KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
227 Nguyễn Văn Cừ, Phường 4, Quận 5, TP.HCM
Điện Thoại: (08) 38.354.266 - Fax:(08) 38.350.096

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

Môn: Nhập môn học máy

BÁO CÁO ĐỒ ÁN THỰC HÀNH: POPULATION METHODS

GVHD: TS.Bùi Tiến Lên

Lớp: 19_21

Nhóm: 18

Thông tin thành viên:

Họ và tên	**MSSV**
Dương Thanh Hiệp	19120505
Nguyễn Nhật Huy	19120528
Nguyễn Tuấn Khanh	19120540
Hồ Công Lượng	19120572
Nguyễn Thanh Minh	19129587

MỤC LỤC

NỘI DUNG BÁO CÁO

1. Intialization:

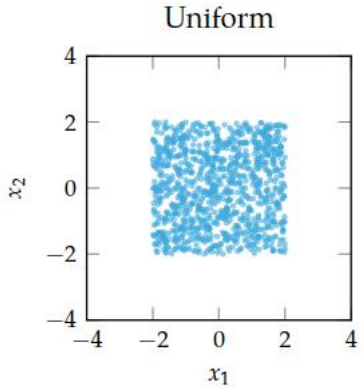
Phương pháp quần thể bắt đầu từ việc khởi tạo quần thể, cũng như phương pháp gradient descent yêu cầu khởi tạo các tham số (design point: $x^{(j)}$). Việc khởi tạo quần thể cần phải trải đều trên không gian mẫu để có thể có cơ hội các design point nằm gần best region (hàm mục tiêu là nhỏ nhất hay cực tiểu toàn cục). Ở đây sẽ trình bày một số phương pháp khởi tạo quần thể cơ bản dựa trên các hàm phân phối quen thuộc.

a.) Khởi tạo quần thể bằng phân phối đều (uniform distribution):

Là phương pháp khởi tạo với xác suất dựa trên phân phối đều, là phân phối mà tất cả các điểm đều có xác suất xuất hiện như nhau, với hàm phân phối xác suất.

$$f(x) = \begin{cases} \frac{1}{\prod_i (b_i - a_i)} \ nếu \ a_i \le x_i \le b_i \ trên \ mỗi \ chiều \\ 0 \end{cases}$$

Khởi tạo quần thể dựa vào phân phối đều sẽ có hình dạng như sau:

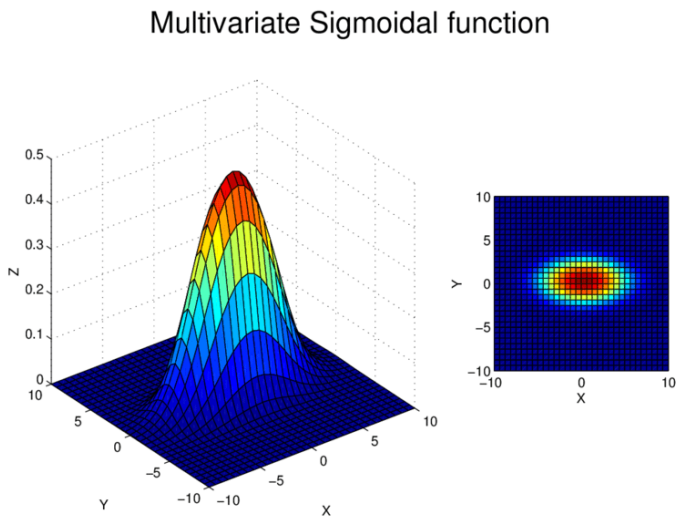


b.) Khởi tạo quần thể bằng phân phối chuẩn (gaussian distribution):

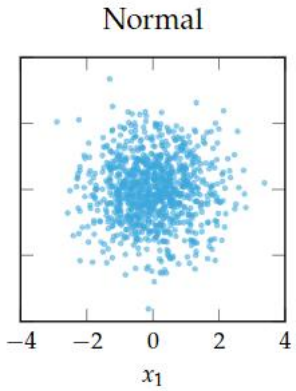
Là một phương pháp khởi tạo với xác suất design point các xa vùng chú ý (vùng có thể mang nghiệm) là thấp hơn so với các giá trị gần tâm vùng chú ý với hàm phân phối xác suất là:

$$f(x) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{p/2}} e^{-(x-\mu)^T \Sigma^{-1}(x-\mu)/2}$$

Với Σ: là ma trận hiệp phương sai, p: là số chiều của dữ liệu, μ: kỳ vọng.



Dưới đây là hình minh họa khi khởi tạo quần thể bằng phân phối chuẩn

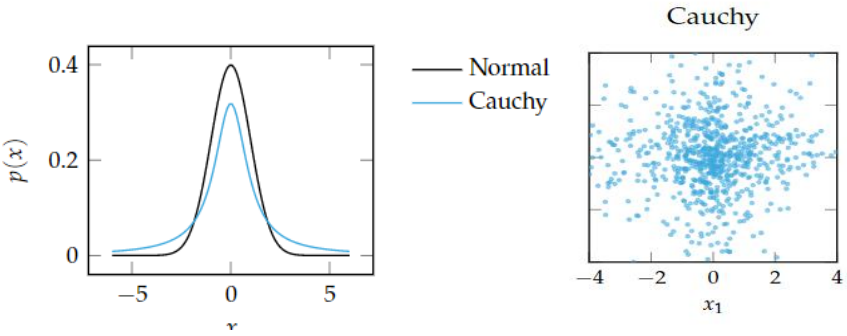


c.) Khởi tạo quần thể bằng phân phối cauchy (cauchy distribution):

Phương pháp sử dụng hàm phân phối cauchy với hàm phân phối tương đương với phân phối student có bậc tự do là 1:

$$f(\mathbf{x};\mu,\mathbf{\Sigma},k)=\frac{\Gamma\left(\frac{1+k}{2}\right)}{\Gamma(\frac{1}{2})\pi^{\frac{k}{2}}|\mathbf{\Sigma}|^{\frac{1}{2}}\left[1+(\mathbf{x}-\mu)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)\right]^{\frac{1+k}{2}}}.$$

Phân phối cauchy cũng thể hiện những điểm gần trung tâm sẽ có xác suất xuất hiện cao hơn những điểm xa trung tâm.



2. Genetic Algorithms:

Lấy cảm hứng từ tiến hóa sinh học, ở đó các cá thể khỏe mạnh sẽ có khả năng cao hơn trong việc truyền gen của mình cho thế hệ tiếp theo.

Khả năng của một cá thể được chọn cho việc truyền gen tỉ lệ nghịch với giá trị của hàm mục tiêu (có nghĩa là giá trị hàm mục tiêu càng nhỏ thì khả năng được chọn của cá thể đó càng lớn).

Các cá thể được chọn cho việc truyền gen sẽ trải qua các quá trình lai tạo (crossover) và đột biến (mutation).

a) Chromosome

- Cách đơn giản nhất để thể hiện là sử dụng các chuỗi nhiễm sắc thể nhị phân (binary string chromosome)

- Tuy nhiên, thỉnh thoảng các chuỗi nhị phân không thể thể hiện được các điểm hợp lệ. Thay vào đó ta sử dụng 1 vector các giá trị thực (real-valued chromosomes) , thường được tạo ra bằng cách random tuân thủ theo các phân phối đã trình bày ở mục 1.

b) Initialization

- Thuật toán di truyền bắt đầu bằng các quần thể được khởi tạo 1 cách ngẫu nhiên. Điều đó có nghĩa là quần thể sẽ bao gồm m chuỗi bit có cùng nhiều dài là n và được khởi tạo bằng câu lệnh:

```
rand_population_binary(m, n) = [bitrand(n) for i in 1:m]
```

c) Selection

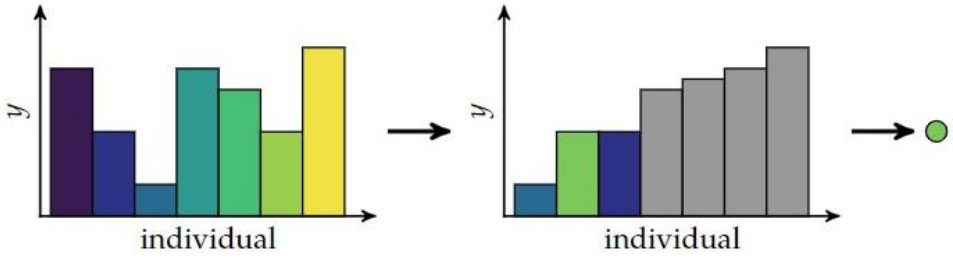
- Là quá trình chọn cá thể cho việc truyền gen.
- Với một quần thể gồm m nhiễm sắc thể thì việc chọn sẽ tạo ra m cặp cha mẹ cho việc

di truyền m cá thể con ở thế hệ tiếp theo. Các cặp được chọn có thể sẽ trùng nhau. Ngoài ra, chúng ta còn có thể chọn một nhóm các cá thể cho việc di truyền.

- Một vài cách tiếp cận cho việc chọn những cá thể phù hợp nhất là: Truncation selection, tournament selection, roulette wheel selection.

- Truncation selection (Lựa chọn cắt bớt):

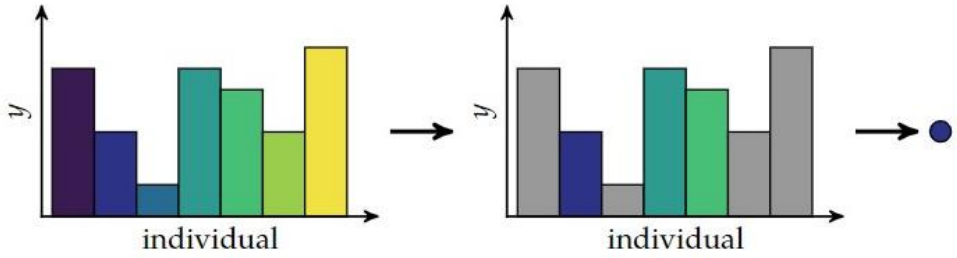
Chúng ta sẽ lựa chọn cha mẹ dựa trên k cá thể tốt nhất trong quần thể.



Ví dụ hình trên, quần thể gồm có 7 cá thể, ta sẽ sắp xếp các cá thể theo giá trị y rồi chọn ra 3 cá thể có giá trị hàm mục tiêu y nhỏ nhất. Sau đó sẽ tiến hành random để tạo ra các cặp cha mẹ ngẫu nhiên từ các cá thể vừa chọn.

- Tournament selection:

Ở cách tiếp cận này, các cặp cha mẹ sẽ là người phù hợp nhất trong số k cá thể được chọn ngẫu nhiên trong quần thể.

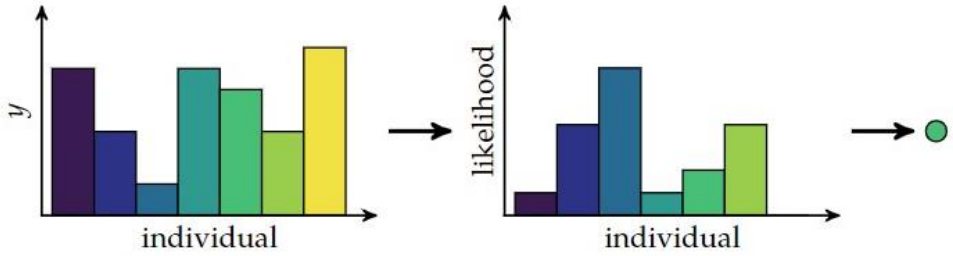


Ví dụ ở hình trên, quần thể gồm 7 cá thể, ta sẽ chọn 3 cá thể ngẫu nhiên rồi chạy một cách tách rời để chọn ra các cặp cha mẹ phù hợp nhất.

- Roulette wheel selection:

Một cặp cha mẹ sẽ được chọn với xác suất tỷ lệ thuận với hiệu suất của nó so với quần thể. Điều đó có nghĩa là các cá thể có giá trị hàm mục tiêu xấu nhất sẽ không có khả năng được chọn.

Cách tiếp cận là lấy giá trị hàm mục tiêu lớn nhất, sau đó trừ đi các giá trị hàm mục tiêu của các cá thể khác, ta sẽ có xác suất được chọn cho cá thể đó.

