Matthew Poncini
J.B. Speed School of Engineering
University of Louisville

26 October 2025

—

CSE-545 Project 5
Genetics and Wisdom of Crowds Hybrid Algorithm for TSP

Abstract—Previous experiments have shown an array of methodologies to provide close to optimal solutions to the traveling salesman problem (TSP). In this experiment, we provide a hybrid algorithm that incorporates a traditional Genetic Algorithm (GA) with a Wisdom of Crowds (WoC) algorithm to produce near optimal solutions to the traveling salesperson problem (TSP) for different sized sets of TSP cities. Our results show that this hybrid implementation produces higher quality tours than the previous GA implementations. The hybrid algorithm is structured like a traditional GA, but utilizes WoC in its crossover operations. Each offspring inherits traits from their parents, but also inherits traits from the general population through WoC.

I. Introduction

In a GA designed to solve the traveling salesman problem (TSP), it starts with a base population. Each member of the population is a tour that represents a solution to TSP. From the base population, members will be paired to create offspring through crossover functions. A new generation will replace the parent population. The new generation will consist of offspring from the previous generation as well as a select number of members of the previous generations; members that provide the most optimal solutions to TSP.

In a Wisdom of Crowds (WoC) solution to TSP, as presented by Sheng Kung, Michael Yi, Mark Steyvers, and Michael D. Lee in their publication "Wisdom of the Crowds in Traveling Salesman Problems", the authors demonstrate that by aggregating multiple suboptimal individual solutions, a collective "crowd" solution could be constructed that often matches or even exceeds the performance of the best individual. This is achieved by building an edge-frequency agreement matrix across participants' tours and then reconstructing a new route that prioritizes high-agreement edges. Edges that were the most common in a set of already close to optimal solutions to TSP were mapped to a low cost, with less frequent edges being assigned a higher cost. Instead of attempting to minimize the Euclidean distant of a tour like more traditional TSP algorithms do, the WoC implementation attempts to minimize the cost associated with an edge's frequency in previous solutions.

In this project, a hybrid algorithm using traditional GA and WoC is used to solve TSP. WoC is applied to GA in two ways in this algorithm: (1) From generation to generation, a fraction of offspring will be formed from an aggregate function that only takes the most repeated edges among the top performing members of the previous generation; (2) The GA crossover functions will incorporate some Aggregate function like behavior when two parents breed an offspring. The aggregate function used in this project is similar to the one described in the Kung, Yi, Steyvers, and Lee publication, but instead of building an edge-frequency agreement matrix, our implementation uses a frequency queue. The Aggregate function is unlikely to build a complete tour that could represent a valid solution to TSP. To address this, our Aggregate algorithm will use a Nearest Neighbors (NN) approach in finding connecting cities that are available to complete the tour and ensure all cities are traveled to. In our Aggregate themed Crossover functions, if the Aggregate portion fails to provide an edge, the Crossover function will take edges from one of the parent tours before drawing edges from NN.

This paper presents the methodology of our WoC themed GA solution TSP. It will cover the mechanics of the Aggregate and Crossover functions in detail. The algorithm will be tried against several TSP sets of different city counts. The results will be compared against the GA algorithm from Project 4, that does not incorporate WoC.

II. Methodology

The methodology will be presented in a hierarchal order. We will start by showing the structure of the GA, break down how each generation is replaced, then present the Aggregate and Crossover algorithms.

Figure 1 shows the flowchart representing the GA. The algorithm begins with importing TSP data, establishes directories for edge distances, and creates the structure for storing populations. Then, the first population of tours is created. Since the Aggregate algorithm depends on drawing edges from top tier tours, the first population is created by taking the 90 best tours from a 1000 tours created randomly. Figure 2 shows the titles of the 6 different methods used to create a new population from a previous population.

