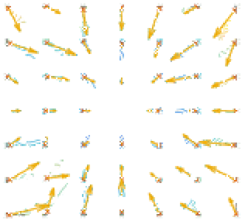
Optimization

Màster de Fonaments de Ciència de Dades

Gerard Gómez

Lecture VII. Two metaheuristic optimization methods:

- **Particle Swarm Optimization**
- **Ant Colony Optimization**



Particle Swarm Optimization

- **Particle swarm optimization** (PSO), developed around 1995 by Kennedy and Eberhart, is a meta-heuristic global optimization method, based on the concept of **swarm intelligence**.
- In analogy to the behavior of bird flocks and fish schools, in PSO the set of candidate solutions to the optimization problem is defined as a **swarm of particles** which may **flow through the parameter space defining trajectories** which are **driven by** their **own and neighbors' best performances**.
- The evolution of the trajectories is based on **cooperation and competition** among individuals **through generations (iterations)**.
- The **flow of information among particles**, which can be limited to a local neighborhood (partial PSO) or extended to the whole swarm (global PSO) is an **essential characteristic** of the algorithm.

Particle Swarm Optimization. The basic algorithm

Consider a maximization problem: given $f : D \subset \mathbb{R}^n \to \mathbb{R}$, find $\boldsymbol{x}^*$ such that

$$f(\boldsymbol{x}^*) \geq f(\boldsymbol{x}), \quad \forall \boldsymbol{x} \in D$$

1. **Initialization**

   Select a certain number N of particles, and for each of the particles:

   1.1 Initialize, for $t = 0$, the position $\boldsymbol{x}_i(0)$, and the velocity $\boldsymbol{v}_i(0)$ for $i = 1, ..., N$

   1.2 Calculate the fitness of each particle $f(\boldsymbol{x}_i(0))$, for $i = 1, ..., N$

   1.3 Initialize the particle's best position to its initial position $\boldsymbol{p}_i(0) = \boldsymbol{x}_i(0)$, for $i = 1, ..., N$

   1.4 Initialize the global best as

   $$\boldsymbol{g}(0) = \boldsymbol{x}_j(0)$$

   if

   $$f(\boldsymbol{x}_j(0)) \geq f(\boldsymbol{x}_k(0)), \quad \text{for all} \quad k = 1, ..., N, \ k \neq i$$

Particle Swarm Optimization. The basic algorithm

2. **Until a stopping criterion is met, repeat the following steps**

   2.1 Update the particle velocity according to

   $$\boldsymbol{v}_i(t+1) = w\,\boldsymbol{v}_i(t) + c_1\boldsymbol{R}_1(\boldsymbol{p}_i(t) - \boldsymbol{x}_i(t)) + c_2\boldsymbol{R}_2(\boldsymbol{g}(t) - \boldsymbol{x}_i(t))$$

   where

   - $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ are two diagonal matrices of random numbers, generated from a uniform distribution in [0,1], updated at each step
   - w, c_1 and c_2 are real-valued fixed constants during all the iterations, usually $w \in [0.5, 0.9]$, $0 \leq c_1,\ c_2 \leq 4$

   2.2 Update the particle position according to

   $$\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t+1)$$

   2.3 Evaluate the fitness of each particle $f(\boldsymbol{x}_i(t+1))$ for $i = 1, ..., N$

   2.4 Test the stopping criterion, and if the stopping criterion is satisfied stop, else continue

   2.5 If $f(\boldsymbol{x}_i(t+1)) \geq f(\boldsymbol{p}_i(t))$, update the particle best position: $\boldsymbol{p}_i(t+1) = \boldsymbol{x}_i(t+1)$

   2.6 If $f(\boldsymbol{x}_i(t+1)) \geq f(\boldsymbol{g}(t))$, update the global best position: $\boldsymbol{g}(t+1) = \boldsymbol{x}_i(t+1)$

Particle Swarm Optimization. The basic algorithm

3. At the end of the iterative process, the best solution is given by $\boldsymbol{g}$

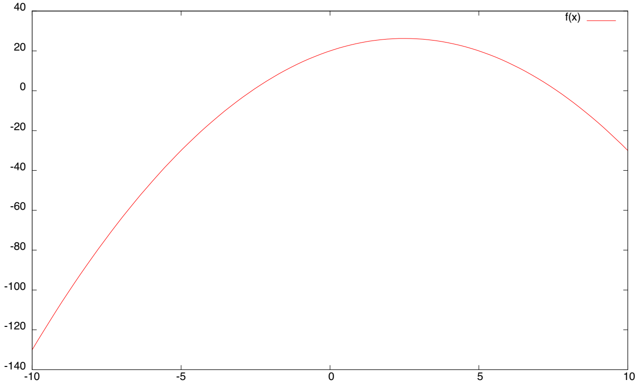
Remarks

- The *inertia weight* w reduces the velocities over time (iterations). If $w \geq 1$ the velocities increase over time and particles can hardly change their direction to move back towards optimum, then the swarm diverges. If $w \ll 1$, then little information is used from the previous step and quick changes of direction appear in the process.
- The constants c_1 and c_2 are called *cognitive coefficient* and *social coefficient*, and modulate the magnitude of the steps taken by the particle in the direction of its personal best and global best respectively.
- The two diagonal matrices $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ of random numbers give a stochastic influence on both the social and the cognitive components of the velocity of each particle.
  Accordingly, the trajectories drawn by the particles are semi-random, as they derive from the contribution of systematic attraction towards the personal and global best solutions and stochastic weighting of these two acceleration terms.

