Ant Colony Optimization with Multiple Objectives

Honghou Zhou

TJHSST Computer Systems Lab 2009-2010

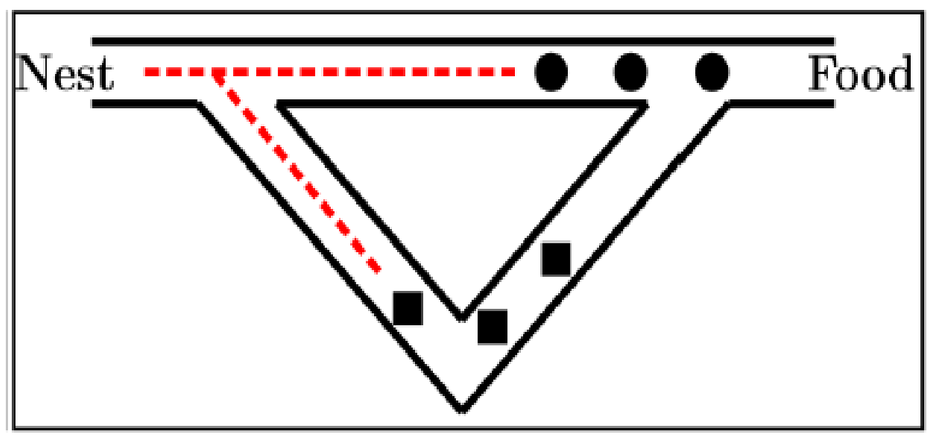
Abstract

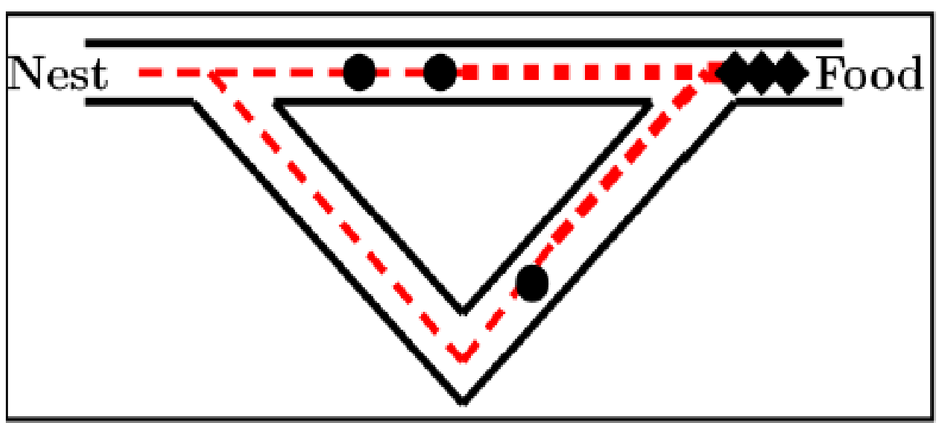
Ant Colony Optimization (ACO) is a useful method to find near optimal paths. The most common algorithms only works toward finding the best path with regard to one condition. However, it is often more realistic and useful to factoring other variables and constraints as well. This project will be to develop a way to consider two objectives and weigh their relative importance.

Introduction/Background

Ant Colony Optimization (ACO) was inspired by, and mimics, how ants find a short path from their colony to a food source. As an ant travels back to its colony, it leaves behind a pheromone trail that other ants follow. Other ants then base their decision for which route to use based on the amount of pheromone they detect. Pheromone builds up faster on shorter path than longer ones, it also evaporates over time. This allows the colony to weed out paths until only a near optimal one remains. The process does not guarantee an optimal solution, but the one it finds should be close.

More than factor generally affect which path is actually the best and focusing on only one means sacrificing others. The idea then is to implement a way to take multiple objectives into account.

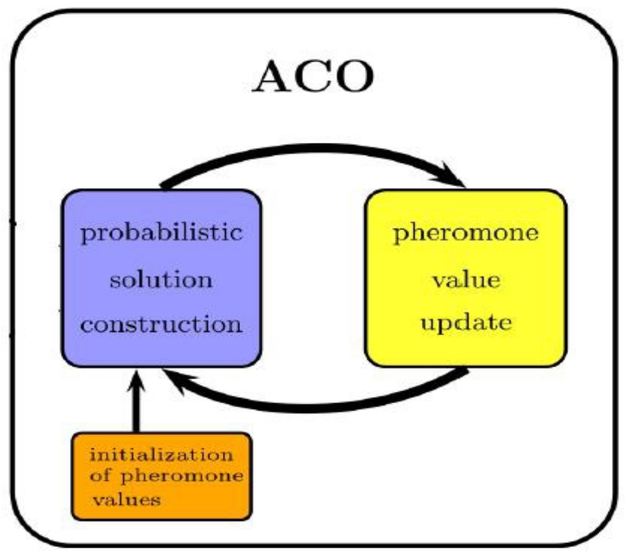




Example of a simple ACO run

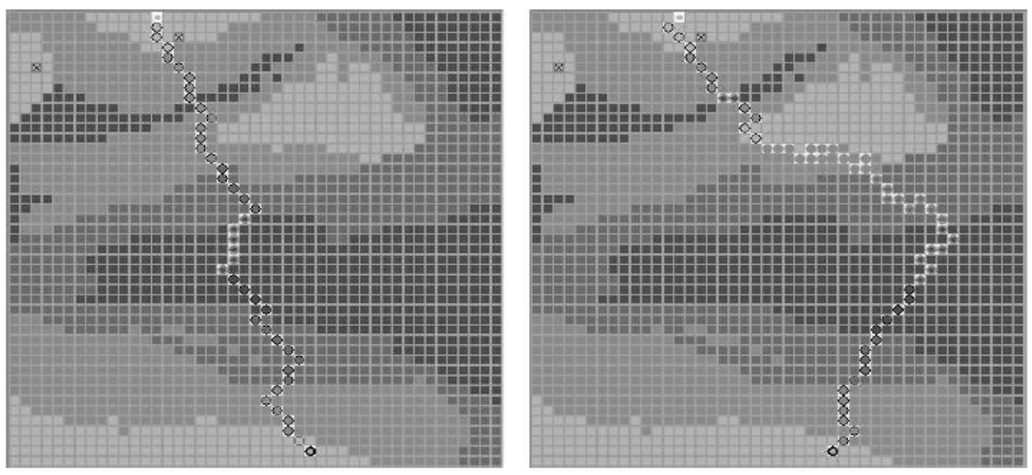
Development

I will first code a working ACO algorithm and test that against the example code I have. The next step would be to implement an additional constraint by adding another pheromone and set of weights (lengths) to the graph.



Expected Results

The result given by the basic ACO algorithm should match already existing code. After adding the second objective, a run should be able to produce different paths based on how important one objective is deemed to be. The results for when one is focused on should be better for that objective. When all the weight is put on one objective, the result should match the basic ACO.



	Combined State Transition Rule				*Dominance State Transition Rule*				*Greedy*
	Fastest ($\lambda = 0.9$)		Safest ($\lambda = 0.1$)		Fastest ($\lambda = 0.9$)		Safest ($\lambda = 0.1$)		
	F_f	F_s	F_f	F_s	F_f	F_s	F_f	F_s	
Best	**70.500**	334.600	84.500	**285.800**	**73.500**	374.300	76.000	**354.600**	
Mean	**75.133**	357.800	105.517	**311.390**	**77.950**	397.280	88.780	**371.890**	NO SOLUTION
	±2.206	±9.726	±17.138	**±19.541**	**±1.724**	±11.591	±13.865	**±7.522**	

Sample results

Conclusion

None at the moment…

References