CS 451: Computational Intelligence
Assignment 2

Muhammad Usaid Rehman, Syed Abbas Haider

March 21, 2021

Contents

1 Graph Coloring using ACO **1**
1.1 The Graph Coloring Problem . . . 1
1.2 Problem Formulation using ACO . . . 2
1.2.1 Updating of Pheromone Trails . . . 2
1.2.2 Heuristics . . . 2
1.2.3 Transition Rule . . . 2
1.2.4 Construction Strategy . . . 3
1.3 The Algorithm . . . 3
1.4 Empirical Analysis . . . 4

2 Know More About Optimization **7**
2.1 What is metaheuristic optimization? . . . 7
2.2 What are the situations in which gradient based optimization techniques do not work? . . . 7
2.3 Briefly explain any one swarm based algorithm that we have NOT discussed in the class . . . 7

References **8**

1 Graph Coloring using ACO

1.1 The Graph Coloring Problem

Graph coloring is a special case of graph labeling. It is an assignment of labels known as colors to elements of a graph (vertices, edges or cycles). We will look at the vertex coloring of a graph in this problem.

In its simplest form, graph coloring assigns colors to vertices such that no two adjacent vertices have the same color. This is an NP-hard problem in graph theory and it has applications in timetabling and resource allocation.

Formally, a k-coloring of a graph can be stated as: given an undirected graph $G = (V, E)$, a k-coloring of G consists of applying a color to each vertex $v \in V(G)$ such that no two adjacent vertices have the same color. The number of colors used are k. The minimum possible number of colors that can be assigned to a graph is called the chromatic number, $\chi(G)$. The Graph Coloring problem is an combinatorial optimization problem that aims to minimize $\chi(G)$. [1]

1.2 Problem Formulation using ACO

Ant Colony Optimization is a metaheuristic to solve hard combinatorial optimization problems that is based on the foraging behavior of ants. There is indirect communication between ants in the form of pheromone trails. A partial solution is constructed by each ant and then a daemon (a higher level control) selects best solutions. [2]

The strategy to construct coloring for each ant that we used is the DSATUR algorithm. It uses a dynamic ordering of vertices based on degree of saturation of the vertices. The degree of saturation or $dsat(v)$ is the measure of the number of colored neighbors of the vertex v. [3]

1.2.1 Updating of Pheromone Trails

The pheromone deposited on a pair of vertices (i, j) is given by $\tau^k(v_i, v_j)$. At the end of an iteration, $\tau^k(v_i, v_j)$ is the pheromone deposited on that edge by each ant. We also define $\Delta\tau(i, j)$ as the amount of pheromone added by all ants.

We store the values of the pheromone trails in a square matrix, where:

$$\tau(v_i, v_j) = \begin{cases} 1 \text{ if } (v_i, v_j) \notin E(G) \\ 0 \text{ if } (v_i, v_j) \in E(G) \end{cases} \tag{1}$$

Evaporation is done using the following equation:

$$\tau(v_i, v_j) = (1 - \rho) \cdot \tau(v_i, v_j) \tag{2}$$

1.2.2 Heuristics

Heuristic information is dependent on our construction strategy. Since we are using DSATUR, our heuristic information η is simply the degree of saturation of the next vertex. [4]

1.2.3 Transition Rule

We place a high importance on information obtained from previous ants. After an elitism strategy is used to update pheromones, the next vertex to color is used based on a

sort of random-proportional rule. The rule is that with some probability q_0, an ant on vertex i moves to vertex j when the product between pheromone trail and the heuristic information is maximum, i.e.

$$v = max\{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}\} \tag{3}$$

1.2.4 Construction Strategy

We initialize pheromone values to 1. Each ant constructs a feasible solution, adding a vertex, color pair to the partial solution until a complete solution is obtained. The construction is done in the following steps:

1. The next vertex is selected using the transition rule.
2. The vertex that is chosen is put into a tabu list to ensure that a colored vertex is not recolored to guarantee feasibility.
3. Pheromone values of pairs of vertices with the same color are reduced in an attempt to reduce the chances of premature convergence.
4. After a complete solution is obtained by each ant, the best one is saved and compared with previous best solutions. Following this, the daemon action makes sure that pheromone is only added to the best solution that was found.

