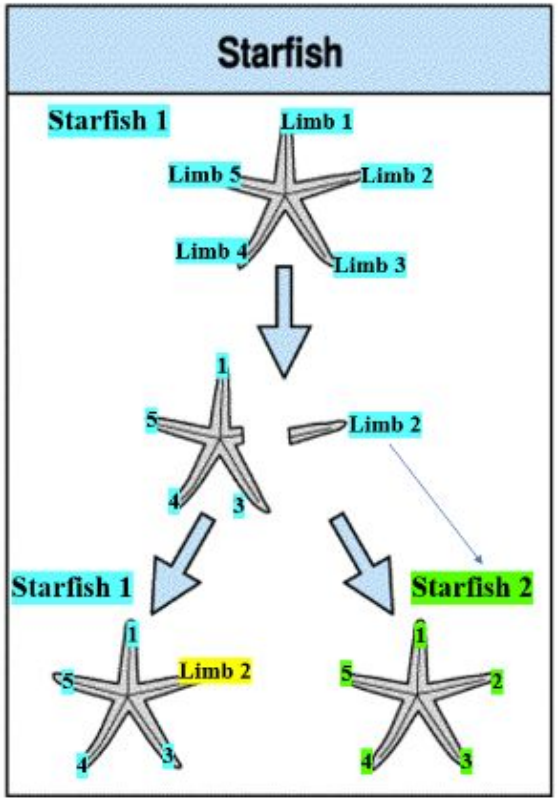
Zombie Starfish vs. Firefly Algorithm On the TSP Problem

MAJ Olin Kennedy
LT Alexandra King
LT Madison Hofmann
LT Sung O

Heuristic Overview

Starfish Regeneration Ability is Terrifying

Zombie Starfish



They say what doesn't kill you makes you stronger…….

What if that meant re-growing your entire body from an arm?

And doing that if each of your arms get chopped off?

Pseudo Code

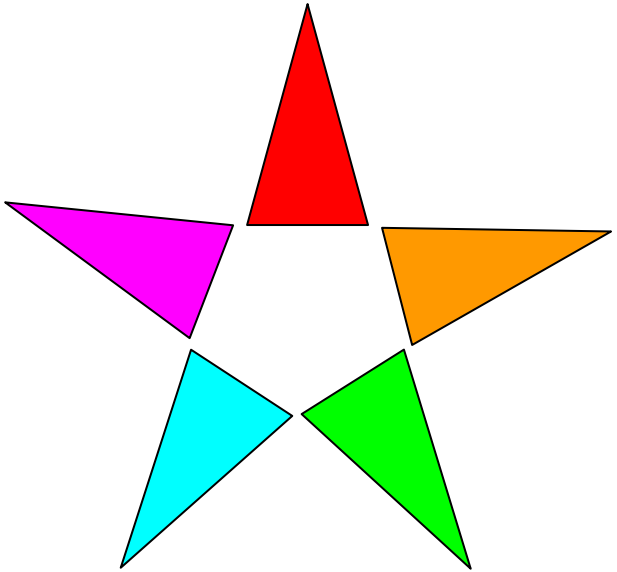
Initialize Zombie Starfish:
    n = number of limbs per starfish
    p = maximum starfish population size
    g = maximum number of generations of starfish
    t = maximum time

While Stopping Criteria not met:
    For each starfish *i*:
        Cut off *n* limbs

        For limb *j* in range(0, *n*):
            IF First Limb:
                reconstruct solution using GRASP
            ELSE:
                reconstruct solution using semi-greedy approach

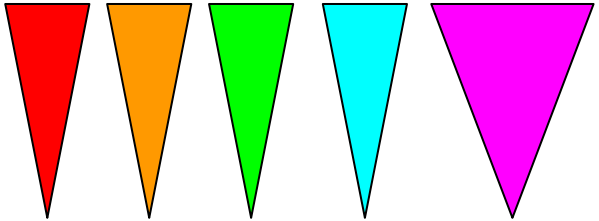
        Evaluate all new solutions
        Discard Worst Solutions
        Check Stopping Criteria
End While
Report Best

Zombie Starfish Small Example



A Tour corresponds to the starfish as so:

