Trí tuệ nhân tạo (Artificial Intelligence)

Tìm kiếm cục bộ

By Hoàng Hữu Việt
Email: viethh@vinhuni.edu.vn
Viện Kỹ thuật và Công nghệ, Đại học Vinh

Vinh, 3/2019

Tài liệu

- Tài liệu chính

  [1] Stuart Russell, Peter Norvig. Artificial Intelligence. A modern approach. 3rd ed. Prentice Hall, 2009.

- Tài liệu khác

  [2] Milos Hauskrecht. Artificial Intelligence, 2013.
  people.cs.pitt.edu/~milos/courses/cs1571-Fall2013/

  [3] Jenna Carr, An Introduction to Genetic Algorithms, 2014.

  [4] Randy L. Haupt, Sue Ellen Haupt, Practical Gennetic Algorithms, 2nd, John Wiley & Sons, 2004.

Nội dung

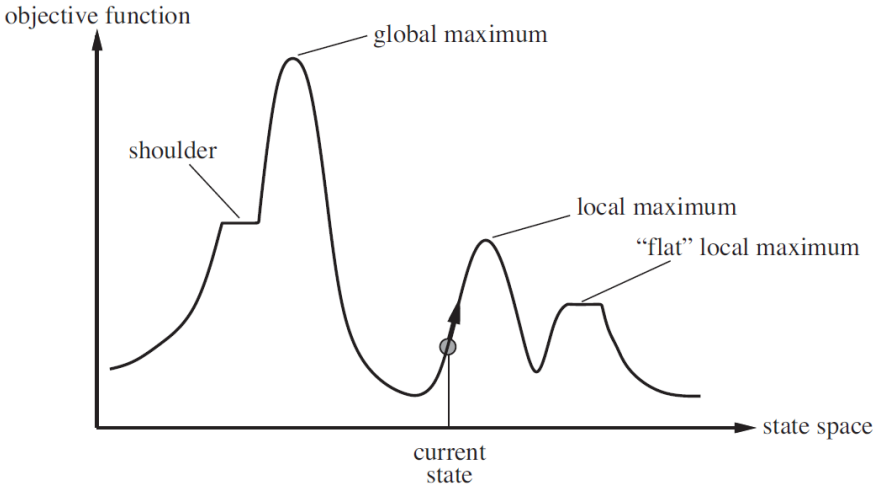
- Tìm kiếm cục bộ (local search)
- Thuật toán leo đồi (hill-climbing search)
- Thuật toán mô phỏng luyện kim (simulated annealing)
- Tìm kiếm Beam (local beam search)
- Thuật toán di truyền (genetic algorithm)
- Bài tập

Tìm kiếm cục bộ

- Các thuật toán tìm kiếm đã biết thiết kế để khám phá các không gian tìm kiếm một cách có hệ thống.
  - Giữ một hoặc nhiều đường trong bộ nhớ.
  - Ghi lại những thay thế đã được khám phá tại mỗi điểm.
  - Khi đích được tìm thấy, đường đến đích là một nghiệm.
- Nhiều bài toán đến đích không liên quan đến nghiệm mà đích chính là nghiệm.
  - Không quan tâm đến đường đi đến nghiệm, ví dụ bài toán tìm min/max của hàm số.
  - Với những bài toán, có thể dùng tìm kiếm cục bộ để giải quyết.

Tìm kiếm cục bộ

- Từ trạng thái hiện thời, di chuyển tới trạng thái tốt nhất trong trạng thái lân cận.



Tìm kiếm cục bộ

- Hoàn chỉnh (completeness)?
  - Tìm được nghiệm của bài toán nếu bài toán tồn tại nghiệm.
- Tối ưu (optimality)?
  - Luôn tìm được một cực trị (cực đại hoặc/cực tiểu) cục bộ (local maximum/minimum).
- Ưu điểm chính?
  - Sử dụng ít bộ nhớ - chỉ lưu giữ trạng thái hiện thời và các láng giềng của nó.
  - Có thể tìm ra nghiệm trong không gian trạng thái lớn hoặc vô hạn (liên tục).
- Có thể dùng để giải các bài toán tối ưu theo dạng tìm trạng thái tốt nhất theo một hàm mục tiêu.

