Astro-TSP: Traveling Salesman Problem Based Solutions for Scheduling Astronomical Observations

Mitchell Humphries

Department of Computer Science

California State Polytechnic University, Pomona

Overview

- Research Problem
- Background
- State of the Art Solutions
  - TSP Variants
- Solutions:
  - Simple Heuristics
  - Genetic Algorithm
- Results and Analysis
- Conclusions and Future Directions

Research Problem

- Increase utilization of the Palomar Observatory by improving the observation schedule.
- Maximize the number of of observations performed.
- Help to advance the field of astronomy and astrophysics.
- A variant of the Travelling-Salesman Problem (TSP) problem

Palomar Observatory/California Institute of Technology

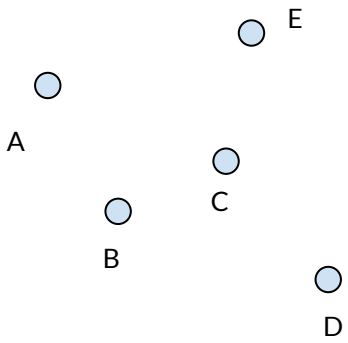
Why This Problem

- Potential advances in the classic TSP problem and solutions
  - Parallel computing application
- A limited amount of observation time is available to astronomers
  - Generalization to other observatories
- Potential generalization of solutions to other physics and computer science problems

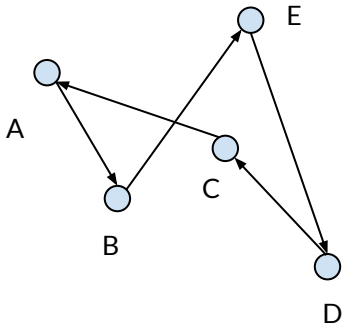
Traveling Salesman Problem (TSP)

- Find the shortest route that visits all cities once and returns to the starting city.
- NP-Hard, thus it is computationally expensive for large problem sizes.

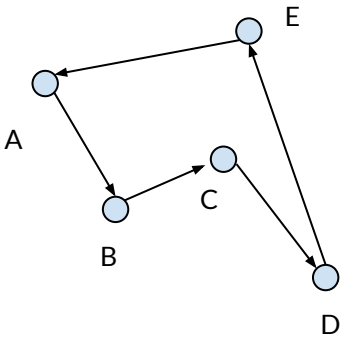
Input Nodes



Non-Optimal
A B E D C

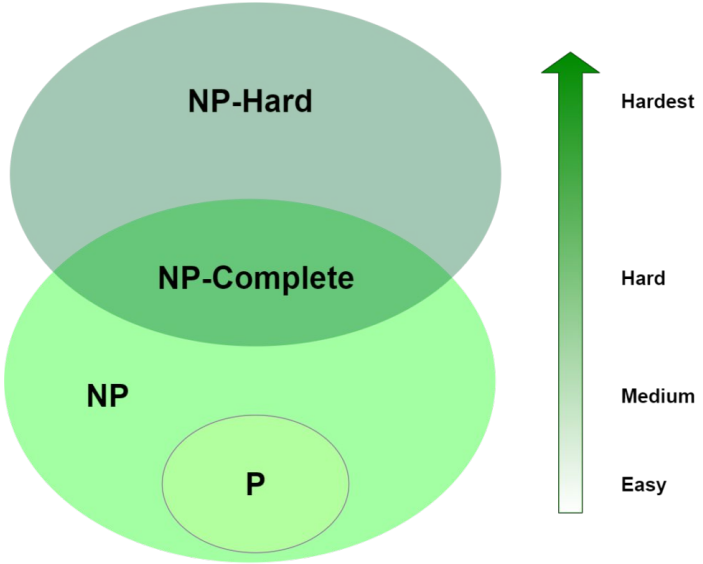


Optimal
A B C D E



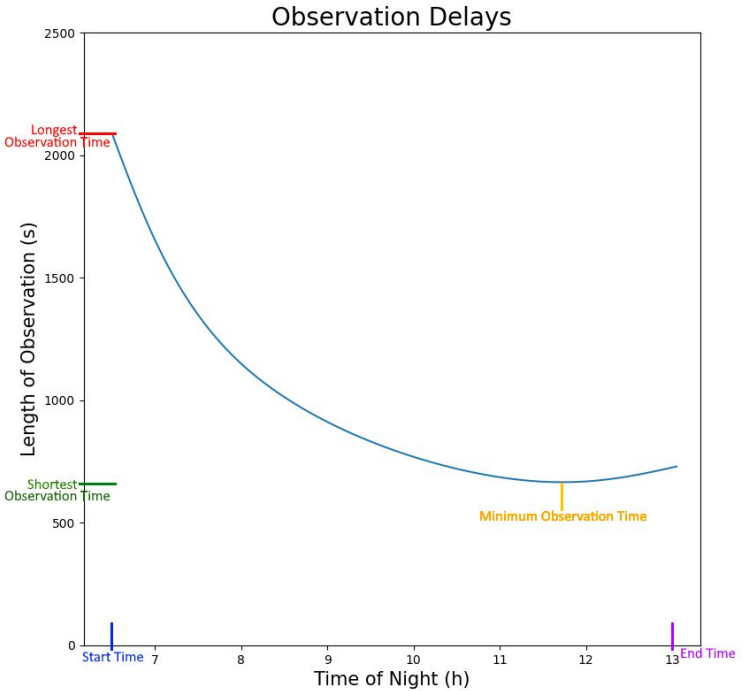
NP-Hard (Nondeterministic Polynomial-time Hard)

- NP - hard problems are the most complex to solve in computer science.
- Includes many problems:
  - Knapsack
  - Bin Packing
  - Subset Sum
  - Job Scheduling on multiple machines



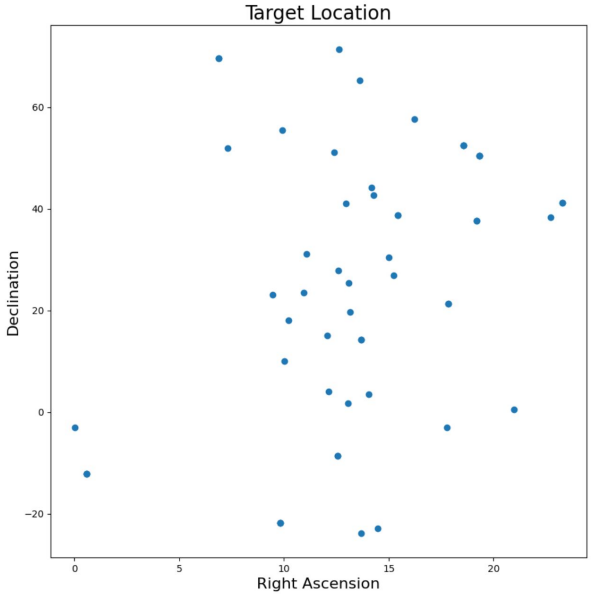
Astro-TSP Research Challenges

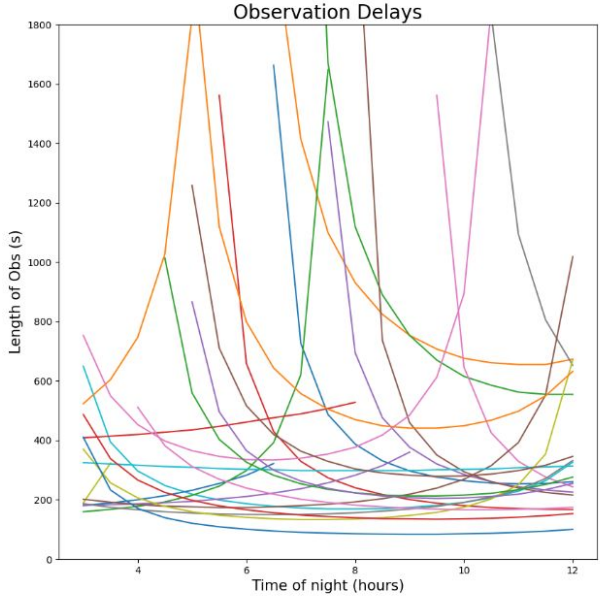
- The Astronomers TSP varies from the original TSP in several key ways, making it more complex:
  - Observations are time dependent
    - Observation time is for a fixed signal-to-noise ratio
  - Observations have time windows
  - Observations have different priorities
- Targeting at least 65 observations in a schedule
  - Search Space for TSP is n!: 65! = 8.24e90
  - 7.26e74 years for computation!!



Astro-TSP Provided Observation Data

- Target coordinates, observation times (at fixed SNR) and slew times were provided by Chaz Shapiro using example target lists and the Exposure Time Calculator for the Next Generation Palomar Spectrograph.