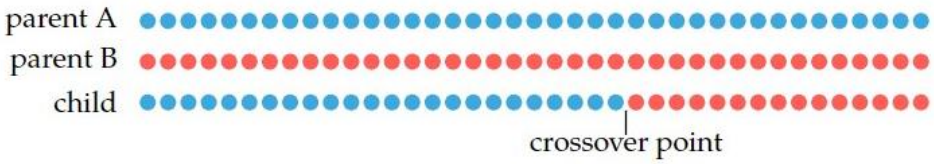


Ví dụ ở hình trên, quần thể gồm 7 cá thể, cá thể màu vàng có giá trị hàm mục tiêu xấu nhất nên xác suất nó được chọn là 0. Sau đó, ta sẽ chạy tách rời để chọn ra các cặp cha mẹ tốt nhất.

d) Crossover

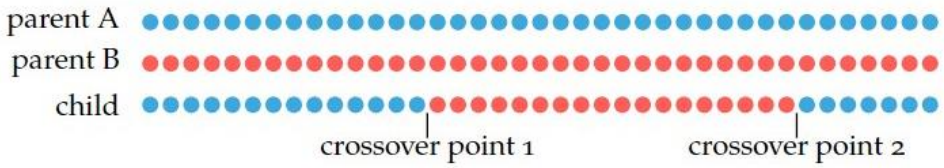
- Kết hợp các nhiễm sắc thể của cha mẹ để tạo thành cá thể con.
- Một vài quá trình lai tạo:
  - Single-point crossover:

Phần đầu tiên trong nhiễm sắc thể của bố A kết hợp với phần sau trong nhiễm sắc thể của mẹ B để tạo ra cá thể con. Điểm giao nhau (crossover point) sẽ được xác định bằng một cách ngẫu nhiên (uniform distribution).



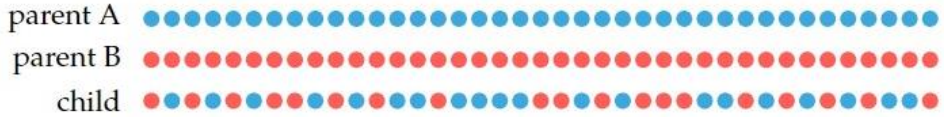
  - Two-point crossover:

Chúng ta sẽ tạo ra 2 điểm giao nhau bằng cách ngẫu nhiên. Phần đầu tiên đến điểm giao nhau 1 và từ điểm giao nhau 2 trở về sau mang nhiễm sắc thể của người bố A kết hợp với phần nhiễm sắc thể ở giữa 2 điểm giao nhau của người mẹ B để tạo ra cá thể con.



  - Uniform crossover:

Mỗi nhiễm sắc thể của cá thể con có thể mang nhiễm sắc thể của người bố hoặc mẹ tại vị trí đó với xác suất như nhau (50%).

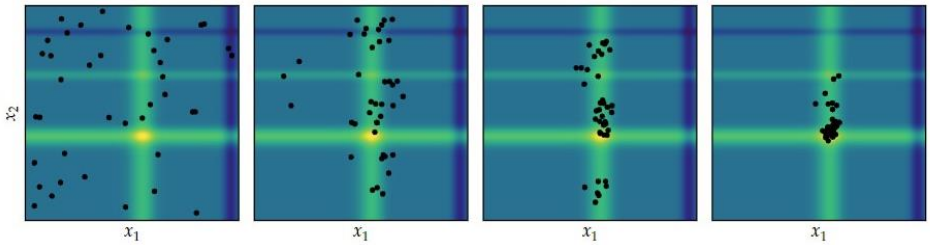


- Những phương pháp lai tạo này cũng có thể áp dụng cho các vector giá trị thực (real-valued chromosomes). Tuy nhiên, chúng ta có thể xác định một quy trình bổ sung lai tạo nội suy giữa các giá trị thực. Ở đây, các giá trị thực được nội suy tuyến tính giữa các giá trị x_a và x_b của cha mẹ theo công thức: $x <- (1 - \gamma)x_a + \gamma x_b$
,trong đó γ là một tham số vô hướng và được thiết lập bằng một nửa của nó.

e) Mutation:

- Nếu 1 cá thể mới được tạo ra chỉ dựa vào quá trình lai tạo (crossover) thì những đặc điểm mà quần thể được khởi ban đầu chưa thể hiện được thì sẽ không bao giờ được thể hiện nữa, và các gen tốt nhất có thể sẽ bão hòa cả quần thể.
- Đột biến cho phép các đặc điểm mới có thể xuất hiện một cách tự nhiên, cho phép các thuật toán di truyền có thể khai phá được nhiều trạng thái hơn.
- Các nhiễm sắc thể của cá thể con sẽ trải qua quá trình đột biến sau khi được lai tạo.
- Mỗi bit trong nhiễm sắc thể nhị phân sẽ có xác suất bị đổi. Với nhiễm sắc thể có m bit thì tỉ lệ đột biến sẽ là 1/m.
- Trong nhiễm sắc thể giá trị thực thì có thể đột biến bằng cách sử dụng bitwise-flips.

⇒ Dưới đây là quá trình 1 thuật toán di truyền thực hiện, bao gồm các bước combine selection, crossover, mutation strategies:

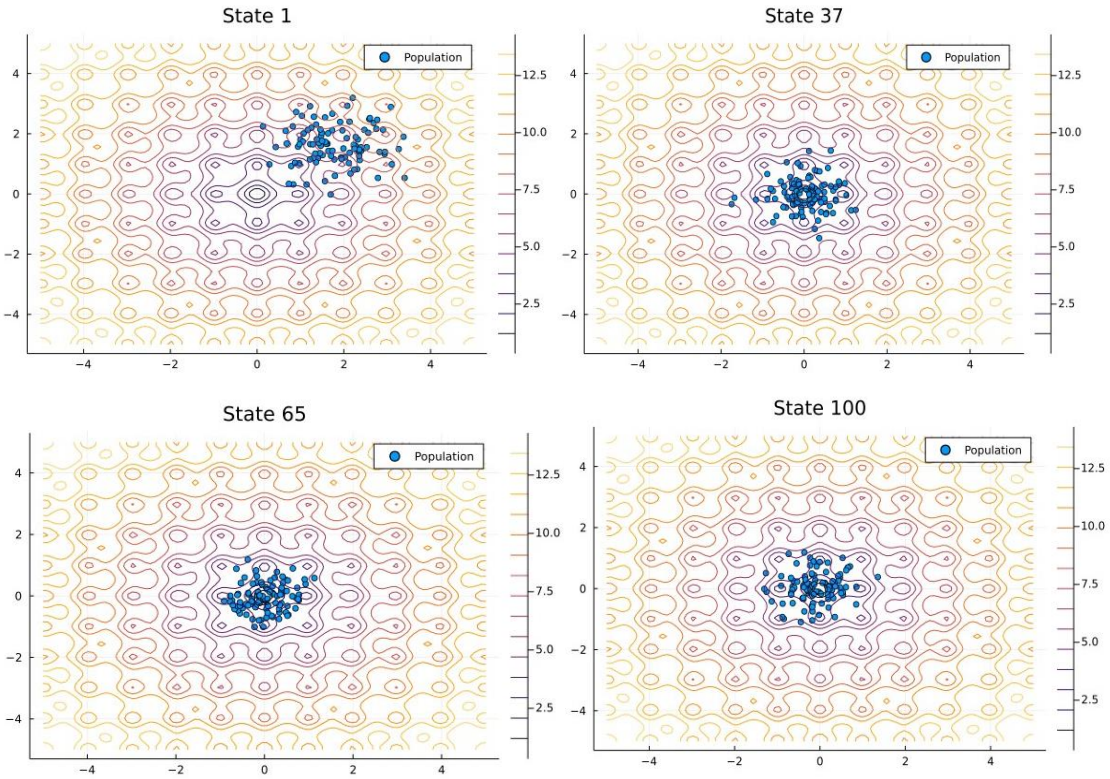


❖ **Ưu nhược điểm của thuật toán di truyền:**

- Ưu điểm:
  + Tìm được các giải pháp phù hợp trong thời gian ngắn.
  + Dễ dàng so sánh với các thuật toán khác trong cùng 1 mục đích.
- Nhược điểm:
  + Khó tiếp cận cho những người mới.
  + Nó có thể không tìm được giải pháp tối ưu nhất cho vấn đề trong một vài trường hợp.
  + Khó chọn một vài thông số như số các thế hệ, kích thước quần thể,...

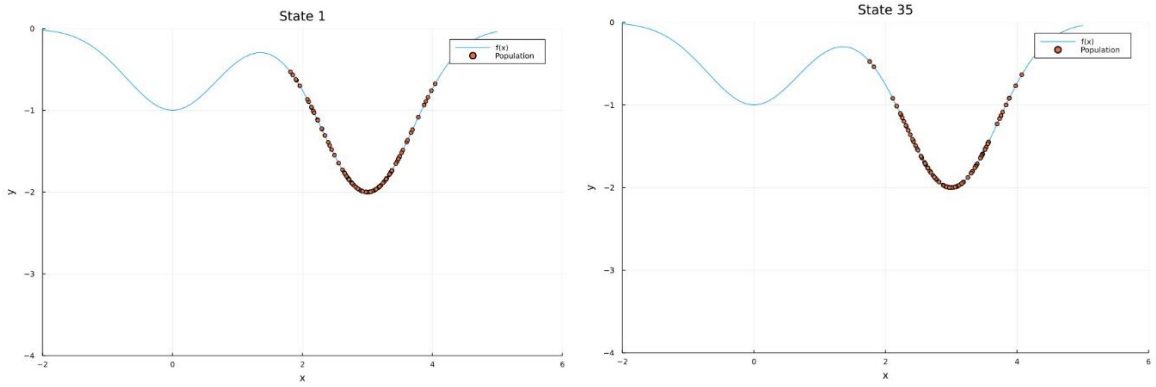
❖ **Kết quả sau khi chạy thuật toán di truyền với TruncationSelection, SinglePointCrossover, GaussianMutation và dựa vào hàm:**

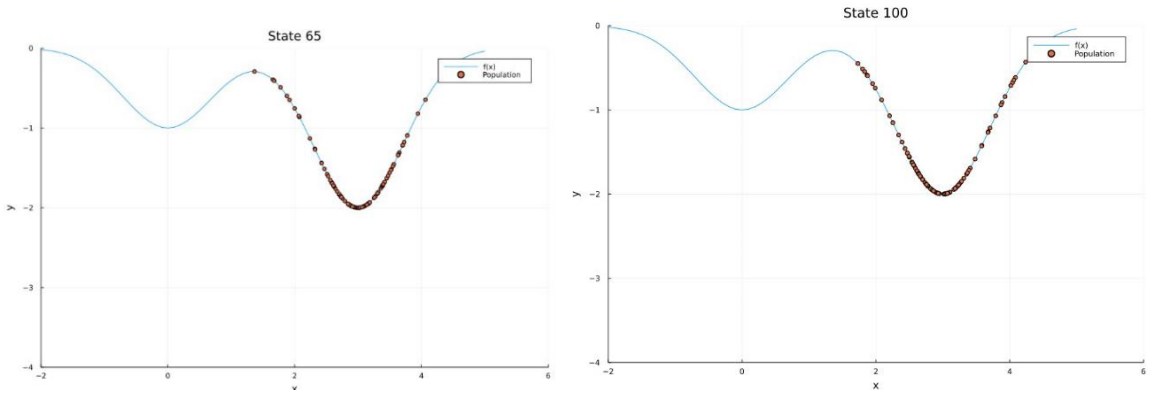
$$ackley(x) = -20e^{-0.4\sqrt{\frac{1}{2}\sum_{i=1}^{2} x_i^2}} - e^{-0.4\frac{1}{2}\sum_{i=1}^{2}\cos(2\pi x_i)} + 20 + e$$



→**Nhận xét**: Nhìn chung, thuật toán chạy khá tốt khi tạo ra những điểm gần với (0,0). Tuy nhiên, sau một vài lần chạy thì các điểm dữ liệu sẽ không còn khuynh hướng hội tụ vào best region nữa. Điều này dễ hiểu bởi vì những hàm lai tạo và đột biến sẽ không hoàn toàn luôn đem về kết quả tốt nhất.