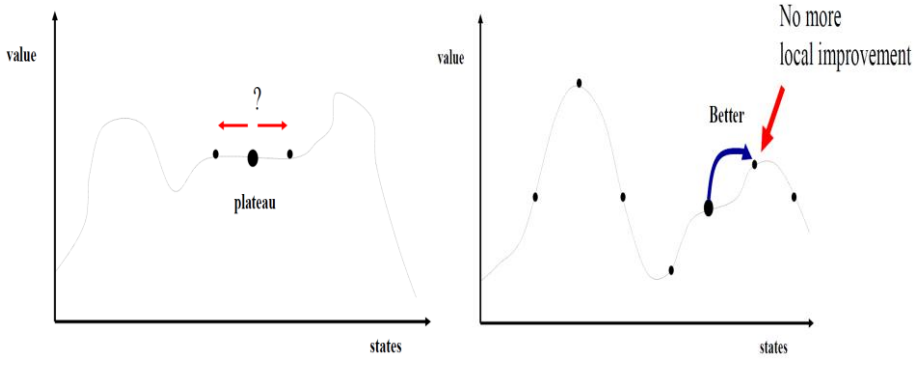
Thuật toán leo đồi

- Thuật toán leo đồi (hill – climbing search) chỉ có một vòng lặp trong đó trạng thái di chuyển theo hướng tăng của giá trị và dừng lại khi leo lên đến đỉnh đồi (steepest ascent version).
- Tên gọi khác: tìm kiếm tham lam cục bộ (greedy local search).

```
function HILL-CLIMBING(problem) returns a state that is a local maximum
  current ← MAKE-NODE(problem.INITIAL-STATE)
  loop do
    neighbor ← a highest-valued successor of current
    if neighbor.VALUE ≤ current.VALUE then return current.STATE
    current ← neighbor
```

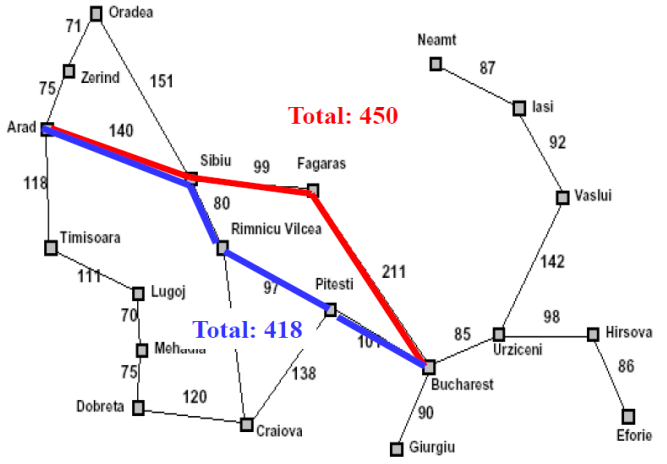
Thuật toán leo đồi

- Nhược điểm:
  - Có thể không tìm được tối ưu, hoặc
  - Chỉ tìm được tối ưu địa phương (cục bộ - local optimum).



Thuật toán leo đồi

- Ví dụ 1. bài toán tìm đường đi với $h(n)$ là khoảng cách Euclidean



Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Thuật toán leo đồi

- Ví dụ 2. bài toán 8 số với hàm heuristic $h(n)$ là tổng khoảng cách giữa các ô trong trạng thái hiện thời và trạng thái đích (G).

2	8	3
1	6	4
7		5

S

→

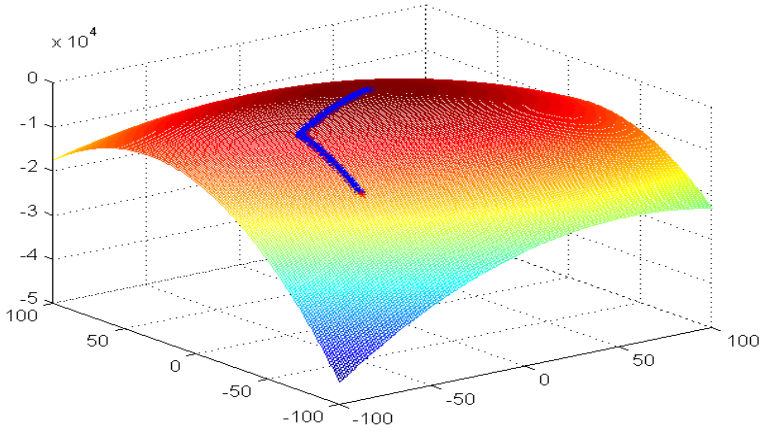
1	2	3
8		4
7	6	5

G

Thuật toán leo đồi

- Ví dụ 3. tìm giá trị lớn nhất/bé nhất của hàm số:

$$f = -[(x-5)^2 + 2(y-10)^2 + xy + x + 3y + 1]$$
$$x \in [-100, 100], y \in [-100, 100]$$