Particle Swarm Optimization

Example. Find the maximum of the function

$$f(x) = -x^2 + 5x + 20 \quad \text{with} \quad -10 \le x \le 10$$



The solution is $x^* = 2.5$, $f(x^*) = 26.25$

In the PSO procedure, we have set for this first example $w = 1$, $c_1 = c_2 = 1$

Example. Initialization $t = 0$

Step 1.1 Initialize the position and the velocity of the population at $t = 0$

$$\begin{array}{lllllllll} x_1(0) & = & -9.6000, & x_2(0) & = & -6.0000, & x_3(0) & = & -2.6000 \\ x_4(0) & = & -1.1000, & x_5(0) & = & 0.6000, & x_6(0) & = & 2.3000 \\ x_7(0) & = & 2.8000, & x_8(0) & = & 8.3000, & x_9(0) & = & 10.0000 \end{array}$$

$$v_1(0) = v_2(0) = v_3(0) = v_4(0) = v_5(0) = v_6(0) = v_7(0) = v_8(0) = v_9(0) = 0$$

Step 1.2 Calculate the fitness of each particle

$$\begin{array}{lllllllll} f(x_1(0)) & = & -120.1600, & f(x_2(0)) & = & -46.0000, & f(x_3(0)) & = & 0.2400 \\ f(x_4(0)) & = & 13.2900, & f(x_5(0)) & = & 22.6400, & f(x_6(0)) & = & 26.2100 \\ f(x_7(0)) & = & 26.1600, & f(x_8(0)) & = & -7.3900, & f(x_9(0)) & = & -30.0000 \end{array}$$

Step 1.3 Initialize the personal best of each particle

$$\begin{array}{lllllllll} p_1(0) & = & -9.6000, & p_2(0) & = & -6.0000, & p_3(0) & = & -2.6000 \\ p_4(0) & = & -1.1000, & p_5(0) & = & 0.6000, & p_6(0) & = & 2.3000 \\ p_7(0) & = & 2.8000, & p_8(0) & = & 8.3000, & p_9(0) & = & 10.0000 \end{array}$$

Step 1.4 Initialize the global best, which is

$$g(0) = x_6(0) = 2.3000$$

since

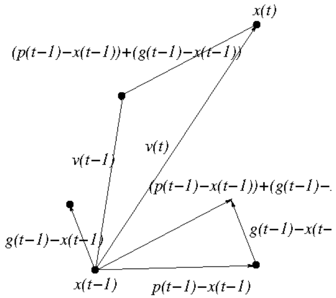
$$f(x_6(0)) \geq f(x_i(0)), \quad \text{for} \quad i = 1, ..., 9$$

Example. First iteration $t = 1$

Step 2.1 Compute two random numbers, $R_1 = 0.213$, $R_2 = 0.876$, in $[0, 1]$, and update the particles velocity according to

$$v_i(t) = w\, v_i(t-1) + c_1 R_1\, [p_i(t-1) - x_i(t-1)] + c_2 R_2\, [g(t-1) - x_i(t-1)]$$

$$\begin{array}{lcllr}
v_1(1) & = & 0 + 0.213\,(-9.6 + 9.6) + 0.876\,(2.3 + 9.6) & = & 10.4244 \\
v_2(1) & = & 0 + 0.213\,(-6 + 6) + 0.876\,(2.3 + 6) & = & 7.2708 \\
v_3(1) & = & 0 + 0.213\,(-2.6 + 2.6) + 0.876\,(2.3 + 2.6) & = & 4.2924 \\
v_4(1) & = & 0 + 0.213\,(-1.1 + 1.1) + 0.876\,(2.3 + 1.1) & = & 2.9784 \\
v_5(1) & = & 0 + 0.213\,(0.6 - 0.6) + 0.876\,(2.3 - 0.6) & = & 1.4892 \\
v_6(1) & = & 0 + 0.213\,(2.3 - 2.3) + 0.876\,(2.3 - 2.3) & = & 0.0000 \\
v_7(1) & = & 0 + 0.213\,(2.8 - 2.8) + 0.876\,(2.3 - 2.8) & = & -0.4380 \\
v_8(1) & = & 0 + 0.213\,(8.3 - 8.3) + 0.876\,(2.3 - 8.3) & = & 5.2560 \\
v_9(1) & = & 0 + 0.213\,(10 - 10) + 0.876\,(2.3 - 10) & = & -6.7452
\end{array}$$