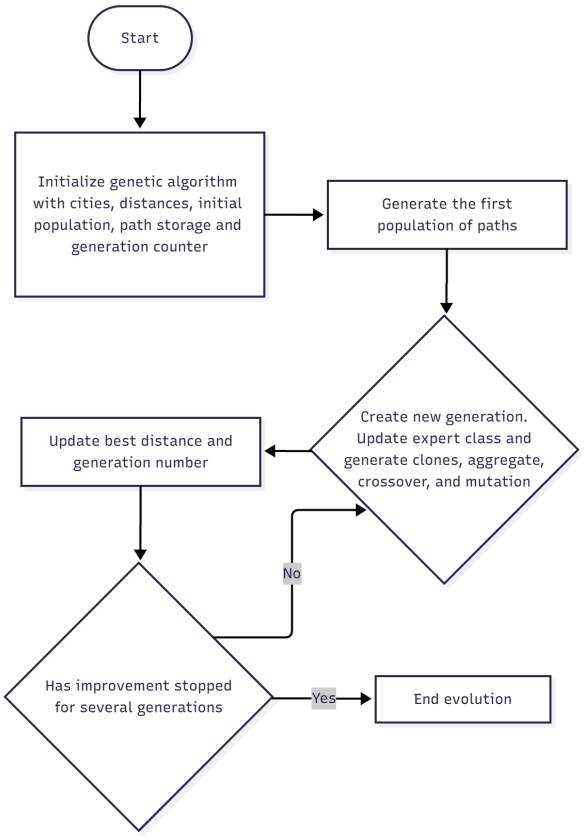

Figure 1: Flow Chart of GA

In this project, we restricted each population to 90 members. In each evolution, the 15 tours with the shortest Euclidean distances are set as the clones. Then using the edge frequencies from the top $p\%$ tours, 15 tours are generated with the Aggregate function with NN completion. Using combinations of the two Crossover and Mutation functions, the other 4 sets of 15 tours are generated.

Aggregate Function Methodology

The Aggregate function generates new offspring tours by combining information extracted from the collective structure of the previous population. Specifically, it identifies the most frequently occurring edges among existing solutions and prioritizes these high-frequency connections when constructing a new path. The algorithm begins with the most common edge and iteratively extends the partial tour by attaching adjacent cities that share an endpoint with the current sequence, ensuring that each city is visited exactly once. When no high-frequency edge can be connected, the method falls back to a NN heuristic based on edge distances. Tours created by the aggregate function incorporate a population-level consensus with local distance optimization.

When creating a new generation, our Aggregate algorithm needs to create 15 tours. For each tour creation, the fraction of top-frequency edges admitted to construction increases by 3%. The first tour created using Aggregate only uses edges from the top 5% of the previous generation tours; the last tour created in each generation will use edges from the top 50% of tours. The purpose of this is to create some diversity among the 15 tours generated. The earlier tours generated (with edges coming from the top 5%) will be most likely be more reliant on NN, but will resemble more the top performers from the previous generation more. The later tours that use edges from the top 50% of the previous generation will rely less on NN, but be influenced more by lower quality tours.

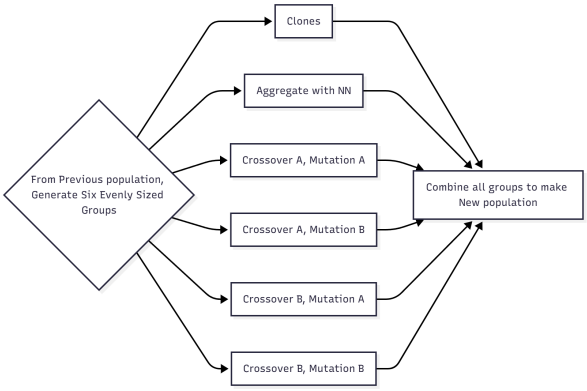

Figure 2: The six ways the members of a new generation are formed

Crossover A Methodology

The Crossover A implements a hybrid crossover mechanism that merges information from both parent tours and the collective edge-frequency knowledge of the population (in Aggregate style). The method begins by identifying the top 20% of high-frequency edges from the previous generation and prioritizes these connections when extending the offspring path. Starting from the first city of the first parent, the algorithm iteratively selects the next city by preferring edges that are both frequently used and short in distance. If no crowd-endorsed edge is available, the operator uses a parent-based greedy rule, selecting the closer of the two possible successor cities inherited from the parents. When neither parent provides feasible edges, the NN heuristic is used to maintain continuity.