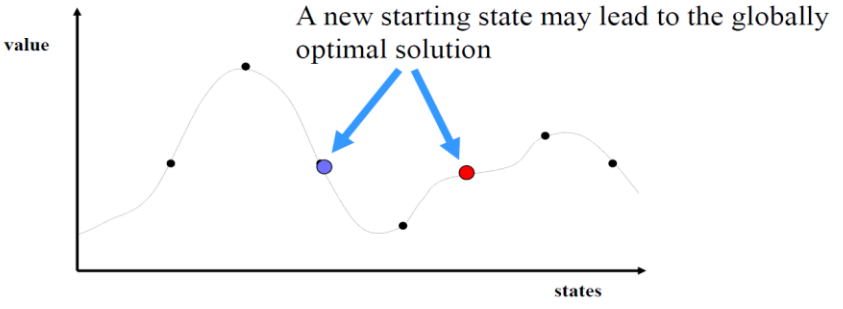


Thuật toán leo đồi

- Thuật toán leo đồi ngẫu nhiên (stochastic hill climbing).
  - Chọn ngẫu leo đồi theo một xác suất.
  - Xác suất chọn leo đồi có thể thay đổi khi di chuyển.
- Thuật toán leo đồi lựa chọn đầu tiên (first-choice hill climbing).
  - Thực hiện thuật toán leo đồi ngẫu nhiên bằng cách sinh các trạng thái ngẫu nhiên tiếp theo cho tới khi một trạng thái được sinh ra tốt hơn trạng thái hiện thời.
  - Đây là chiến lược tốt khi một trạng thái có rất nhiều láng giềng.

Thuật toán leo đồi

- Thuật toán leo đồi khởi tạo lại ngẫu nhiên (random-restart hill climbing)
  - Nhiều khởi động lại các thuật toán leo đồi từ trạng thái ban đầu khác nhau.
  - Lấy nghiệm tốt nhất của các thuật toán được khởi tạo tại các trạng thái ban đầu.

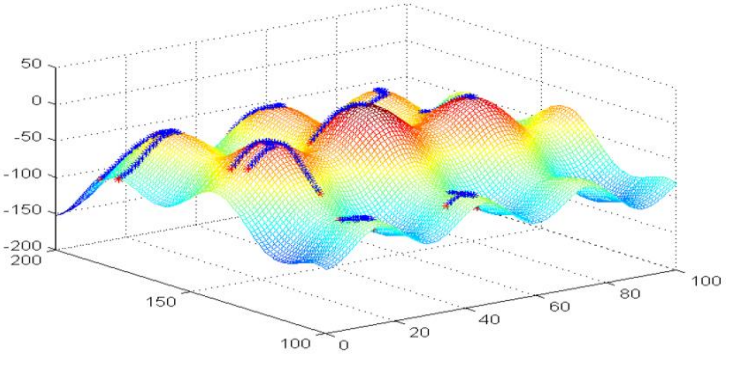


Thuật toán leo đồi

- Bài tập 1. lập trình tìm giá trị lớn nhất của hàm số:

$$f = -\left[\sqrt{(x-50)^2 + 4(y-150)^2} + 20\cos(2\pi x/30) + 20\cos(2\pi y/40)\right]$$

với x, y nguyên và $x \in [0,100]$, $y \in [100,200]$.