❖ **Kết quả sau khi chạy thuật toán di truyền với TruncationSelection, SinglePointCrossover, GaussianMutation dựa vào hàm:** f(x) = $-e^{-x^2} - 2e^{-(x-3)^2}$





→**Nhận xét:** Thuật toán di truyền rất tốt khi tạo ra được những điểm rất gần với best region. Hàm đi qua cực tiểu toàn cục một cách ngẫu nhiên khi trải qua các bước lai tạo và đột biến.

3. Differential Evolution:

❖ **Định nghĩa**:

Different evolution là thuật toán tìm kiếm meta-heuristic trên quần thể giúp tối ưu hóa hàm chi phí bằng cách cải thiện từng cá thể trong quần thể dựa vào quá trình tiến hóa (kết hợp các cá thể riêng biệt trong quần thể với nhau).

❖ **Mã giả:**

1. Chọn 3 phần riêng biệt không trùng lặp a, b, c trong quần thể đã khởi tạo.
2. Tạo cá thể thay thế $z = a + w * (b - c)$
3. Chọn ngẫu nhiên $j \in [1, \dots, n]$ để xác định vị trí chắc chắn được thay thế
4. Xây dựng x' bằng binary crossover ở thuật toán generic

$$x_i' = \begin{cases} z_i \text{ nếu } i = j \text{ hay với xác suất } p \text{ cho trước} \\ x_i \end{cases}$$

5. Chọn giữa x và x' nếu phần tử nào cho kết quả tốt hơn.

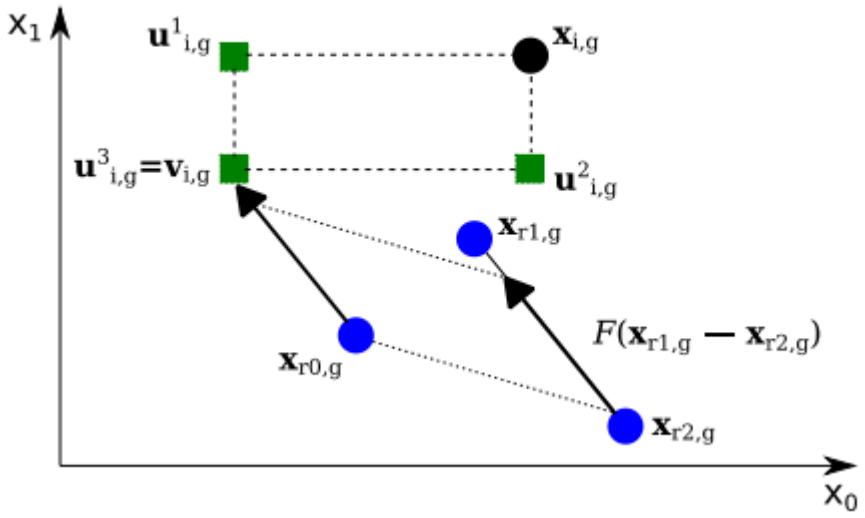
❖ **Ý tưởng**

Trước tiên chúng ta khởi tạo quần thể. Sau đó với số vòng lặp và và từng cá thể riêng biệt x trong quần thể chúng ta thực hiện:

- Mutation: tạo ra vector đột biến

Tạo cá thể đột biến bằng cách cộng vector khác biệt giữa hai cá thể được chọn ngẫu nhiên vào một cá thể thứ ba được chọn ngẫu nhiên khác với mong muốn tạo cá thể đột biến có xu hướng gần điểm cực tiểu (w: trọng số vi sai, thường từ 0.4 → 1).

z = a + w * (b - c)



- Recombination: tạo quần thể thử nghiệm

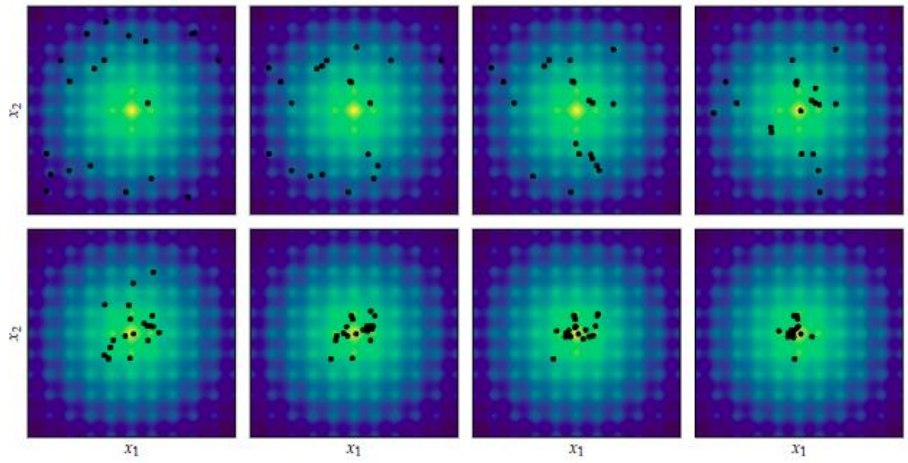
  Thực hiện lai ghép x với z bằng cách hoán đổi các thành phần dựa vào xác suất chéo hoặc xác suất ngẫu nhiên. Từ đó ta được cá thể mới là x'.

- Selection: chọn lọc quần thể thế hệ tiếp theo

  Đánh giá và thay thế cá thể tốt hơn vào quần thể.

❖ **Nhận xét:**

- Với ý tưởng tạo ra quần thể thế hệ tiếp theo tốt hơn hoặc bằng thế hệ trước, các cá thể sẽ có xu hướng gần hơn tới các điểm cực tiểu, và việc lai ghép vector đột biến cá thể cũ ngẫu nhiên sẽ tạo ra cá thể có thể leo đồi để thoát khỏi cực tiểu địa phương.
- Việc tạo cá thể mới phụ thuộc lớn vào cách lai ghép chúng ta chọn, độ lớn trọng số vi sai, xác suất lai ghép và cũng như số quần thể ta tạo ra.

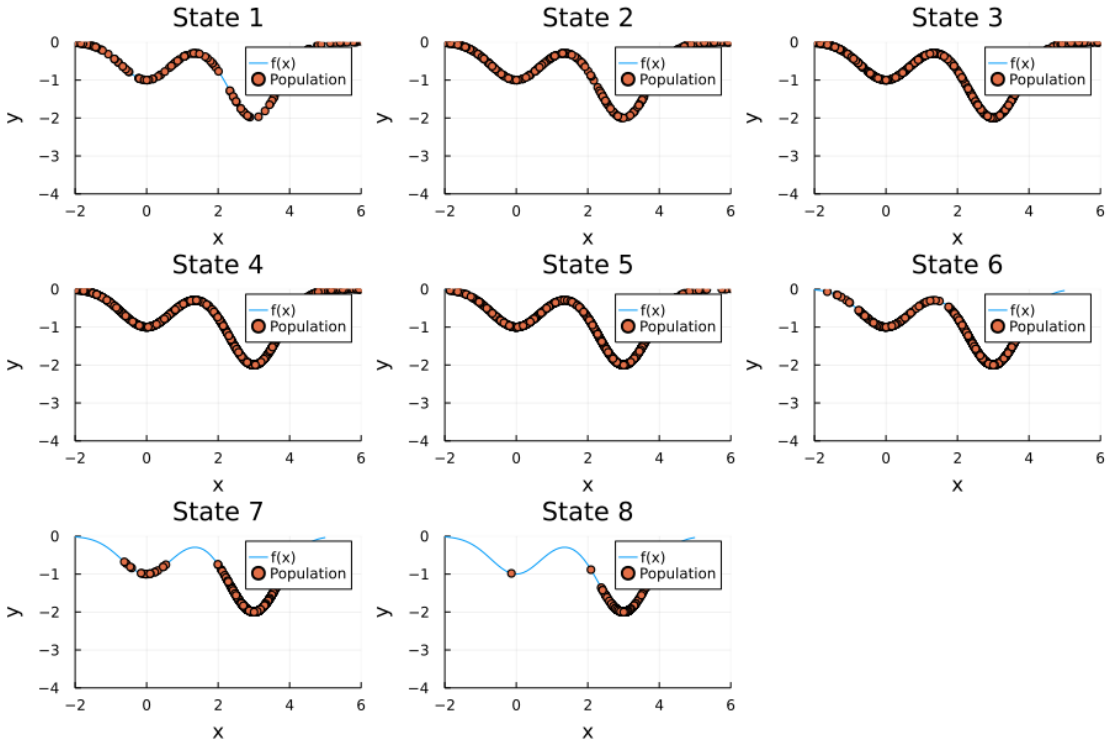


Differential evolution with p = 0.5 and w = 0.2 applied to Ackley's function

❖ **Ưu, nhược điểm:**

- Ưu điểm:
  - Differrential evolution là một phương pháp tiếp cận tốt hơn Genetic Algorithms khi chúng ta cố gắng tối thiểu hàm chi phí. Genetic Alogorithms cũng có thể thực hiện điều này nhưng chúng ta cần phải lựu chọn crossover scheme thích hợp.
  - Có thể giúp chúng ta thoát khỏi cực tiểu cục bộ.
  + Cho kết quả hội tụ tốt hơn và lợi về mặt tính toán.
- Nhược điểm:
  + Hiệu suất bị ảnh hưởng bởi các tham số điều khiển
    - Kích thước quần thể
    - Cách các vector ngẫu nhiên được chọn
    - Hệ số w
    - Cách phép lai ghép được thực hiện
    - Số vòng lặp tạo quần thể thế hệ tiếp theo
    - …
  + Kết quả thay đổi theo mỗi lần chạy.
  + Nên thử nghiệm nhiều lần để rút ra được tham số phù hợp với mô hình.

❖ **Kết quả thuật toán khi chạy trên hàm $-e^{-x^2} - 2e^{-(x-3)^2}$:**

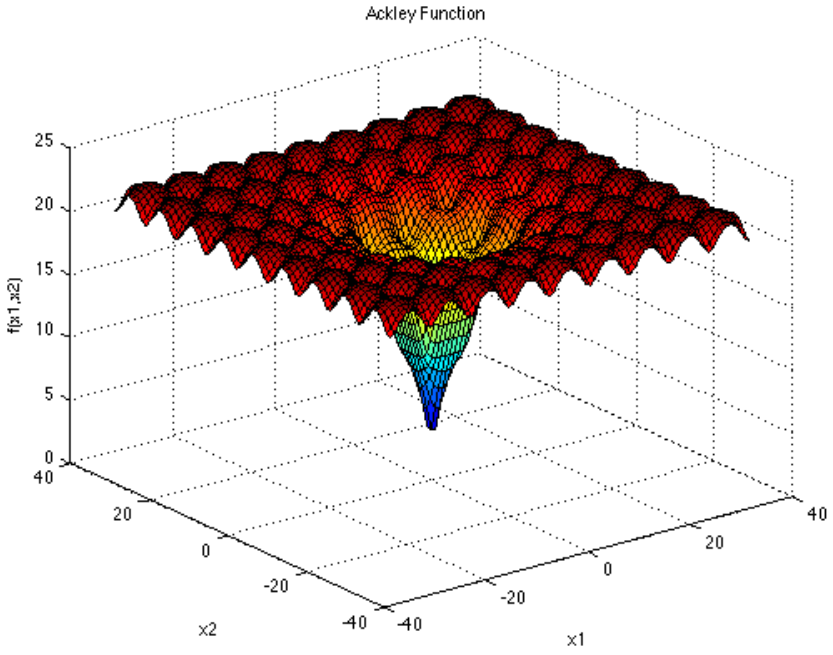


❖ **Nhận xét:**

- Trực giác cho thấy thuật toán cho kết quả có tính hội tụ, chính xác cao và có khả năng vượt qua các cực tiểu địa phương liền kề nhau để tiến tới cực tiểu toàn cục (0).
- Khi thử với số lần lặp là 10 thì thuật toán đã hội tụ.

