UNIVERSITAS
GADJAH MADA

Optimization Methods

Proposal of New Creative Method Assignment

Incremental Grid Search Optimization (IGSO)

By Sasha A. and Kreshnayogi Dava B.

UNIVERSITAS
GADJAH MADA

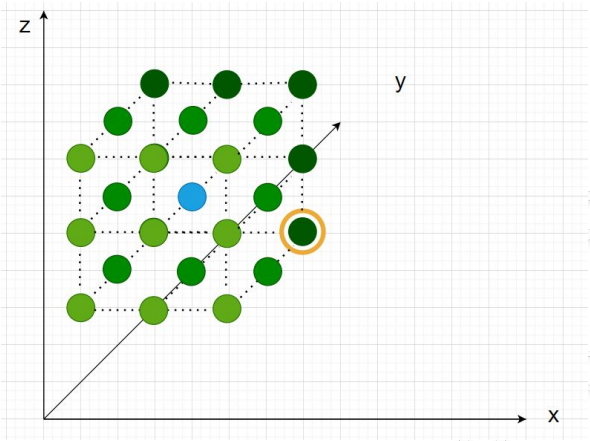
IGSO CONCEPT

Intuition and Algorithm Formulation

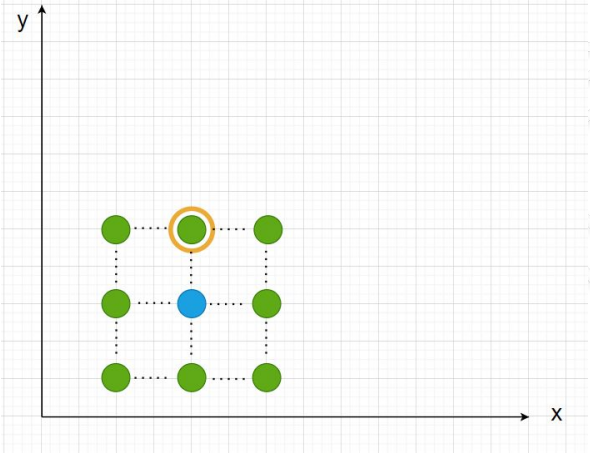
Proposed Concept

The concept of ***neighbors*** have been popular in discrete graph problems. However, not so much in optimization for continuous function space, such as what is intended here.

The idea is simple → to utilize a grid of candidate points or "nodes" around an initial starting point, where the next point with lowest/highest ***f*** value is chosen for the next iteration.



f(x,y,z)



f(x,y)

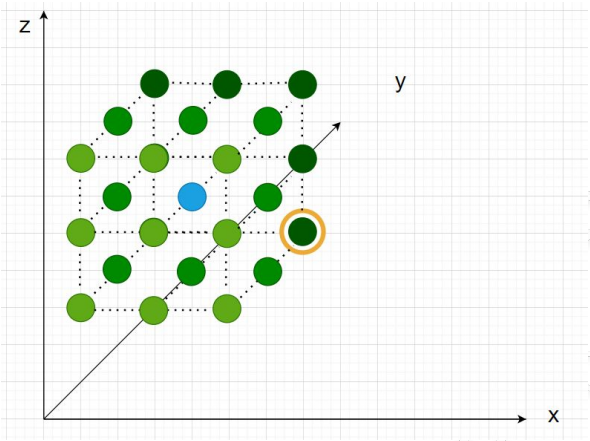
Proposed Concept

For the algorithm to learn effectively, an important factor is the adaptive step size.