State of the Art | Current Solution

- The current solution is to create schedules by hand.
- West to East sorting is a good rule of thumb to catch targets before they set.
  - Improvements come with experience and practice.
- Simple Sort based on observation time

Hale Telescope dome composite.
(Palomar/Caltech/Annie Mejía)

State of the art | TSP Variants

- TSP Decision Problem
  - Given a set of cities and a specified distance of k
  - Does a tour exist with a total distance less than k?
  - NP-Complete
- Time-Dependent Travelling Salesman Problem (TDTSP):
  - Given a set of cities whose cost changes with respect to time or after any city is visited
  - Find the shortest path that visits all cities once and returns to the starting city
- Travelling Salesman Problem with Time Windows (TSPTW):
  - Given a set of cities, each with a specific time window that represents when the city may be visited.
  - Find the shortest path that visits each city exactly once within its time window and returns to the starting city.
- Prize Collecting Travelling Salesman Problem (PCTSP):
  - Given a set of N cities, each with an assigned prize value and a specified distance of k
  - Find the shortest path that collects the most priority and returns to the starting city while traveling a total distance of less than k.

State of the Art | Operations Research

Linear Programming

- Clímaco *et al.* [2] compared both a Branch and Cut Algorithm and a mixed integer linear programing solution to the prize collecting TSP. Solved problems in seconds with up to 50 cities.
- Laporte *et al.* [7] proposed an integer linear programing formulation to the selective TSP (also known as the prize collecting TSP) capable of solving instances between 10 and 90 cities in under 100s.
- Kara *et al.* [6] proposed a Mixed Integer Linear Programing formulation for TSP with time windows that is an extension of Bakers formulation that was able to solve problem instances up to 100 nodes to near optimicity in a matter of seconds.

TTP | Linear Programming

CalPolyPomona College of Science Computer Science

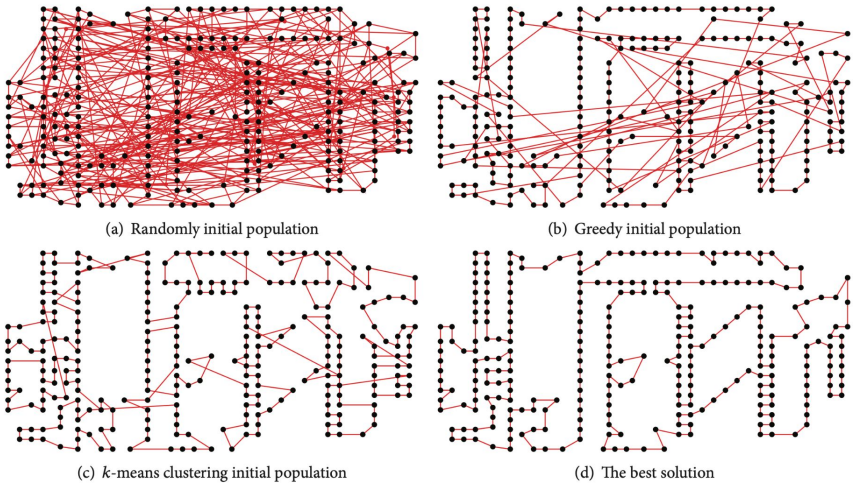
- “Solving the Traveling Telescope Problem with Mixed Integer Linear Programming.”
  - By: Luke B. Handley, Erik A. Petigura, Velibor V. Mišić
- Traveling telescope problem:
  - Targets must be accessible for at least part of the observing duration
  - Slew time may be a function of time
    - All slew times must be precomputed
- MIP formulation is made up of 9 constraints
- No inclusion of priority to the problem
  - “global optimality becomes less intuitive when targets have different numerical priorities.” [13]
- Achieved a 5x reduction in slew time compared to random ordering

Unpublished work available at: https://arxiv.org/abs/2310.18497

State of the Art | Genetic Algorithms

Genetic Algorithm

- Razali and Geraghty [9] examined various selection strategies for the TSP, showing that tournament selection is optimal while achieving an average deviation of only 0.9% from known optimal solutions.
- Deng *et al.* [4] demonstrated and purposed initialization methods that were able to frequently achieve within 10% of a known optimal solution without the need to perform crossover.

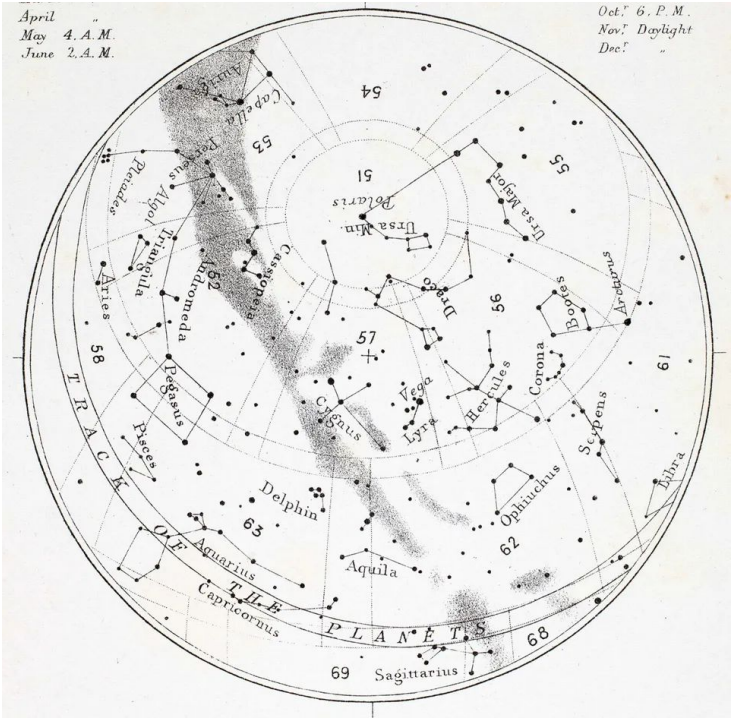


Deng et al. [4]

Solving Astro-TSP

- A schedule is considered best if it:
  - Maximizes the number of observations (accounting for priority)
  - In the event two schedules have the same total score, the best schedule is taken to be the one that minimizes the time spent observing.

- Methods investigated:
  - Simple Heuristics
    - Simple Ordering
    - Look-Ahead Greedy
  - Genetic Algorithm



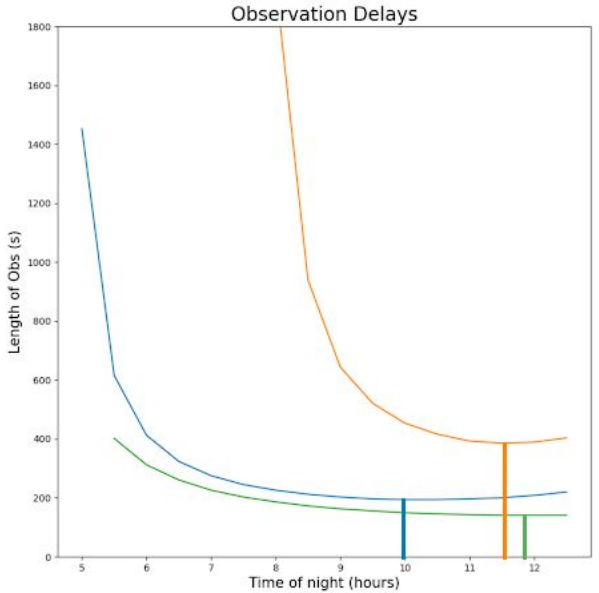
1892 Astronomy North Sky Star Map - Polaris Leo Cassiopeia

Optimization Criteria

- The best schedule should:
  - Maximize the total priority of the observations.
  - Maximize the number of unique observations.
  - Minimize the time spent observing.
- Additionally, we want to minimize the required computing time to find the solution to be less than 30 minutes

Simple Ordering

- Order the nodes based on the start times of the lowest observation time.
- Make sure we can perform the scheduled observations.
  - Check the time windows are respected for each observation
- Does not account for an observation priority



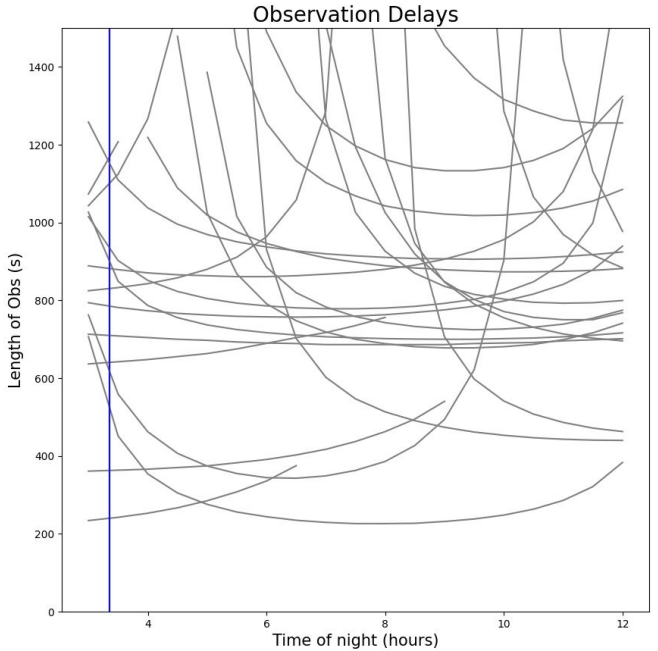
Order: Blue, Orange, Green

Look-Ahead Greedy

- **Value**: The ratio of Priority:Observation Time
- **Look-Ahead Time**: How far out to check observations