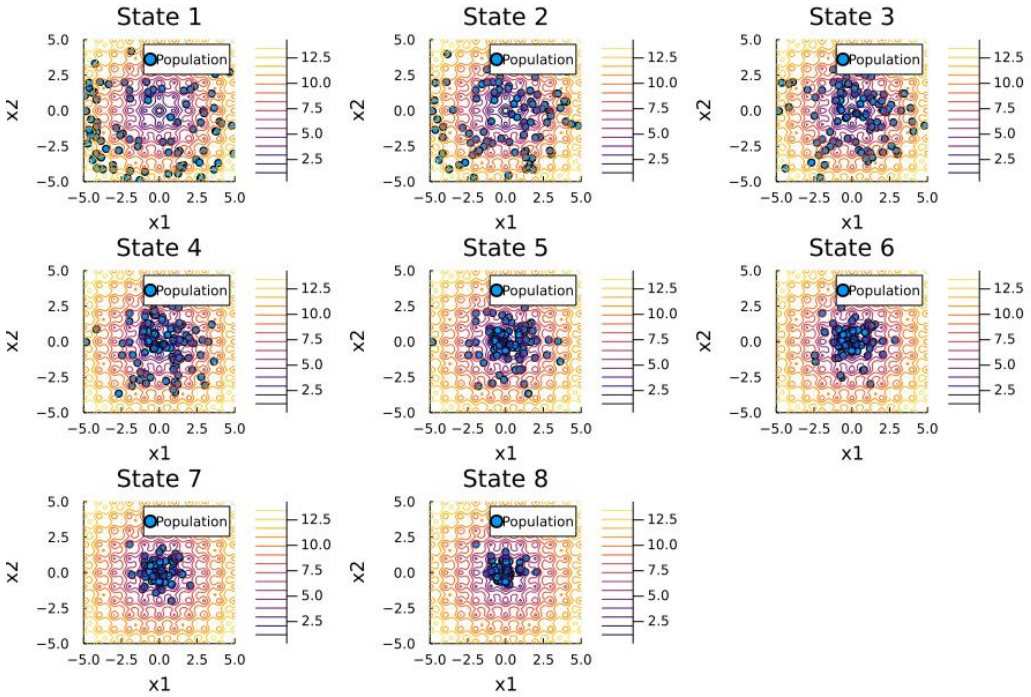
❖ **Kết quả thuật toán khi chạy trên hàm Ackley với a = 20, b = 0.2 and c = 2π**:

$$f(x) = 20e(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2} - e\left(\frac{1}{d}\sum_{i=1}^{d}\cos(2\pi x_i)\right) + 20 + e$$



→ **Nhận xét:** Ở dạng hai chiều Ackey Function có dạng bằng phẳng bên ngoài và lõm ở giữa nên nó đánh giá rất tốt các thuật toán tối ưu, đặc biệt là các thuật toán leo đồi vì rất dễ mắc kẹt ở các điểm cực tiểu địa phương khác nhau.

❖ **Two – dimensional form:**



→ Nhận xét:

- Kết quả tương tự khi kiểm tra với Ackey ở dạng 2 chiều, các quần thể thế hệ tiếp theo cho kết quả tối ưu hơn các thế hệ trước, cũng vượt qua các cực tiểu địa phương và tiến tới cực tiểu toàn cục là (0,0)
- Khi thử với số lần lặp là 10 thì thuật toán đã hội tụ.

4. Particle Swarm Optimization:

Particle swarm optimization (tối ưu bầy đàn) sẽ tối theo một moment sao cho có khuynh hướng hội tụ về cực tiểu. Mỗi phần tử trong quần thể, hay một hạt (particle) sẽ mang thông tin về vị trí hiện tại, vector vận tốc, và vị trí tốt nhất. Từ đó cho phép mỗi phần tử trong tích lũy tốc độ về hướng thuận lợi, không phụ thuộc vào nhiễu loạn địa phương.

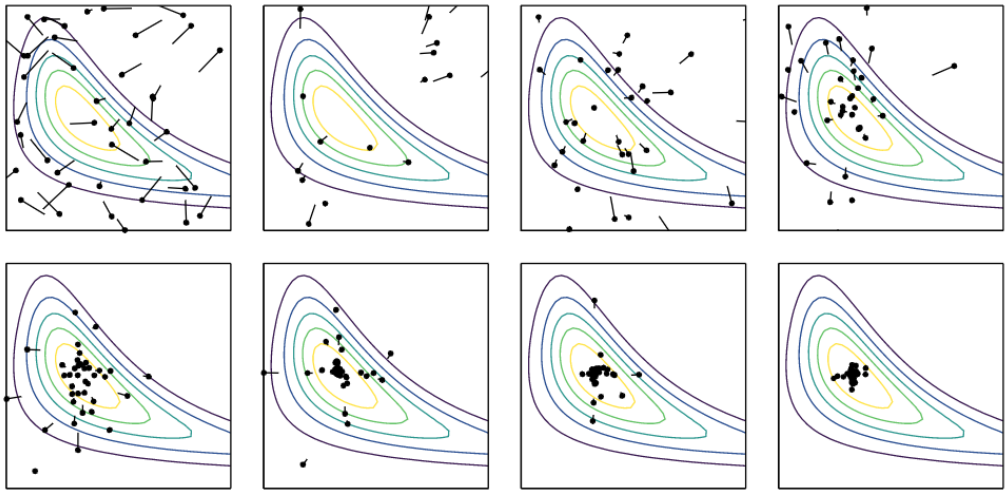
```
mutable struct Particle
    x
    v
    x_best
end
```

Tại mỗi lần lặp mỗi phần tử được tịnh tiến với sự kết hợp của hai yếu tố: vị trí tốt nhất toàn cục (cho toàn bộ giá trị phần tử khởi tạo) và vị trí tốt nhất với chính nó (để đảm bảo một hạt mới được tạo ra bị lệch khỏi hàm mục tiêu quá xa vẫn có thể đưa về lại được). Công thức để cập nhật các giá trị như sau.

$$x^{(i)} \leftarrow x^{(i)} + v^{(i)}$$

$$v^{(i)} \leftarrow wv^{(i)} + c_1 r_1\left(x_{best}^{(i)} - x^{(i)}\right) + c_2 r_2\left(x_{best} - x^{(i)}\right)$$

Với $x_{best}^{(i)}$ là giá trị tốt nhất cho mỗi phần tử, và x_{best} là giá trị tốt nhất trên toàn bộ phần tử được khởi tạo trong quần thể. x_{best} và $x_{best}^{(i)}$ sẽ được cập nhật qua mỗi vòng lặp nếu trong quá trình tịnh tiến có một điểm có kết quả hàm mục tiêu f(.) là tốt hơn cả $f(x_{best})$ đối với x_{best} và cho $f(x_{best}^{(i)})$ đối với $x_{best}^{(i)}$. Từ đó hướng tịnh tiến sẽ được cải thiện qua mỗi vòng lặp, đưa các phần tử về hướng tốt được cho là tốt nhất.



Nhìn vào hình thì các điểm sẽ có khuynh hướng dịch chuyển về điểm cực tiểu từ vòng lặp thứ nhất, sau các vòng lặp tiếp theo các điểm đã hội tụ về vùng tốt nhất best region.

❖ **Ưu, nhược điểm**

- Ưu điểm:

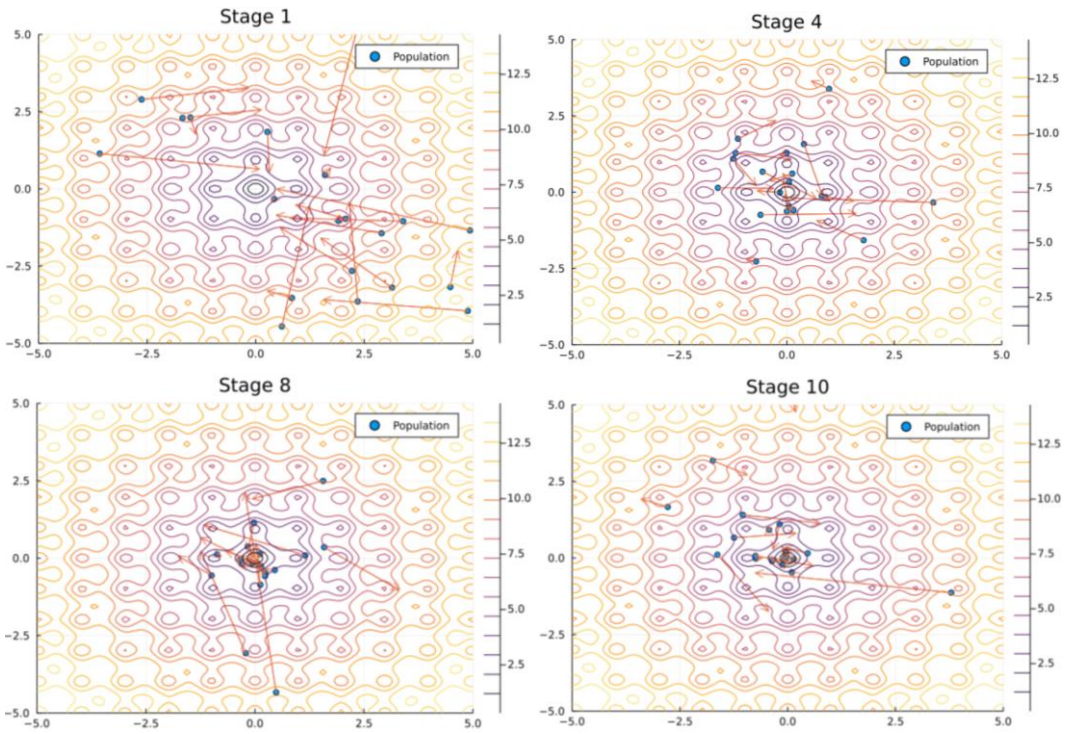
  \+ Không còn yếu tố ngẫu nhiên quá nhiều trong thuật toán, mà các điểm sẽ được tịnh tiến về hướng có được xác định là tốt nhất tại thời điểm. Đồng thời giá trị để tịnh tiến cũng được cập nhật dựa trên các thông tin mang tính tổng quát như hướng cực tiểu. Có dáng vấp của gradient descent.

- Nhược điểm:

  \+ Nếu như trong bầy đàn được khởi tạo (quần thể được khởi tạo) nằm trong điểm cực trị địa phương thì sẽ không thể thoát ra được khỏi vùng cực trị đó, xem như bài toán chỉ tối ưu cục bộ.

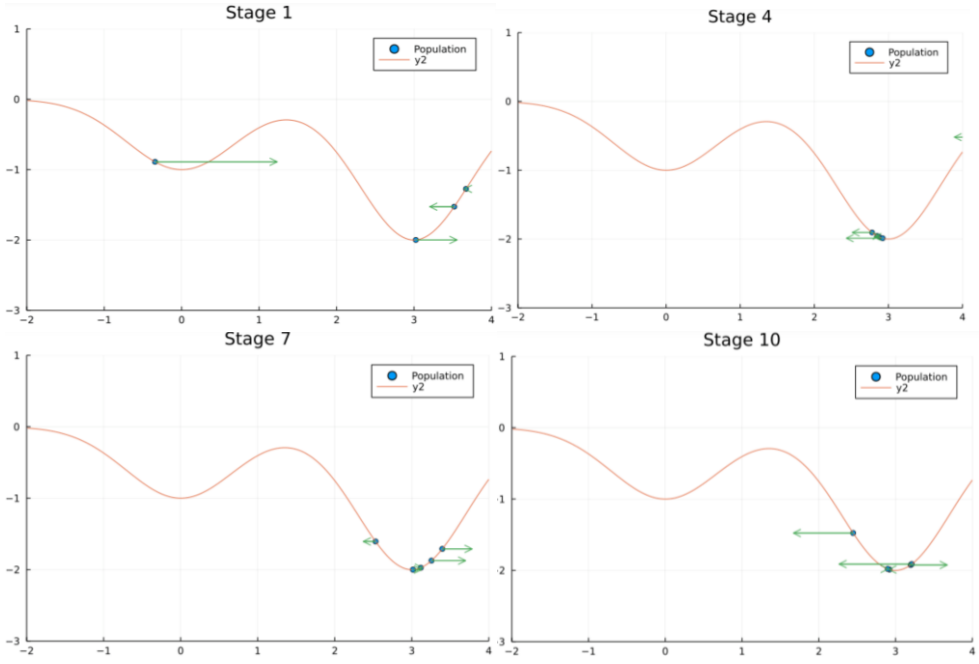
❖ **Kết quả thuật toán khi chạy trên hàm:**

$$ackley(x) = -20e^{-0.4\sqrt{\frac{1}{2}\sum_{i=1}^{2} x_i^2}} - e^{-0.4\times\frac{1}{2}\sum_{i=1}^{2}\cos(2\pi x_i)} + 20 + e$$

