Heuristics Final Project

MAJ Olin Kennedy, 2nd Lt Alex King, 2nd Lt Sung O, 2nd Lt Madison Hofmann

December 7, 2022

1 Overview

1.1 Background

Zombie Starfish is a new population-based meta-heuristic algorithm modeled after the natural ability of starfish to regenerate limbs. The algorithm simultaneously conducts exploration and exploitation. The algorithm should globally explore the search space with randomness being a key aspect. The exploitation phase focuses on investigating search spaces that seem promising.

1.2 Traveling Salesman Problem (TSP)

The traveling salesman problem, also referred to as TSP, is one of the most common problems in operations research. It has many applications and consequently is intensively studied in optimization. This NP-hard problem is in combinatorial optimization and therefore gets incredibly difficult with computational complexity. The TSP essentially asks the question: "What is the shortest route given a list of cities and distances between each pair of cities where each city is visited exactly once and returns to the origin city?" We can define a tour as just that, a route in which every city is visited once and returns to the origin city. The connection between the traveling salesman problem and the proposed Zombie Starfish model will be further explained in Section 2.

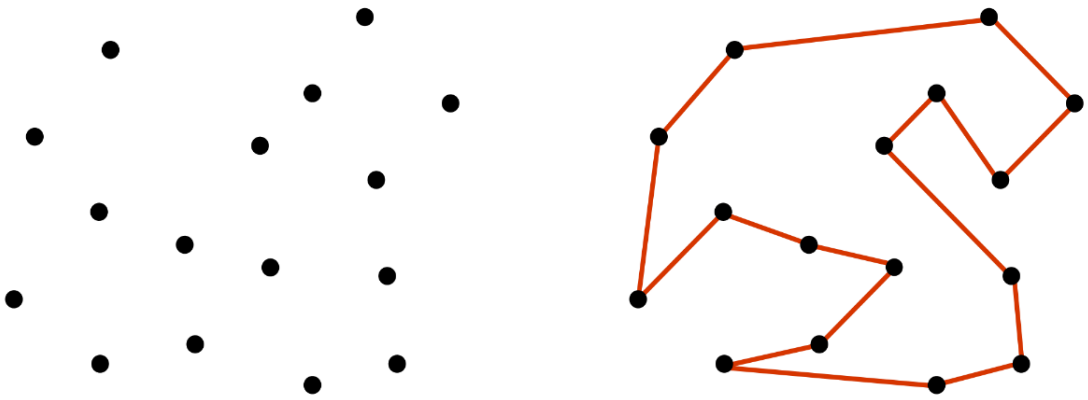

Figure 1: Traveling Salesman Problem Visualization

1.3 Project Description

The project consists of using heuristic search methods to find good solutions to the Traveling Salesman Problem. In this article, we will implement a pre-existing meta-heuristic. In our case, this pre-existing meta-heuristic will be the Firefly heuristic. We will then devise and implement our own original natural based meta-heuristic, which we named Zombie Starfish. We aim to meet the qualifications of a good heuristic: efficient, robust to multiple instances of the traveling salesman problem, and appropriate for tuning parameters. The code should take inputs of different TSP instances with little

to no changes and output the tour order and its tour length. Regarding our original meta-heuristic, we will explain the nature and methodology, relating it back to the natural behavior of starfish. The pseudocode will then follow, allowing for better understanding and implementation by other researchers. After the intricacies of the heuristic is described and applied, the article will delve into optimality gaps, run times, in the context of a performance comparison of Zombie Starfish with the pre-existing heuristic of the Firefly. We will then conclude with considerations for improvement as well as potential for further research.

2 Zombie Starfish

2.1 Nature

Starfish are a species that have the ability to complete three unique functions: Limb loss, regeneration, and re-population. First, a starfish can lose one or more limbs without dying. Each individual limb can be lost and the starfish will still survive. Secondly, after they have lost one or more limbs, each individual limb can regenerate a new body. Any limb that is removed can start to regrow its own body and become a separate and completely new starfish(Figure 2). Thirdly, their species repopulates through this regeneration. Starfish are able to lose and regenerate limbs because their vital organs are in their arms, and therefore they do not lose important bodily functions with regeneration [2]. It's common knowledge that lizards can lose their tail and subsequently regrow it. What is amazing about the Starfish is that if the lizard had the same ability as the starfish, the lizards tail would regenerate into a brand new lizard! The Zombie Starfish meta-heuristic incorporates all three unique functions of the starfish, as explained in Section 2.2.