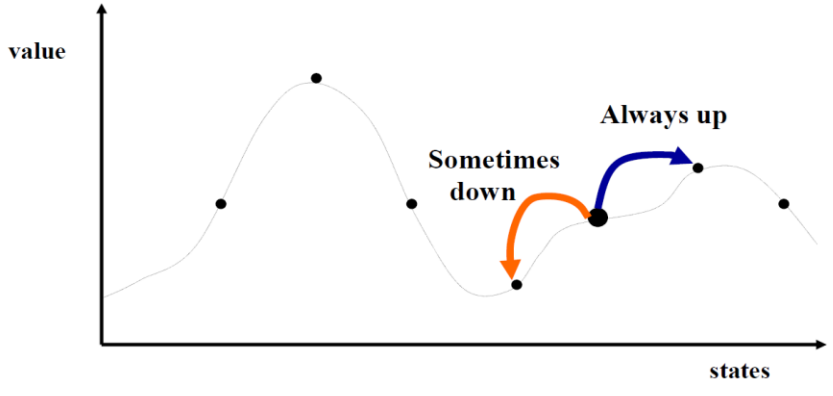


Thuật toán mô phỏng luyện kim

- Thuật toán leo đồi di chuyển tới trạng thái tốt nhất trong các trạng thái lân cận của trạng thái hiện thời.
  - Dẫn tới bị ”kẹt” trong một vùng cục bộ ⇒ thuật toán không tìm được nghiệm (không hoàn chỉnh).
- Thuật toán mô phỏng luyện kim (simulated annealing) lấy ý tưởng từ quá trình tôi ủ trong luyện kim.
  - Trong luyện kim, tôi ủ (annealing) là quá trình để làm cứng kim loại và thủy tinh bằng cách nung nóng chúng đến một nhiệt độ cao và sau đó làm lạnh dần dần chúng.

Thuật toán mô phỏng luyện kim

- Thuật toán mô phỏng luyện kim kết hợp leo đồi với di chuyển ngẫu nhiên.
- Cho phép di chuyển đến các trạng thái với nghiệm tồi hơn để có thể tránh được các điểm tối ưu cục bộ.



Thuật toán mô phỏng luyện kim

- Thay vì chọn trạng thái tốt nhất tiếp theo, chọn một trạng thái ngẫu nhiên:
  - Nếu trạng thái được chọn tốt hơn trạng thái hiện thời thì chuyển đến trạng thái đã chọn.
  - Ngược lại, chuyển đến trạng thái đã chọn với xác suất $e^{\Delta E/T}$.

```
function SIMULATED-ANNEALING(problem, schedule) returns a solution state
  inputs: problem, a problem
          schedule, a mapping from time to "temperature"

  current ← MAKE-NODE(problem.INITIAL-STATE)
  for t = 1 to ∞ do
      T ← schedule(t)
      if T = 0 then return current
      next ← a randomly selected successor of current
      ΔE ← next.VALUE – current.VALUE
      if ΔE > 0 then current ← next
      else current ← next only with probability e^(ΔE/T)
```

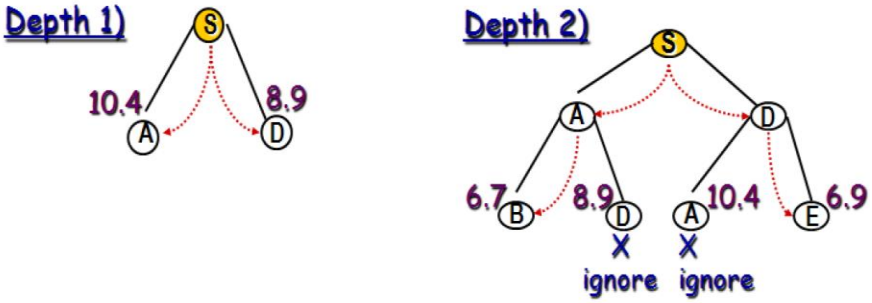
Thuật toán mô phỏng luyện kim

- Xác suất chọn trạng thái "tồi" giảm khi ΔE nhỏ hoặc khi nhiệt độ T cao.
- Nếu *schedule* làm T giảm đủ chậm, thuật toán sẽ tìm được một nghiệm tối ưu tổng thể với xác suất gần bằng 1.

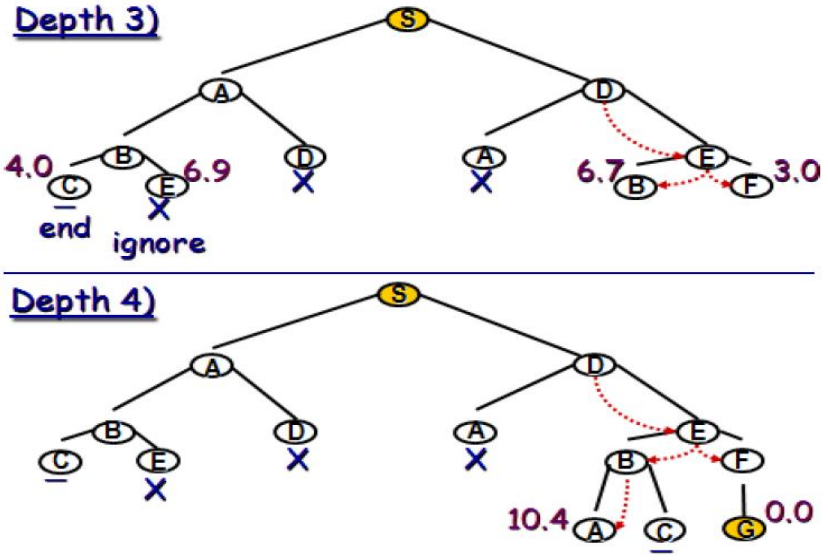
```
function SIMULATED-ANNEALING(problem, schedule) returns a solution state
  inputs: problem, a problem
          schedule, a mapping from time to "temperature"

  current ← MAKE-NODE(problem.INITIAL-STATE)
  for t = 1 to ∞ do
      T ← schedule(t)
      if T = 0 then return current
      next ← a randomly selected successor of current
      ΔE ← next.VALUE – current.VALUE
      if ΔE > 0 then current ← next
      else current ← next only with probability e^(ΔE/T)
```