Example. First iteration $t = 1$

Step 2.2 Update the particles position according to

$$x_i(t) = x_i(t-1) + v_i(t)$$

$$\begin{array}{llllll} x_1(1) = & 0.8244, & x_2(1) = & 1.2708, & x_3(1) = & 1.6924 \\ x_4(1) = & 1.8784, & x_5(1) = & 2.0892, & x_6(1) = & 2.3000 \\ x_7(1) = & 2.3620, & x_8(1) = & 3.0440, & x_9(1) = & 3.2548 \end{array}$$

Step 2.3 Calculate the fitness of each particle $x_i(1)$ using

$$f(x) = -x^2 + 5x + 20$$

$$\begin{array}{llllll} f(x_1) = & 23.4424, & f(x_2) = & 24.7391, & f(x_3) = & 25.5978 \\ f(x_4) = & 25.8636, & f(x_5) = & 26.0812, & f(x_6) = & 26.2100 \\ f(x_7) = & \color{red}{26.2310}, & f(x_8) = & 25.9541, & f(x_9) = & 25.6803 \end{array}$$

Step 2.4 If the stopping criterion is satisfied stop, else continue

Example. First iteration $t = 1$

Step 2.5 Update the personal best for each particle, $i = 1, \dots, 9$ according to

$$p_i(1) = \begin{cases} p_i(0) & \text{if} \quad f(x_i(1)) < f(p_i(0)) \\ x_i(1) & \text{if} \quad f(x_i(1)) \geq f(p_i(0)) \end{cases}$$

$$\begin{array}{llllll} p_1(1) = & 0.8244, & p_2(1) = & 1.2708, & p_3(1) = & 1.6924 \\ p_4(1) = & 1.8784, & p_5(1) = & 2.0892, & p_6(1) = & 2.3000 \\ p_7(1) = & 2.362, & p_8(1) = & 3.0440, & p_9(1) = & 3.2548 \end{array}$$

Step 2.6 Find the global best

$$g(1) = x_7(1) = 2.3620$$

Example. Second iteration $t = 2$

Step 2.1 Compute two random numbers $R_1 = 0.113$, $R_2 = 0.706$ in $[0, 1]$, and update the particles velocity according to

$$v_i(t) = w\, v_i(t-1) + c_1 R_1 \left[p_i(t-1) - x_i(t-1)\right] + c_2 R_2 \left[g(t-1) - x_i(t-1)\right]$$

$$\begin{array}{rcrrcrrcr}
v_1(2) & = & 11.5099, & v_2(2) & = & 8.0412, & v_3(2) & = & 4.7651 \\
v_4(2) & = & 3.3198, & v_5(2) & = & 1.6818, & v_6(2) & = & 0.0438 \\
v_7(2) & = & -0.4380, & v_8(2) & = & -5.7375, & v_9(2) & = & -7.3755
\end{array}$$

Step 2.2 Find the new values of $x_i(2)$ for $i = 1, ..., 9$ using

$$x_i(2) = x_i(1) + v_i(2)$$

$$\begin{array}{rcrrcrrcr}
x_1(2) & = & 12.3343, & x_2(2) & = & 9.3120, & x_3(2) & = & 6.4575 \\
x_4(2) & = & 5.1982, & x_5(2) & = & 3.7710, & x_6(2) & = & 2.3438 \\
x_7(2) & = & 1.9240, & x_8(2) & = & -2.6935, & x_9(2) & = & -4.1207
\end{array}$$

Example. Second iteration $t = 2$

Step 2.3 Calculate the fitness of each particle $x_i(2)$ using

$$f(x) = -x^2 + 5x + 20$$

$$\begin{array}{lcrlcrlcr}
f_1^2 & = & -70.4644, & f_2^2 & = & -20.1532, & f_3^2 & = & 10.5879 \\
f_4^2 & = & 18.9696, & f_5^2 & = & 24.6346, & f_6^2 & = & 26.2256 \\
f_7^2 & = & 25.9182, & f_8^2 & = & -0.7224, & f_9^2 & = & -17.5839
\end{array}$$

Step 2.4 If the stopping criterion is satisfied stop, else continue

Step 2.5 Update the personal best for each particle, $i = 1, ..., 9$

$$\begin{array}{lcrlcrlcr}
p_1(2) & = & 0.8244, & p_2(2) & = & 1.2708, & p_3(2) & = & 1.6924 \\
p_4(2) & = & 1.8784, & p_5(2) & = & 2.0892, & p_6(2) & = & 2.3438 \\
p_7(2) & = & 2.3620, & p_8(2) & = & 3.0440, & p_9(2) & = & 3.2548
\end{array}$$