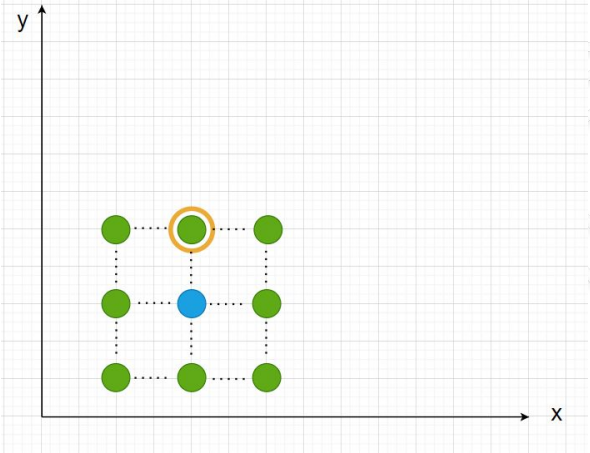
Step size is the distance between starting point and a particular neighbor.

In this algorithm, the size of grid (step size) determines the range of exploration, which will shrink or expand based on difference in ***f***.

To summarize, the "grid" is able to adapt, exploring further nodes when necessary and minimizing the search space as solution reaches the desired optimum.



f(x,y,z)



f(x,y)

Proposed Algorithm

Assume a function $f(x,y) = (x-2)^2 - (y-3)^2$ where we intend to find the minimum.

1. Initialize ***stepSize***, ***maxStepSize***, ***minStepSize***, and threshold ***T***.
2. Initialize ***main node*** (x_0, y_0)
3. Initialize ***n*** neighbors where ***n*** = (num_of_variables)^3 - 1. For instance :
    a. $(nx_1, ny_1) = (x_0 - stepSize, y_0 - stepSize)$
    b. $(nx_2, ny_2) = (x_0 + stepSize, y_0 - stepSize)$
    c. $(nx_3, ny_3) = (x_0 - stepSize, y_0 - stepSize)$
    d. and continued.

Proposed Algorithm

4. Calculate f value for all neighbors.
5. Between all neighbors and the main node itself, choose the node with lowest f value as the next ***main node***.
6. Knowing that $\Delta f = f_{new\ main\ node} - f_{previous\ main\ node}$, update the next iteration's step size accordingly :
   a. If Δf > ***T*** then new ***stepSize = stepSize*** * 1.2
   b. If Δf <= ***T*** then new ***stepSize = stepSize*** * 0.8
   c. Condition → ***stepSize*** cannot increase to be greater than ***maxStepSize*** or decrease to be smaller than ***minStepSize***