1.3 The Algorithm

Following is the rough pseudocode for the algorithm:

Algorithm 1: ACO_GCP

```
Initialize parameters and pheromone trails
while stopping criterion not reached do
    Place ants on starting vertices
    foreach ant ∈ ants do
        Choose next vertex to color using transition rule
        Update the tabu list
        Update pheromone trails
    Select best solution
    Update pheromones using daemon action
    Evaporate necessary pheromones
return Best solution
```

1.4 Empirical Analysis

The graph upon which the algorithm was tested looks like this:

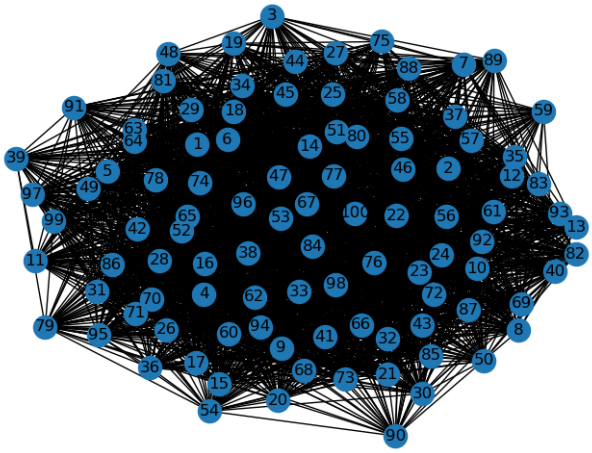


Figure 1: A graph with 100 vertices

When the optimization was performed for a long period using default values of the parameter the following graph was obtained. It can be seen that the average fitness stabilized earlier and did not fluctuated slightly around 19.5-20.0. Whereas the Best fitness reached the lowest optimal value of 17.0, it did however from time to time fluctuate back to 18.0 in some iterations.

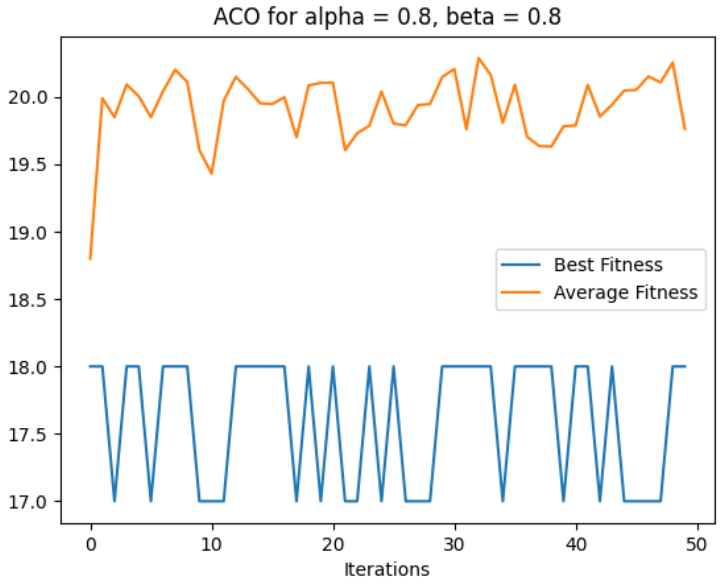
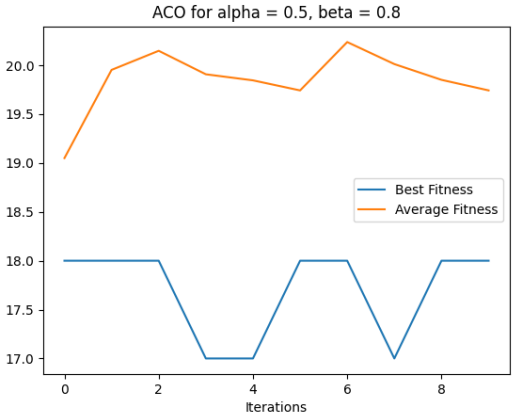
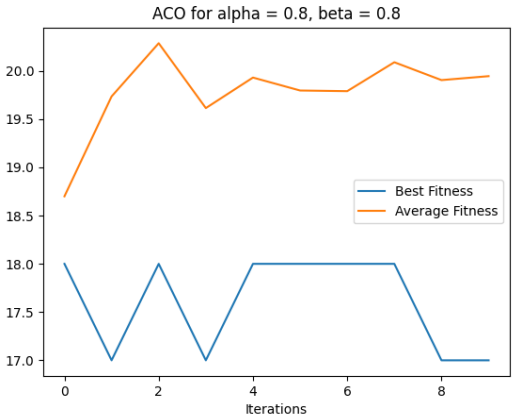
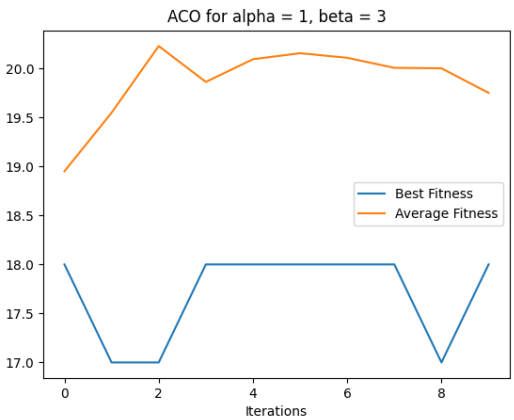