- The look-ahead time is set at 10 minutes
  - Setting it too low: risks mimicking simple ordering
  - Setting it too high: risks waiting for valuable observations while ignoring possible ones

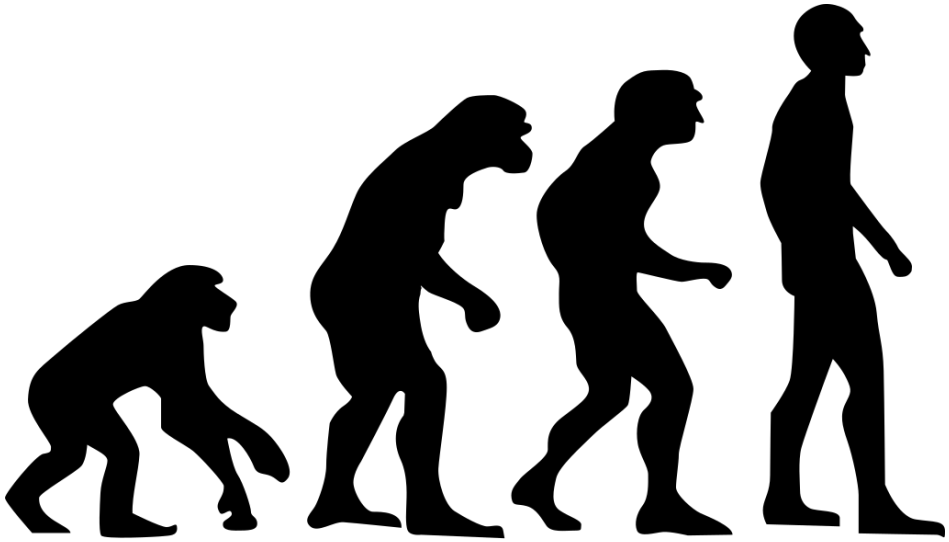
- Find the most valuable, doable, observation within the look-ahead window and add it to the schedule.
  - If no observations are possible, increase the look-ahead time until one is.



Genetic Algorithm Overview

- Metaheuristic technique inspired by biological evolution [3] [10]

- Every potential solution [schedule] is represented as an individual
- Special considerations for combinatorial optimization problems such as the TSP.
- Stochastic Algorithm
  - Relies on pseudo-random number generation
  - Non-deterministic
- No guarantee the optimal solution will be found

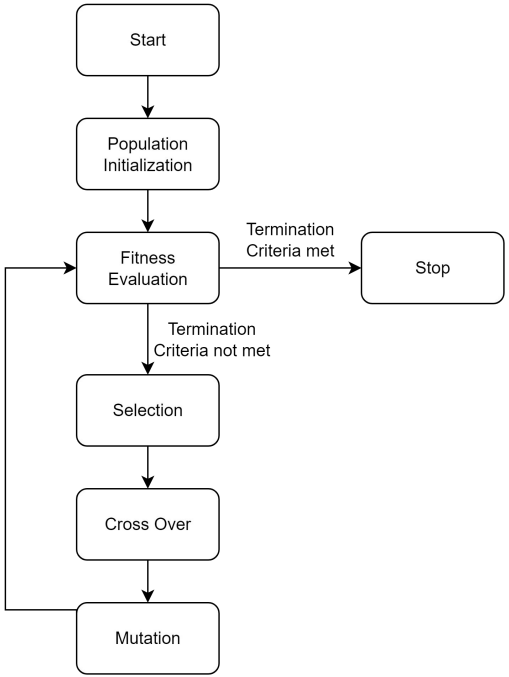


Genetic Algorithm Terminology

Combinatorial Optimization	Genetic Algorithm	Astro-TSP
Encoded Solution	Chromosome	Permutation of all observations
Solution	Decoded Chromosome	Schedule of observations
Set of Encoded Solutions	Population	A set of permutations of all observations
Objective Function	Fitness Function	A method to evaluate how strong a schedule is.

Genetic Algorithm Process Breakdown

- **Initialization**: Creation of the initial population [list of possible schedules]
- **Fitness Evaluation**: Determine how strong each individual [schedule] is and assign a numerical score
- **Selection**: Process of selecting individuals to be used to create the next generation
- **Crossover**: Process of combining and mixing two selected individuals to create a new one
- **Mutation**: Randomly changing the new individual



Encoding and Fitness Function

- **Encoding**: Representation of a solution [schedule] as an individual within the population.
- Astro-TSP: Object-based encoding, a solution is a list of objects [observations]:

Obs A	Obs B	Obs C	Obs D	Obs E

- **Fitness Function**: Used to determine a fitness value for every solution.
- Astro-TSP:

$$Fitness = Priority\ Gathered + \frac{1}{Time\ Spent\ Observing}$$

Decoding Algorithm

DECODING ALGORITHM

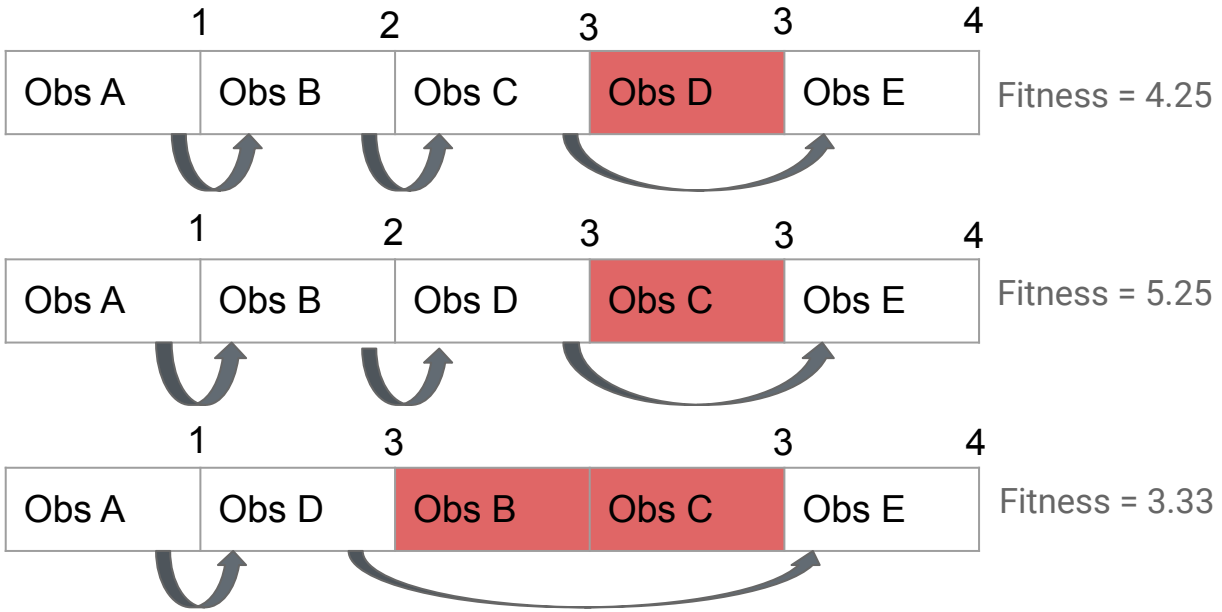
```
    Input: Observation_List # An ordered  list of all Observations
    Output: Schedule # Ordered set or subset of Observation_List
    # Create an empty Schedule
1   Schedule
    # Loop Through each observation in Observation_List
2   For Obs in Observation_List:

        # Check if Observation can be performed
3       If can_perform_observation(Obs):
            # Append Obs to the schedule
4           Schedule.append(Obs)

            # Update time and telescope position
5           update_time_and_position(Obs)
    Return Completed Schedule
6   End
```

Fitness Example

Obs	Priority	Time Window
A	1	0 - 1
B	1	1 - 2
C	1	2 - 3
D	2	2 - 3
E	1	3 - 4



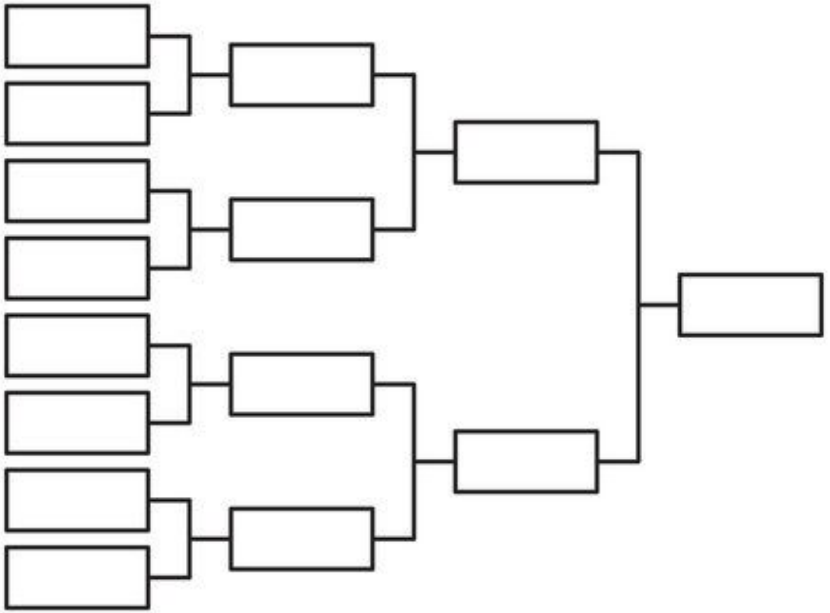
Initialization

- Creating the initial population.
- For Astro TSP: Take advantage of West to East natural order
  1. Select baseline schedule of either: modified simple schedule, or modified Look-Ahead Greedy
  2. Apply local shuffling to introduce diversity among the individuals

Obs A	Obs B	Obs C	Obs D	Obs E	No Shuffle
Obs B	Obs A	Obs C	Obs E	Obs D	Low Shuffle Parameter
Obs C	Obs D	Obs A	Obs E	Obs B	High Shuffle Parameter

Selection

- Selecting which individuals will be used to generate the next population
- Astro TSP: Tournament Selection
  - Select ‘n’ number of individuals and the most fit is selected
- Elitism: Ensure the most fit individual is copied to the next generation.