UNIVERSITAS
GADJAH MADA

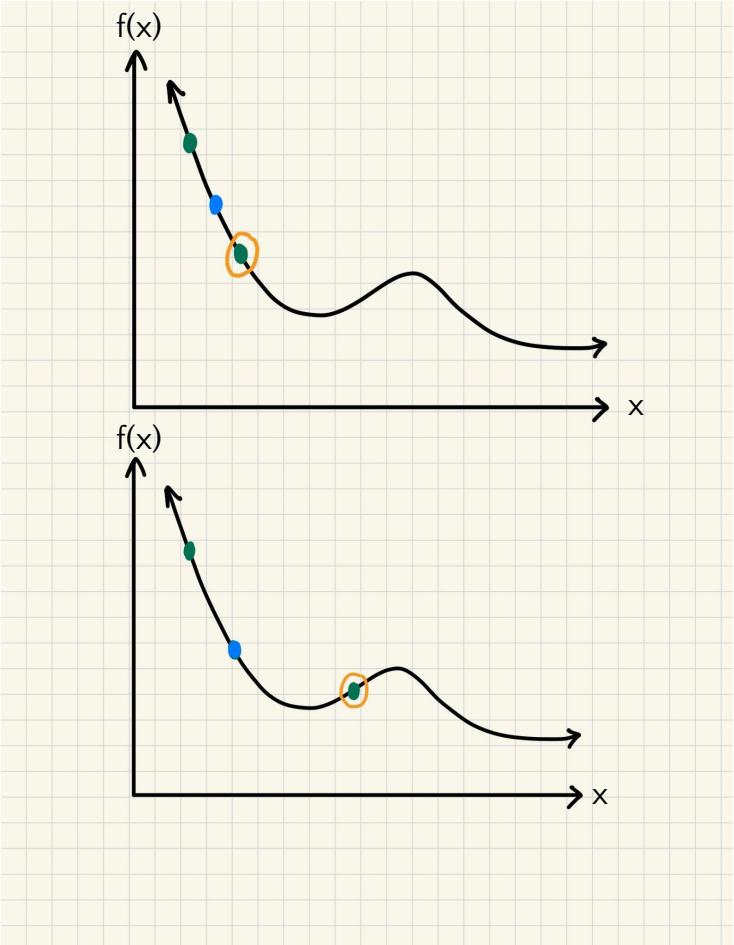
IGSO EXAMPLE

Function f(x)

BLUE is ***main node***.

GREEN is neighbor nodes.

YELLOW is node with lowest ***f***.



Take a look at this case with function **f(x)** where we intend to find its **minimum**.

In the 1st iteration, assume $\Delta f > T$. Here :

- Choose the node with lowest ***f*** value as the next ***main node***
- Increase ***stepSize*** for new neighbors on next iteration

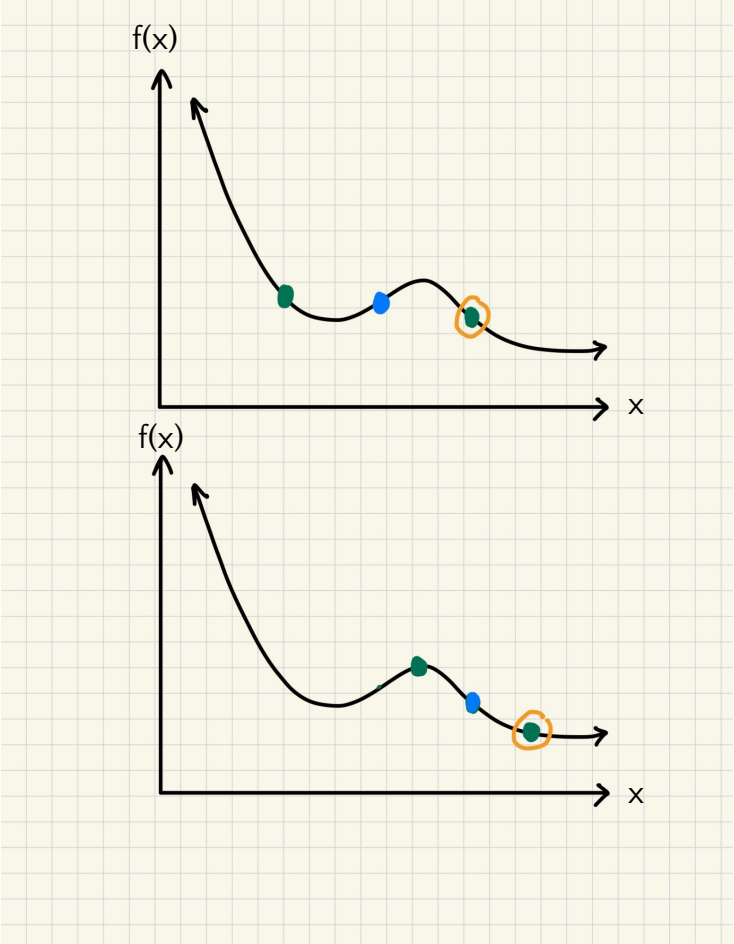
As seen in the 2nd iteration, the neighbor nodes have greater distance from the main node.

Afterwards, assume that the 2nd iteration's Δf is smaller than ***T***. Therefore, the next ***stepSize*** will be smaller.

BLUE is ***main node***.

GREEN is neighbor nodes.

YELLOW is node with lowest ***f***.



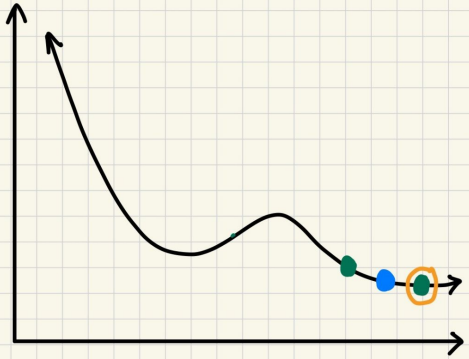
Now, in this 3rd iteration, the Δf is smaller than the ***T***, by intuition.

