Cheetah Optimizer Algorithm

By: Alex Taylor and Om Patel

Agenda

- Overview
- Original Algorithm
- Benchmarking for Original Algorithm
- Modifications for TSP
- Benchmarking for TSP Algorithm
- TSP Art

Overview

Inspired by the cheetahs' qualities:

- Long searching/scanning followed by fast chase
- Built-in exploration vs exploitation
- Cheetah's energy limits, random step sizes, prey movements when chasing

Original Algorithm

Main Points:

- Send out some of the cheetahs on a hunt
- Searching
- Sit-and-Wait
- Attack
- Giving up the hunt
- Resting at home before new hunts

Algorithm 1: *The CO Algorithm*

```
Define the problem data, dimension (D), and the initial population size (n)
Generate the initial population of cheetahs X_i (i = 1, 2, ..., n) and evaluate the fitness of each cheetah
Initialize the population's home, leader and prey solutions
t ← 0
it ← 1
MaxIt ← desired maximum number of iterations
T ← 60 × ⌈D/10⌉
while it ≤ MaxIt do
    Select m (2 ≤ m ≤ n) members of cheetahs randomly
    for each member i ∈ m do
    Define the neighbor agent of member i
        for each arbitrary arrangement j ∈ {1, 2, ..., D} do
            Calculate r̂, ř, α, β, and H
            r_2, r_3 ← random numbers are chosen uniformly from 0 to 1
            if r_2 ≤ r_3 then
                r_4 ← a random number is chosen uniformly from 0 to 3
                if H ≥ r_4 then
                    Calculate the new position of member i in arrangement j using Equation (3) //Attack
                else
                    Calculate the new position of member i in arrangement j using Equation (1) //Search
                end
            else
                Calculate the new position of member i in arrangement j using Equation (2) //Sit-and-wait
            end
        end
        Update the solutions of member i and the leader
    end
    t ← t + 1
    if t > rand × T and the leader position doesn't change for a time, then //Leave the prey and go back home
        Implement the leave the prey and go back home strategy and change the leader position
        Substitute the position of member i by the prey position
        t ← 0
    end
    it ← it + 1
    Update the prey (global best) solution
end
```

Benchmarking for Julia Conversion

- Implemented 8 functions from CEC 2005
  - 3 Unimodal functions for exploitation
  - 5 Multimodal functions for exploration
    - 3 Basic Multimodal functions
    - 2 Expanded Multimodal functions
- Overlapping functions from CEC 2010 & CEC 2013

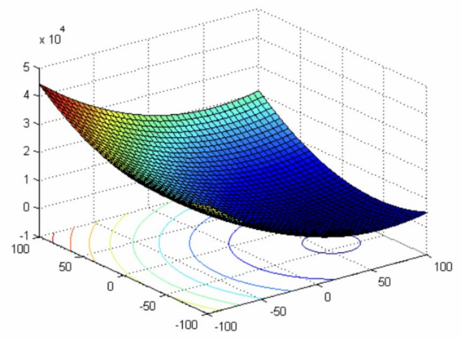


Figure 2-1 3-*D* map for 2-*D* function

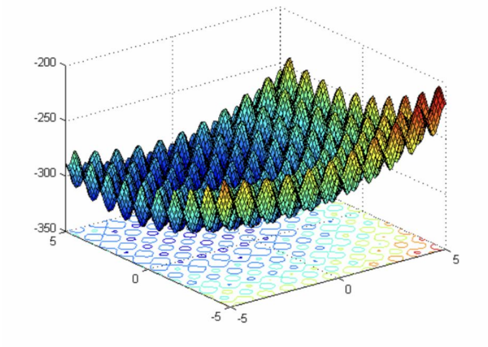


Figure 2-9 3-*D* map for 2-*D* function

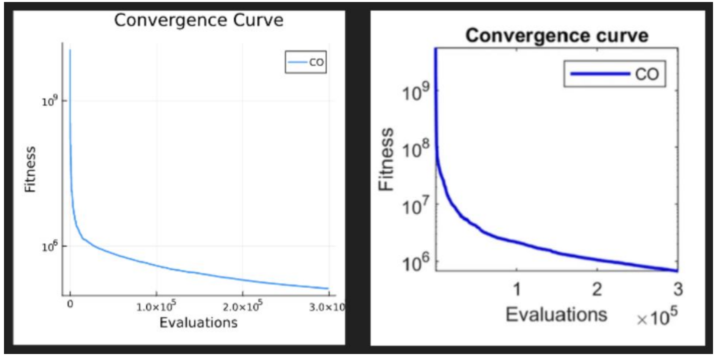
Benchmarking for Julia Conversion

- Replicated the experiment settings and parameters
  - Function Evaluations/Iterations: 300,000
  - Population Size: 6
  - Search Agents: 2
  - Dimensions: 30
- Low Replication Transparency
  - No reasoning for the number of function evaluations
    - Led to us picking our own number of simulations - 10
  - No Information about tuning random variables