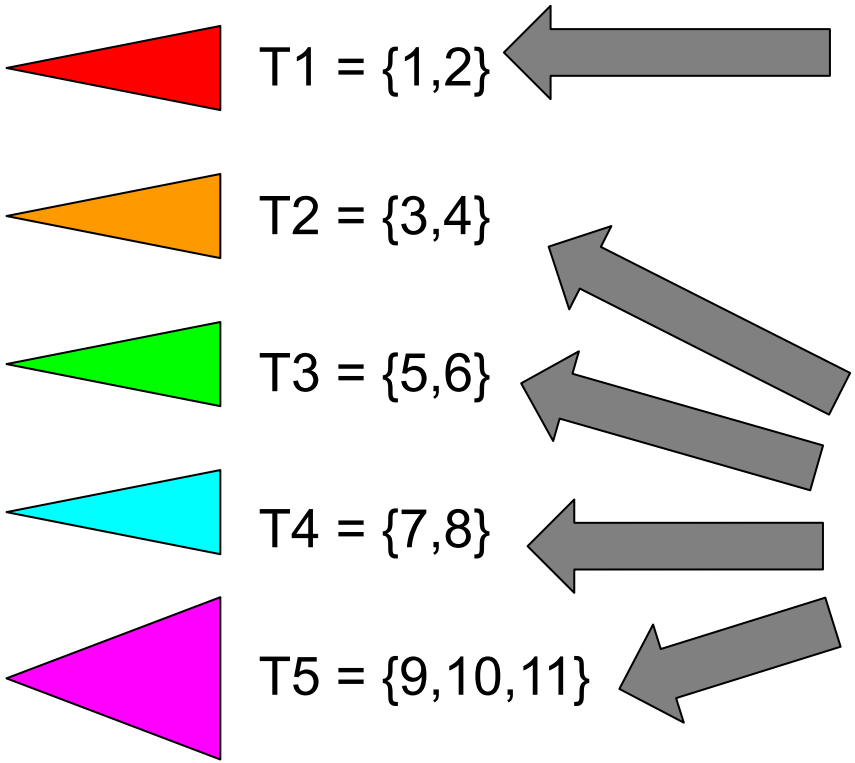
T = {1,2, 3,4, 5,6, 7,8, 9,10,11}



Length of a starfish leg is gotten by floor division. Last leg accumulates the remainder.