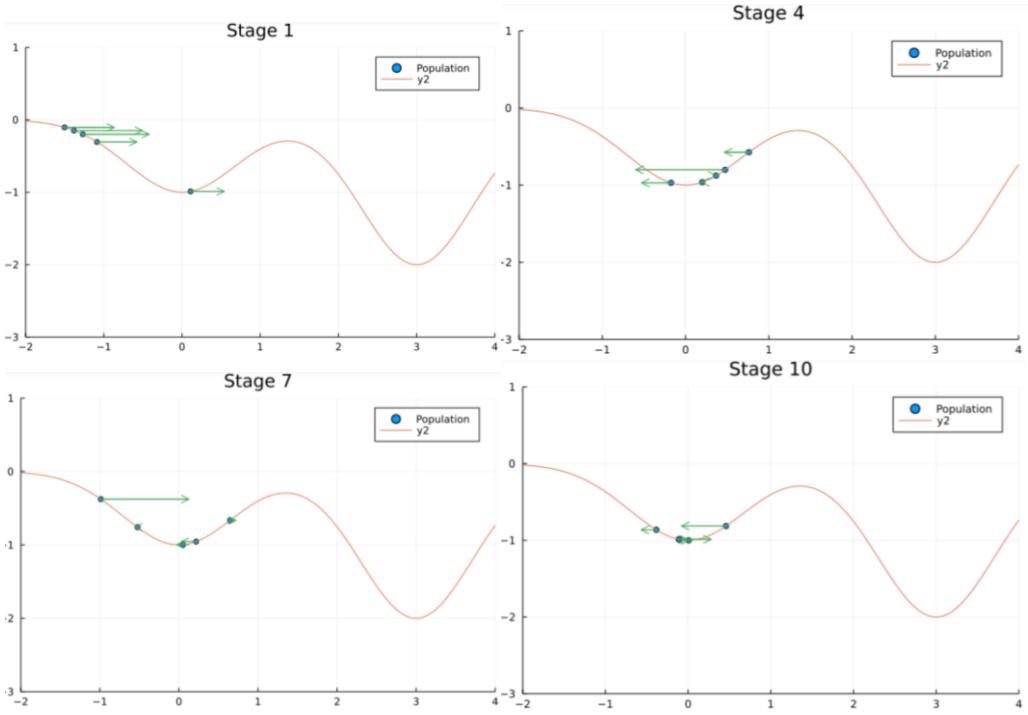


→ **Nhận xét:** Các phần tử trong quần thể do mang thông tin về hướng tốt nhất, nên có thể bỏ qua cực trị địa phương mà leo đồi nên khiến cho các phần tử đồng thời cùng hội tụ về best region (0, 0).

❖ **Kết quả thuật toán khi chạy trên hàm $-e^{-x^2} - 2e^{-(x-3)^2}$:**



→**Nhận xét:** Do khi giai đoạn khởi tạo, trong cá thể sẽ có một cá thể lọt vào best region khiến cho việc hội tụ trở nên dễ dàng, tuy nhiên trường hợp tiếp theo sẽ cho thấy việc khởi tạo cá thể không đủ lớn hay không lọt vào best region chỉ khiến nó hội tụ ở cực trị địa phương.



5. Firefly Algorithm:

Thuật toán đom đóm là một thuật toán metaheuristic dựa trên bầy đàn được giới thiệu bởi Xin-She Yang(Senior Research Scientist at National Physical Laboratory). Thuật toán đom đóm được lấy cảm hứng từ hành vi bay ra của đom đóm.

❖ **Giả thiết được đưa ra:**

- Những con đom đóm bị hút vào nhau.
- Độ hấp dẫn tỷ lệ thuận với độ sáng.
- Độ hấp dẫn tỷ lệ nghịch với khoảng cách.
- Con đom đóm sáng mạnh hơn sẽ hút con đom đóm sáng yếu hơn về phía nó.
- Nếu độ sáng của cả hai con là như nhau, thì đom đóm di chuyển ngẫu nhiên.

❖ **Các tham số sử dụng trong thuật toán:**

- Số cá thể của quần thể đom đóm: n

- Cá thể đom đóm: x_i (i = 1:n)
- Số lần lặp tối đa: k_max
- Số thuộc tính của mỗi con đom đóm: m
- Hệ số hấp thụ ánh sáng: γ
- Hệ số học: α
- Cường độ sáng nguồn: β_0

❖ **Các công thức toán được sử dụng**

- Khoảng cách giữa 2 cá thể: $r = \sqrt{(xi - xj)2}$
- Độ hấp dẫn: $I(r) = e^{-\gamma r2}$
- Di chuyển x_i tới x_j: $x_i += \beta_0 * I(r) * (xj - x_i) + \alpha * \varepsilon_i$
- ε_i = rand(MvNormal(Matrix(1.0I, m, m)))

❖ **Thuật toán đom đóm:**

1) Xác định hàm mục tiêu: f(x);
2) Khởi tạo quần thể gồm n cá thể đom đóm: x_i (i = 1, 2,….,n).
3) Xác định hàm tính cường độ ánh sáng
4) Khai báo các tham số: ngưỡng của các thuộc tính, hệ số hấp thụ ánh sáng, hệ số học, biến lưu kết quả,...
5) Đánh giá quần thể thông qua hàm mục tiêu, chọn ra cá thể tốt nhất ban đầu.
6) Chạy vòng lặp

while (chưa đạt tối đa vòng lặp)
  for i = 1 : n (n con đóm đóm)
    for j = 1 : n (n con đóm đóm)
      if (f(x_i) > f(x_j)): (tùy thuộc vào bài toán đẻ xem cá thể nào tốt hơn)
        Di chuyển x_i đến x_j
        Cập nhật lại giá trị hàm mục tiêu của cá thể x_i : f(x_i)
        Chọn lại kết quả tốt nhất của hiện tại. min(f(x))
      end if
    end for j
  end for i
end while

7) Trả về kết quả tốt nhất

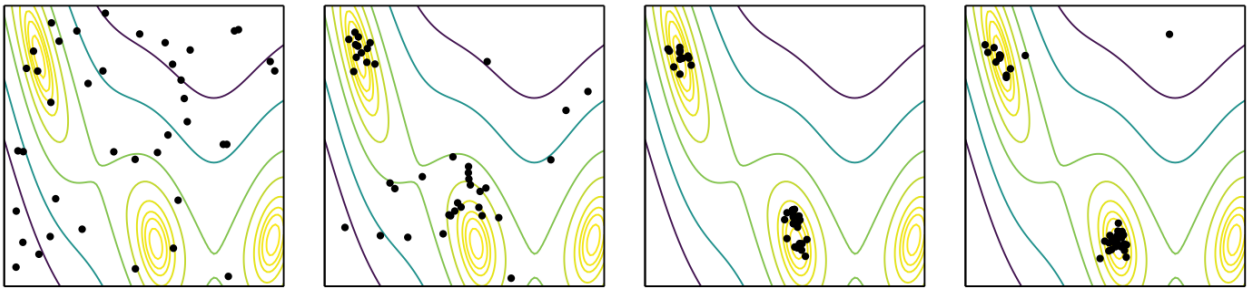


Figure 9.15. Firefly search with $\alpha = 0.5$, $\beta = 1$, and $\gamma = 0.1$ applied to the Branin function (appendix B.3).

→**Nhận xét:** Hình trên cho thấy sau nhiều vòng lặp, các con đom đóm tập trung về những điểm sáng gần chúng nhất

❖ **Ứng dụng thực tế:**

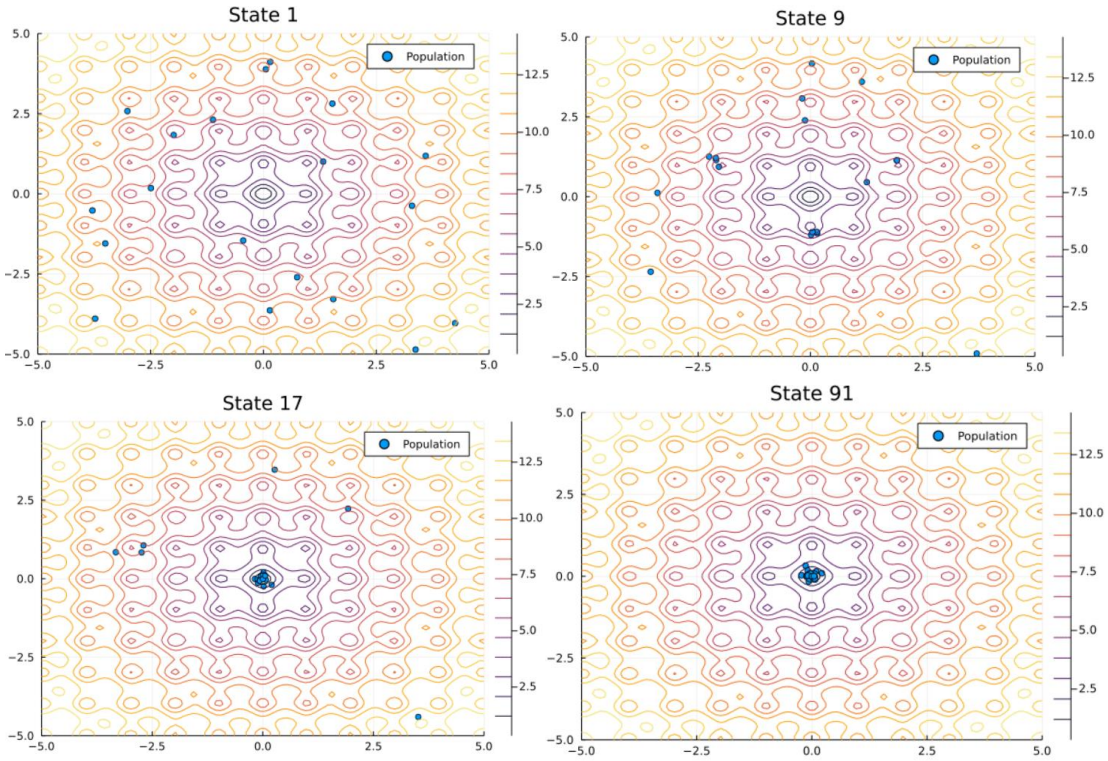
- Giải quyết các bài toán tối ưu hóa
- Nén hình ảnh, xử lí hình ảnh
- Tìm đường đi ngắn nhất,...
- Tìm cực tiểu

❖ **Ưu, nhược điểm:**

- **Ưu điểm:**
  + Cơ chế chia sẽ thông tin giúp thuật toán hội tụ nhanh hơn trong một số điều kiện nhất định
  + Xác xuất rơi vào trường hợp cục bộ thấp hơn do sử dụng các tham số ngẫu nhiên
- **Nhược điểm:**
  + Kết quả phụ thuộc vào các tham số đầu vào
  + Số lần lặp lớn làm tăng thời gian thực thi