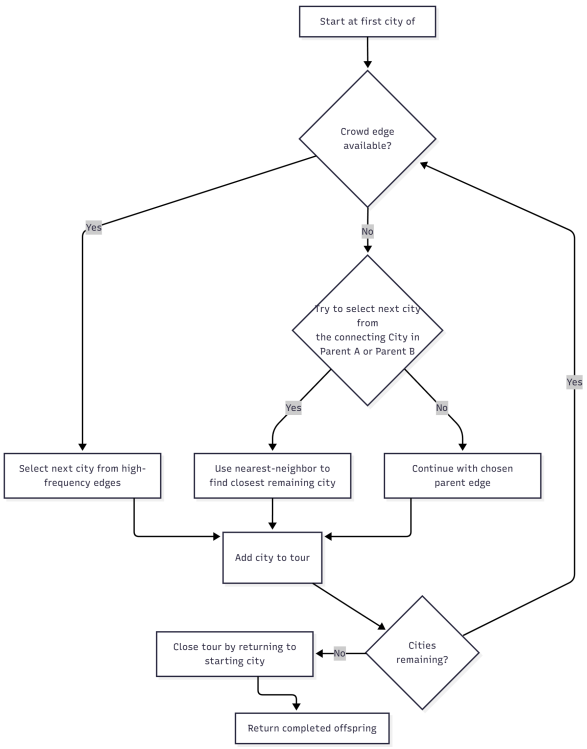

Figure 3: Flow Chart of Crossover A

The resulting tour is then closed to form a valid TSP cycle. This procedure blends population-level consensus with parental inheritance and local optimization, promoting both exploitation of learned structural patterns and exploration of new route configurations.

Crossover B Methodology

Crossover B is similar to Crossover A in that it utilizes collective edge-frequency known- edge, but differs in how the offspring inherits edges from its parent tours. Crossover B ranks each parent's edges by length and retains the top 25% quality parental edges. Like Crossover A, it identifies the top 20% of high-frequency edges from the previous generation and prioritizes these connections. Offspring construction is started by selecting the most frequent occurring edge in the crowd-sourced set of edges. The tour then grows by a prioritized expansion rule applied at each step: (i) attaches any crowd-endorsed edge that connects to either end of the current partial tour; if a crowd-sourced edge is unavailable, (ii) it attaches an edge drawn from the top-ranked subsets of Parent A or Parent B; if edges are not available from this pool, (iii) it uses a nearest-neighbor fallback that inserts the city minimizing the distance to the closer tour endpoint. This staged preference exploits population consensus first, leverages high-quality parental structure second, and heuristically completes the tour to be a valid TSP solution when needed. The process continues until all cities are included exactly once, after which the tour is closed to form a valid TSP cycle.

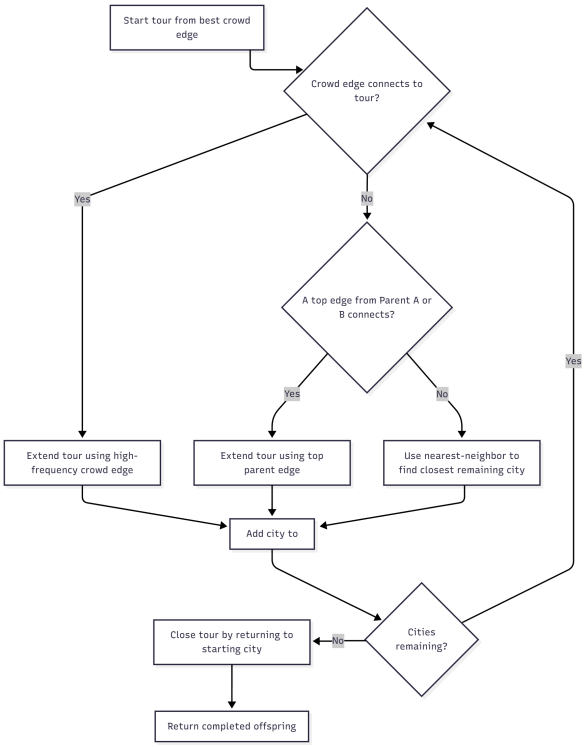

Figure 4: Flow Chart of Crossover B

Mutations

Only offspring created by Crossovers A and B received mutations. Offspring that received Mutation A, had the following procedure implemented to it's tour. Two cities were randomly selected in the tour. If swapping the city positions in the tour yielded a shorter distance, then the two cities would be swapped. Otherwise, the cities would remain in their present location. The number of times this was repeated depended on the tour size. For a tour with n cities, the process was repeated approximately $n/5$ times. Mutation B is the same as Mutation A, but only selects consecutive cities in the tour.

Often times when looking at visualizations of solutions to TSP, the naked eye can see shorter routes possible just by swapping two adjacent cities. As generations evolve, Mutation A is less likely to cre-

ate better tours. However, Mutation B could will have a better opportunity in later generations.