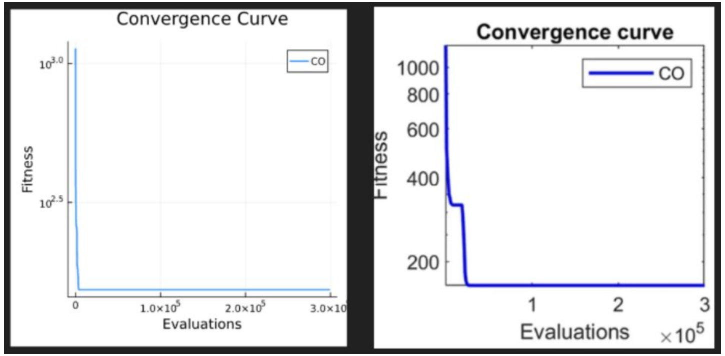
Benchmarking for Julia Conversion

Function	Template Paper Mean	Template Paper STD	Cheetah Mean	Cheetah STD
f1	5.7e-13	1.76e-12	2.78607e-11	3.41241e-11
f2	3.25e-5	1.27e-5	1.8998e-6	1.3808e-6
f3	802000.0	411000.0	2.53999e5	84051.9
f6	51.0	66.1	23.784	32.457
f9	2.03e-10	8.37e-10	1.79093	1.30992
f10	199.0	55.3	169.153	34.2995
f13	1.32	0.304	13.2556	10.0221
f14	12.8	0.332	0.658255	0.42255

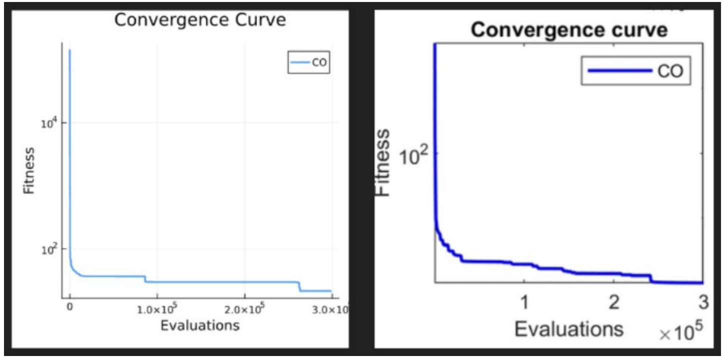
Benchmarking for Julia Conversion



Convergence Curves for f3(unimodal)
Julia implementation pictured on left



Convergence Curves for f10(multimodal)
Julia implementation pictured on left



Convergence Curves for f13(multimodal)
Julia implementation pictured on left

Algorithm TSP Modifications

1. The original algorithm iterates through dimensions
   a. No longer search vs attack per dimension, instead one choice for the whole cheetah that iteration.
2. How do we implement Search and Attack for the whole Cheetah then?
   a. Some sort of swaps?
      i. One swap = Too Few
      ii. Every city swap = Too Many
      iii. Solution: dim/3 swaps

Search Modifications

- Simple Implementation
- Random Search of the Search Space
- Conclusion: Random swap of the cities' within one cheetah's order

$$X_{i,j}^{t+1} = X_{i,j}^{t} + \hat{r}_{i,j}^{-1} . \alpha_{i,j}^{t}$$

Attack Modifications

- Simple Implementation
- Based on leader and neighbor values
- Conclusion: Find pairs of cities present in the leader and neighbor cheetahs and put those cities together within the current cheetah
  - Side note: Leader cheetah weighted heavier than neighbor(7:1)

$$X_{i,j}^{t+1} = X_{B,j}^{t} + \check{r}_{i,j}.\beta_{i,j}^{t}$$

Algorithm TSP Modifications

3. Certain calculations omitted
    a. Checks between leader and prey
    b. Clamping bounds
    c. Other small optimization calculations
    d. Solution: Keep the core ideas present
4. M Value altered
    a. A higher number of search agents lead us to better results almost across the board

TSP Benchmarking

Benchmarking Metaheuristic Optimization Algorithms on Travelling Salesman Problems

Deniz Tosoni, University of Applied Sciences and Arts Northwestern Switzerland, Switzerland, deniz.tosoni@students.fhnw.ch
Christopher Galli, University of Applied Sciences and Arts Northwestern Switzerland, Switzerland, christopher.galli@students.fhnw.ch
Thomas Hanne, University of Applied Sciences and Arts Northwestern Switzerland, Switzerland, thomas.hanne@fhnw.ch
Rolf Dornberger, University of Applied Sciences and Arts Northwestern Switzerland, Switzerland, rolf.dornberger@fhnw.ch