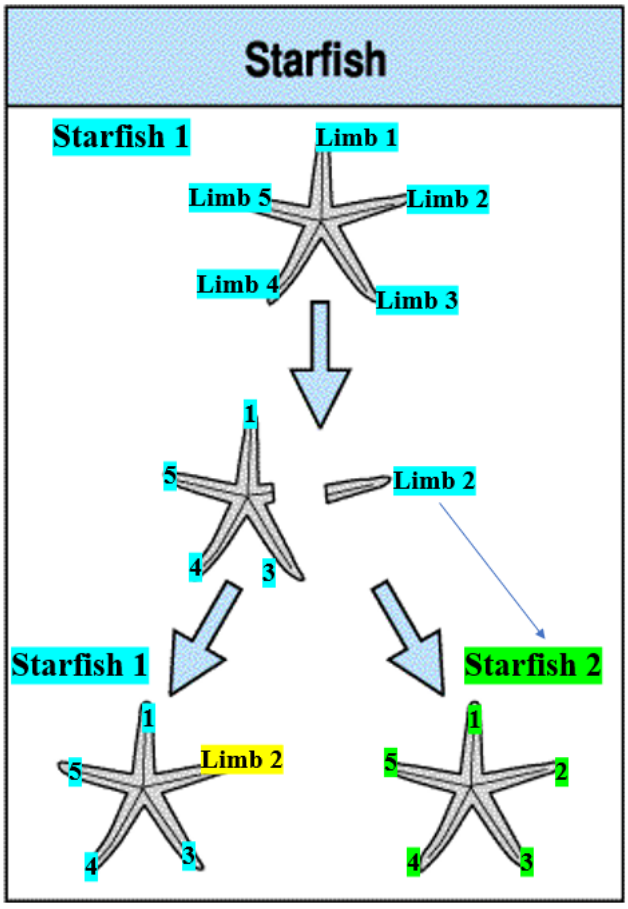

Figure 2: Starfish Regeneration Flowchart[1]

2.2 Methodology

This section connects the three unique functions of a starfish explained above to the methodology of Zombie Starfish. For ease of explanation and connecting the Zombie Starfish application to the real-world TSP, each Zombie Starfish term will be explained in terms of a simplistic TSP example at hand:

- **Semi-Complete Starfish:** A path of length 15: [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]
- **Limb(s):** A segment(s) of the path split into five even sections of 3 'cities'. In this example, the starfish has 5 limbs.
  - Limb 1: [1,2,3]
  - Limb 2: [4,5,6]
  - Limb 3: [7,8,9]
  - Limb 4: [10,11,12]
  - Limb 5: [13,14,15]
- **Complete Starfish:** All limbs, plus the addition of the first element of the first limb. In other words, a complete tour of length 16 after returning to the 'home' city: [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,1]

2.2.1 Limb Loss

In this meta-heuristic, a limb is a segment, or portion, of an existing solution. The length of each limb is essentially a floor division of the number of cities in the tour (length of the tour), divided by the number of limbs in the starfish. Any remainder(s) as a result of the floor division is accumulated in the last limb. The Zombie Starfish meta-heuristic equates cutting off a limb with 'losing' all portions of its solution by segment. In relation to this specific TSP example, the path would 'lose' all 5 segments. Each segment will be removed from the solution and split off into its own new solution space to begin searching to regenerate its own new solution.

2.2.2 Regenerate body

This regeneration phase simultaneously conducts exploration and exploitation. Just as a lost starfish limb can regenerate to become its own new starfish, a 'lost' segment of the solution can 'regrow' any element that is not currently in the solution to become its own, new and complete solution. In the specific TSP example, when Limb 4 is 'lost' from the original solution, it must regenerate a solution that contains the cities that were in Limbs 1,2,3 and 5 to become a Semi-Complete Starfish (and eventually it will append the first element of Limb 4 to the end of the tour to become a Complete Starfish). These segments should not necessarily be added in order, and Zombie Starfish utilizes either 2-OPT (within GRASP) or Greedy Randomized Adaptive Search Procedure (GRASP) to reconstruct the best new solution.