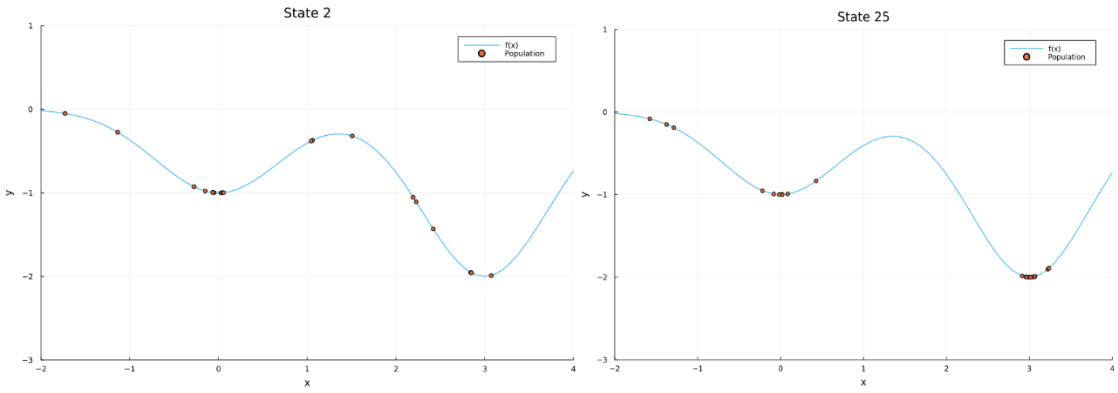
❖ **Kết quả khi chạy trên hàm:**

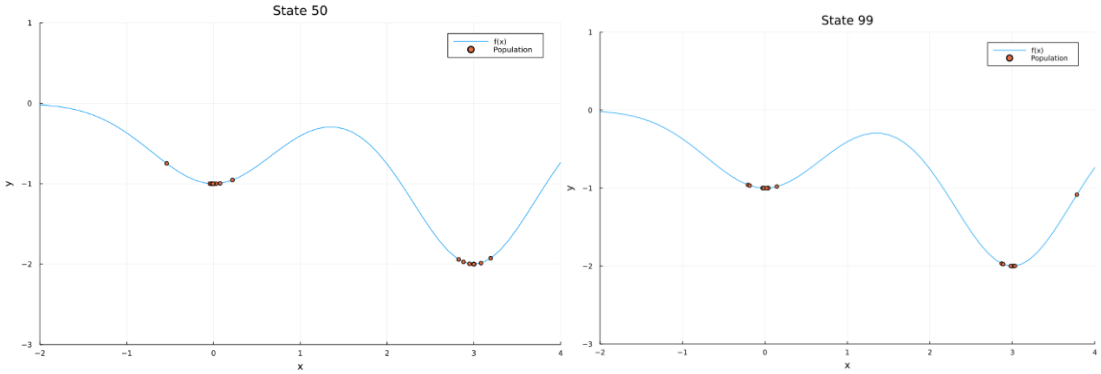
$$ackley(x) = -20e^{-0.4\sqrt{\frac{1}{2}\sum_{i=1}^{2} x_i^2}} - e^{-0.4\times\frac{1}{2}\sum_{i=1}^{2}\cos(2\pi x_i)} + 20 + e$$



→ **Nhận xét:** Mỗi chấm xanh đại điện cho 1 con đom đóm(cặp giá trị (x1,x2)). Sau mỗi lần lặp, các con đom đóm có xu hướng di chuyển về nơi giá trị hàm f(x1,x2) là nhỏ nhất. Ở trường hợp này là vị trí (0,0)

❖ **Kết quả thuật toán khi chạy trên hàm:** $-e^{-x^2} - 2e^{-(x-3)^2}$

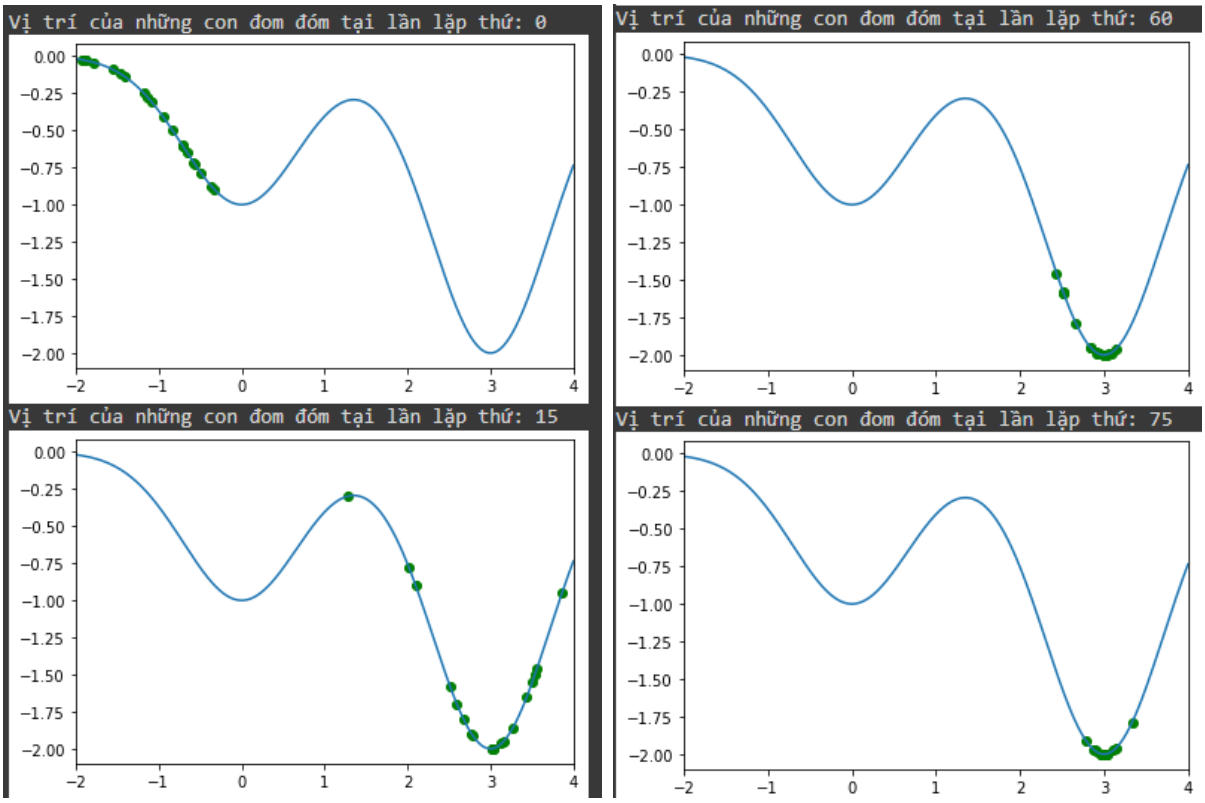




→ **Nhận xét:**

- Mỗi chấm xanh đại điện cho 1 con đom đóm(x). Sau mỗi lần lặp, các con đom đóm có xu hướng di chuyển về nơi giá trị hàm f(x) nhỏ nhất. Tốn nhiều vòng lặp để hội tụ các con đom đóm về cực tiểu toàn cục.
- Ở một giai đoạn nào đó có thể bị mắc kẹt ở cực tiểu cực bộ nhưng vì tính di chuyển ngẫu nhiên khi cường độ sáng bằng nhau nên chỉ cần 1 cá thể may mắn bay đến gần cực tiểu toàn cục thì quần thể vẫn hội tụ về cực tiểu toàn cục.

Ví dụ: khởi tạo quần thể ban đầu toàn những cá thể không nằm gần cực tiểu toàn cục $-2 < x < 0$. Các cá thể vẫn di chuyển về đúng cực tiểu toàn cục.



6. Cukoo Search:

Là một thuật toán khác dựa vào lấy cảm hứng từ tự nhiên được đặt tên theo con chim cukoo, đặc điểm của chúng sẽ đẻ trứng vào tổ chim của loài khác. Khi điều đó xảy ra chim chủ sẽ phá hủy nó khi phát hiện trứng của loài cukoo hoặc sẽ tạo tổ ở một nơi mới. Tuy nhiên vẫn có cơ hội cho trứng của cukoo được chấp nhận và nuôi lớn bởi chim chủ.

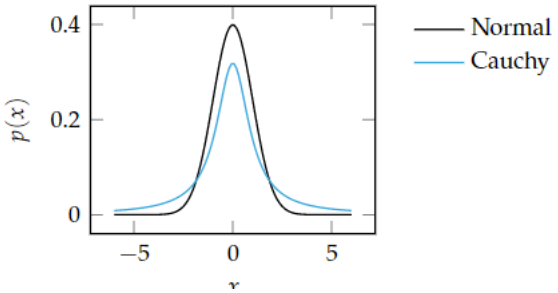
Trong cukoo search, mỗi tổ được biểu diễn bằng một design point. Và điểm mới sẽ được sinh ra bằng cách ngẫu nhiên tịnh tiến theo kết quả của một phân phối cho trước. Điểm mới sẽ chiếm tổ nếu nó mang lại giá trị tối ưu cho hàm f(.) tốt hơn, điều này tương ứng với với việc trứng cuckoo thay thế trứng loài chim khác ở thực tế.

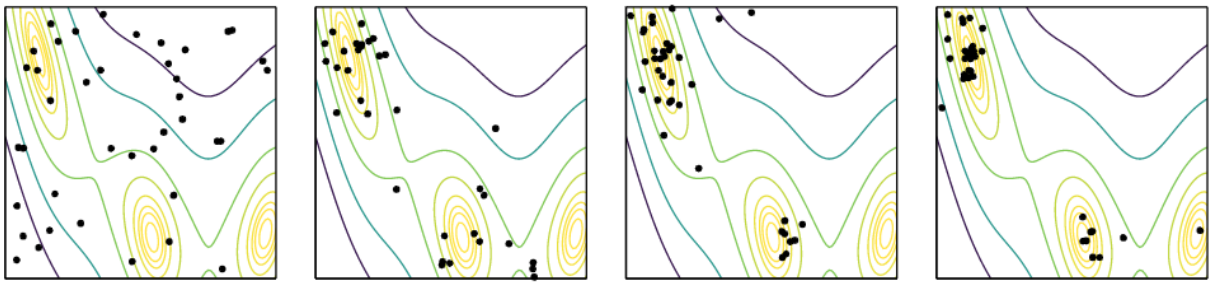
❖ **Những điều luật cốt lõi:**

+ Chim cuckoo sẽ đẻ một quả trứng ngẫu nhiên trên các tổ
+ Tổ tốt nhất với quả trứng tốt nhất sẽ tồn tại qua thế hệ tiếp theo (hàm mục tiêu tối ưu hơn)
+ Trứng cuckoo có cơ hội bị phát hiện bởi chim chủ, trong trường hợp đó trứng cuckoo bị phá hủy.

Cuckoo dựa vào chuyến bay ngẫu nhiên để đẻ trứng, hay đúng hơn là các điểm sẽ được chọn ngẫu nhiên để đẻ, và việc đẻ ở một vị trí nào sẽ dựa vào bước đi hay giá trị của một

phân phối, ở đây xem xét phân phối cauchy và phân phối chuẩn. Việc sử dụng phân phối chuẩn khiến giá trị bước đi sẽ xung quanh một vùng tập trung, khó có thể đi xa khiến việc hội tụ chậm hơn, thay vào đó phân phối cauchy với phần đuôi trải ra rộng hơn trong đồ thị hàm phân phối giúp các điểm có thể hội tụ nhanh hơn. Ngoài ra, phân phối Cauchy đã được chứng minh là thể hiện cho các chuyển động của các động vật khác trong tự nhiên.





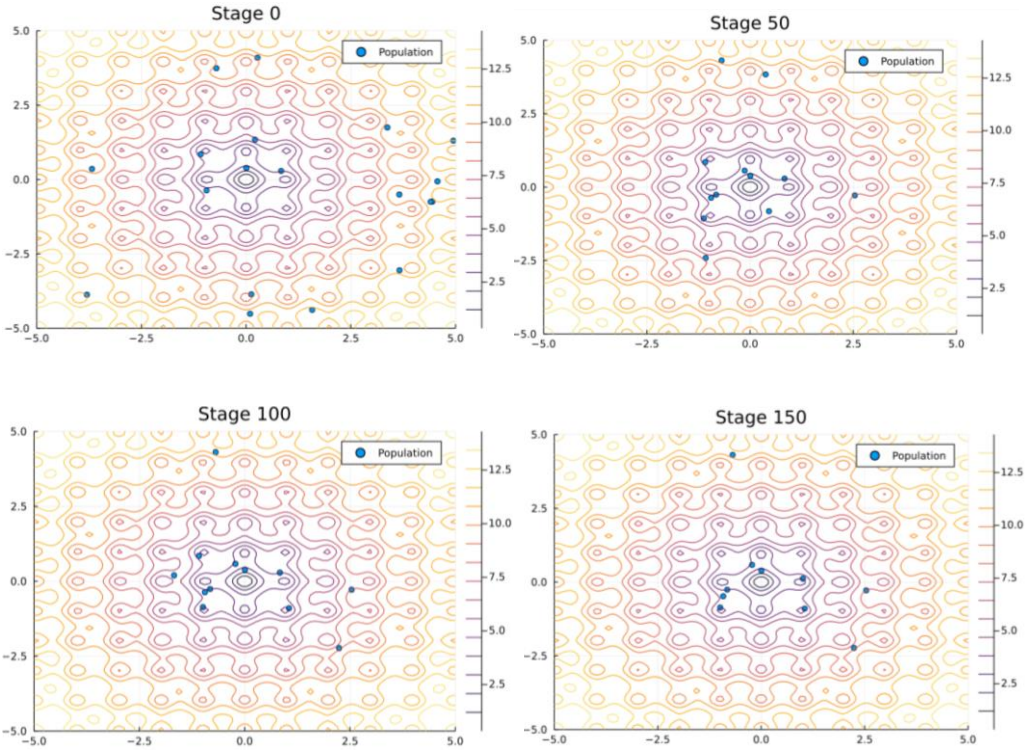
→**Nhận xét:** Với hình trên trứng cuckoo đã đẻ vào được những trứng về hướng cực tiểu được giữ lại và sau nhiều vòng lặp, các trứng đã hội tụ về được các cực trị.