TSP Benchmarking

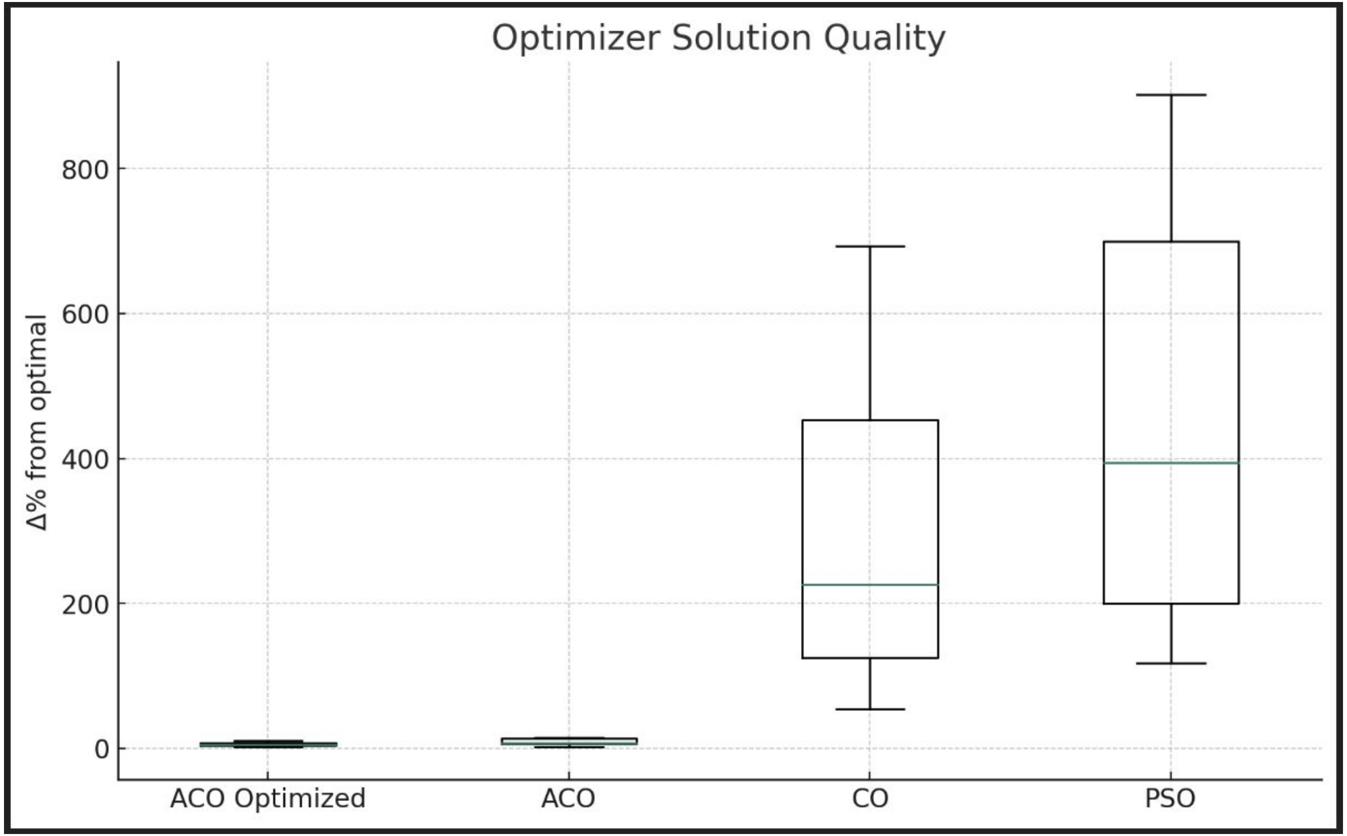
- 10 instances of the from TSPLIB were used
  - Ranged in complexity and size (various scenarios)
- 6 algorithms were compared against
  - ACO Optimized, ACO, GA, PSO, SA, TS
- Benchmarking Approach
  - 10 simulations on each instance
  - 1000 iterations for each simulation
  - 100 populations size for population-based methods

```julia
tsp = readTSPLIB(:Eil51)

fitness_values = []

for _ in 1:10
    best_fitness = cheetah_TSP(tsp.weights, tour_cost, size(tsp.nodes, 1), iter, pop_size, agents)
    push!(fitness_values, best_fitness)
end

# Compute metrics
best_fitness = minimum(fitness_values)
average_fitness = mean(fitness_values)
avg_delta_percentage = ((average_fitness - tsp.optimal) / tsp.optimal) * 100

println("Best: ", best_fitness)
println("Average: ", average_fitness)
println("Average Delta: ", avg_delta_percentage)


# Store results
results[tsp_instances[1]] = (best = best_fitness, average = average_fitness, avg_delta = avg_delta_percentage)
```