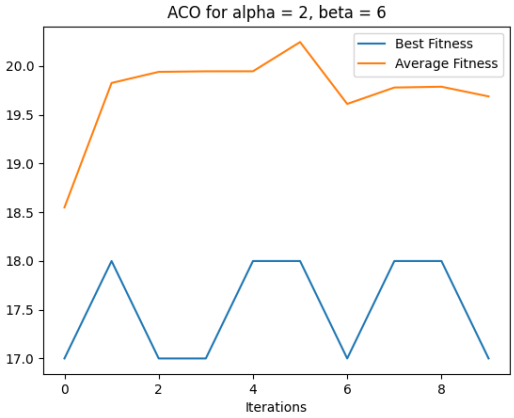
Figure 2: Results with default values for 50 iterations

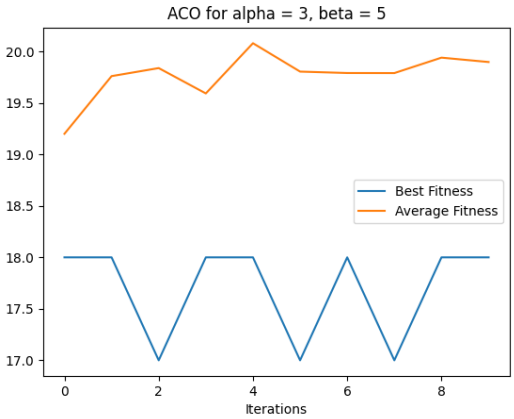
We also repeated the experiment with different parameter values of alpha and beta, the figures below show how the results varied for each parameter configuration when it was run for ten iterations. The results did not vary as much on using different parameters.

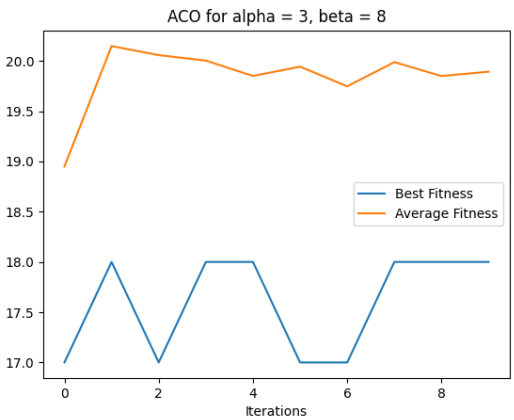

ACO for alpha = 0.5, beta = 0.8
Best Fitness
Average Fitness
Iterations


ACO for alpha = 0.8, beta = 0.8
Best Fitness
Average Fitness
Iterations


ACO for alpha = 1, beta = 3
Best Fitness
Average Fitness
Iterations


ACO for alpha = 2, beta = 6
Best Fitness
Average Fitness
20.0
19.5
19.0
18.5
18.0
17.5
17.0
0
2
4
6
8
Iterations