Thuật toán tìm kiếm Beam

- Ý tưởng
  - Sinh ra k trạng thái ngẫu nhiên.
  - Sinh ra tất cả các các trạng thái tiếp theo của k trạng thái.
    - Nếu một trong các trạng thái được sinh ra là đích thì dừng.
    - Ngược lại chọn k trạng thái tốt nhất và lặp lại 2.



Thuật toán tìm kiếm Beam



Thuật toán di truyền

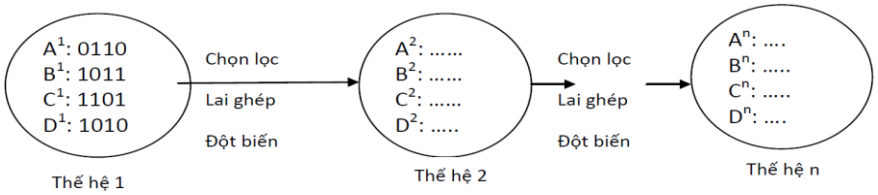
- Xuất xứ
  - Thuật toán di truyền (Genetic algorithm - GA) được đề xuất bởi John Holland trong thập niên 1960s.
  - Được phát triển bởi John Holland và các sinh viên của ông ở trường ĐH Michigan trong thời gian 1960s-1970s
- Ứng dụng
  - Lập lịch (Scheduling).
  - Robot (thiết kế, tìm đường đi,..)
  - Mạng máy tính (định tuyến đường truyền,...).
  - Trò chơi trên máy tính.

Thuật toán di truyền

- Ý tưởng bắt chước sự chọn lọc tự nhiên và di truyền.
- Mỗi cá thể có cấu trúc gen đặc trưng cho phẩm chất của cá thể đó.
- Các cá thể có khả năng thích nghi tốt với môi trường sẽ được tái sinh và nhân bản ở các thế hệ sau.
- Các cá thể con có thể thừa hưởng các phẩm chất của cả cha và mẹ.
- Trong quá trình tiến hóa, có thể xảy ra hiện tượng đột biến.
  - Cấu trúc gen của cá thể con có thể chứa các gen mà cả cha và mẹ đều không có.

Thuật toán di truyền

- Các khái niệm:
  - Cá thể (individual): được biểu diễn bởi cấu trúc gen, ví dụ được biểu diễn bởi một dãy các bit.
  - Quần thể (population): tập hợp các cá thể (individual).
  - Thế hệ (generation): một giai đoạn phát triển của một quần thể.
- Từ thế hệ ban đầu, thuật toán di truyền bắt chước chọn lọc (selection), sinh sản (reproduction) và đột biến (mutation) để tạo ra các thế hệ.



Thuật toán di truyền

- Các toán tử cơ bản để biến đổi các thế hệ:
  - Toán tử chọn lọc (selection): các cá thể tốt được chọn lọc để đưa vào thế hệ sau.
    - Sự chọn lọc được thực hiện dựa vào hàm thích nghi (fitness function) với môi trường của mỗi cá thể.
  - Toán tử sinh sản (reproduction): hai cá thể cha và mẹ trao đổi các gen để tạo ra một cá thể con.
  - Toán tử đột biến (mutation): Một cá thể thay đổi một số gen để tạo thành cá thể mới.