Crossover

- Combining two selected individuals to create unique offspring to add to the next generation

Mutation

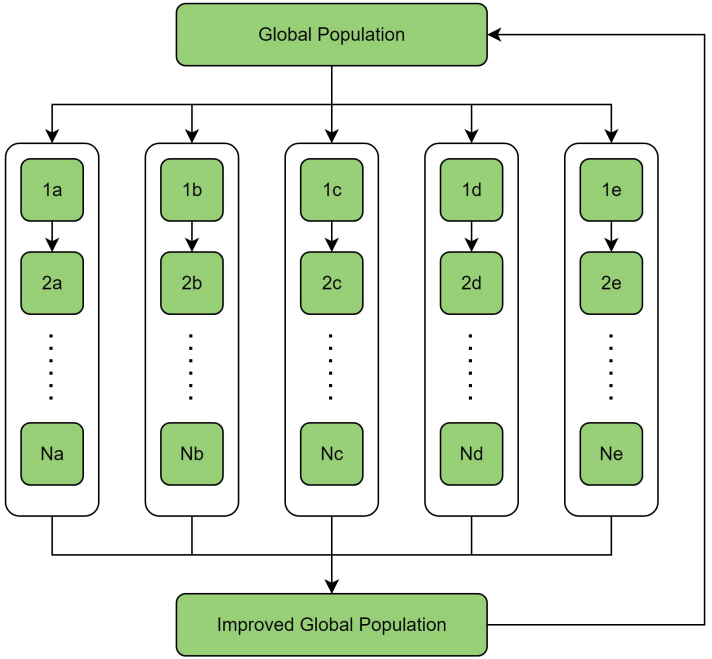
- Randomly alerting an individual
- Used to help ensure diversity is not lost among solutions
- Prevents becoming stuck in local maxima/minima
- Astro-TSP: Random neighbor swap
  - Ensures less ideal observations are still be considered

Before mutation.

After mutation

Parallelization

- Our GA solution is inherently parallelizable
  - Large populations and generations lead to better results.
- Hybrid approach for parallelization
  - Divide the global population into subpopulations dependent on the system resources
  - Run each subpopulation on its own CPU process
- Number of subpopulations should be less than or equal to the number of processors available.
- Implemented with a shared memory pool
  - Python concurrent futures



Hypothesis and Expectations of Genetic Algorithm

- Our proposed genetic algorithm will gather more priority then the Simple Solver or the Look-Ahead Greedy when:
  - The schedule is overly subscribed
  - Observations have varying priority
- The proposed genetic algorithm may not perform as well as the Simple Solver or the Look-Ahead Greedy when there is a limited number of observations.

Results Overview

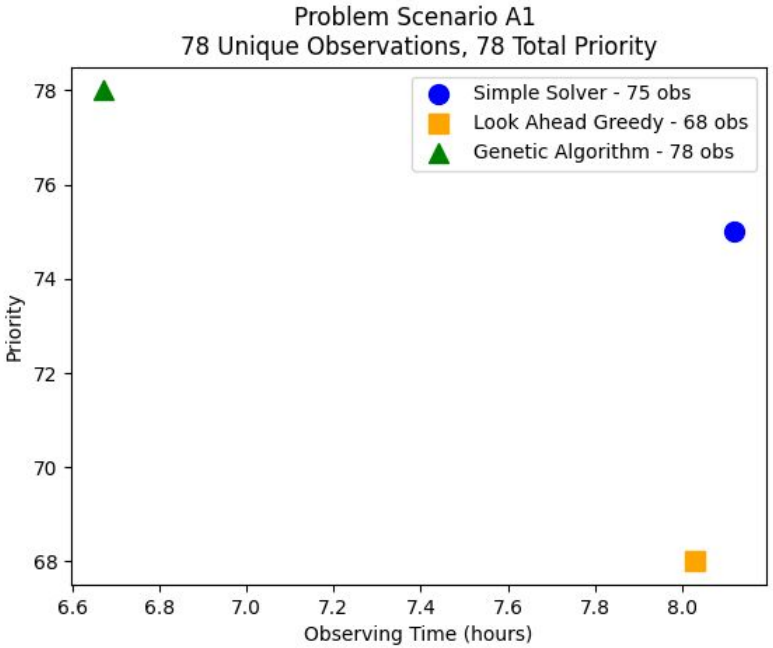
- Comparison of 5 possible scenarios
  - A1: Base Line Problem
  - A2: Priority
  - A3: Under subscription
  - A4: Over subscription
  - A5: Over subscription
- All scenerios were solved 5 times for GA

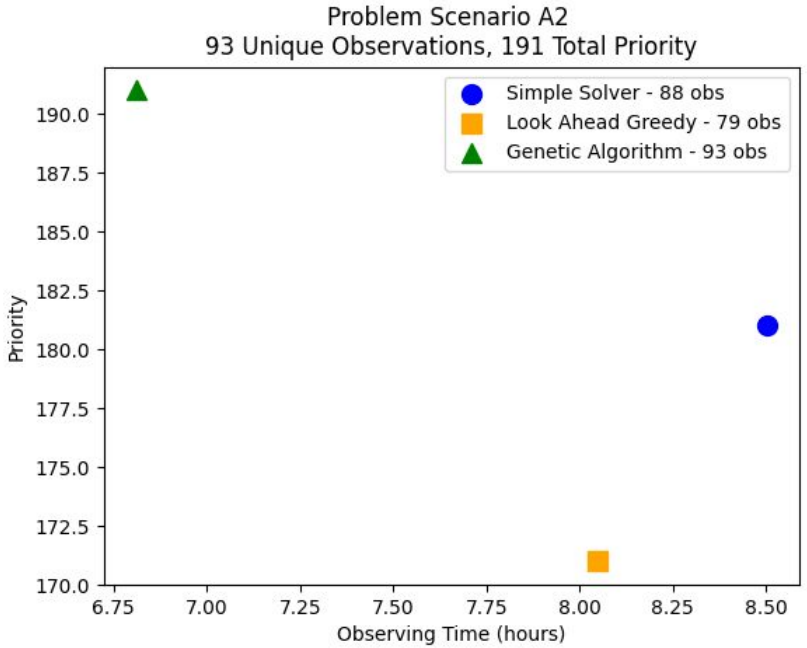
- System Resources:
  - Ryzen 9 3900X - 12 Cores, 4.1Ghz
    - Limited to 10 Cores
  - 32GB of RAM
- Key Parameters
  - 500 individuals per subpopulation
  - 10 subpopulations
  - 5 Global improvement cycles

Sample Output

Observation	Right Ascension (RA)	Declination (DECL)	Start Time	End Time
ZTF22abfulti	13.92	22.53	03:00:00	03:03:49
ZTF21abhyqlv	15.48	50.84	03:04:22	03:13:39
ZTF20acpjbgc	18.37	23.48	03:14:11	03:29:23
…	…	…	…	…