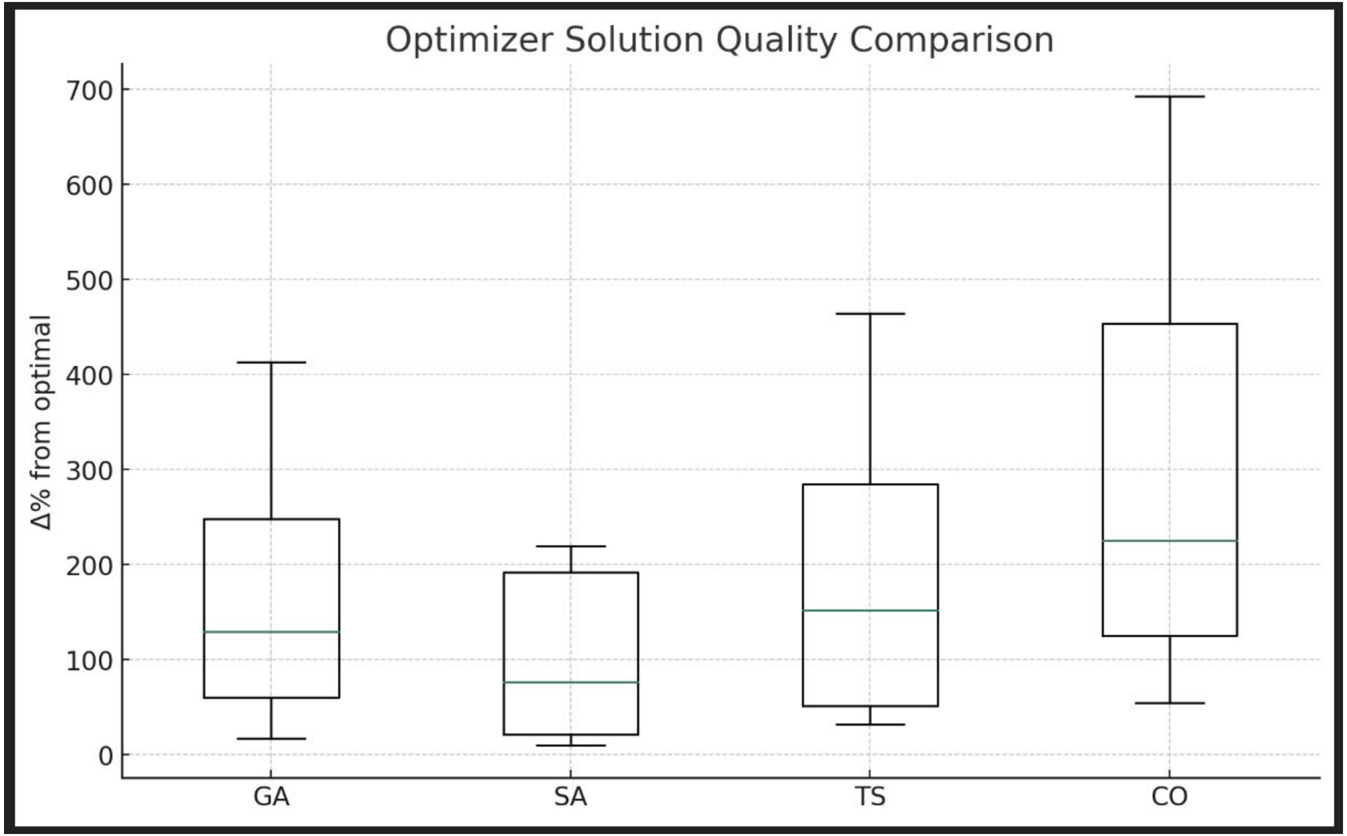
TSP Model	Route Cost	ACO Optimized Best	ACO Optimized Avg	ACO Optimized Avg Δ%	ACO Best	ACO Avg	ACO Avg Δ%	PSO Best	PSO Avg	PSO Avg Δ%	CO Best	CO Avg	CO Avg Δ%
Eil51	426	441	449	5.4%	451	457	7.3%	856	961	125.59%	596.0	657.1	54.2488
Berlin52	7542	7548	7689	1.95%	7757	8001	6.1%	14786	16460	118.2%	11010.0	11644.7	54.398
Eil76	538	553	562	4.48%	577	579	7.6%	1396	1561	190.22%	1058.0	1181.3	119.572
Eil101	629	669	691	9.9%	704	725	15.18%	2163	2421	284.94%	1667.0	1820.7	189.459
Pr76	108159	114169	116751	7.94%	122567	124262	14.89%	316878	359243	232.14%	242133.0	2.61278e5	141.569
Pr107	44303	45970	46240	4.37%	45768	45993	3.81%	312766	349933	686.86%	142531.0	1.60332e5	261.9
Pr124	58537	60203	60992	4.19%	64462	65631	12.12%	408894	470387	703.57%	314959.0	3.43521e5	481.943
Pr136	96772	105322	107519	11.11%	110390	111386	15.10%	515092	584403	503.90%	434923.0	4.53825e5	368.964
Pr144	58537	59705	60342	3.08%	59525	59729	2.04%	512336	573211	879.23%	429347.0	4.57936e5	682.301
Pr152	73682	78180	78784	6.92%	77650	78422	6.43%	668545	738097	901.73%	556877.0	5.8407e5	692.69

