-\exp(-w_{attractant} \|\theta - \theta^i\|_2^2) + \exp(-w_{repellent} \|\theta - \theta^i\|_2^2) \right], \text{ Let } y = \|\theta - \theta^i\|_2^2$$

$$= -d_{attractant} \sum \left[\exp(-w_{attractant} y) - \exp(-w_{repellent} y) \right], \text{ AND } w_{repellent} > w_{attractant}$$





Analysis



1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
0
2
4
6
8
10
np.exp(-x *10)
np.exp(-x *1)
np.exp(-x *1) - np.exp(-x *10)
-np.exp(-x *1) + np.exp(-x *10)

Chi-Square Performance Criteria

- ❑ Formulating resource sharing problem of evolutionary algorithm, BFOA, as multi-arm bandit problem.
- ❑ Then get the quantitative and qualitative results of the multi-niche convergence of the population.
- ❑

Revenue Optimization

- ❑ Original BFO algorithm has been applied to revenue optimization problem.
- ❑ We do not touch the core evolutionary algorithm.
- ❑ We modelled the fitness function to be aligned with the need of revenue optimization objective function.
- ❑ Specially using constraints and boundary conditions we try to optimize the results.
- ❑ A new hyper-parameters have been used to modelled and control the fitness function, so that algorithm could find the most optimal solution.

Problem Formulation

- We have basic parameters, Number of room types(R), Number of Booking Buckets (T), Number of Market Segments (M) and Number of pricing tags (lrm).
- Also we have a Room Capacity (Nr) for each room type.
- We also have derived parameters, Dimensionality of the searching problem (Dim)
  - Dim = (T+1)*M
- Since, room types formulate a independent problems we can use same algorithm iteratively with different parameters and available room capacity (Nr)
- Since we search through possible allocation/assignments, we use price tags to get the maximum profit over given assignment
- All R, T, M, lrm parameter space is stored as 4 dimensional tensor.
- Prmk is the pricing tag for each cel (P)l. Also, there is a Demand (D) for each cell.

Simulation

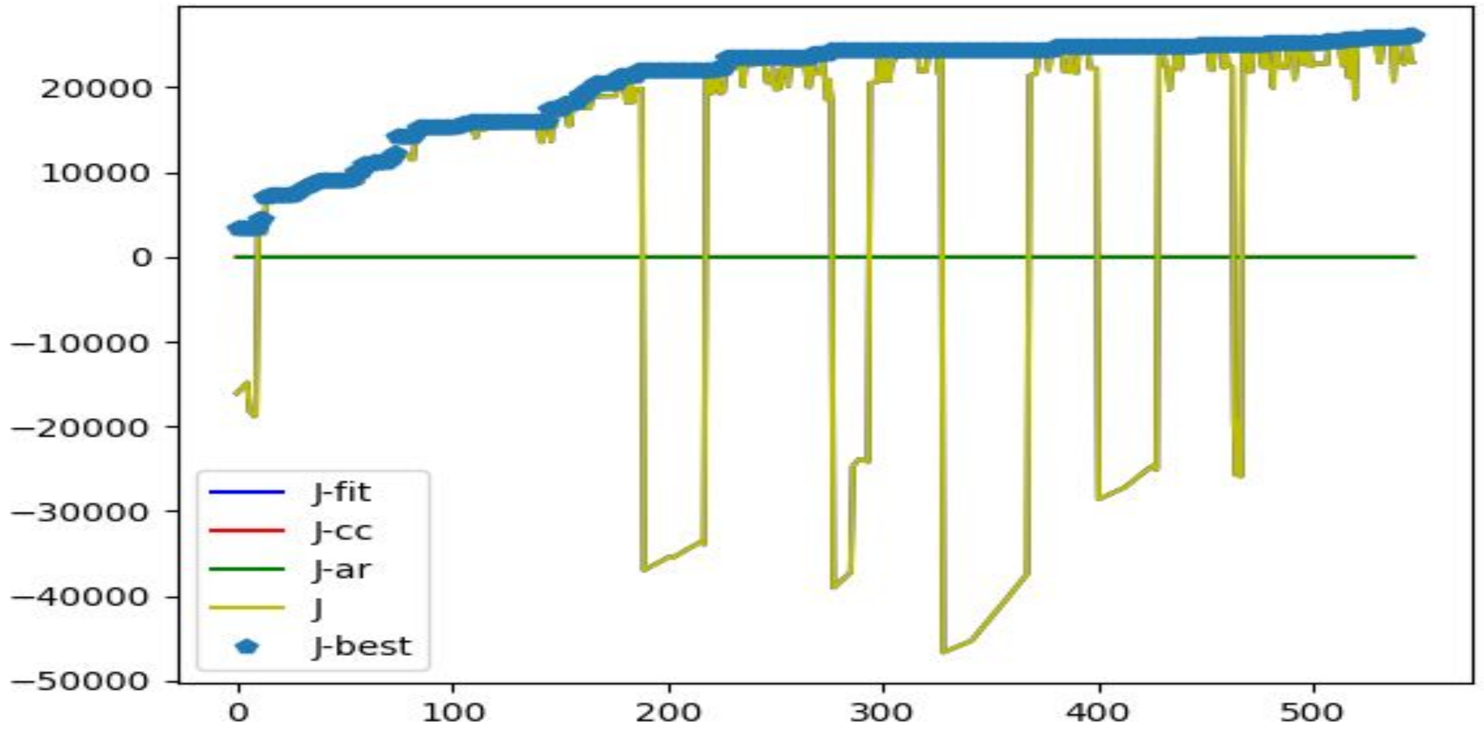
- Parameters

Dimension = (T+1) * M

```
M = 3
R = 2
lrm = 3
T = 2

Nr = [(i + 1) * 400 for i in range(R)]
```

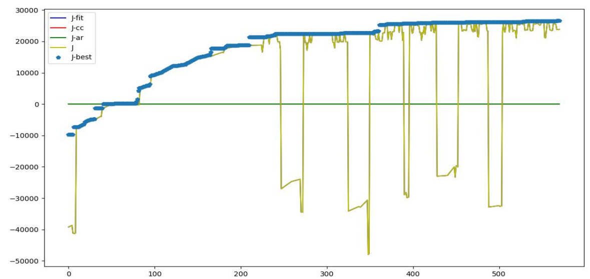
```
for r in range(R):
    for t in range(T):
        for m in range(M):
            for k in range(lrm):
                D[r][t][m][k] = 50 * (r + t + m + k)

for r in range(R):
    for t in range(T):
        for m in range(M):
            for k in range(lrm):
                P[r][t][m][k] = 5000 / D[r][t][m][k]
```

Simulation



20000
10000
0
−10000
−20000
−30000
−40000
−50000
0
100
200
300
400
500
J-fit
J-cc
J-ar
J
J-best

Simulation



Allocation **[49, 46, 62, 49, 74, 0, 62, 6, 50]**

Total Allocations: 398

Demands [50.0, 50.0, 100.0, 50.0, 100.0, 0, 100.0, 150.0, 200.0]

Profits [4900.0, 4600.0, 3100.0, 4900.0, 3700.0, 0, 3100.0, 200.0, 1250.0]

Tags [1, 0, 0, 0, 0, 0, 0, 0, 0]

Best Fitness (Profit) : 25750.0

Searched JFit Value 22998.6875

Processing Time
14.750471115112305 s (5-15 s average)

Memory in MB :
82.82421875(75-8 5MB average)