- 2 - OPT: 2 Opt is a local search algorithm. It starts at an initial solution and looks for improvement in the neighborhood of that solution. Under the premise of the traveling salesman problem, 2 Opt will take two arcs from the route and then reconnect the arcs with each other. After calculating the new travel distance of the route, the algorithm determines if the new total travel distance has gotten shorter from the previous current route. 2 Opt continues this process until no improving swaps are remaining.
- GRASP: GRASP is a semi-greedy approach characterized by three distinct phases. First, a restricted candidate list (RCL) constructs a full solution in a semi-greedy manner. Then, a local search heuristic is implemented to find a better solution. If this solution is better than any other currently found, it is kept. Otherwise it is thrown away. Lastly, this is repeated for m iterations.

Zombie Starfish uses GRASP during the regeneration phase for the first limb in the original starting solution, so that at least one new solution is guaranteed to improve. This is tantamount to exploitation. All other limbs are built upon using the Nearest Neighbor algorithm with an incorporated restricted

candidate list (RCL). It restricts the valid candidate list to the nearest three 'neighbors' and with equal probability decideds to add one of those three 'neighbors' to the path, where 'neighbors' is equivalent to any element of any limb that has not been added to the starfish yet. This is to ensure that each new solution created is still semi-greedy, and that randomness is introduced to enhance the exploration of the feasible space. Rebuilding limbs 2 through n could best be thought of as a 'focused' exploration of the solution space since that exploration is limited by the size of the RCL, given the initial solution. Utilizing the TSP example, if Segment 4 is regenerating through GRASP, than any element ('city') of Segments 1,2,3 or 5 that have not been added to the starfish can be a 'neighbor'. Once each element of each limb of the starfish has been added it becomes a Semi-Complete Starfish. Then, the first element of the current path is added to the end of the limbs to create a complete starfish.

2.2.3 Repopulation

Repopulation occurs naturally in this heuristic because each starfish in the original population and subsequent generations goes through the process above for g generations. Each starfish first splits into its n number of limbs. Each lost limb regenerates into a completely new starfish, essentially taking the characteristics of its 'parent' limb and adding 'children' limbs until it is a completely new starfish. The end goal is to progressively create better starfish overtime, because each starfish stems from the parent, and the Starfish population (set of solutions) are likely to get better and better overtime. Because GRASP is used on the first arm of the starfish, this is equivalent to exploitation of the current solution of the starfish before it was cut up. Subsequently, using semi-greedy construction on the remaining arms is equivalent to a limited exploration because 2-opt is not applied to those arms.

Eventually, rebuilding all of these starfish limbs into new starfish leads to an explosion of population. Zombie Starfish deals with this by culling the population down to the maximum population size, using elitism within the population size to retain good solutions already found.

2.3 Tune-able Elements

The tune-able elements are broken down into three sections below: Zombie Starfish Tunable Elements, GRASP Tunable Elements, and generic stopping criteria tunable elements. Some pros of these tunable elements are that they are easy for the user to pick and understand. However, we discuss in the further research section that there could be some cons associated with the choice of n, number of limbs per starfish, and the RCL. Maybe it would be better to study the optimal number n, instead of arbitrarily picking 5 for this small-scale class example, to improve the meta-heuristic. As for the RCL, we chose to use equal probabilities to select candidates out of the RCL, but a weighted roulette method could have been implimented instead.

2.3.1 Zombie Starfish Specific Tunable Elements

n: number of limbs per starfish (number of cities in the tour, with an inherent specified length)

p: maximum starfish population size (number of iterations)

g: maximum number of generations of starfish

2.3.2 GRASP Specific Tunable Elements

p: Restricted Candidate List Criteria

Semi-Greedy Construction Procedure (Nearest Neighbor with RCL)

Probabilities associated with picking from RCL (equal probabilities in this example)

2.3.3 Stopping Criteria Tunable Elements

g: maximum number of generations of starfish

t: maximum length of time the algorithm should run

f: found best tour (This is a stopping criteria that exists because TSP solutions are provided to check this meta-heuristic against).

2.4 Pseudocode

Initialize Zombie Starfish

n: number of limbs per starfish

p: maximum starfish population size

g: maximum number of generations of starfish

t: maximum time

While Stopping Criteria not met:

for each starfish i:

Cut off all n limbs

for limb j in range(0,n):

IF j=1 for starfish i : *(If it is the first limb of starfish i)*

Reconstruct solution using construction and 2-OPT (GRASP)

ELSE:

Reconstruct solutions using Semi-Greedy Approach

Evaluate all new solutions

Discard Worst Solutions

Check Stopping Criteria

End While

Report Best

3 Performance of Zombie Starfish

In order to evaluate the performance or our newly developed heuristic, we will be comparing it to the Firefly Heuristic which is another nature based algorithm. The following section will provide a brief overview of the heuristic to understand how it functions.

