Self Organizing Systems Exercise 2

Alexander Dobler 01631858
Thomas Kaufmann 01129115

December 22, 2020

Abstract

In the second exercise of the lecture Self Organizing Systems we are implementing a simple Particle Swarm Optimization (PSO) algorithm and experimenting with different parameters, fitness functions, constraints and constraint handling method. More specifically, we are given a PSO framework in NetLogo for optimizing functions f from $\mathbb{R}^2$ to $\mathbb{R}$. Our task is to implement 3 different fitness functions, 3 different constraints and a constraint handling method using penalization. Furthermore, we are to conduct several experiments, to observe different effects of parameters on the performance of the convergence behaviour of the PSO algorithm. Here we are inspecting *population size*, *particle speed limit*, *particle inertia* and the difference between constraint handling using penalization and rejection.

1 Implementation

In this section we will describe how we implemented the required tasks given in the exercise. We will divide this section into explanations for *constraints*, *fitness functions* and *constraint handling with penalization*.

1.1 Constraints

As we are already given skeletons for constraints, implementing is as easy as returning *true*, if the constraint is violated and *false* otherwise. We opted for implementing the following constraints.

1. $x^2 + y^2 < 6000$
2. $x^2 + y^2 < 9000$ and $x^2 + y^2 > 4000$
3. $\tan(2x) < \tan(4y)$

So, if for example for the first constraint it holds that $x^2 + y^2 >= 6000$, the constraint is violated and we return true. We selected these constraints, as we wanted to have functions with one connected region (constraint 1 and 2) and also constraints with miltiple separated regions (constraint 3).

1.2 Fitness Functions

Here we have a similar setting as for constraints, because we already have skeletons for fitness functions. We opted to implement the following functions.

1. Schaffer function
2. Booth's function
3. Schwefel function

Scaling x and y variables for the input of the functions is done as already shown in the template. It is also important to mention that the NetLogo sin-function expects angles in degrees as input, so we had to convert radians to degrees first. For the Schwefel function we had to set $n := 2$ and $x_1 = x, x_2 = y$ for our purpose of two dimensions.

We chose these functions as we wanted to have both optima at the corners of the grid and optima in the middle of the grid. Furthermore we also have a diversity of how many local optima the search-space of the different functions have.

1.3 Constraint Handling with Penalization

The implementation for penalization is more interesting. First we created 4 different functions to calculate penalization values for each constraint (the example constraint included). So for each constraint we can compute its penalty value at patch $x, y \in \mathbb{R}^2$, if the constraint is violated at this patch. This penalty value is $C(x, y) \cdot d$ for constraints $C(x, y) < 0$ and a constant

d. For example for constraint 1 we have $(x^2 + y^2 - 6000) \cdot d$ as penalty value if $x^2 + y^2 \geq 6000$. Furthermore we wanted penalty values to be between 0 and 1, so we chose constants d appropriately. So for example for constraint 1 we set $d := \frac{1}{14000}$ as $C(x, y) = x^2 + y^2 - 6000$ can be as big as 14000 for $x = y = 100$.

We then add these penalty values at position (x, y) to the value of the patch at position (x, y) if the selected contraint is violated at position (x, y). This, of course, is only done if penalization is selected as constraint handling method. Due to this variant of implementation, we do not have to update anything in the *update-particle-positions*, as fitness-functions at patches are already considering penalization by the selected constraint.

2 Experiments

3 Results and Analysis

4 Conclusion

References