TSP Model	Route Cost	GA Best	GA Avg	GA Avg Δ%	SA Best	SA Avg	SA Avg Δ%	TS Best	TS Avg	TS Avg Δ%	CO Best	CO Avg	CO Avg Δ%
Eil51	426	8349	8930	17.51%	445	467	9.65%	537	562	32%	630.0	662.5	55.5164
Berlin52	7542	8390	8498	18.4%	8498	9279	23.03%	9438	10208	35.35%	10873.0	11716.0	55.3434
Eil76	538	784	860	59.87%	601	628	16.65%	745	783	45.48%	1105.0	1189.8	121.152
Eil101	629	1122	1240	97.12%	715	761	21.05%	963	1071	70.19%	1742.0	1829.5	190.859
Pr76	108159	161513	175464	62.23%	150864	168180	55.49%	226031	261316	141.60%	246928.0	2.60515e5	140.863
Pr107	44303	105358	116134	162.13%	110474	138476	212.57%	113250	139636	215.18%	139038.0	1.55417e5	250.805
Pr124	58537	179185	210652	259.86%	130517	151577	158.94%	213808	238626	307.65%	329739.0	3.49454e5	491.994
Pr136	96772	280366	303433	213.56%	165789	190849	97.21%	204531	254191	162.67%	428091.0	4.52932e5	368.04
Pr144	58537	277456	300546	413.43%	151187	177458	203.16%	298368	330267	464.20%	409448.0	4.5334e5	674.451
Pr152	73682	337027	355993	383.15%	209951	23529	219.34%	257020	315339	327.97%	563426.0	5.90565e5	701.505