3.1 Firefly Heuristic Overview

The Zombie Starfish Algorithm results will be compared to the results of the Firefly Heuristic, a pre-existing meta-heuristic that we decided to work with. The Firefly Heuristic is modeled after fireflies' mating behaviors (they are unisex, meaning that every firefly is attracted to every other firefly). Within the heuristic, fireflies are only attracted to brighter fireflies and will move towards them, which makes the firefly in question brighter. This heuristic can be applied to the TSP using a discrete version, where each 'firefly' is a feasible TSP tour or solution. The distance from one 'firefly' to another can be identified as the number of edges that differ between each specific solution, and the attractiveness is determined using a function of the 'distance' between tours and how 'bright' or good each solution is. The firefly with the worse solution will move towards a a better, or more 'attractive', solution. Each time a 'firefly' moves towards another brighter one, it will create a certain number of new solutions. Each iteration of this heuristic has each firefly create however many (a preset number) new solutions, and from those solutions it will select a new population made up of the best solutions. This will continue for a set number of iterations, keeping track of the overall best solution found.

3.2 Performance Comparison

This study utilizes three datasets to apply and compare the two metaheuristics upon.

Dataset 1: BAYS29

Dataset 2: EIL51

Dataset 3: KROA100

		Firefly	**Zombie Starfish**
Dataset 1	Average Time To Complete	28.13926	27.10222
	Standard Deviation	6.78139	13.36264
Dataset 2	Average Time To Complete	54.48979	116.97725
	Standard Deviation	14.50349	49.40832
Dataset 3	Average Time To Complete	257.64200	489.50543
	Standard Deviation	86.65273	211.02921

Table 1: Time Results (sec)

At first glance of Table 1, Zombie Starfish appears to be incredibly disappointing when only looking at the time to complete. It only narrowly beats out the Firefly algorithm using Dataset 1. With the other datasets that we compare the algorithms on, Zombie Starfish takes signficantly longer than the established heuristic. However, this only tells part of the story. It does not take into account the quality of the solution that the heuristics come to. Another reason for the speed of the firefly algorithm is that it converges and then stops, which results in a shorter runtime compared to Zombie Starfish which does not converge.