Therefore, the ***stepSize*** decreases for the 4th iteration again, shown in the last bottom graph.

BLUE is ***main node***.

GREEN is neighbor nodes.

YELLOW is node with lowest f.



The algorithm stops when it reaches the minimum step size for certain number of iterations (no improvement).

UNIVERSITAS
GADJAH MADA

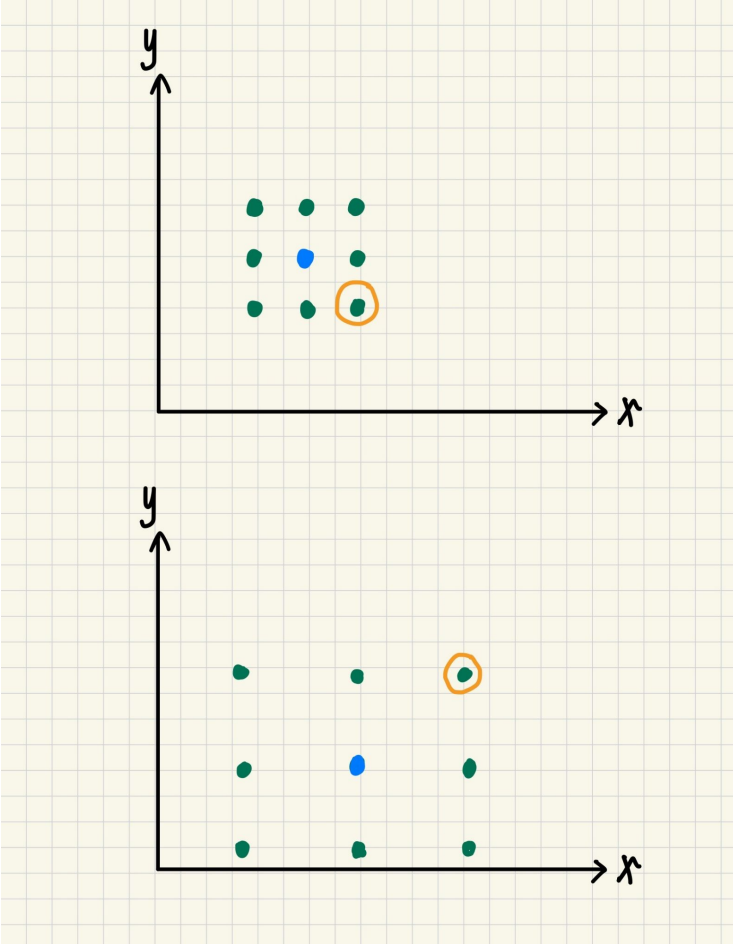
IGSO EXAMPLE

Function f(x,y)

BLUE is ***main node***.

GREEN is neighbor nodes.

YELLOW is node with lowest ***f***.



*REMINDER : the f(x,y) is not represented. These illustrations show the **x-y plane** only.*