III. Results

The GA and WoC hybrid algorithm was used to provide solutions to 6 TSP city sets, all with different numbers of cities. Table 1 shows the performance of the hybrid GA–WoC algorithm across multiple TSP datasets of varying city sizes, while Table 2 presents the baseline results obtained from the standard GA implementation used in Project 4. When comparing the two, a clear pattern emerges: the hybrid GA–WoC approach consistently produces either equivalent or improved tour distances while achieving these results in fewer generations and with significantly reduced runtime for smaller datasets.

For instance, in the 11-city and 22-city problems, the GA–WoC method achieves identical best distances to the standard GA but with roughly one-third of the runtime. As the number of cities increases, the performance advantage becomes more nuanced. For the 44-city dataset, GA–WoC produces a slightly longer tour than the baseline GA but does so with lower average runtime variability and faster convergence, indicating better computational efficiency and stability.

For larger problem instances (77 cities and above), the hybrid method demonstrates scalability but also begins to exhibit runtime growth similar to the baseline GA, as expected in population-based metaheuristics. Nevertheless, the algorithm continues to deliver near-optimal routes, showing that integrating Wisdom of Crowds principles enhances exploration while preserving solution quality.

44 City Results

Figure 5 shows the convergence behavior of the GA–WoC hybrid algorithm for the 44-city TSP instance. The plot displays the minimum, average, and maximum path distances observed across successive generations. Early in the evolution, the population exhibits high variability, with maximum path distances exceeding 2500 while the best tours remain near 1800. As generations progress, both the best and average tour distances rapidly decrease, indicating that the population is collectively moving toward higher-quality solutions.

Around generation 20, the improvement rate slows as the algorithm begins to refine existing good tours rather than discovering entirely new structures. By generation 50, the curves converge tightly, with all members of the population producing near-optimal paths around 600–700 in total distance. The gradual reduction in the spread between the maximum and minimum distances illustrates effective population convergence and reduced diversity. This is the expected outcome as selection pressure increases and the WoC aggregate mechanism guides the search toward consensus edge structures.

Table 1: GA WoC Results Across City Sizes

Number of Cities	GA WoC Best Distance	GA WoC Avg Generation	GA WoC Avg Distance	GA WoC Avg Runtime (s)	Runtime Std. Dev.
11	351.04588	16.4	351.04588	0.6585142	0.1260696
22	416.094642	43.7	418.3103846	6.2952944	1.6533453
44	558.786336	65.0	584.0896674	80.9374732	24.1036796
77	784.219374	14.0	—	42.134347	—
97	870.653955	15.0	—	85.788623	—
222	1240.628816	21	—	1110.933788	—

Table 2: GA Baseline Results Across City Sizes (From Project 4)

Number of Cities	GA Best Distance	GA Avg Generation	GA Avg Distance	GA Avg Runtime (s)	GA Runtime Std. Dev.
11	351.04588	13.6	351.6032605	1.8708531	0.6926769
22	416.094642	28.0	419.369635	28.519665	14.359862
44	552.9840449	38.2	575.061841	111.6569093	40.5010431
77	798.508927	28	—	708.329455	—

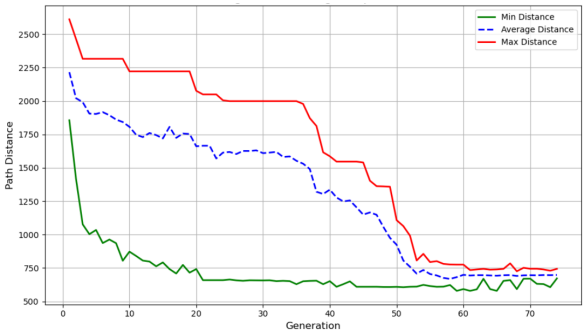

Figure 5: Convergence of GA–WoC for the 44-city TSP. The plot shows the minimum (green), average (blue dashed), and maximum (red) path distances across generations.

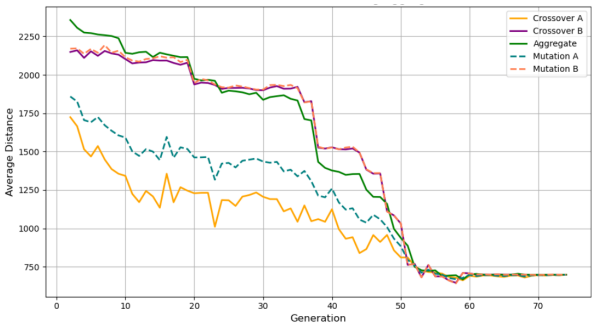

Figure 6: Average path distances across generations for each GA–WoC operator on the 44-city TSP instance.