Thuật toán di truyền

```
function GENETIC-ALGORITHM(population, FITNESS-FN) returns an individual
  inputs: population, a set of individuals
          FITNESS-FN, a function that measures the fitness of an individual

  repeat
      new_population ← empty set
      for i = 1 to SIZE(population) do
          x ← RANDOM-SELECTION(population, FITNESS-FN)
          y ← RANDOM-SELECTION(population, FITNESS-FN)
          child ← REPRODUCE(x, y)
          if (small random probability) then child ← MUTATE(child)
          add child to new_population
      population ← new_population
  until some individual is fit enough, or enough time has elapsed
  return the best individual in population, according to FITNESS-FN
```

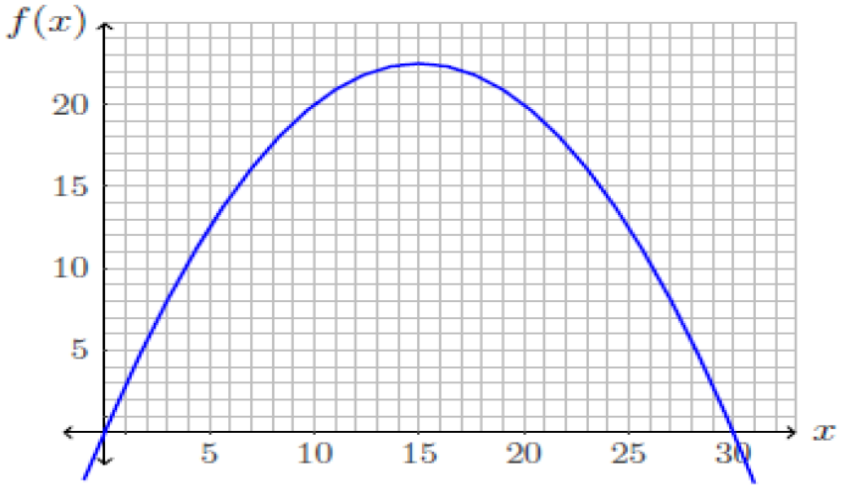
```
function REPRODUCE(x, y) returns an individual
  inputs: x, y, parent individuals

  n ← LENGTH(x); c ← random number from 1 to n
  return APPEND(SUBSTRING(x, 1, c), SUBSTRING(y, c + 1, n))
```

Thuật toán di truyền

- Hàm thích nghi là hàm mà thuật toán cần tối ưu.
- Điều kiện kết thúc vòng lặp:
  - Có thể là một số thế hệ đủ lớn.
  - Hoặc độ thích nghi của các cá thể tốt nhất trong các thế hệ kế tiếp nhau khác nhau không đáng kể.
- Khi thuật toán dừng, cá thể tốt nhất trong thế hệ cuối cùng được chọn làm nghiệm cần tìm.

Thuật toán di truyền

- Ví dụ 1. tìm giá trị lớn nhất của hàm số $f(x) = x^2/10 + 3x$ với x nguyên và $0 \le x \le 31$.



Thuật toán di truyền

- Mã hoá mỗi số nguyên $x \in [0,31]$ như một cá thể bởi một số nhị phân có độ dài 5 bits: 0 = 00000,1 = 00001, . . . , 31 = 11111.
- Định nghĩa hàm thích nghi, ví dụ hàm thích nghi là hàm $f(x) = -x^2/10 + 3x$.
- Định nghĩa quần thể ban đầu, ví dụ chọn ngẫu nhiên 10 cá thể gồm: 11, 26, 2, 14, 12, 30, 22, 9, 3, 17.
- Chọn lọc cá thể với xác suất được định nghĩa:

$$p(x_i) = \frac{f(x_i)}{\sum_{k=1}^{10} f(x_k)}, i = 1, 2, ..., 10$$

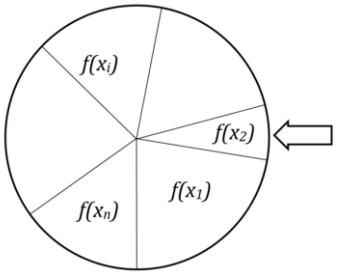
Thuật toán di truyền

- Các cá thể khởi tạo, giá trị hàm thích nghi và xác suất chọn lọc như bảng sau:

Chromosome Number	Initial Population	x Value	Fitness Value $f(x)$	Selection Probability
1	01011	11	20.9	0.1416
2	11010	26	10.4	0.0705
3	00010	2	5.6	0.0379
4	01110	14	22.4	0.1518
5	01100	12	21.6	0.1463
6	11110	30	0	0
7	10110	22	17.6	0.1192
8	01001	9	18.9	0.1280
9	00011	3	8.1	0.0549
10	10001	17	22.1	0.1497
		Sum	147.6	
		Average	14.76	
		Max	22.4	

Thuật toán di truyền

- Chọn lọc các cá thể bằng thủ tục quay bánh xe:
  - Có một cái bánh xe được chia thành n phần ứng với độ thích nghi của các cá thể và một mũi tên chỉ vào bánh xe.
  - Quay bánh xe, khi bánh xe dừng, mũi tên chỉ vào phần nào thì cá thể ứng với phần đó được chọn.
- Với cách chọn lọc này, các cá thể có độ thích nghi càng cao càng có khả năng được chọn ở thế hệ sau.