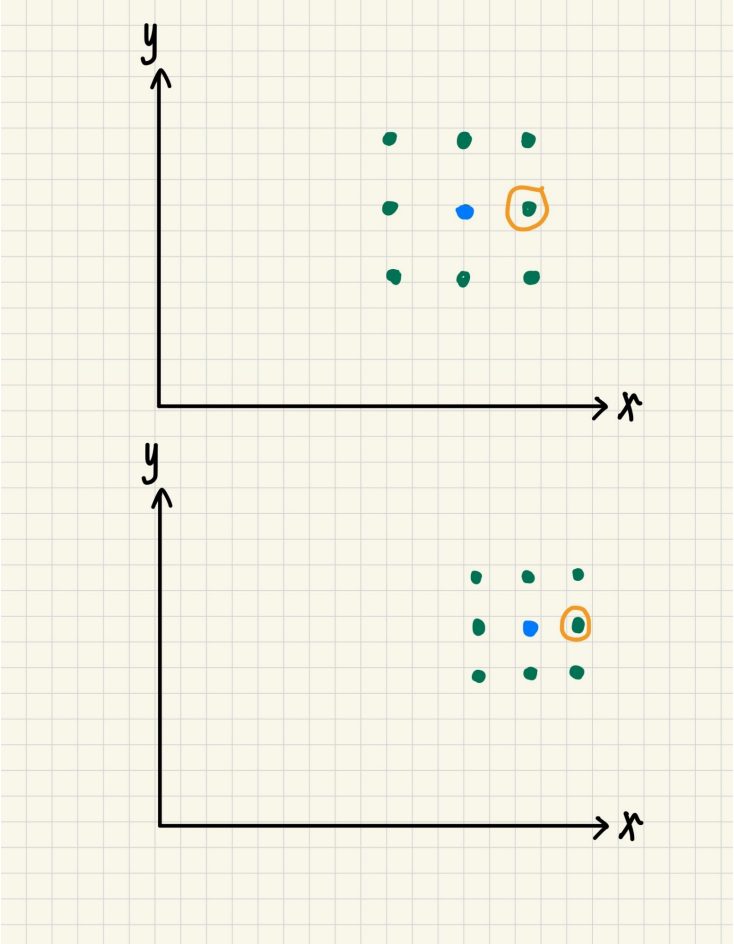
Assume on the 1st iteration, $\Delta f > T$. The ***stepSize*** increases for 2nd iteration.

In the next iteration, assume $\Delta f < T$. Thus, the ***stepSize*** for the third iteration is smaller.

BLUE is ***main node***.

GREEN is neighbor nodes.

YELLOW is node with lowest f.

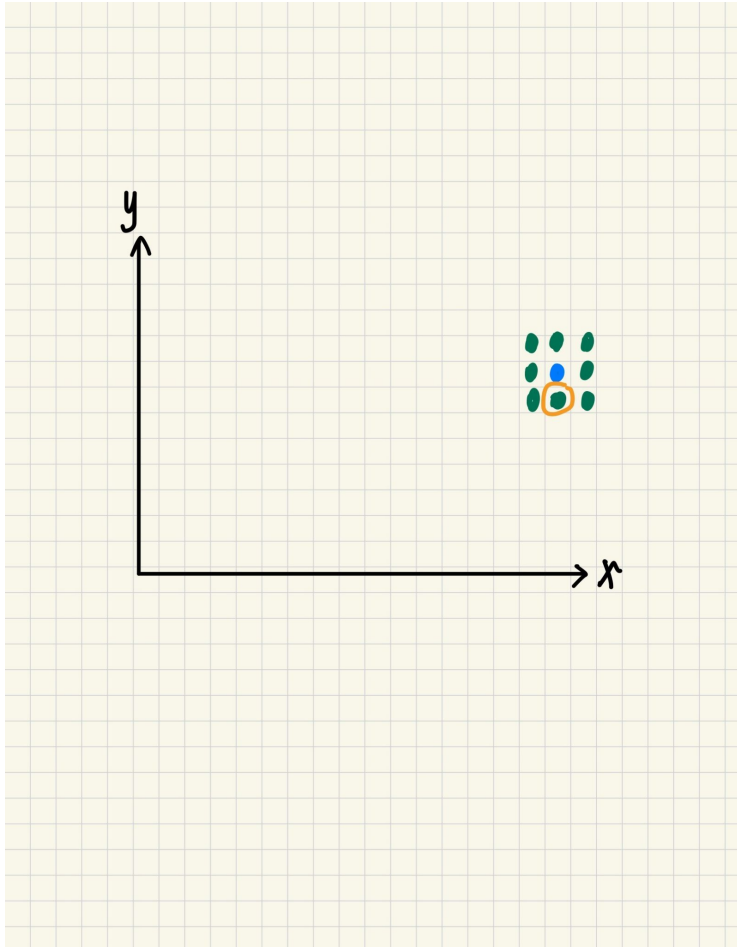


For the 3rd and 4th iteration shown in these graphs, assume $f < T$ sequentially in both cases, resulting in ***stepSize*** to continuously decrease as well.