11 floor_divide 5 = 2

Implementation Details



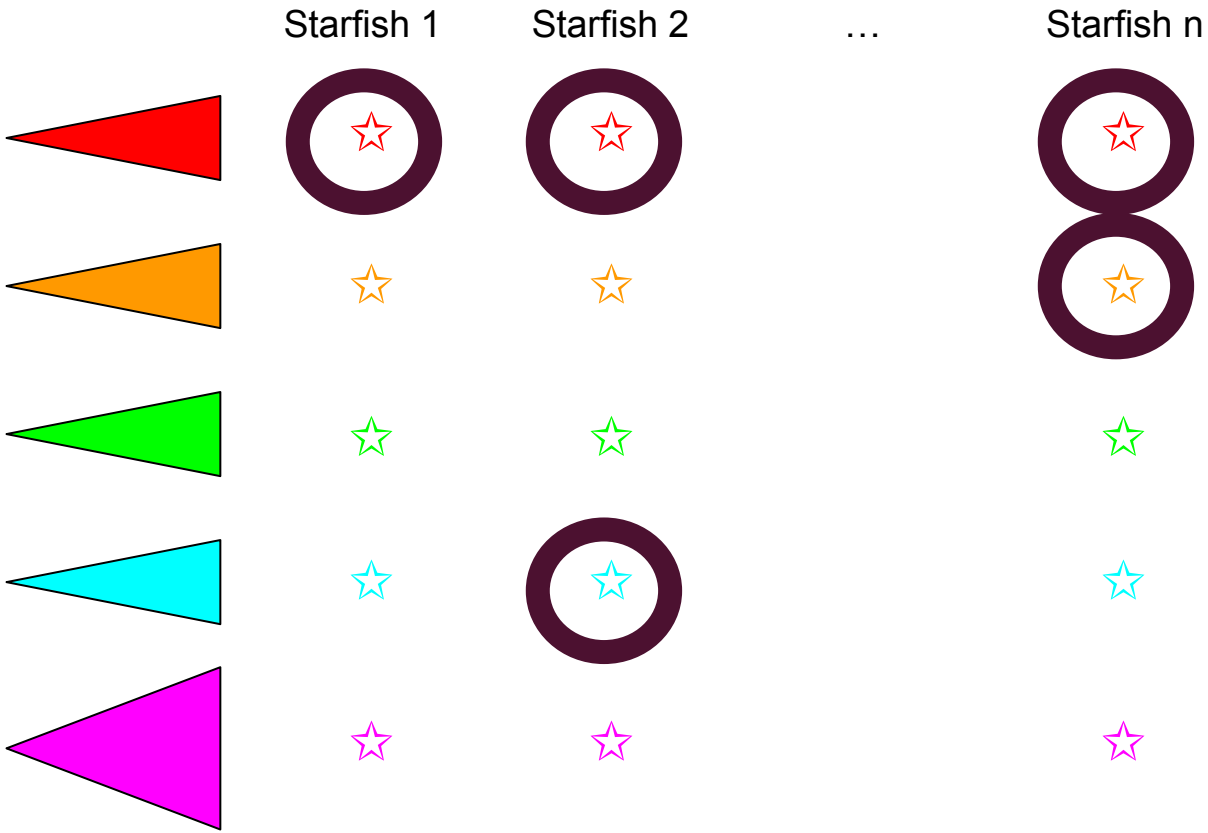
Use Grasp on first leg

- Semi-greedy Construction
- Then 2-opt
- Intensification

Use Semi-greedy Construction on remaining legs

- Diversification

Implementation Details



☆ Cull the herd!

☆ Select the Best Starfish

In this example:
Limbs/Starfish = 5
Max_Pop = 5

Continue the loop until you hit max generations or hit max time.

Testing Performance

We tested on 3 Datasets:

- Bays 29
- EIL 51
- Kroa 100

5 Runs per Dataset

2 Measures of Performance: Speed and Optimality Gap

Comparing Apples to Oranges:

- Firefly converges and stops
- Zombie Starfish always go to max gens or max runtime.

Performance Results: Speed

		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

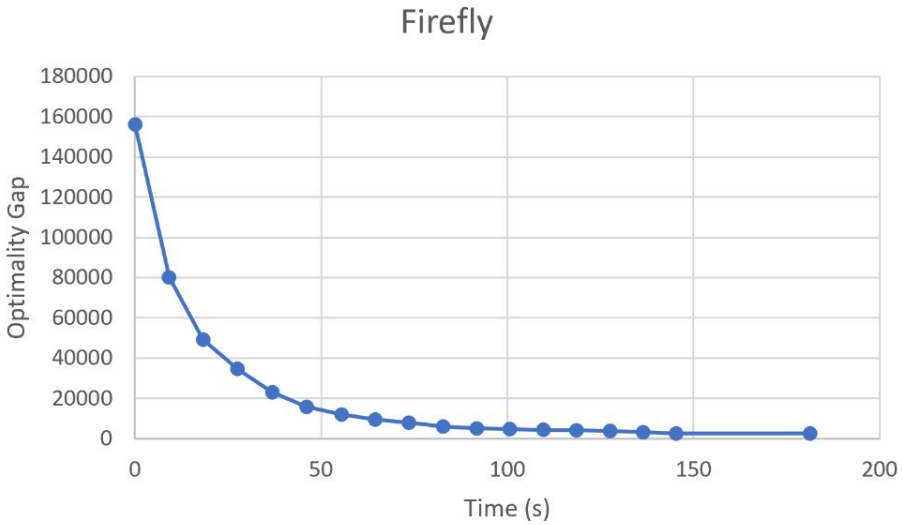
- First glance
  - Zombie Starfish performs significantly worse except on smallest dataset
- Based on completion time, not factoring in quality of solution

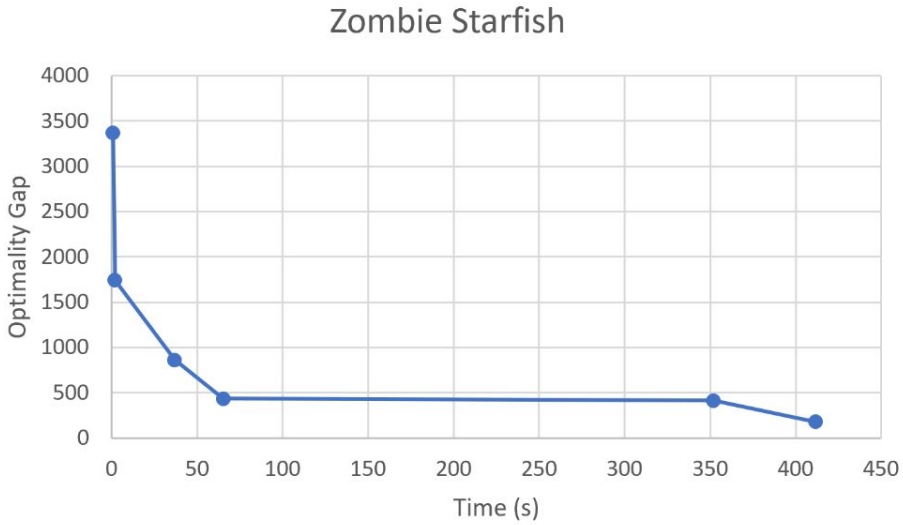
Performance Results: Optimality Gap

		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

- Zombie Starfish performs very well
- Difficult for Firefly to outperform Zombie Starfish at the earlier generations