TSP Benchmarking



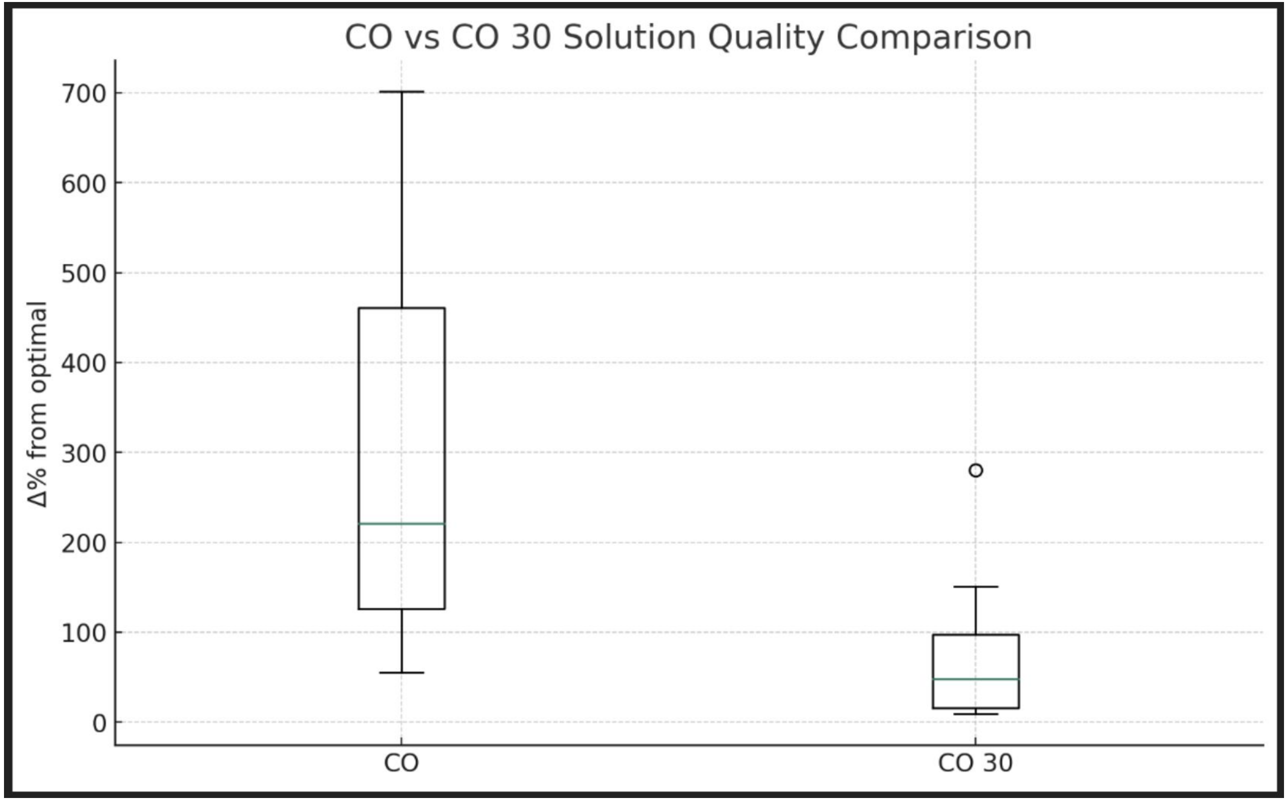
TSP Benchmarking



TSP Benchmarking

TSP Model	Route Cost	CO Best	CO Avg	CO Avg Δ%	CO 30 Best	CO 30 Avg	CO 30 Avg Δ%
Eil51	426	630.0	662.5	55.5164	457.0	475.3	11.5728
Berlin52	7542	10873.0	11716.0	55.3434	7792.0	8302.0	10.0769
Eil76	538	1105.0	1189.8	121.152	665.0	701.6	30.4089
Eil101	629	1742.0	1829.5	190.859	904.0	1008.0	60.2544
Pr76	108159	246928.0	2.60515e5	140.863	136618.0	1.47728e5	36.5843
Pr107	44303	139038.0	1.55417e5	250.805	46734.0	48361.3	9.16033
Pr124	58537	329739.0	3.49454e5	491.994	98833.0	1.08478e5	83.7679
Pr136	96772	428091.0	4.52932e5	368.04	169516.0	1.9575e5	102.279
Pr144	58537	409448.0	4.5334e5	674.451	180482.0	2.22957e5	280.882
Pr152	73682	563426.0	5.90565e5	701.505	111475.0	184890.0	150.93

TSP Benchmarking



Overall Algorithm Thoughts

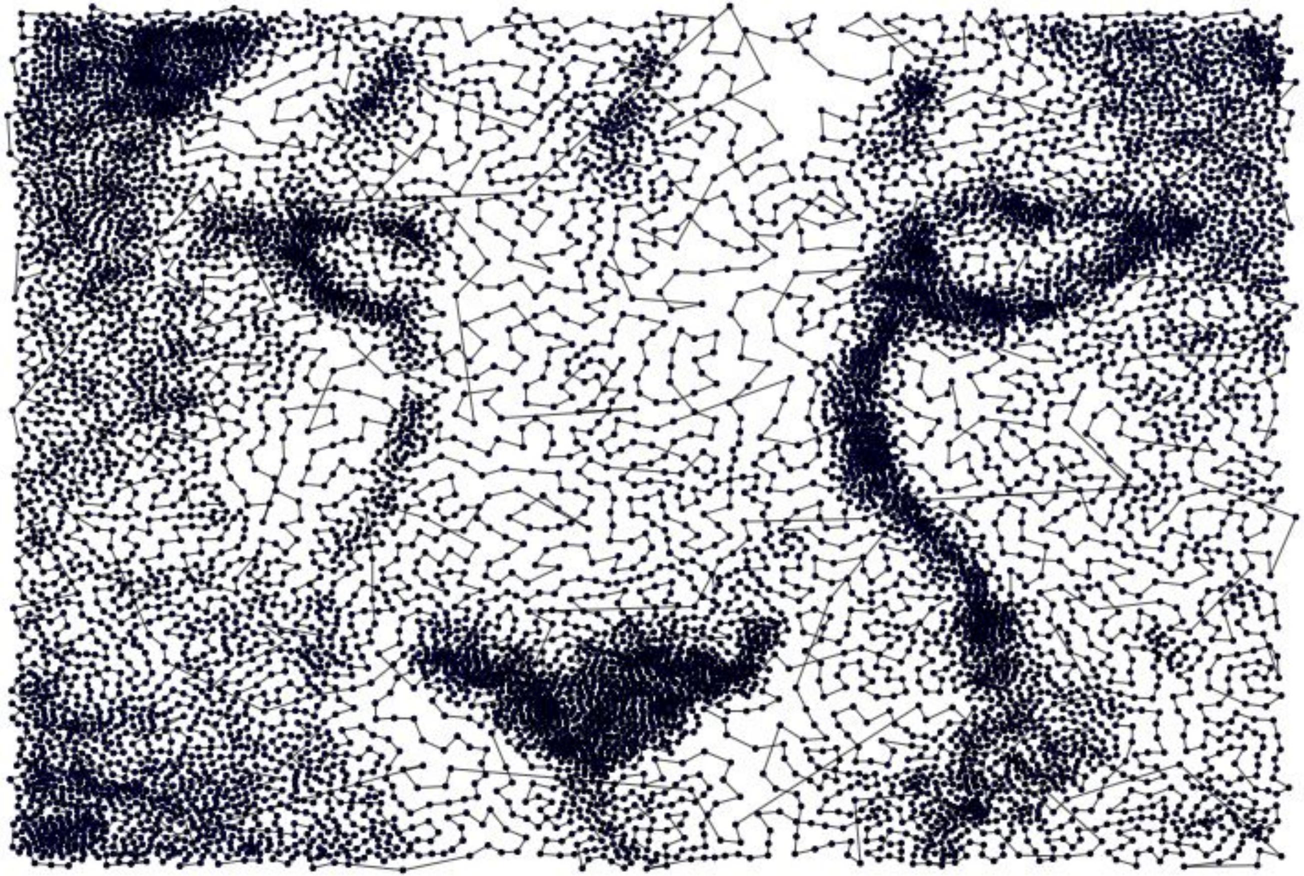
- Not too many unique ideas, just rephrased older ideas
  - Ex. Hunt Time = SA’s temperature
- Outperformed by existing algorithms
- Interesting in theory but in execution doesn’t provide much innovation
- Can still be used to create meaningful art!