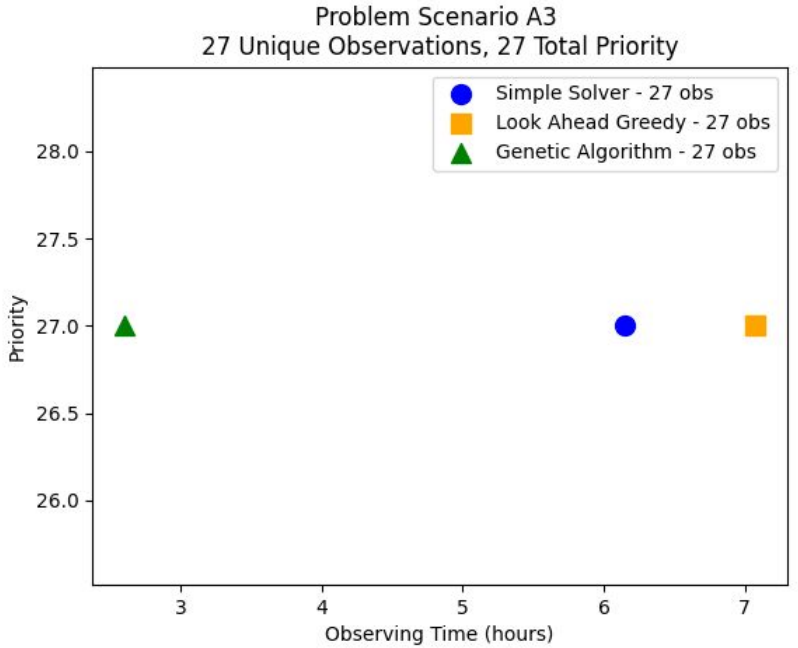
A1: Baseline | A2: Priority



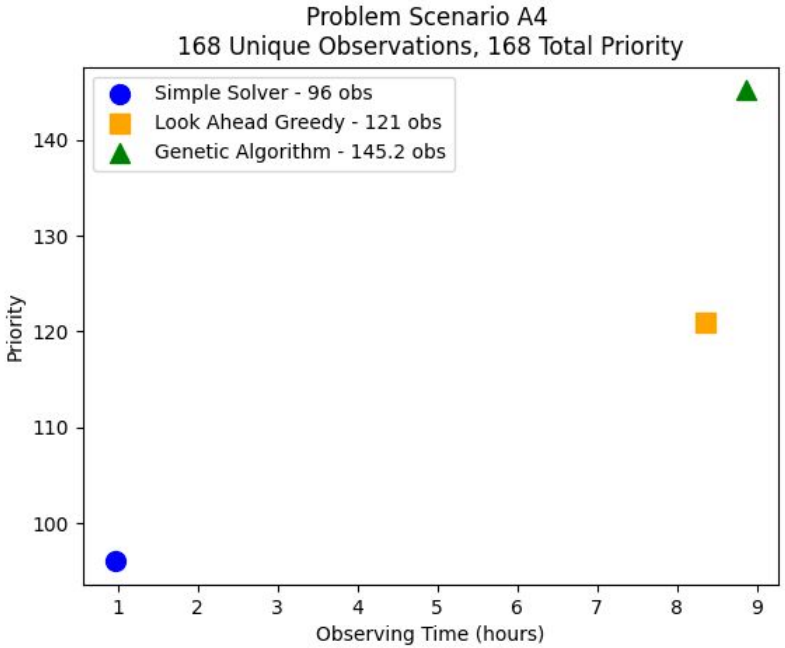


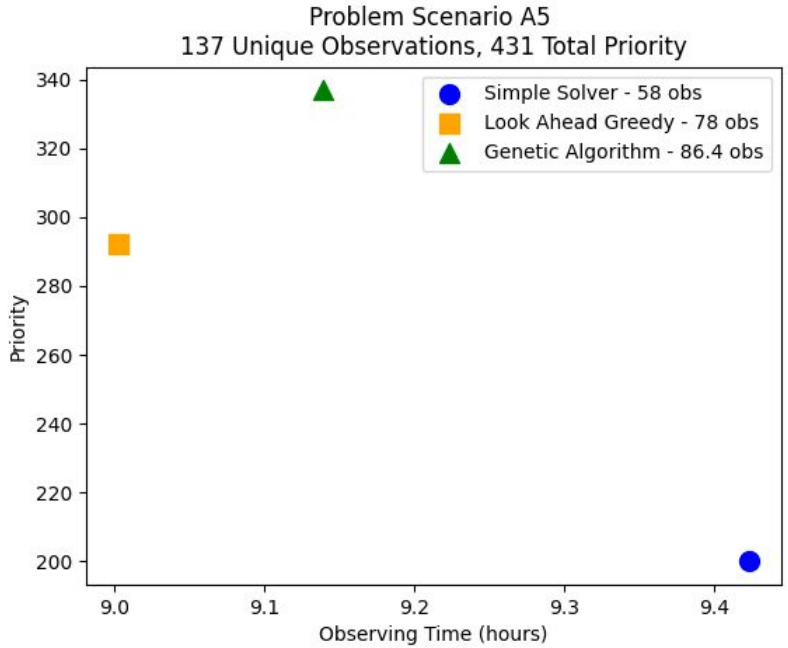
A3: Under subscription

- All three scheduling methods scheduled all observations
- GA required significantly less observing time
  - Included a long waiting time



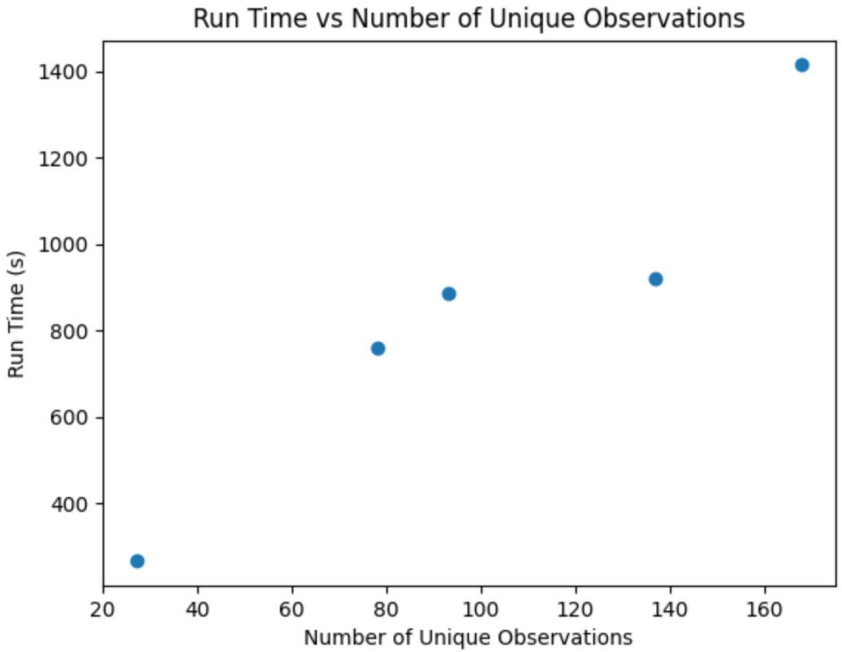
A4 & A5 Over subscription





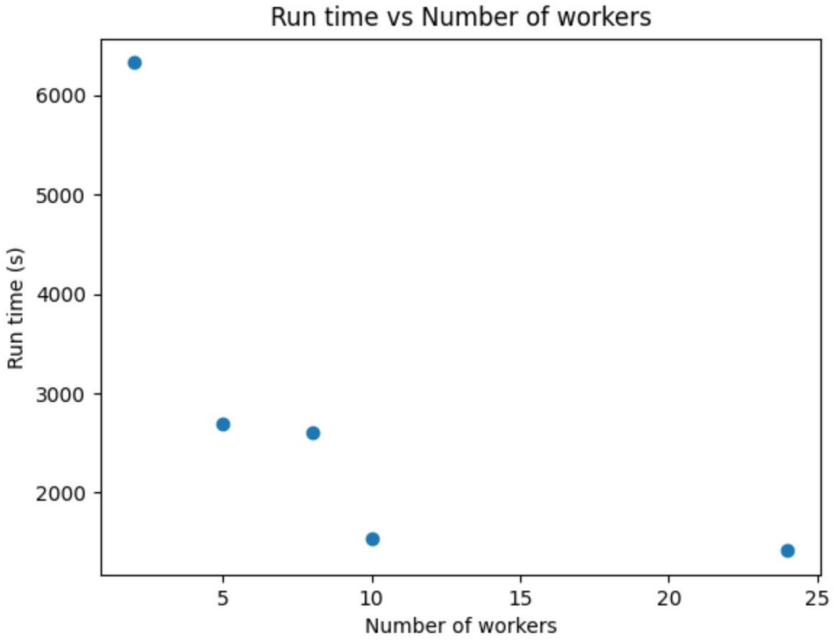
Run Time Analysis

- The runtime is affected by several configurable variables
  - Number of individuals in the population
  - Number of Generations
  - Number of observation
- Increasing the number of observations has a near-linear effect on the runtime.



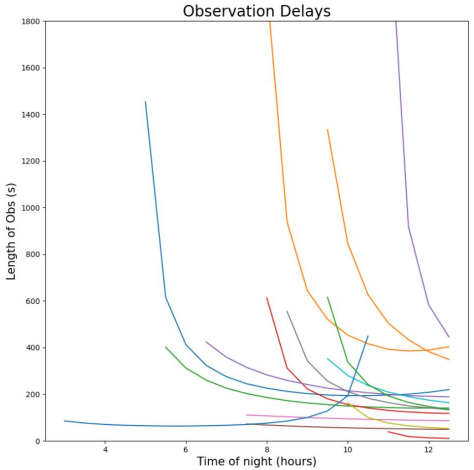
Parallelization Analysis

- The number of workers (processes) available has a clear effect on the runtime.
- Ideally, the number of processes is greater than the number of subpopulations
  - Failure to do so results in the sequential solving of subpopulations

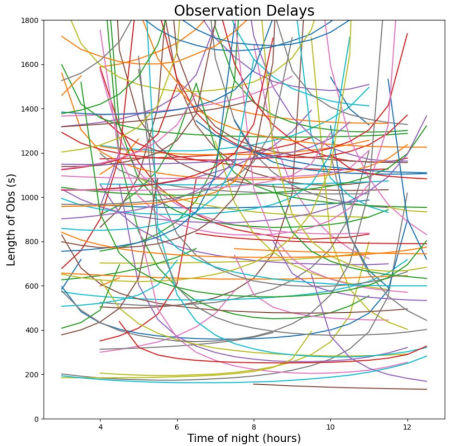


Genetic Algorithm Analysis

- Generally performs best for oversubscribed nights.
  - High Observation Times
  - Many Unique Observations
- Priority can effectively encourage certain observation to be scheduled.