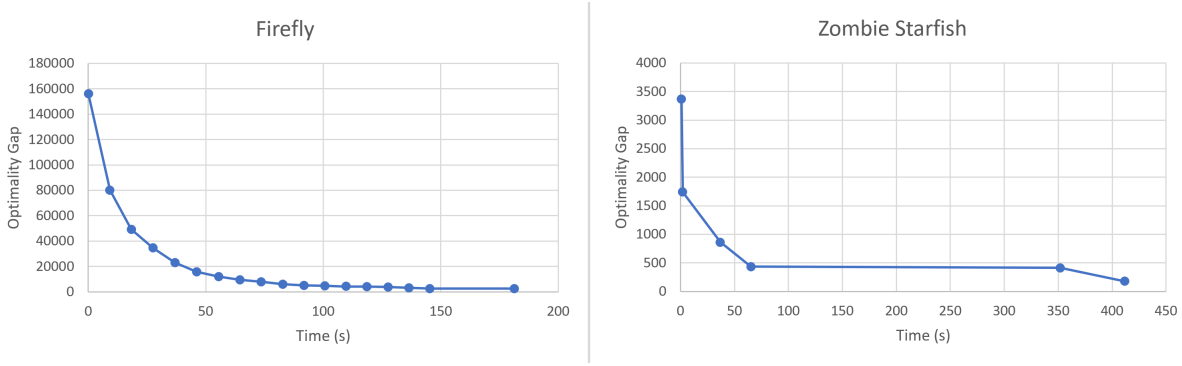

Figure 3: Optimality Gap Comparison

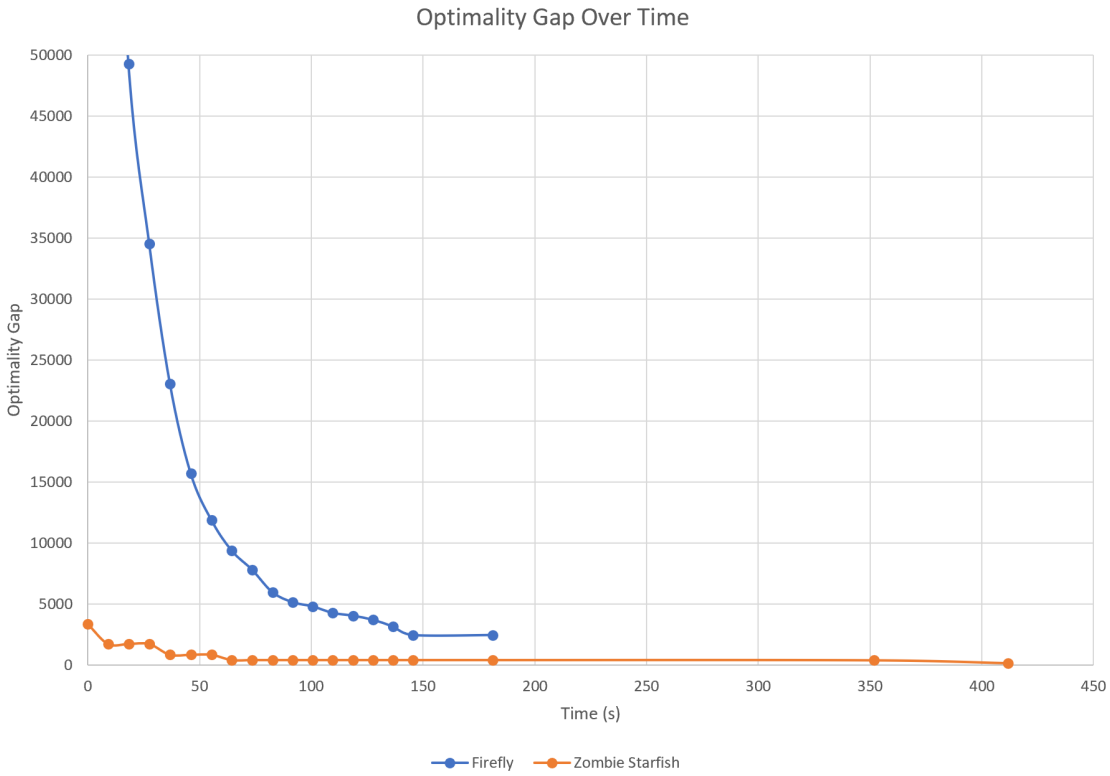

Figure 4: Optimality Gap Combined

Figure 3 and 4 show the optimality gap comparison for a run with Dataset 3, which contains 100 cities. This was chosen due to the ease of displaying the difference in the two heuristics. The conclusion is similar of other datasets and other runs. From only looking at Table 1 and Figure 3, we can see that the average time to complete the algorithm favors the Firefly metaheuristic significantly. However, once we look at Figure 4, Zombie Starfish starts to fare much better. In this specific example, the first iteration by the Firefly heuristic is at 156027 for the optimality gap. For Zombie Starfish, the optimality gap sits at 3372 in the first iteration. For the Firefly heuristic to make its optimality gap down to 3372, it would require around 130 seconds. Inspecting just the second iteration of the Zombie Starfish results, the optimality gap is cut to 1741. As can be seen in Figure 4, the Firefly heuristic does not even reach that low of an optimality gap in its entire run. Even more revealing is the time that Zombie Starfish took to reach that 1741 at just 9.19 seconds. For reference, the lowest optimality gap the Firefly and Zombie Starfish converged to was 2486 and 179, respectively. The Firefly cannot even get to an optimality gap as low as the second iteration of the Zombie Starfish in this specific instance. It is true that the Firefly heuristic completed its process much quicker than the original Zombie Starfish. However, the quality of the solution that Zombie Starfish settled on is much better. Table 2 outlines the difference in the quality as well. The average optimality gaps of the initial generations are magnitudes higher on the Firefly heuristic than on Zombie Starfish.