How was it made?

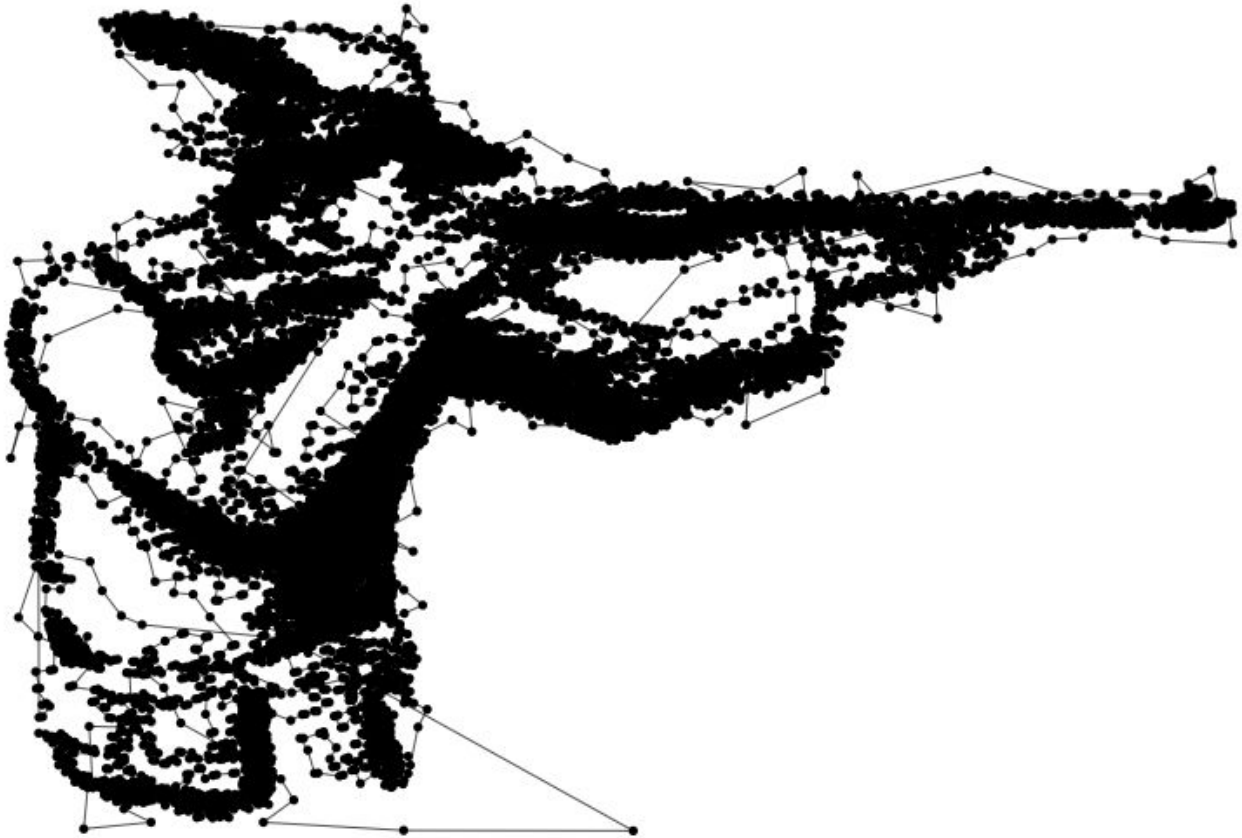
- Stipples were created by StippleGen2
- Converted SVG to coordinates using PathToPoints tool
- ChatGPT to transform points into a distance matrix
- Greedy approach to initialize the population position
  - Produced more efficient & quicker results
- Plotted the tour with using a scatter for cities dots and lines to visualize to connect the cities