Non Ideal Observation Delays



Ideal Observation Delays

CalPolyPomona
College of Science
Computer Science

Conclusion

- Shown to be a TSP Problem
- Three unique scheduling algorithms were investigated.
  - Simple Ordering
  - Look-Ahead Greedy
  - Genetic Algorithm
- Experimentally validated the viability and scalability of the Genetic Algorithm
  - The Genetic Algorithm performs best for saturated schedules
  - Capable of considering 150+ observations

Future Direction

Future development for this project would allow for:

- Improvements and optimizations of the GA-based solutions
  - Investigate additional crossover, mutation, and selection methods
  - CUDA Fitness calculations
  - Message Passing Interface (MPI)
- Additional validation with more data sets
- Development and deployment (Python package, GUI system, etc.)
  - Production-level code
  - Complete and improved documentation

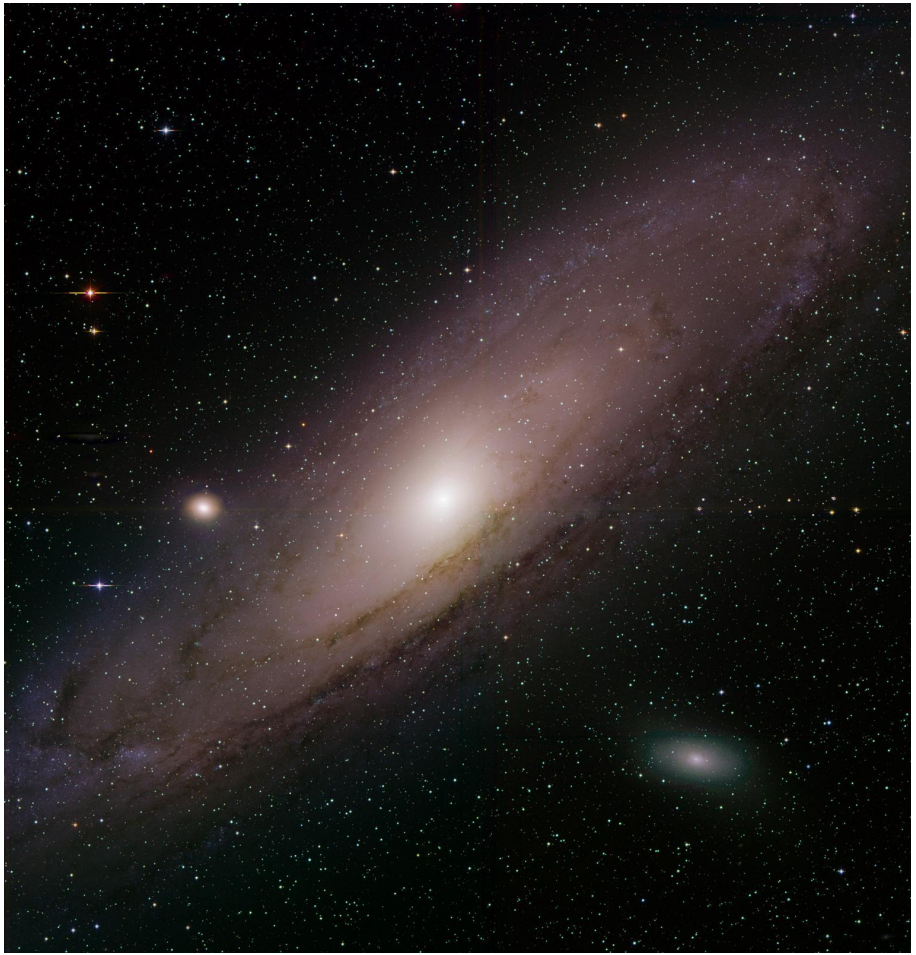
Palomar Observatory/California Institute of Technology

Acknowledgments

JPL University Crowdsourcing Initiative (JUCI)

Dr. Chaz Shapiro - JUCI Advisor

George Matta - Research Team Member



The Andromeda Galaxy as imaged by ZTF, 2019. (Caltech/Palomar/ZTF)

References

[1] Akter, S., Murad, M. W., Chaity, R. H., Sadiquzzaman, M., & Akter, S. (2020). Genetic algorithm with updated multipoint crossover technique and its application to TSP. 2020 IEEE Region 10 Symposium (TENSYMP). https://doi.org/10.1109/tensymp50017.2020.9231017

[2] Clímaco, G., Simonetti, L., & Rosseti, I. (2021). A branch-and-cut and MIP-based heuristics for the prize-collecting travelling salesman problem. RAIRO - Operations Research, 55. https://doi.org/10.1051/ro/2020002

[3] Coello, C. A., Dhaenens, C., & Jourdan, L. (2010). Multi-objective combinatorial optimization: Problematic and context. Advances in Multi-Objective Nature Inspired Computing, 1–21. https://doi.org/10.1007/978-3-642-11218-8_1

[4] Deng, Y., Liu, Y., & Zhou, D. (2015). An improved genetic algorithm with initial population strategy for symmetric TSP. Mathematical Problems in Engineering, 2015, 1–6. https://doi.org/10.1155/2015/212794

[5] Hungerländer, P., & Truden, C. (2018). Efficient and easy-to-implement mixed-integer linear programs for the traveling salesperson problem with time windows. Transportation Research Procedia, 30, 157–166. https://doi.org/10.1016/j.trpro.2018.09.018

[6] Kara, I., & Derya, T. (2015). Formulations for minimizing tour duration of the traveling salesman problem with time windows. Procedia Economics and Finance, 26, 1026–1034. https://doi.org/10.1016/s2212-5671(15)00926-0

CalPolyPomona
College of Science
Computer Science

References

[7] Laporte, G., & Martello, S. (1990). The selective travelling salesman problem. Discrete Applied Mathematics, 26(2–3), 193–207. https://doi.org/10.1016/0166-218x(90)90100-q

[8] Razali, N., & Geraghty, J. (2011). Genetic Algorithm Performance with Different Selection Strategies in Solving TSP. E World Congress on Engineering, 2.

[9] Steele, D. (2008). Eyes to the skies getting bigger and bigger. photograph. Retrieved 2023, from https://www.nbcnews.com/id/wbna22999753.

[10] Verma, S., Pant, M., & Snasel, V. (2021). A comprehensive review on NSGA-II for multi-objective combinatorial optimization problems. IEEE Access, 9, 57757–57791. https://doi.org/10.1109/access.2021.3070634

[11] Yuan, Y., Cattaruzza, D., Ogier, M., & Semet, F. (2020). A branch-and-cut algorithm for the generalized traveling salesman problem with time windows.

[12] Rick, R. (n.d.). MMT Observatory. photograph, Center for Astrophysics. Retrieved 2023, from https://www.cfa.harvard.edu/facilities-technology/telescopes-instruments/mmt-observatory.

[13] Handley, L. B., Petigura, E. A., & Mišić, V. V. (n.d.). Solving the Traveling Telescope Problem with Mixed Integer Linear Programming. Department of Physics & Astronomy and Anderson School of Management, University of California Los Angeles. Retrieved 2023. from: https://arxiv.org/abs/2310.18497

Q&A

Additional Slides Below

(Could be used in Q & A)

Linear Programming

- A mathematical representation of the problem as a series of linear equations.
- Made up of 3 main components:
  - Objective function
  - Decision variables
  - Constraints
- Implemented Time-Window TSP from Kara *et. al*.[5]

Definitions and Objective Function

Definitions:

$i = 0, 1, 2, \ldots, n + 1$

t_i : *the time node i is visited*

a_i : *the earlist time node i can be visited*

b_i : *the latest time node i can be visited*

x_{ij} : *is the path from node i to node j taken*

C_{ij} : *the cost to travel from node i to node j*

Objective function:

Minimize t_{n+1}

Example constraints:

$t_i - C_{0i} x_{0i} \geq 0 \quad , i = 1, 2, \ldots, n$

$t_i \geq a_i \quad , i = 1, 2, \ldots, n$

$t_i \leq b_i \quad , i = 1, 2, \ldots, n$

- Complete formulation made up of 9 constraints

Kara et. al. [6]

Pros and Cons | Integer Linear Programming

Pros:

- Deterministic
- Extensive, pre-existing, research
- Highly optimized ILP solvers
  - Gurobi*

Cons:

- Current TSP solutions do not take into account:
  - Time-dependency
  - Not visiting every node

Attempting the ILP formulation for Astro-TSP resulted in:

- A better understanding of the problem
- A better understanding of the provided dataset
  - East-to-West pattern
  - Similar positions have similar ideal time of night.