Step 2.6 Find the global best

$$g(2) = x_7(1) = 2.3620$$

Example. Third iteration $t = 3$

Step 2.1 Compute two random numbers $r_1 = 0.178$, $r_2 = 0.507$ in $[0, 1]$, and update the particles velocity according to

$$v_i(t) = w\, v_i(t-1) + c_1 R_1 \left[p_i(t-1) - x_i(t-1)\right] + c_2 R_2 \left[g(t-1) - x_i(t-1)\right]$$

$$\begin{array}{lclclclcl} v_1(3) & = & 4.4052, & v_2(3) & = & 3.0862, & v_3(3) & = & 1.8405 \\ v_4(3) & = & 1.2909, & v_5(3) & = & 0.6681, & v_6(3) & = & 0.0530 \\ v_7(3) & = & -0.1380, & v_8(3) & = & -2.1531, & v_9(3) & = & -2.7759 \end{array}$$

Step 2.2 Find the new values of $x_i(3)$ for $i = 1, ..., 9$ using

$$x_i(3) = x_i(2) + v_i(3)$$

$$\begin{array}{lclclclcl} x_1(3) & = & 16.7395, & x_2(3) & = & 12.3982, & x_3(3) & = & 8.2980 \\ x_4(3) & = & 6.4892, & x_5(3) & = & 4.4391, & x_6(3) & = & 2.3968 \\ x_7(3) & = & 1.7860, & x_8(3) & = & -4.8466, & x_9(3) & = & -6.8967 \end{array}$$

Example. Third iteration $t = 3$

Step 2.3 Calculate the fitness of each particle $x_i(2)$ using

$$f(x) = -x^2 + 5x + 20$$

$$\begin{array}{rclrclrcl}
f_1(3) & = & -176.5145, & f_2(3) & = & -71.7244, & f_3()3 & = & -7.3673 \\
f_4(3) & = & 10.3367, & f_5(3) & = & 22.4900, & f_6(3) & = & 26.2393 \\
f_7(3) & = & 25.7402, & f_8(3) & = & -27.7222, & f_9(3) & = & -62.0471
\end{array}$$

Step 2.4 If the stopping criterion is satisfied stop, else continue

Step 2.5 Update the personal best for each particle, $i = 1, ..., 9$

$$\begin{array}{rclrclrcl}
p_1(3) & = & 0.8244, & p_2(3) & = & 1.2708, & p_3(3) & = & 1.6924 \\
p_4(3) & = & 1.8784, & p_5(3) & = & 2.0892, & p_6(3) & = & 2.3968 \\
p_7(3) & = & 2.3620, & p_8(3) & = & 3.0440, & p_9(3) & = & 3.2548
\end{array}$$

Step 2.6 Find the global best

$$g(3) = x_6(3) = 2.3968$$

Example. Iterations $t = 4, \ldots, 8$

```
Iteration          4
x    -2.8614   -1.2849    0.2040    0.8609    1.6054    2.4498    2.5688    4.9774    5.7218
v   -19.6009  -13.6831   -8.0940   -5.6282   -2.8337    0.0530    0.7828    9.8240   12.6185
f    -2.4945   11.9247   20.9786   23.5634   25.4497   26.2475   26.2453   20.1126   15.8698
p     0.8244    1.2708    1.6924    1.8784    2.0892    2.4498    2.5688    3.0440    3.2548
g     2.4498
Iteration          5
x   -14.9298   -9.7061   -4.7727   -2.5961   -0.1294    2.5028    3.2626   11.0434   13.5101
v   -12.0684   -8.4212   -4.9767   -3.4571   -1.7348    0.0530    0.6938    6.0660    7.7883
f -277.5471 -122.7393  -26.6417    0.2793   19.3362   26.2500   25.6685  -46.7398  -94.9730
p     0.8244    1.2708    1.6924    1.8784    2.0892    2.5028    2.5688    3.0440    3.2548
g     2.5028
Iteration          6
x   -12.8315   -8.2265   -3.8773   -1.9586    0.2160    2.5558    3.3363   10.0656   12.2402
v     2.0983    1.4796    0.8954    0.6376    0.3454    0.0530    0.0737   -0.9778   -1.2699
f -208.8041  -88.8073  -14.4201    6.3712   21.0334   26.2469   25.5506  -30.9884  -68.6212
p     0.8244    1.2708    1.6924    1.8784    2.0892    2.5028    2.5688    3.0440    3.2548
g     2.5028
Iteration          7
x    -9.6439   -5.9889   -2.5370   -1.0141    0.7119    2.6046    3.3489    8.5294   10.2554
v     3.1876    2.2376    1.3403    0.9445    0.4958    0.0488    0.0126   -1.5362   -1.9848
f -121.2237  -45.8116    0.8787   13.9011   23.0526   26.2390   25.5293  -10.1040  -33.8960
p     0.8244    1.2708    1.6924    1.8784    2.0892    2.5028    2.5688    3.0440    3.2548
g     2.5028
Iteration          8
x     1.3342    1.6746    1.9961    2.1379    2.2987    2.5833    2.8012    3.0268    3.1875
v    10.9781    7.6635    4.5331    3.1520    1.5868   -0.0214   -0.5477   -5.5026   -7.0678
f    24.8909   25.5687   25.9961   26.1189   26.2095   26.2431   26.1593   25.9725   25.7773
p     1.3342    1.6746    1.9961    2.1379    2.2987    2.5028    2.5688    3.0268    3.1875
g     2.5028
```