BLUE is ***main node***.

GREEN is neighbor nodes.

YELLOW is node with lowest f.



As intuitively seen, the “grid” becomes smaller as the solution reaches closer to local minimum.

UNIVERSITAS
GADJAH MADA

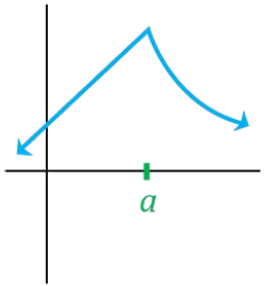
IGSO Algorithm Analysis

Advantages, Disadvantages, and Improvements

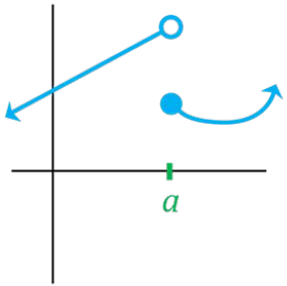
Advantages and Disadvantages

The main advantage of this algorithm is that unlike Gradient Descent and similar methods, IGSO has the ability to find optimum (local max/min values) of **non-differentiable** functions → the process does not rely on whether the function can be derived.

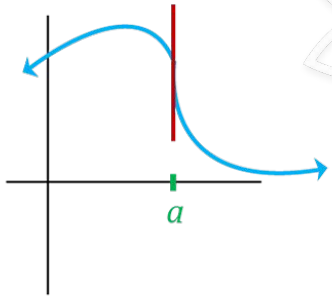
In addition, the algorithm can be altered manually to either encourage high or low exploration, depending on the initial goal.



Cusp / Corner



Discontinuous

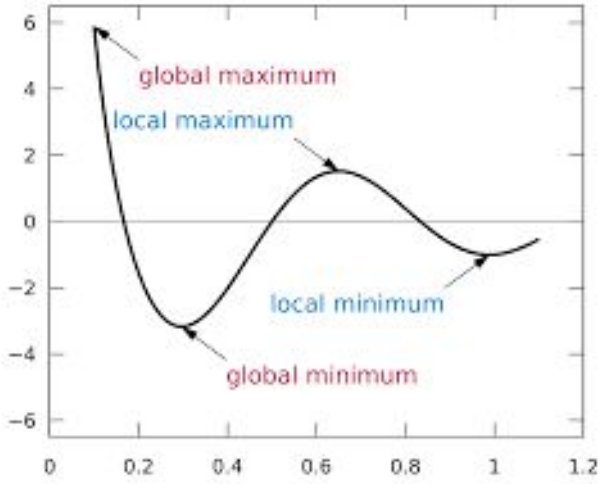


Vertical Tangent

Advantages and Disadvantages

Some disadvantages have been found, however :

- Its number of iterations (runtime) is hypothesized to increase exponentially as the number of variables (e.g. *x, y, z, w,...*) increases.
  - Number of neighbors will is in power of 3
- Similar to the case with Gradient Descent, IGSO still cannot guarantee that it finds global optimum (curse of local optimum)
  - However, the algorithm is designed to **minimize** it being "stuck" within a local minimum.
- The efficiency of the algorithm highly depends on the initialized variables (e.g. step size, T, and so on).



Trial Code

A trial program has been created to simulate how IGSO algorithm works.

In this example case, the function is $f(x,y) = (x-2)^2 + (y-3)^2$ and the goal is to evaluate whether the ***minimum*** found is the same with the already known mathematically to be (2,3).

IGSO Algorithm Implementation Example (LINK)

UNIVERSITAS
GADJAH MADA

Thank You.