*https://www.gurobi.com/

ILP- TW Constraints (1 of 2)

$i = 0, 1, 2,..., n + 1$

t_i : *the time node i is visited*

a_i : *the earlist time node i can be visited*

b_i : *the latest time node i can be visited*

x_{ij} : *is the path from node i to node j taken*

C_{ij} : *the cost to travel from node i to node j*

$$(1) \quad t_i - C_{0i}x_{0i} \geq 0 \quad , i = 1, 2,..., n$$

$$(2) \quad t_i \geq a_i \quad , i = 1, 2,..., n$$

$$(3) \quad t_i \leq b_i \quad , i = 1, 2,..., n$$

$$(4) \quad t_i - t_j + (b_i - a_j + C_{ij})x_{ij} \leq b_i - a_j$$

$$i, j = 1, 2,..., n; \; i \neq j$$

[6] Kara et. al.

ILP – TW Constraints (1 of 2)

$i = 0, 1, 2,..., n + 1$

t_i : *the time node i is visited*

a_i : *the earlist time node i can be visited*

b_i : *the latest time node i can be visited*

x_{ij} : *is the path from node i to node j taken*

C_{ij} : *the cost to travel from node i to node j*

$$(5) \quad \sum_{j=0}^{n} x_{ij} = 1$$

$$(6) \quad \sum_{j=0}^{n} x_{ji} = 1$$

$$(7) \quad t_i + C_{i0} \leq t_{n+1} \quad , i = 1, 2,..., n$$

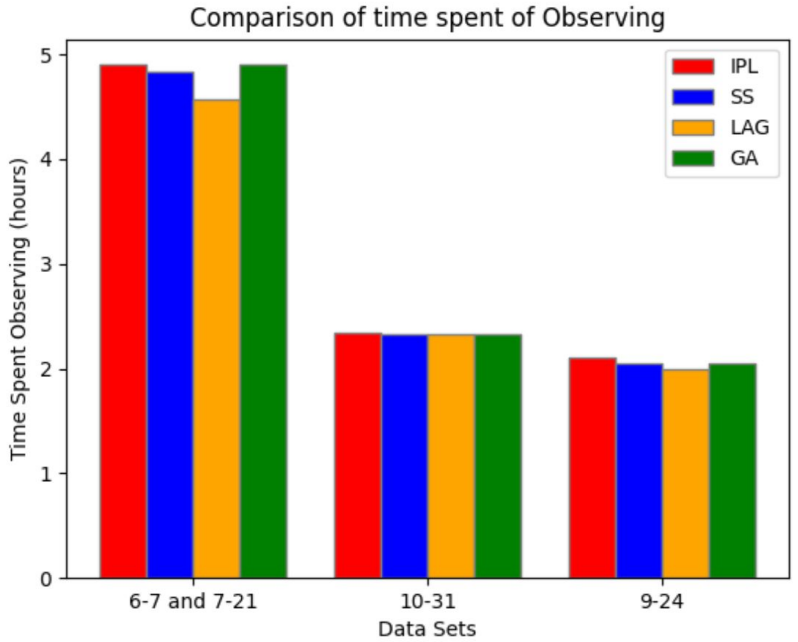
$$(8) \quad x_{ii} = 0 \quad i = 1, 2,..., n$$

$$(9) \quad x_{ij} \in \{0, 1\}, \forall (i, j) \in A$$

[6] Kara et. al.

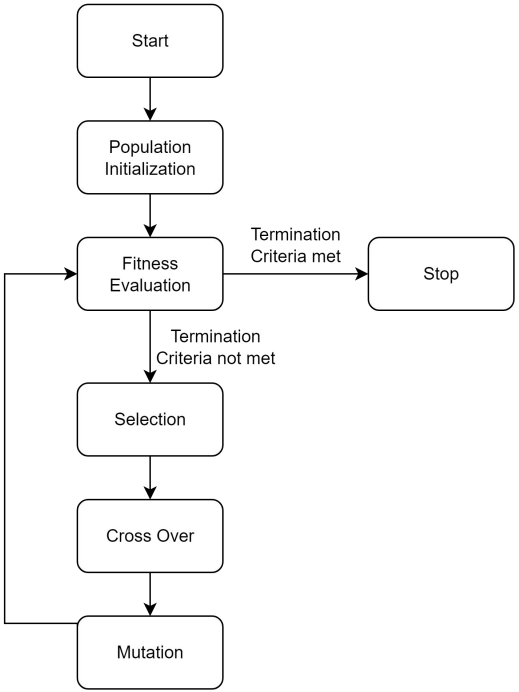
ILP Results and Comparison

- The only “fair” comparison for the ILP is to compare to the total time spent observing.
- Required conditions:
  - All Observations must be scheduled
  - Time window only
  - Equal Priority
  - no repeats



Genetic Algorithm Terminology

GA-Terminology	Astro Equivalent	
Individual	Schedule	A single schedule - the list of observations to preform.
Population	List of Schedules	A list of schedules to calculate fitnesses for.
Generation	Iteration of Schedule List	The Genetic Algorithm process



Example of Input File

	ZTF22aaykjqo						
	Name	RA	RECL	OTMstart	OTMairmass_start	OTM exptime	dt
518	ZTF22aaykjqo	5.77	31.92	2022-08-07T10:00:00	5.17	1009.04	7
519	ZTF22aaykjqo	5.77	31.92	2022-08-07T10:30:00	3.5	513.97	7.5
520	ZTF22aaykjqo	5.77	31.92	2022-08-07T11:00:00	2.64	355.45	8
521	ZTF22aaykjqo	5.77	31.92	2022-08-07T11:30:00	2.13	280.07	8.5
	ZTF22aadoihx						
	Name	RA	RECL	OTMstart	OTMairmass_start	OTMexptime	dt
527	ZTF22aadoihx	11.48	58.56	2022-08-07T4:00:00	2.01	267.75	1
528	ZTF22aadoihx	11.48	58.56	2022-08-07T4:30:00	2.28	305.05	1.5

*No Priority given, assume equal priority

2 Observations from 08-07

Sample Explanation of reading output

10:00:00 OBSERVE: ZTF22aasffwi for: 961 s, in 2 parts. For priority of: 1

Part 1: 10:00:00 to 10:08:15

Part 2: 10:08:25 to 10:16:01

10:16:01 MOVE from: (4.4, 37.52) to: (3.46, -12.68)

10:16:51 OBSERVE: ZTF22abcknjm for: 259 s, in 1 parts. For priority of: 1

10:21:10 MOVE from: (3.46, -12.68) to: (3.48, 5.79)

10:21:36 OBSERVE: ZTF22abgnagw for: 665 s, in 1 parts. For priority of: 1

10:32:41 MOVE from: (3.48, 5.79) to: (0.75, 41.77)

10:33:20 OBSERVE: ZTF22aaksqtn for: 156 s, in 1 parts. For priority of: 1

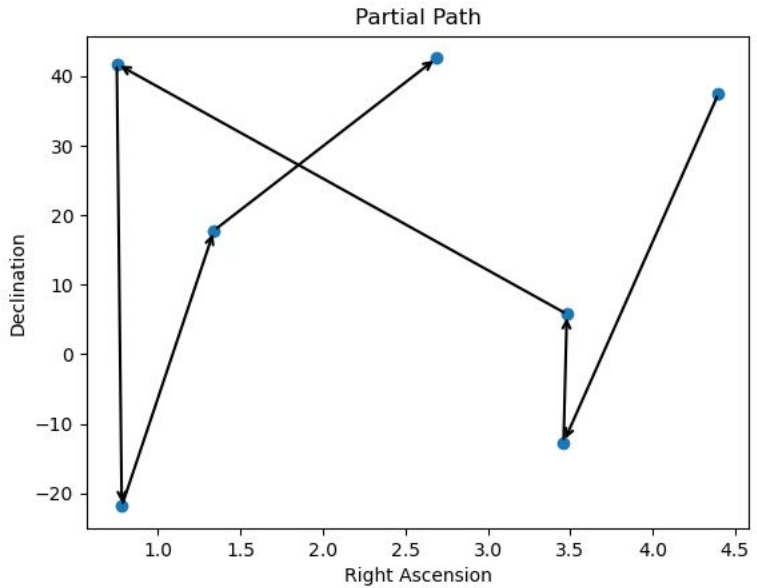
10:35:57 MOVE from: (0.75, 41.77) to: (0.78, -21.85)

10:36:57 OBSERVE: ZTF22aaytppn for: 91 s, in 1 parts. For priority of: 1

10:38:28 MOVE from: (0.78, -21.85) to: (1.34, 17.83)

10:39:10 OBSERVE: ZTF22aavbfhz for: 362 s, in 1 parts. For priority of: 1

10:45:12 MOVE from: (1.34, 17.83) to: (2.69, 42.6)