Figure 6 compares the performance of the different evolutionary operators used within the GA–WoC framework for the 44-city TSP instance. Each curve represents the average path distance of tours generated by a particular method across successive generations. Early in the evolution, all methods begin with relatively high distances exceeding 2000, but their convergence trajectories differ markedly.

Crossover A (orange) demonstrates the fastest early improvement, achieving substantial distance reductions within the first 15 generations. This behavior reflects its greedy structure, which strongly prioritizes high-frequency and short edges, allowing it to exploit population-level knowledge quickly. In contrast, Crossover B (purple) and the Aggregate method (green) show slower but steadier convergence, maintaining consistency across generations. The Aggregate operator, in particular, yields stable and reliable improvement as it progressively integrates crowd-informed edges into the population, demonstrating the benefit of collective learning.

Both Mutation A (teal dashed) and Mutation B (coral dashed) primarily act as fine-tuning mechanisms rather than major contributors to early improvement. Their influence becomes more apparent in later generations (after generation 40), where they help smooth convergence and refine near-optimal routes.

By generation 60, all methods converge toward the same near-optimal distance (≈ 700), confirming that the hybridized combination of crowd-informed and parent-based exploration strategies collectively leads to convergence on high-quality solutions. The figure highlights that Crossover A drives early exploitation, the Aggregate operator ensures stable global guidance, and the mutation operators preserve diversity and prevent premature convergence.

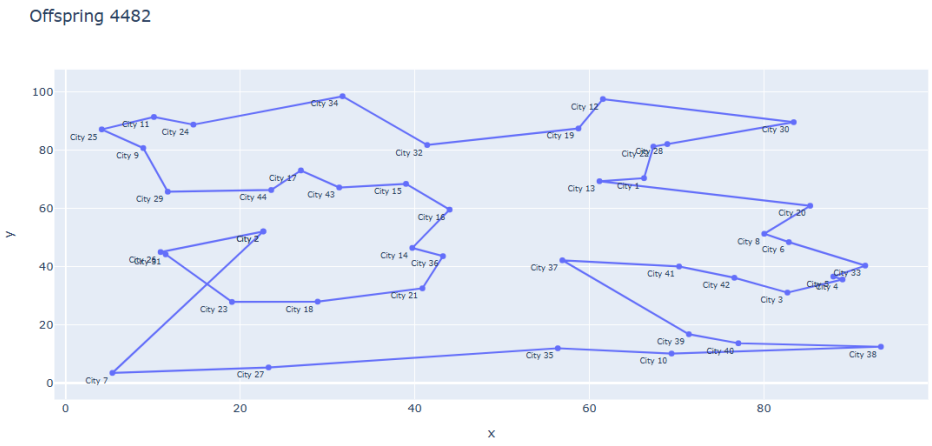

Figure 7: Optimized 44-city tour for offspring 4482 generated by the GA–WoC hybrid algorithm.

Figure 7 visualizes the tour of the best offspring (Offspring 4482) generated during the GA–WoC run for the 44-city TSP instance. Each labeled point represents a city's (x, y) coordinate, and the connecting lines indicate the final traversal sequence of the tour. The tour demonstrates clear geometric efficiency: cities are visited in clusters with minimal long cross-region jumps, and the overall path exhibits smooth, continuous transitions between local neighborhoods.

IV. Conclusion

In conclusion, the hybrid Genetic Algorithm–Wisdom of Crowds model effectively combines evolutionary learning with collective intelligence to produce near-optimal solutions for the Traveling Salesman Problem. By integrating aggregate edge-frequency information into both crossover and standalone tour generation, the algorithm balances exploration and exploitation more efficiently than a standard GA. The results show faster convergence, reduced runtime variability, and strong solution quality across multiple TSP sizes. The aggregate-based operators enable population-level consensus building, while mutation and crossover maintain genetic diversity. Overall, this hybridization demonstrates that collective solution structures can meaningfully guide evolutionary search, yielding a more adaptive and computationally efficient approach to complex combinatorial optimization problems.

—

Submitted by Matthew Poncini on 26 October 2025.