Example. Iterations $t = 9, ..., 13$

```
Iteration          9
x   12.4679   9.4484   6.5967   5.3386   3.9127   2.5336   2.1631  -2.5456  -3.9715
v   11.1337   7.7738   4.6006   3.2006   1.6140  -0.0496  -0.6381  -5.5724  -7.1590
f  -73.1086 -22.0303   9.4672  18.1926  24.2543  26.2489  26.1365   0.7917 -15.6301
p    1.3342   1.6746   1.9961   2.1379   2.2987   2.5028   2.5688   3.0268   3.1875
g    2.5028
Iteration         10
x    9.7213   7.5354   5.4710   4.5602   3.5280   2.4443   2.0217  -1.1474  -2.1796
v   -2.7466  -1.9130  -1.1257  -0.7784  -0.3847  -0.0893  -0.1414   1.3982   1.7919
f  -25.8966   0.8949  17.4234  22.0057  25.1933  26.2469  26.0213  12.9467   4.3514
p    1.3342   1.6746   1.9961   2.1379   2.2987   2.5028   2.5688   3.0268   3.1875
g    2.5028
Iteration         11
x   -1.5398  -0.3197   0.8325   1.3409   1.9170   2.4198   2.4425   4.5266   5.1027
v  -11.2610  -7.8551  -4.6384  -3.2193  -1.6109  -0.0245   0.4208   5.6739   7.2823
f    9.9303  18.2991  23.4696  24.9065  25.9101  26.2436  26.2467  22.1430  19.4760
p    1.3342   1.6746   1.9961   2.1379   2.2987   2.5028   2.4425   3.0268   3.1875
g    2.5028
Iteration         12
x  -10.1234  -6.3057  -2.7000  -1.1093   0.6935   2.4506   2.9025   8.8592  10.6620
v   -8.5836  -5.9859  -3.5326  -2.4502  -1.2235   0.0308   0.4600   4.3326   5.5593
f -133.0994 -51.2895  -0.7903  13.2229  22.9866  26.2476  26.0880 -14.1891 -40.3680
p    1.3342   1.6746   1.9961   2.1379   2.2987   2.5028   2.4425   3.0268   3.1875
g    2.5028
Iteration         13
x   -7.2226  -4.2880  -1.5165  -0.2938   1.0920   2.5320   2.9361   7.3687   8.7544
v    2.9008   2.0176   1.1835   0.8155   0.3985   0.0814   0.0336  -1.4905  -1.9076
f  -68.2790 -19.8276  10.1175  18.4447  24.2675  26.2490  26.0598   2.5462 -12.8678
p    1.3342   1.6746   1.9961   2.1379   2.2987   2.5028   2.4425   3.0268   3.1875
g    2.5028
```

Particle Swarm Optimization. Additional comments

1. **The stopping criterion**

   The stopping criterion mainly depends on the problem and can be:

   1.1 A prespecified total number of iterations
   1.2 A maximum number of iterations since the last update of global best
   1.3 A predefined target value of the fitness function

2. **Position and velocity initialization**

   PSO requires an initial estimate of the positions $\boldsymbol{x}_i = (x_{i,1}, \dots, x_{i,n})$ and velocities $\boldsymbol{v}_i = (v_{i,1}, \dots, v_{i,n})$ of the particles.
   The way $\boldsymbol{x}_i$ and $\boldsymbol{v}_i$ are initialized has an important role in the probability that particles travel outside the feasible set.
   The best option is that the initial particles' positions cover as uniformly as possible the feasible set

   $$x_{i,j}(0) \sim U(x_{j,min}, x_{j,max}), \quad i = 1, \dots, N \quad j = 1, \dots, n$$

   A good option to set the initial velocities to zero, or to very small random numbers, as the exploration ability is still guaranteed by the choice of the initial positions

Particle Swarm Optimization. Additional comments

3. **Choice of the inertia weight** w

   The inertia weight can be implemented either as a fixed value or dynamically changing values (which is much better).

   The large inertia value is high at first, which allows all particles to move freely in the search space at the initial steps, and decreases over time.