Performance Results: Optimality Gap

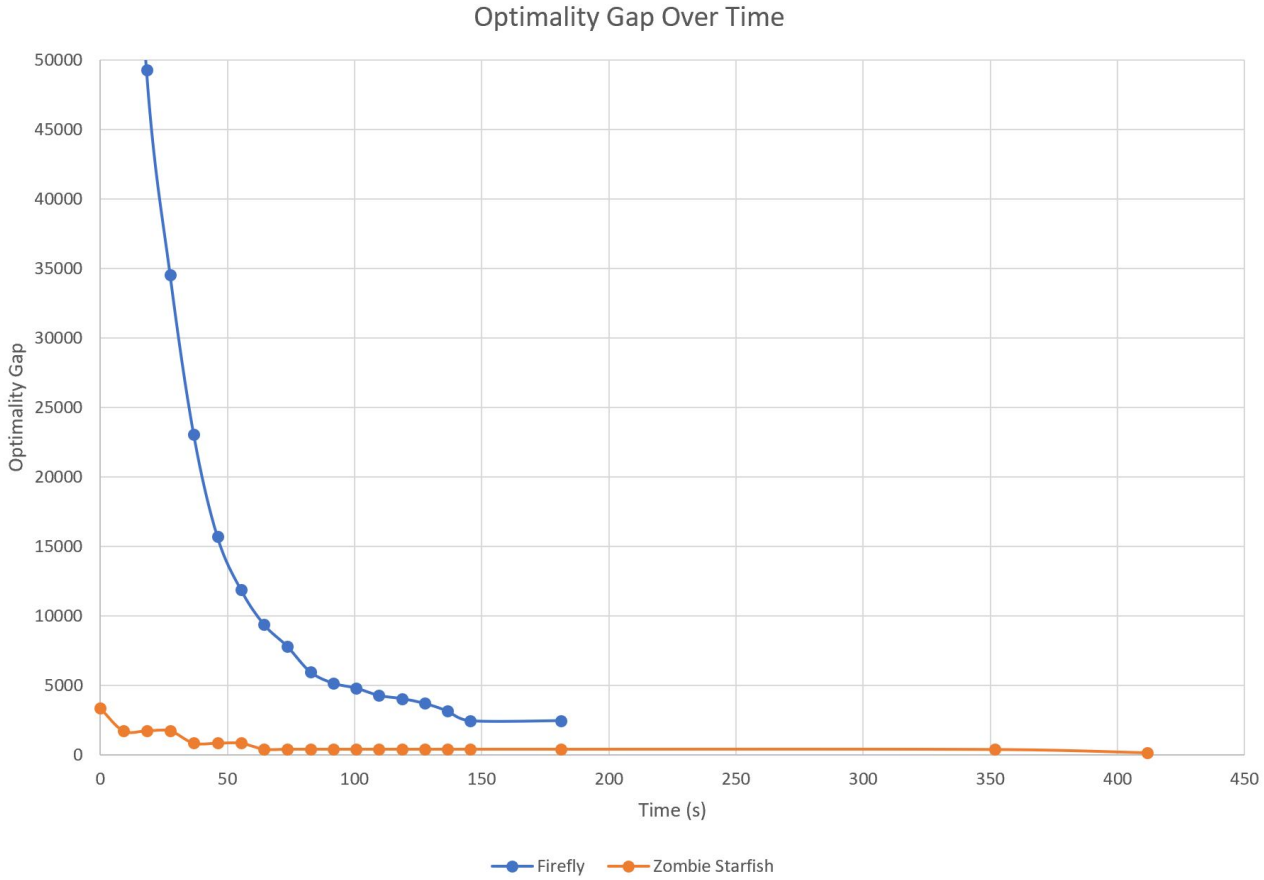




For Comparison:

KROA 100
Run 5

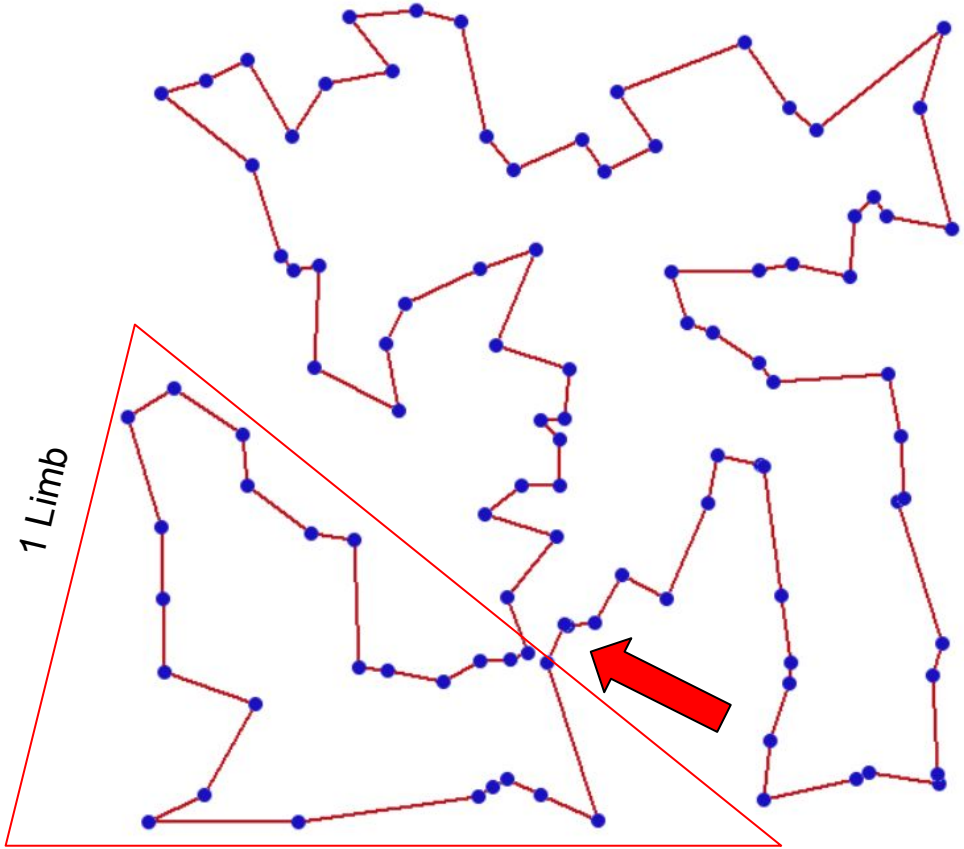
Performance Results: Optimality Gap



For Comparison:

KROA 100
Run 5

A Theoretical Explanation on why we think Zombie Starfish works



In an optimal TSP solution, there are "non-optimal" edges of the graph that must be included to be in the local area of the optimal solution.

Zombie Starfish uses a combination of a Graspy Nearest Neighbor and two-opt to give you lots of chances to get the right "non-optimal" edges in your solution and then build a solution of that.

The basic hypothesis is that if you get the right starting tour that is better described than simple GRASP (i.e. single starting point), then you'll get better solutions overall.

Another way to think about Zombie Starfish

Traditional Grasp from a single starting point

n=4 starting points,
limb length of 1

Zombie Starfish Single Starting Points with limb length of 2

X			
	X		
		X	
			X

n * (n-1) starting points for limb length of 2

$$\prod_{len=1}^{L} ((n+1)\text{-length})$$

***** Less Restrictive!**

Tunable Elements

- n = number of limbs per starfish (number of cities in the tour, with an inherent specified length)
  - This is key because it sets how we subdivide the solution space
- p = maximum starfish population size (number of iterations)
- g = maximum number of generations of starfish
- Inherited Tunable Element: Size of the restricted candidate list (RCL) for the semi-greedy construction
  - We used RCL size of 3, equal probability of choosing any item in the RCL

Possible Future Research

Zombie Starfish is better than Firefly on TSP. Is it better than plain GRASP?

Hypothesis: Yes, it'd be better since you build from previous generation good solutions

Is there an optimal length to a starfish arm? Number of arms per starfish? How is this dependent on the size of the TSP instance?

Don't Know

Our semi-greedy construction used an RCL of size 3. If we were to parameterize this, would there be an generalized, optimal in most cases RCL size?

Maybe?

Questions?