		Firefly	**Zombie Starfish**
Dataset 1	Average Initial Generation	4287	283.4
	Average Final Generation	20.2	1.2
Dataset 2	Average Initial Generation	1180.8	64.4
	Average Final Generation	40.2	7.2
Dataset 3	Average Initial Generation	144703.6	3765
	Average Final Generation	2580.8	311.2

Table 2: Initial and Final Optimality Gap

3.3 Heuristic Assessment

- Simplicity: Given that this a nature-inspired meta-heuristic, it is easy to understand the regeneration of starfish limbs and therefore fairly simple to apply it to regenerating new solutions for a more real-world example like the TSP.
- Reasonable Storage Requirements: Any TSP data set could be inputted into the model and Zombie Starfish would run and produce an answer. Storage requirements can be side-stepped by controlling for the max size of the starfish population.
- Speed: Zombie Starfish performs slower than the Firefly Heuristic, but provides much better answers. Overall speed is still in polynomial time since the underlying construction method and local search heuristic are both in polynomial time.
- Good answers most of the time: Since at the end of each iteration we are always selecting the best new solutions created, we will typically converge upon a good answer; furthermore, we have enough randomization included that it avoids getting stuck in a local optima.
- Low Variance: We always reached close to optimality when using Zombie Starfish, so the variance in our answers for an instance was low.
- Robustness: This meta-heuristic would need to be slightly edited to be applied to different problem types, since it was designed and implemented specifically for the TSP. However, making these slight changes would not be very complicated so long as a semi-greedy solution construction method exists and a local search heuristic (such as 2-opt) exists for that problem type
- Multiple Starting Points: By definition, Zombie Starfish is robust to multiple starting points. In effect, cutting off the limbs and rebuilding new starfish can be thought of as selecting multiple starting points from the original solution.
- Good stopping criteria: The stopping criteria do provide a trade off between exploration and exploitation as long as the user provides a large enough number of iterations (number of generations) to allow for a good solution to be reached.
- Interactive: There are tune-able elements that the analyst user can experiment with, but Zombie Starfish could add more interactive elements (at the expense of model simplicity, however).

4 Further Research

Instead of comparing Zombie Starfish to another meta-heuristic that was created in class, it would be interesting to compare the results of the Zombie Starfish meta-heuristic against a purely GRASP approach. This could be advantageous because GRASP is well-founded and has years of supporting research. This would be interesting because Zombie Starfish closely resembles GRASP, except it doesn't use arbitrary starting points (inheriting them from previous starfish) and not applying a local search heuristic to every instance (to improve exploration of the solution space).

Additionally, Zombie Starfish arbitrarily chooses n, the number of limbs per starfish, to be 5, given the test data from the course and small-scale example size. However, it would be important to study if there is an optimal number of limbs per starfish, or if n can always be arbitrary. Furthermore, researching the relationship between n and the size of the TSP problem at hand could be important in fine tuning this meta-heuristic.

Further research could also study whether or not picking the "First" limb for 2-OPT would affect optimality. Weighted roulette could be implemented to somewhat randomly choose various limbs to go through the 2-OPT portion of GRASP instead of arbitrarily picking the first limb of all starfish.

Lastly, the code for Zombie Starfish does not explore different size RCLs within GRASP. Larger RCLs would incorporate more randomness and exploration into this meta-heuristic.

5 Conclusion

Zombie Starfish is a newly developed meta-heuristic that effectively implements multiple instances of the TSP. It is modeled after the starfish species' ability to lose limbs, regenerate limbs, and repopulate. It performs better than the previously discussed meta-heuristic, Firefly. There is no guarantee of good solutions every time or good solutions as fast in Firefly, but Zombie Starfish utilizes GRASP to allow for guaranteed improvement upon at least one limb for every starfish. The applications of the two heuristics to specific datasets in this study do hint to getting to a smaller optimality gap quicker. Zombie Starfish is especially strong in its simplicity. It is simple because following the natural tendency of starfish to regenerate limbs is straightforward in applying it to searching new solution space in the TSP example. Zombie Starfish also has a strong ability to produce good answers most of the time. At the end of each iteration, it will typically converge upon a good answer and there is enough randomization included to avoid getting stuck in a local optima. Zombie Starfish could be improved with further research over time, but its current methodology lends reasonable results for instances of the real-world TSP.

References

[1] Bastiani, Mike. *Regenerative Abilities: Planarian, Hydra, Starfish. University of Utah*, https://bastiani.biology.utah.edu/courses/3230/db20lecture/Lectures/a8Pattern.html. Accessed 2 Dec. 2022.

[2] Sartore, Joel. "Starfish." *National Geographic*, https://tinyurl.com/2p82wuhr. Accessed 29 Nov. 2022.