ACO for alpha = 3, beta = 5
Best Fitness
Average Fitness
20.0
19.5
19.0
18.5
18.0
17.5
17.0
0
2
4
6
8
Iterations


ACO for alpha = 3, beta = 8
Best Fitness
Average Fitness
20.0
19.5
19.0
18.5
18.0
17.5
17.0
0
2
4
6
8
Iterations

2 Know More About Optimization

2.1 What is metaheuristic optimization?

Recent literature refers "metaheuristics" as algorithms with stochastic components, meaning randomness exists in the algorithm. The term was introduced by Fred Glover in his seminal paper (Glover 1986). By his convention this referred to all modern nature inspired algorithms, in his words they are a *"master strategy that guides and modifies other heuristics to produce solutions beyond those that are normally generated in a quest for local optimality"* (Glover and Laguna 1997).

Metaheuristic optimization intends to seek a globally best solution to a global optimization problem using metaheuristic algorithms. It can bring about quality solutions to hard optimization problems better than gradient-based algorithms and derivative-free algorithms, but it cannot be verified if it reaches the optimal solution.

The two major constituents of such algorithms are exploration and exploitation. Exploration means to explore the search space on a global scale and bring about diverse solutions; this is achieved via randomization and allows the search to escape from the local optima. On the other hand, exploitation refers to concentrating search on local region where a known current good solution lies; this convergence to an optimum is ensured through selection of the best solutions. A suitable mix of these two components will usually make global optimality achievable.

2.2 What are the situations in which gradient based optimization techniques do not work?

It is very difficult to apply gradient based solutions for non-linear, multi-modal, multivariate functions. Moreover, some objective functions can have discontinuities which would hinder the performance of gradient based algorithms. It may also be computationally expensive to calculate derivatives accurately, especially with higher number of decision variables and multiple peaks in the function. Furthermore, noise or uncertainty in the objective and constraint functions can also be detrimental to gradient based approaches.

2.3 Briefly explain any one swarm based algorithm that we have NOT discussed in the class

The Bee Algorithm is part of the trend of algorithms that utilize swarm intelligence. It is inspired by the food forging behaviour of honey bees. Forager scouting bees from the hive explore different random areas in search of flowers with abundant and good pollen. When they discover a location with a good nectar source they bring the nectar back to the hive and communicate the direction, distance, and quality of the food source to their fellow bees in the hive through a 'waggle dance' that acts as a signaling system. Using

this information more bees are recruited to follow the specified locations for more nectar. This strategy allows efficient exploration of nectar over huge distances.

This algorithm works in a similar way to this principle, it initializes a population of randomly distributed solutions, and it defines neighbourhood regions around some of those solutions primarily around those of a high fitness value, which are marked as 'elite bees' and are saved for the next generation . It searches more solutions in each neighbourhood region and finds the solution with the highest value in each such region adds them to the next generation as well. The remaining population is filled by randomly assigning new solutions, and the process is repeated again for several iterations until the population converges to certain solutions.

This was initially created at Oxford, under the name of Honey Bee Algorithm, for allocating computers among different web-hosting servers and clients. Later on other variants were developed such as Virtual Bee Algorithm (VBA) to solve continuous optimization problems and Artificial Bee Colony (ABC) algorithm for numerical function optimization. Bee algorithms in general are better suited for discrete and combinatorial optimization, but can be applied to a wide range of problems.

References

1. Lewis, R. *A guide to graph colouring* ISBN: 9783319257280 (Springer Berlin Heidelberg, New York, NY, 2015).
2. Engelbrecht, A. P. *Computational intelligence: an introduction* 2nd ed. OCLC: ocn133465571. ISBN: 9780470035610 (John Wiley & Sons, Chichester, England ; Hoboken, NJ, 2007).
3. Karthikeyan., Geetha, D. T. & Kumar, S. *Ant Colony System for Graph Coloring Problem* in (2017).
4. Costa, D. & Hertz, A. Ants Can Colour Graphs. *The Journal of the Operational Research Society* **48,** 295 (1997).