Predator

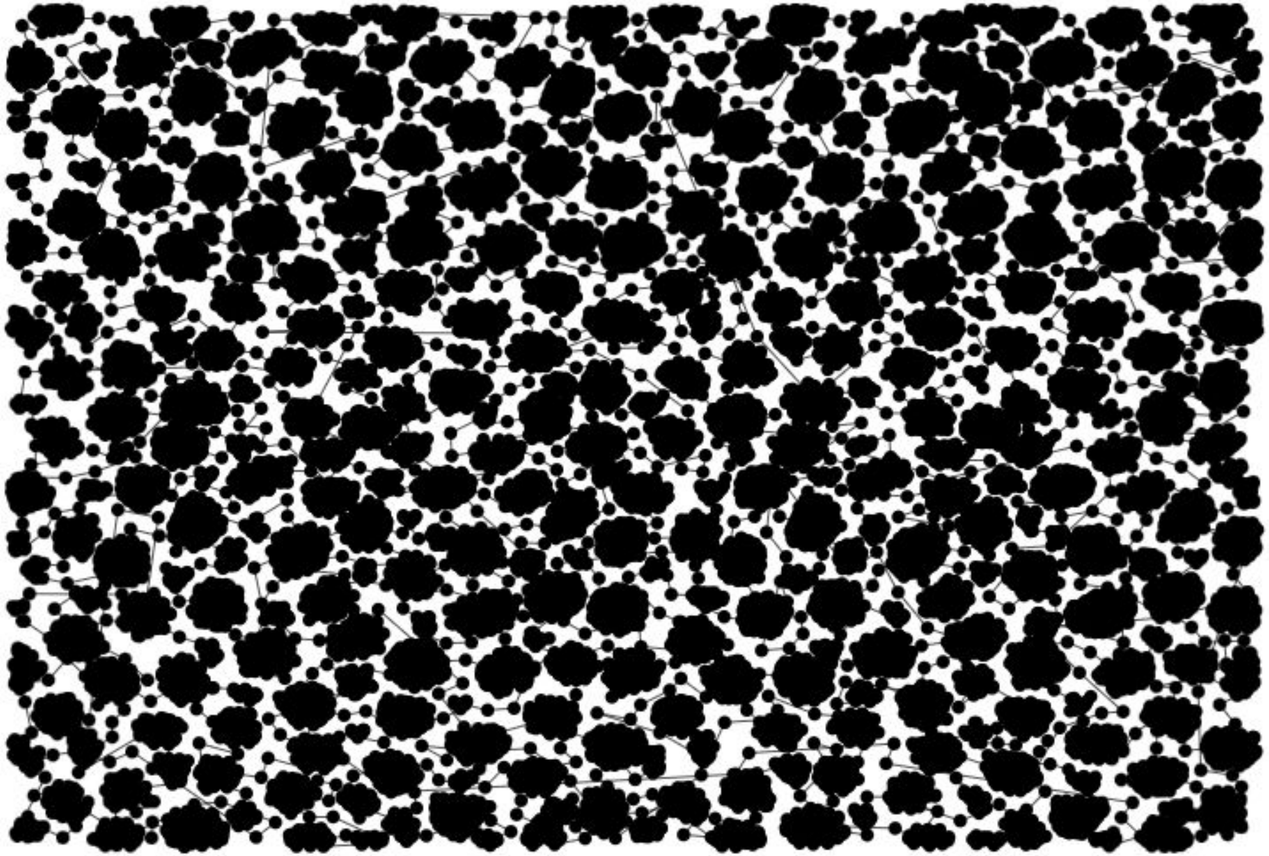
TSP Art



TSP Art



TSP Art



Thank you! Any questions?