   Usually, the inertia weight value decreases linearly with the iteration number according to

   $$w^{t+1} = w_{max} - \frac{w_{\max} - w_{\min}}{t_{\max}} t$$

   with $w_{\max} \approx 0.9$ and $w_{\max} \approx 0.3$.

4. **Choice of the acceleration constants** c_1 **and** c_2

   c_1 and c_2 govern the extent to which the particles move towards the individual and global best particle, modulating the relative contributions of the social and cognitive terms In general, it has been shown that the conditions

   $$c_1 = c_2 = 2$$

   work well for most of the applications.

Particle Swarm Optimization. Additional comments

5. **Avoiding the velocity explosion**

   If we take $c_1 = c_2 = 2$, then both the terms $c_1 \boldsymbol{R}_1$ and $c_2 \boldsymbol{R}_2$ will be uniformly distributed in [0, 2] with average value equal to 1. As a consequence, it may happen that the trajectory of a particle crosses the boundaries of the feasible set.
   To avoid this situation, a velocity threshold is introduced in the algorithm, so that

   $$\begin{array}{cccc} \text{if} & v_{i,j} > v_j^{\max} & \text{then} & v_{i,j} = v_j^{\max} \\ \text{if} & v_{i,j} < -v_j^{\max} & \text{then} & v_{i,j} = -v_j^{\max} \end{array}$$

   where

   $$v_j^{\max} = k \frac{x_j^{\max} - x_j^{\min}}{2}, \quad j = 1, \dots, n, \quad k \in (0, 1]$$

Particle Swarm Optimization. Additional comments

5. **Swarm population**

   The size N of the population is another factor that has an impact on the performances of the PSO algorithm.
   A large population increases the computational efforts but also the diversity of the swarm and its exploration ability; it also increases the probability of premature convergence.
   In most cases it has been demonstrated that when the number of individuals is larger than 50, PSO is not sensitive to the size of the population

Particle Swarm Optimization. Additional comments

6. **Network topology**

   The basic PSO algorithm, may be easily trapped in a local optimum.

   Indeed, fast convergence is often achieved as all the particles tend to be attracted simultaneously to the portion of the search space where the global best is.

   If the global optimum is not close to the best particle, this characteristic may hinder the possibility of the swarm to explore other areas.

   One way of limiting the probability of a premature convergence to local optima is to define the social component of the velocity update equation not in terms of the global best g but just based on the best known position l (local best) of a sub-swarm "around" the particle that is moved (its neighborhood).

   The advantage of a local best swarm (partial PSO) is that while neighbors are closely connected, the individuals that are topologically distant are also relatively independent of one another, so they may search different portions of the feasible set or explore different local optima without the overall swarm being trapped in any of them.

Example. Iterations $t = 10, ..., 14$

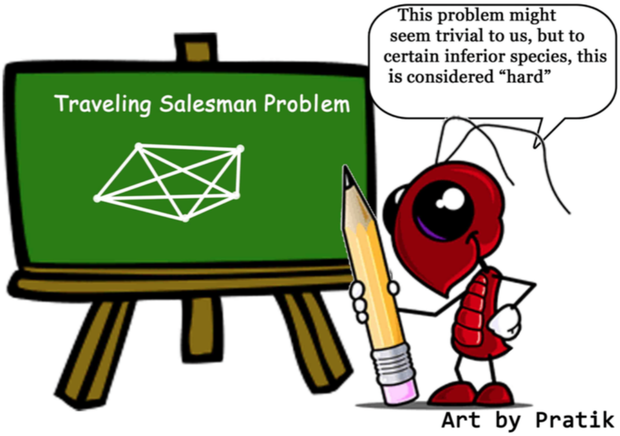
Values are used for the velocity computation: w=0.5, $c_1 = 0.5$, $c_2 = 0.9$

```
Iteration          10
 x     3.0562    2.7475    2.6576    2.6180    2.5875    2.4924    2.4811    2.2615    2.1860
 v     0.1815   -0.0230   -0.0145   -0.0107   -0.0307    0.0174    0.0374   -0.0953   -0.1216
 f    25.9406   26.1888   26.2252   26.2361   26.2424   26.2499   26.2496   26.1931   26.1514
 p     2.8747    2.7475    2.3535    2.4079    2.5142    2.4925    2.4973    2.3568    2.3076
 g     2.5270
 Iteration          11
 x     2.6310    2.5420    2.4512    2.4744    2.4986    2.5316    2.5447    2.4739    2.4590
 v    -0.4252   -0.2055   -0.2064   -0.1436   -0.0888    0.0392    0.0636    0.2124    0.2730
 f    26.2328   26.2482   26.2476   26.2493   26.2500   26.2490   26.2480   26.2493   26.2483
 p     2.6310    2.5420    2.4512    2.4744    2.4986    2.4925    2.4973    2.4739    2.4590
 g     2.5270
 Iteration          12
 x     2.3569    2.4304    2.3929    2.4338    2.4711    2.5308    2.5447    2.6116    2.6358
 v    -0.2742   -0.1116   -0.0583   -0.0406   -0.0276   -0.0008   -0.0000    0.1377    0.1768
 f    26.2295   26.2452   26.2385   26.2456   26.2492   26.2490   26.2480   26.2375   26.2316
 p     2.6310    2.5420    2.4512    2.4744    2.4986    2.4925    2.4973    2.4739    2.4590
 g     2.5270
 Iteration          13
 x     2.3813    2.4503    2.4336    2.4620    2.4876    2.5151    2.5210    2.5996    2.6204
 v     0.0245    0.0199    0.0407    0.0283    0.0165   -0.0157   -0.0237   -0.0120   -0.0154
 f    26.2359   26.2475   26.2456   26.2486   26.2498   26.2498   26.2496   26.2401   26.2355
 p     2.3813    2.5420    2.4512    2.4744    2.4986    2.4925    2.4973    2.4739    2.4590
 g     2.5270
 Iteration          14
 x     2.4130    2.5130    2.4745    2.4906    2.5062    2.4983    2.4990    2.5256    2.5253
 v     0.0316    0.0627    0.0410    0.0285    0.0187   -0.0168   -0.0220   -0.0740   -0.0951
 f    26.2424   26.2498   26.2494   26.2499   26.2500   26.2500   26.2500   26.2493   26.2494
 p     2.4130    2.5130    2.4745    2.4906    2.4986    2.4983    2.4990    2.5256    2.5253
 g     2.5270
```