Thuật toán di truyền

- Thủ tục quay bánh xe (roulette wheel selection) 10 lần:
  - Tính tổng xác suất q_i cho mỗi cá thể x_i:

$$q_i = \sum_{j=1}^{i} p(x_j), i = 1, 2, ..., 10.$$

với

$$p(x_i) = \frac{f(x_i)}{\sum_{k=1}^{10} f(x_k)}, i = 1, 2, ..., 10$$

  - Sinh ra một số ngẫu nhiên $r \in [0,1]$.
  - Kiểm tra nếu $r < q_1$ thì chọn x_1, ngược lại chọn x_i để thõa mãn $q_{i-1} < r < q_i$.

Thuật toán di truyền

- Cặp đôi các cá thể ngẫu nhiên, với mỗi cặp, sinh ra một số nguyên ngẫu nhiên k trên đoạn [1, 5], k là vị trí điểm ghép.
- Cặp gồm hai cá thể:

$$x = (x_1 . . . x_k x_{k+1} . . . x_5),\ y = (y_1 . . . y_k y_{k+1} . . . y_5)$$

sinh ra một cá thể con:

$$child = (x_1 . . . x_k y_{k+1} . . . y_5)$$

hoặc sinh ra hai cá thể con:

$$child_1 = (x_1 . . . x_k y_{k+1} . . . y_5)$$
$$child_2 = (y_1 . . . y_k x_{k+1} . . . x_5)$$

Thuật toán di truyền

- Ví dụ sau khi chọn lọc và lai ghép lần lặp 1

Chromosome Number	Mating Pairs	New Population	x Value	Fitness Value $f(x)$
5	01|100	01010	10	20
2	11|010	11100	28	5.6
4	0111|0	01111	15	22.5
8	0100|1	01000	8	17.6
9	0001|1	01010	10	20
2	1101|0	11011	27	8.1
7	10110	10110	22	17.6
4	01110	01110	14	22.4
10	100|01	10001	17	22.1
8	010|01	01001	9	18.9
			Sum	174.8
			Average	17.48
			Max	22.5

Thuật toán di truyền

- Thực hiện toán tử đột biến trên các cá thể có được sau quá trình sinh sản.
  - Đột biến là thay đổi trạng thái một số gen nào đó của cá thể.
- Mỗi gen chịu đột biến với xác suất p_m (thường được chọn rất bé).
- Với mỗi vị trí *i* của cá thể $child = (x_1 \ldots x_i \ldots x_5)$ sinh ra một số thực ngẫu nhiên p_i trong [0,1].
- Đột biến *child* thành *child'*, trong đó:

$$child'_i = child_i \text{ nếu } p_i \geq p_m$$

và

$$child'_i = 1 - child_i \text{ nếu } p_i < p_m.$$

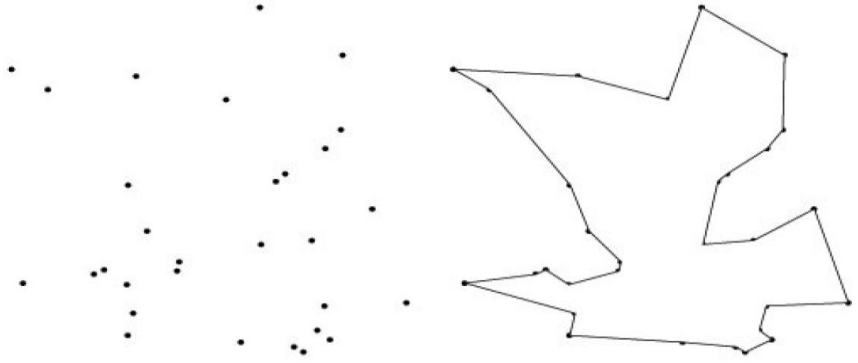
Thuật toán di truyền

- Giả sử chọn xác suất đột biến $p_m = 0$, khi đó sau vòng lặp 1, thuật toán tìm được giá trị lớn nhất của hàm số.