❖ **Ưu, nhược điểm:**

- Ưu điểm: hoạt động tốt, loại bỏ được các kết quả mang lại hàm tối ưu lớn giúp hội tụ tốt hơn.
- Nhược điểm: phụ thuộc vào không gian quần thể khởi tạo (phụ thuộc vào tính ngẫu nhiên), nếu các điểm ở quá xa cực tiểu có thể xảy ra trường hợp không bao giờ hội tụ về cực trị toàn cục mà chỉ hội tụ về cục bộ, hoặc sẽ tốn rất nhiều vòng lặp.

❖ **Kết quả thuật toán khi chạy trên hàm:**

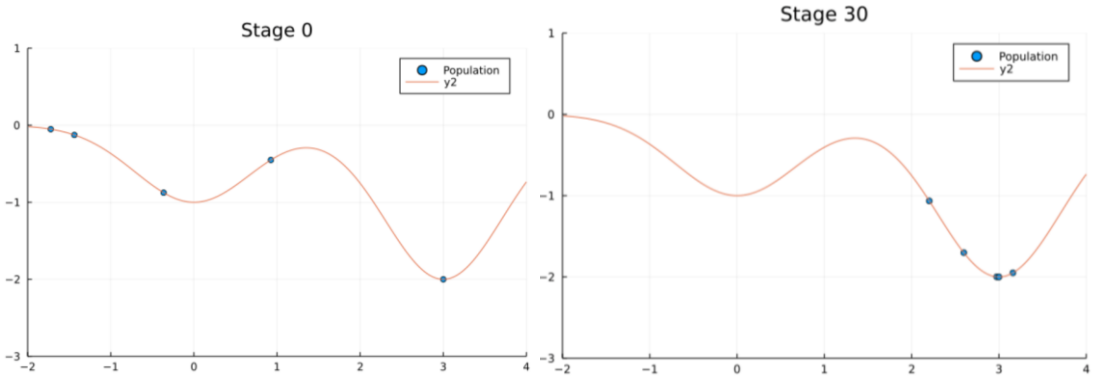
$$ackley(x) = -20e^{-0.4\sqrt{\frac{1}{2}\sum_{i=1}^{2}x_i^2}} - e^{-0.4\times\frac{1}{2}\sum_{i=1}^{2}\cos(2\pi x_i)} + 20 + e:$$

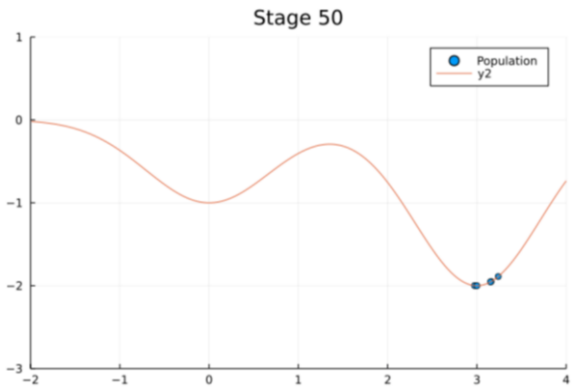


→ **Nhận xét:** Có thể thấy rõ ở các bước để phần tử về best region tốn khá nhiều chi phí hơn các thuật toán khác, do xác phải cần số vòng lặp lớn để trưng được đẻ gần best region (số lượng cá thể lớn). Và nghiệm tốt nhất sau 150 vòng lặp vẫn không gần (0, 0) như các thuật toán trên.

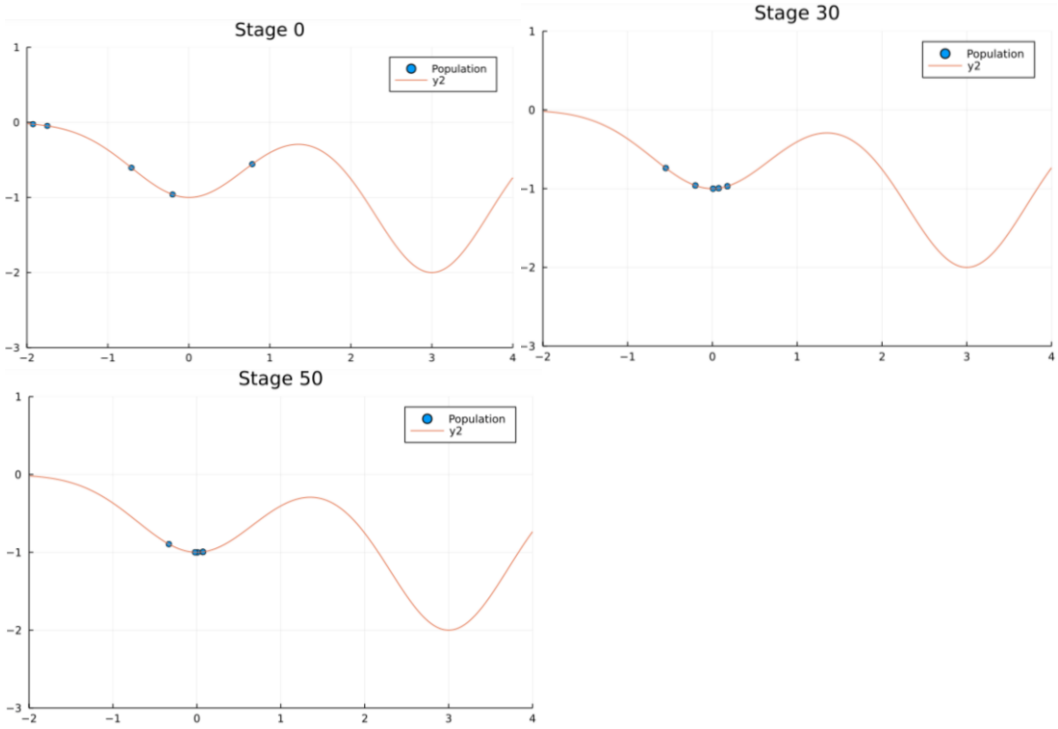
❖ **Kết quả thuật toán khi chạy trên hàm $-e^{-x^2} - 2e^{-(x-3)^2}$:**

Các trứng sinh ra ở các vị trí tốt nhất sau nhiều vòng lặp sẽ khiến các phần tử trong quần thể hội tụ về cực tiểu toàn cục.

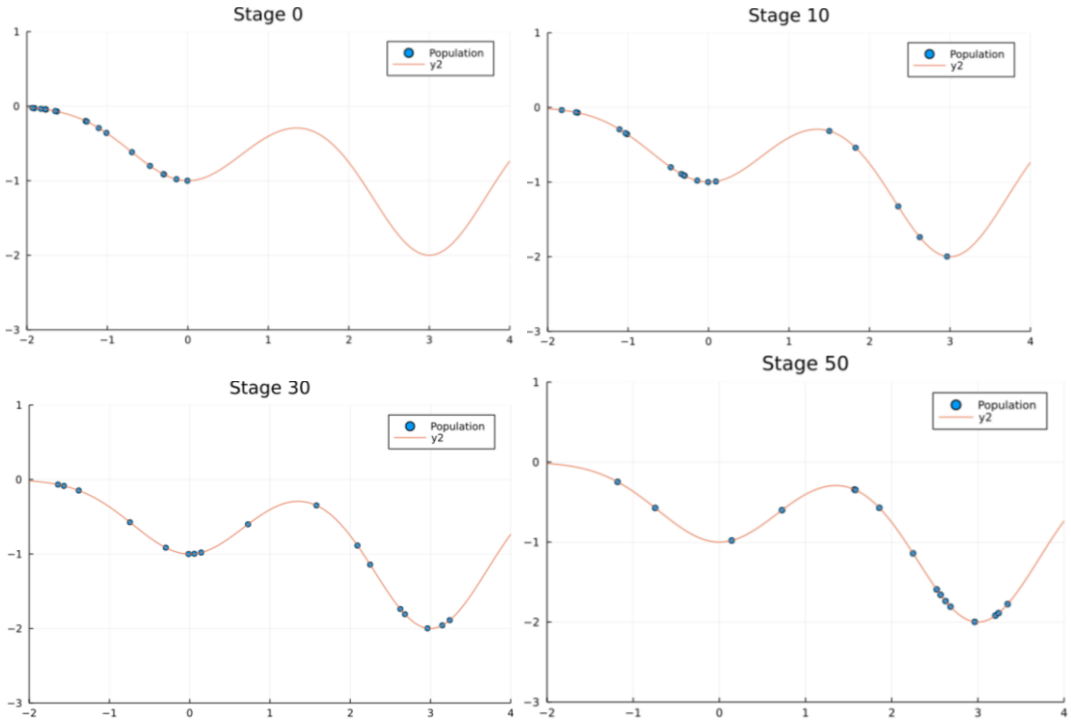




Các điểm khởi tạo nằm trong vùng cực trị cục bộ khiến bài toán không về tối ưu toàn cục được.



Nhưng nếu dữ liệu đủ lớn, tính ngẫu nhiên sẽ xuất hiện giúp cho các phần tử trong quần thể có khả năng hội tụ về cực trị toàn cục.



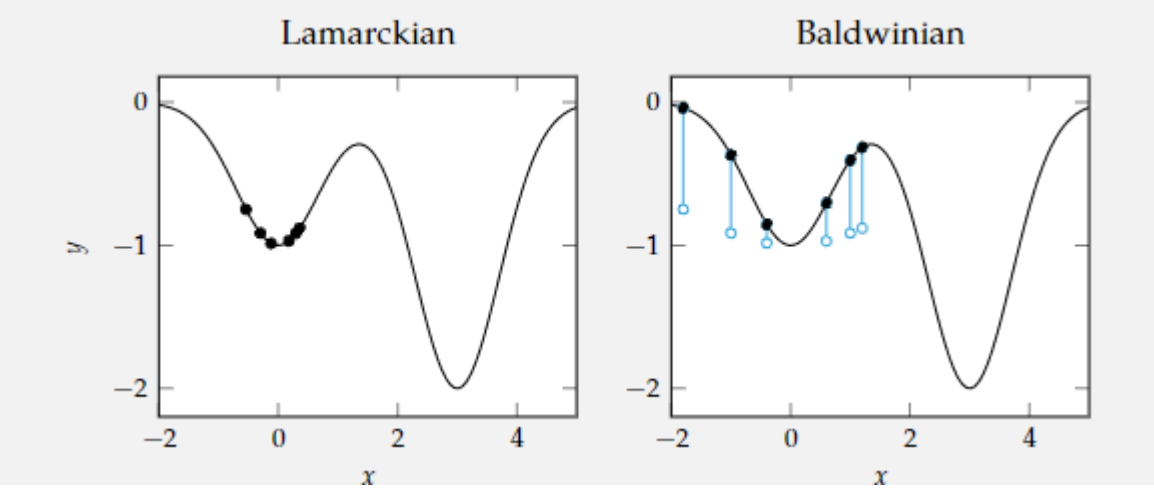
7. Hybrid Methods:

❖ **Định nghĩa:**

Hybrid Methods là phương pháp kết hợp Population Methods với phương pháp tìm kiếm cục bộ nhằm cải thiện khả năng, tốc độ tìm kiếm cực trị toàn cục. Các phương thức, kỹ thuật thường được dùng để kết hợp với Population Methods bao gồm: Lamarckian learning, Baldwinian learning, Lamarckian – Baldwinian, …

Lamarckian learning: Kết hợp tìm kiếm cục bộ, thay đổi cá thể ban đầu và giá trị hàm mục tiêu với cá thể, giá trị tối ưu đã tìm kiếm được.

Baldwinian learning: Tương tự như Lamarckian learning nhưng các cá thể sẽ không bị thay đổi, nhưng giá trị hàm mục tiêu của từng cá thể sẽ dựa vào giá trị tối ưu tìm kiếm được.