Ant Colony Optimization[1]

[1] Introduced by Marco Dorigo in his PhD thesis in 1992

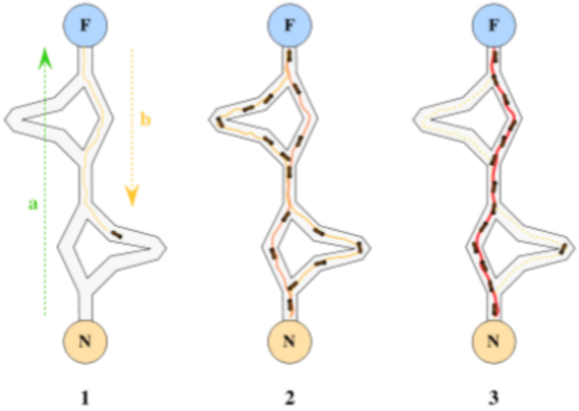
Ant Colony Optimization


Traveling Salesman Problem
This problem might seem trivial to us, but to certain inferior species, this is considered "hard"
Art by Pratik

Ant Colony Optimization

- Shortest path is discovered via pheromone trails.
- Ants navigate from nest to food source. Ants are blind!
- While moving, ants leave a chemical pheromone trail on the ground.
- Ants can smell pheromone.
- When choosing their way, they tend to choose, in probability, paths marked by strong pheromone concentrations.
- As soon as an ant finds a food source, it evaluates the quantity and the quality of the food and carries some of it back to the nest.
- During the return trip, the quantity of pheromone that an ant leaves on the ground may depend on the quantity and quality of the food.
- The pheromone trails will guide other ants to the food source.

Ant Colony Optimization



A colony of ants has several paths to go from the nest **N** to the food **F**. After some time, almost all endup using the shortest one

Ant Colony Optimization

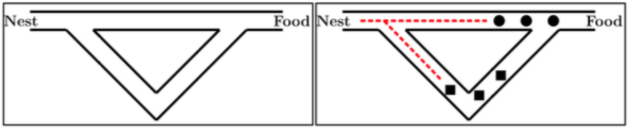
- Pheromones accumulate on path segments
- Path is selected at random based on the amount of ”trail” present on possible paths from starting node
- Ant reaches next node, selects next path
- Continues until reaches the final node
- The finished tour is a solution.
- Tour is analyzed for optimality

Ant Colony Optimization. A simplified model

Consider the following discretized and simplified model defined by a graph

$$G = (N, L)$$

- N consists of two nodes, namely n_d (representing the nest of the ants), and n_f (representing the food source)
- L consists of two links, namely e_1 and e_2, between n_d and n_f.



50% of the ants take the short path (circles) and 50% the long path (circles)

- To e_1 we assign a length of l_1, and to e_2 a length of e_2, such that $l_2 > l_1$
- Since ants deposit pheromone on the paths on which they move, the chemical pheromone trails must be modeled.
- We introduce an artificial pheromone value τ_i for each of the two links e_1 and e_2 indicating the strength of the pheromone trail on the corresponding path.

Ant Colony Optimization. A simplified model

- We introduce a certain number n_a of artificial ants.

- Each ant behaves as follows: Starting from n_d (i.e., the nest), an ant chooses with probability

$$p_i = \frac{\tau_i}{\tau_1 + \tau_2}, \quad i = 1, 2$$

  between path e_1 and path e_2 for reaching the food source n_f.