Chromosome Number	Mating Pairs	New Population	x Value	Fitness Value $f(x)$
5	01|100	01010	10	20
2	11|010	11100	28	5.6
4	0111|0	01111	15	22.5
8	0100|1	01000	8	17.6
9	0001|1	01010	10	20
2	1101|0	11011	27	8.1
7	10110	10110	22	17.6
4	01110	01110	14	22.4
10	100|01	10001	17	22.1
8	010|01	01001	9	18.9
			Sum	174.8
			Average	17.48
			Max	22.5

Thuật toán di truyền

- Ví dụ 2. bài toán người du lịch (traveling salesman problem)
  - Một người du lịch cần tìm đường đi ngắn nhất có thể để thăm *n* thành phố với mỗi thành phố thăm đúng một lần và trở về thành phố ban đầu.



Thuật toán di truyền

- Ký hiệu n thành phố là $c_1, c_2, \ldots, c_n$.
- Ký hiệu $d(c_i,c_j)$ là khoảng cách giữa thành phố c_i và c_j, $i \in [1,n]$, $j \in [1,n]$.
- Giả sử $d(c_i,c_j) = d(c_j,c_i)$ với $i \in [1,n]$, $j \in [1,n]$.
- Phương pháp truyền thống:
  - Tìm tất cả các chu trình và sau đó tìm chu trình ngắn nhất.
  - Số chu trình cần tìm là rất lớn: $(n-1)!/2$ chu trình.
  - Ví dụ nếu có 20 thành phố thì cần tìm 19!/2 = 60,822,550,204,416,000 chu trình.

Thuật toán di truyền

- Định nghĩa hàm thích nghi:

$$D = \sum_{k=1}^{n} d(c_k, c_{k+1}) \text{ với } c_{n+1} = c_1.$$

- Mỗi cá thể là một vector biểu diễn một hoán vị từ 1 đến n. Ví dụ với $n = 5$, các cá thể có thể là [3,5,1,2,4], [1,4,5,3,2].
- Nếu sử dụng toán tử sinh sản thông thường thì sẽ sinh ra lỗi. Ví dụ [3,5|1,2,4] lai ghép với [1,4|5,3,2] sinh ra [3,5,5,3,2] và [1,4,1,2,4].

Thuật toán di truyền

- Có nhiều cách định nghĩa toán tử sinh sản cho bài toán, chẳng hạn:
  - Chọn 1 vị trí ngẫu nhiên trong cá thể và 2 cá thể hoán đổi 2 số nguyên ở vị trí được chọn để tạo ra 2 cá thể mới.
  - Tiếp tục hoán đổi 2 số nguyên trong 2 cá thể tạo ra nếu bị trùng giá trị cho đến khi không có giá trị trùng trong mỗi cá thể.

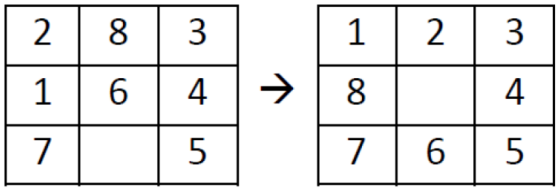
Parents	Offspring (step 1)	Offspring (step 2)	Offspring (step 3)	Offspring (step 4)
4 1 5|3|2 6	4 1 5 **2**|2|6	4|1|5 **2 1** 6	|4|**4 5 2 1** 6	**3 4 5 2 1** 6
3 4 6|2|1 5	3 4 6 **3**|1|5	3|4|6 **3 2** 5	|3|**1 6 3 2** 5	**4 1 6 3 2** 5

Thuật toán di truyền

- Tóm lại, để giải quyết một bài toán bằng thuật toán di truyền, cần thực hiện các bước:
  - Mã hóa các đối tượng cần tìm bởi một cấu trúc dữ liệu.
  - Thiết kế hàm thích nghi. Trong các bài toán tối ưu, hàm thích nghi được xác định dựa vào hàm mục tiêu.
  - Trên cơ sở cấu trúc của cá thể, thiết kế các toán tử lai ghép/sinh sản và đột biến.
  - Xác định cỡ của quần thể và khởi tạo quần thể ban đầu.
  - Xác định xác suất lai ghép và xác suất đột biến.

Bài tập

1) Sử dụng thuật toán tìm kiếm tham lam và thuật toán leo đồi tìm nghiệm của bài toán 8 số với hàm $h(n)$ là tổng khoảng cách giữa các ô trong trạng thái hiện thời và trạng thái đích (G).



S G

2) Vẽ không gian trạng thái và sử dụng thuật toán leo đồi để tìm giá trị lớn nhất của hàm số $f(x,y) = -[(x-1)^2 + y^2]$ với điều kiện x và y là các số nguyên và $-2 \leq x, y \leq 2$, biết trạng thái đầu s_0 = [0,-1] (tức là $x_0 = 0$ và $y_0 = -1$).