→ **Nhận xét:** Có thể thấy như nhóm trình bày, Lamarckian sẽ thay đổi và di chuyển các cá thể về điểm cực tiểu địa phương, còn khi sử dụng phương pháp Baldwinian thì vị trí các cá thể vẫn được giữ nguyên nhưng giá trị đánh giá sẽ dựa vào giá trị các điểm 'trắng" như trong hình.

❖ **Ví dụ cụ thể**: **Hybrid Genetic Algorithm**

Mô tả thuật toán:

- B1. Xác định một hàm mục tiêu / phù hợp và đặt các toán tử GA (chẳng hạn như kích thước quần thể, tỷ lệ bố mẹ / con cái, phương pháp chọn lọc, số lần lai chéo và tỷ lệ đột biến).
- B2. Tạo ngẫu nhiên quần thể ban đầu là quần thể bố mẹ hiện tại.
- B3. Đánh giá hàm mục tiêu đối với từng cá thể (nhiễm sắc thể hoặc dung dịch) trong quần thể ban đầu.
- B4. Tạo ra một quần thể con bằng cách sử dụng các toán tử GA (chẳng hạn như chọn lọc / giao phối, trao đổi chéo và đột biến).
- B5. Đánh giá chức năng mục tiêu của từng cá thể trong quần thể con.
- B6. Thực hiện tìm kiếm cục bộ trên từng con cái, đánh giá giá trị fit function của từng vị trí mới và thay thế con cái nếu có giải pháp cải thiện cục bộ (Lamarckian learning)
- B7. Quyết định đưa những cá thể nào vào quần thể tiếp theo. Bước này được gọi là "thay thế" trong đó các cá thể từ quần thể bố mẹ hiện tại được "thay thế" bởi một quần thể mới bao gồm các cá thể đó từ quần thể con và / hoặc quần thể bố mẹ.

- B8. Nếu một tiêu chí dừng được thỏa mãn, thì quy trình sẽ được tạm dừng. Nếu không, hãy chuyển sang Bước 4.

- **Bước 6** chính là bước chúng ta kết hợp phương thức Lamarckian learning khi thay thế từng cá thể mới bằng cá thể tốt hơn. Còn lại thì Hybrid Genetic Algorithm vẫn giữ các bước của Genetic Algorithms.

❖ **Tổng thể:**

Cải thiện được hiệu năng của Population Methods (hội tụ sớm hơn) nhờ vào việc kết hợp tìm kiếm cục bộ, vẫn gữ được những đặc điểm của Population Methods tuy nhiên tùy thuộc cách tìm kiếm cục bộ, việc thay đổi cấu trúc di truyền của cá thể khiến Genetic Search có thể hội tụ quá sớm ở một cực trị địa phương, điều này là rất xấu nếu chúng ta cần tìm kiếm cực trị toàn cục.

8. Summary:

– Population methods sử dụng một tập hợp các cá thể (quần thể) trong không gian thiết kế để phát triển bài toán theo những hướng tối ưu.

– Genetic Algorithms sử dụng các biện pháp lựa chọn, lai tạo, đột biến để tạo ra những thế hệ con tốt hơn.

– Khác với những thuật toán xác định như Hill-Climbing, Newton-Raphson tuy cho các phép xử lí nhanh và hiệu quả một số vấn đề nhất định nhưng khả năng cao bị kẹt ở cục bộ. Việc sử dụng những tham số ngẫu nhiên ở các thuật toán meta-heuristic như: Firefly Algorithm, Cuckoo Search, Partical Swarm Optimization giúp chúng có khả năng thoát khỏi cục bộ và có cơ hội tìm kiếm lớn hơn trên toàn cục.

– Population methods sử dụng cơ chế hội tụ quần thể về những cá thể tốt nhất nhưng vẫn giữ được không gian trạng thái thích hợp.

– Population methods có thể kết hợp với các phương pháp tìm kiếm cục bộ như: Lamarckian learning, Baldwinian learning,..để cải thiện tốc độ, độ hiệu quả.

– Nhược điểm của các Population method là tốc độ hội tụ phụ thuộc vào các tham số đầu vào và việc khởi tạo thế hệ ban đầu → Việc thiết kế các tham số cụ thể cho từng bài toán có vai trò quan trọng.

9. Application:

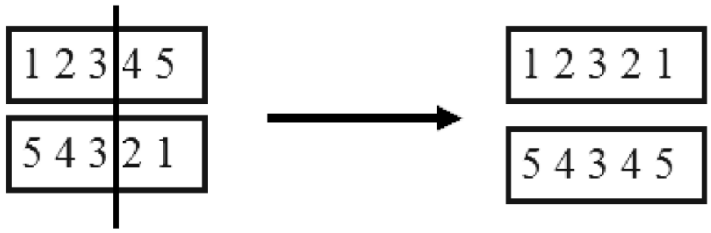
Nhóm em xin trình bày ứng dụng cho bài toán thực tế cụ thể là phương pháp Genetic Algorithm dùng cho bài toán **Travelling Salesman Problem (TSP)**: là bài toán tìm hành trình tối ưu cho một người bàn hàng (hay có thể là shipper), mỗi thành phố chỉ được đi qua một lần, ngoại trừ thành phố bắt đầu cũng là nơi kết thúc chuyến đi sao cho chi phí là

thấp nhất. Từ đó có thể xác định được nơi chọn làm cửa hàng khi đồ vận chuyển cho nhiều nơi cùng một lúc, phân cho người nhân viên đang ở đâu tuyến giao hàng nào…

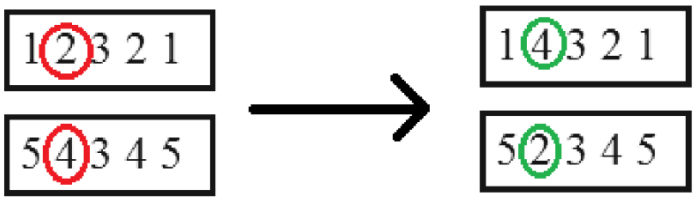
- Để giải quyết bài toán, ta cần xác định các thành phần chính cho thuật toán:
  - Chromosomes (nhiễm sắc thể): sẽ là dãy các số đại diện cho n địa điểm, ở đây là 8 thành phố.

[0,1,3,2,4,6,5,7]

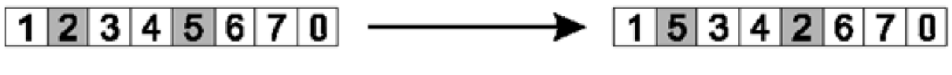
  - Crossover (lai tạo): thay vì việc lai tạo theo bình thường sẽ làm các thành phố sẽ trùng lặp nhau



  - Thay vào đó ta sẽ cải tiến thêm một bước bằng cách tìm những địa điểm trùng ở từng cá thể đem lai tạo swap cho nhau, từ đó lỗi có thể được fix, đồng thời tạo ra được cá thể mới từ việc lai tạo.

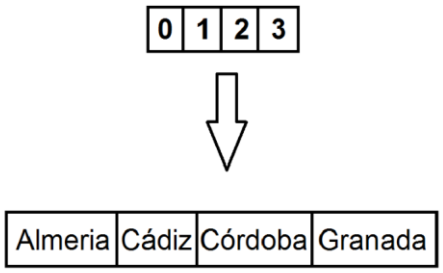


  - Mutation (đột biến): có thể trao đổi vị trí của các gene trong nhiễm sắc thể ở cách xa hoặc gần



  - Decode: tương ứng với mỗi thành phố ta sẽ có các địa điểm thật như sau

```
cities = {
    0:'Almería',
    1:'Cádiz',
    2:'Cordoba',
    3:'Granada',
    4:'Huelva',
    5:'Jaen',
    6:'Málaga',
    7:'Sevilla'}
```



- Ngoài ra ta còn sẽ có thêm thông tin về khoảng cách giữa các thành phố phục vụ cho hàm tối ưu:

```
w0=[999,454,317,165,528,222,223,410]
w1=[453,999,253,291,210,325,234,121]
w2=[317,252,999,202,226,108,158,140]
w3=[165,292,201,999,344,94,124,248]
w4=[508,210,235,346,999,336,303,94]
w5=[222,325,116,93,340,999,182,247]
w6=[223,235,158,125,302,185,999,206]
w7=[410,121,141,248,93,242,199,999]
```

- Với hàm tối ưu được sử dụng là:

$$\text{Penalty}(Path) = 100\ x\ |Gens\ -\ Path|$$

$$\text{Fitness}(Path) = 2\ x \sum_{i,j=0}^{7}(w_{i,j}) + 50\ x\ Penalty(Path)$$

- Với Penalty(.) thực chất muốn đánh vào những nhiễm sắc thể có đường bị trùng lập sẽ loại đi những trường hợp lỗi nên việc đánh trọng số 100*50*x sẽ là rất lớn, khiến những nhiễm sắc thể sai yêu cầu đề sẽ bị loại bỏ.

```
def penalty(chromosome):
        actual = chromosome
        value_penalty = 0
        for i in actual:
            times = 0
            times = actual.count(i)
            if times > 1:
                value_penalty+= 100 * abs(times - len(actual))
        return value_penalty
```

- Và hàm tối ưu sẽ là hàm có chi phí đường đi thấp nhất, cũng chính là lời giải bài

toán.

- Và đây là kết quả khi chọn k_max số vòng lặp là 10 và show ra kết quả tốt nhất ở mỗi lần lặp.

```
Chromosome:  [7, 2, 5, 3, 0, 6, 1, 4]
Solution: (['Sevilla', 'Cordoba', 'Jaen', 'Granada', 'Almeria', 'Malaga', 'Cadiz', 'Huelva'], 1269)
Chromosome:  [3, 0, 5, 2, 4, 1, 7, 6]
Solution: (['Granada', 'Almeria', 'Jaen', 'Cordoba', 'Huelva', 'Cadiz', 'Sevilla', 'Malaga'], 1384)
Chromosome:  [2, 4, 7, 1, 6, 0, 3, 5]
Solution: (['Cordoba', 'Huelva', 'Sevilla', 'Cadiz', 'Malaga', 'Almeria', 'Granada', 'Jaen'], 1273)
Chromosome:  [2, 5, 3, 0, 6, 1, 7, 4]
Solution: (['Cordoba', 'Jaen', 'Granada', 'Almeria', 'Malaga', 'Cadiz', 'Sevilla', 'Huelva'], 1273)
Chromosome:  [4, 2, 5, 3, 0, 6, 1, 7]
Solution: (['Huelva', 'Cordoba', 'Jaen', 'Granada', 'Almeria', 'Malaga', 'Cadiz', 'Sevilla'], 1273)
Chromosome:  [4, 7, 2, 5, 3, 0, 6, 1]
Solution: (['Huelva', 'Sevilla', 'Cordoba', 'Jaen', 'Granada', 'Almeria', 'Malaga', 'Cadiz'], 1269)
Chromosome:  [0, 3, 5, 2, 7, 4, 1, 6]
Solution: (['Almeria', 'Granada', 'Jaen', 'Cordoba', 'Sevilla', 'Huelva', 'Cadiz', 'Malaga'], 1275)
Chromosome:  [3, 0, 6, 1, 7, 4, 2, 5]
Solution: (['Granada', 'Almeria', 'Malaga', 'Cadiz', 'Sevilla', 'Huelva', 'Cordoba', 'Jaen'], 1273)
Chromosome:  [4, 7, 1, 6, 0, 3, 5, 2]
Solution: (['Huelva', 'Sevilla', 'Cadiz', 'Malaga', 'Almeria', 'Granada', 'Jaen', 'Cordoba'], 1273)
Chromosome:  [1, 4, 7, 2, 5, 3, 0, 6]
Solution: (['Cadiz', 'Huelva', 'Sevilla', 'Cordoba', 'Jaen', 'Granada', 'Almeria', 'Malaga'], 1269)
Chromosome:  [0, 3, 5, 2, 7, 4, 1, 6]
Solution: (['Almeria', 'Granada', 'Jaen', 'Cordoba', 'Sevilla', 'Huelva', 'Cadiz', 'Malaga'], 1275)

Total time:  4.008964538574219  secs.
```

10. Tham khảo:

1. Mykel J. Kochenderfer và Tim A. Wheeler, Algorithms for Optimization, (2019).
2. Fernando Méndez Requena, fermenreq/TSP-VRP-GENETICS-ALGORITHMat 46e0a4befcf1d0c1b8dd94922a996cc23b6ab2ca (github.com), (2018).

HẾT