- Obviously, if $\tau_1 > \tau_2$, the probability of choosing e_1 is higher, and vice versa.

- For returning from n_f to n_d, an ant uses the same path as it chose to reach n_f, and it changes the artificial pheromone value associated to the used edge according to

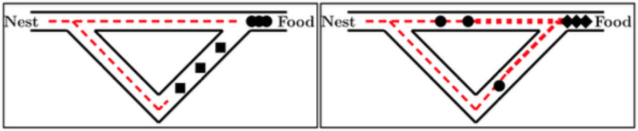
$$\tau_i \to \tau_i + \frac{Q}{l_i}$$

  where the positive constant Q is a parameter of the model. In other words, the amount of artificial pheromone that is added depends on the length of the chosen path: the shorter the path, the higher the amount of added pheromone.

Ant Colony Optimization. A simplified model

This model is iteratively simulated as follows:

- At each step (iteration) all the ants are initially placed in node n_d.
- Each ant moves from n_d to n_f as outlined above.



The circles arrive earlier to n_f, therefore, when returning, the probability totake again the short path is higher The pheromone trail on the short path receives, in probability, a stronger reinforcement, and the probability to take this path grows. Due to the evaporation of the pheromones, the whole colony will, in probability, use the short path

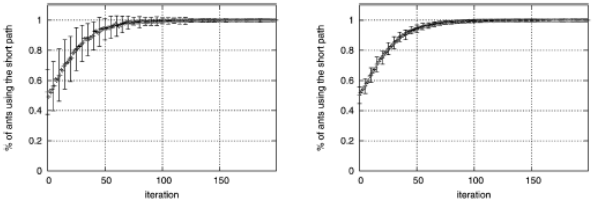
- The pheromone evaporation in the model is simulated as follows:

$$\tau_i \to (1-\rho)\tau_i, \quad i = 1, 2$$

where $\rho \in (0, 1)$ is a parameter that regulates the pheromone evaporation.

Ant Colony Optimization Some numerical results

Using $l_1 = 1$, $l_2 = 2$, $Q = 1$, $\tau_1 = \tau_2 = 0.5$ and $\rho = 0$, the figure shows the results of the simulation using $n_a = 10$ (left) and $n_a = 100$ initial ants.



The x-axis shows the iterations, and the y-axis the percentage of the ants using the short path, the error bars show the standard deviation for each 5th iteration[2]

[2]C. Blum, Physics of Life Reviews 2 (2005) 353–373

Ant Colony Optimization. Differences between real ants and the ones of the model

- While real ants move in their environment in an asynchronous way, at each iteration of the simulated system each of the artificial ants moves from the nest to the food source and follows the same path back.
- While real ants leave pheromone on the ground whenever they move, artificial ants only deposit artificial pheromone on their way back to the nest.
- The foraging behavior of real ants is based on an implicit evaluation of a solution (i.e., a path from the nest to the food source). By implicit solution evaluation we mean the fact that shorter paths will be completed earlier than longer ones, and therefore they will receive pheromone reinforcement more quickly. In contrast, the artificial ants evaluate a solution with respect to some quality measure which is used to determine the strength of the pheromone reinforcement that the ants perform during their return trip to the nest.

Ant Colony Optimization algorithm

- To implement the procedure, we need a graph $G = (N, L)$, where N is the set of nodes and L the set of links between them.
- Each arc (i, j) of the graph has an associated variable τ_{ij} called the pheromone trail.
- The intensity of the pheromone is an indicator of the utility of that arc to build better solutions.
- At each node, stochastic decisions are taken to decide on the next node.
- Initially, a constant amount of pheromone (i.e., $\tau_{ij} = 1$, for all $i, j \in N$) is allocated to all the arcs.
- The probability of the kth ant at node i choosing node j using the pheromone trail τ_{ij} is given by

$$p_{ik}(k) = \begin{cases} \dfrac{\tau_{ij}^{\alpha}}{\sum_{i \in N_i^k} \tau_{ij}^{\alpha}} & \text{if} \quad j \in N_i^k \\ 0 & \text{if} \quad j \notin N_i^k \end{cases}$$

where N_i^k is the neighbourhood of ant k when sitting at the ith node, and α is a parameter that controls the influence of τ_{ij}

Ant Colony Optimization algorithm

- The neighbourhood N_i^k of the ith node contains all nodes directly connected to it excepting the predecessor node. This ensures unidirectional movement of the ants. As an exception for the destination node, where N_i^k should be null, the predecessor of node i is included.

- The pheromone level at each iteration is updated by

$$\tau_{ij}(k+1) = \rho\tau_{ij}(k) + \Delta\tau_{ij}(k)$$

  where $0 \leq \rho < 1$ and $1 - \rho$ represent the pheromone evaporation rate, and $\Delta\tau_{ij}$ is related to the performance of each ant.