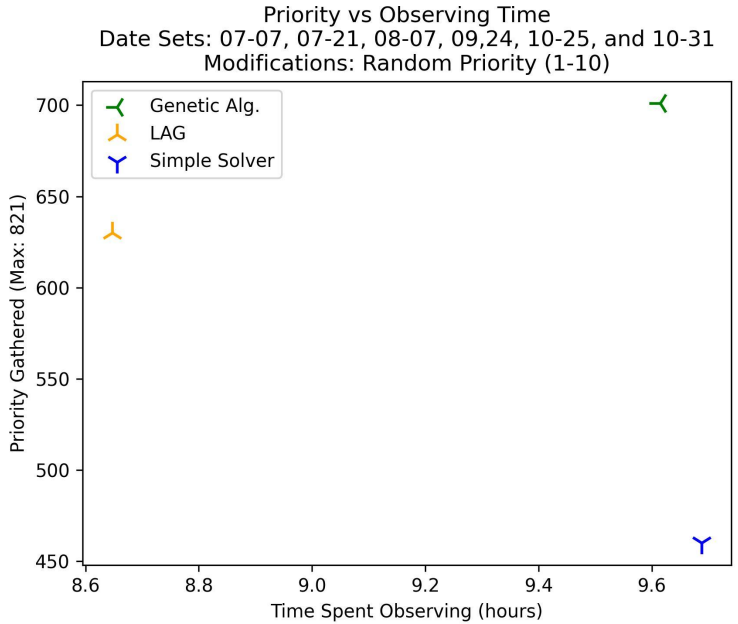
Sample Growth Slide

CalPolyPomona
College of Science
Computer Science

Growth by Global Generation

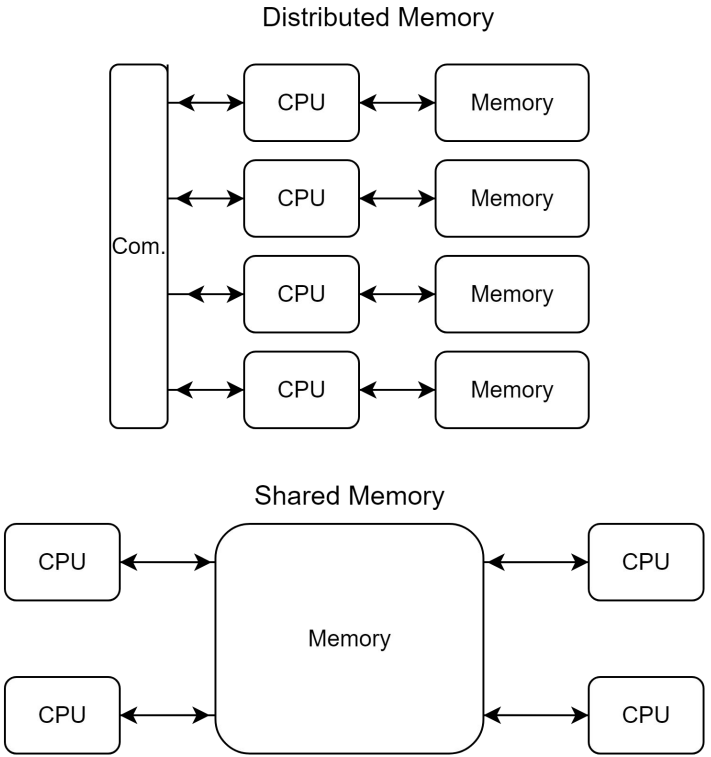
Generation	Priority
1	687
2	687
3	692
4	694
5	695

6	701
7	701
8	701
9	701



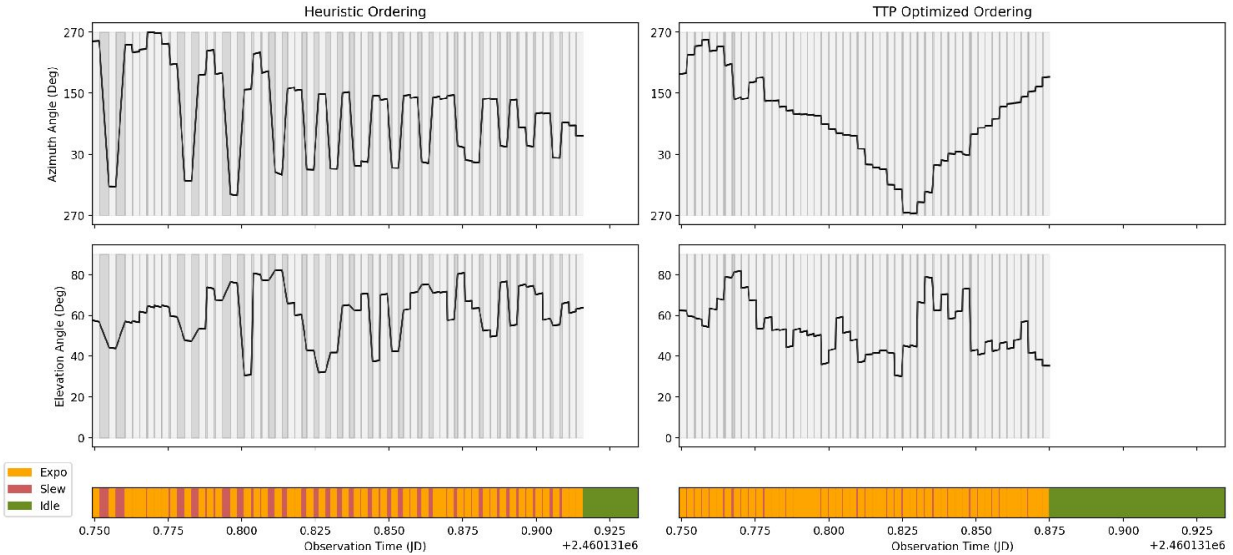
Distributed and Shared Memory

- Distributed Memory
  - Each CPU has its own memory
  - Sharing information between CPUs is more difficult
  - Typically faster access times
  - Common for clusters and supercomputers
- Shared Memory
  - CPUs share memory
  - Requires use of locks, sephehouras or other management techniques
  - Easier to code
  - Common on standard PC hardware



TTP | Linear Programming

- Schedule 100 observations in 10 mins
- Reduction in slew overhead of 5x compared to random ordering

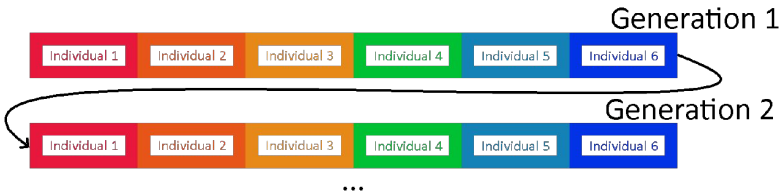


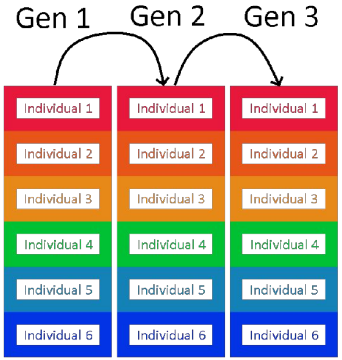
Schedule Analysis [13]

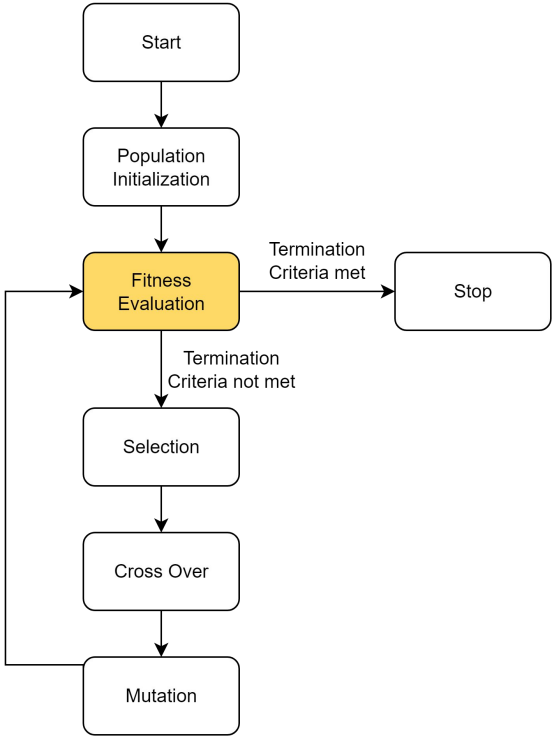
Unpublished work available at: https://arxiv.org/abs/2310.18497

CUDA Application

- Using a GPU (graphical processing unit)
  - Built for parallelization
  - Artificial Intelligence applications
- Calculating Fitness for each individual for every generation takes a long time
- Using CUDA, we can calculate the fitnesses for *all* the individuals in a generation **simultaneously!**
  - And perform the rest of the process with multiprocessing on a CPU as usual
- Expect higher scores as population sizes increase

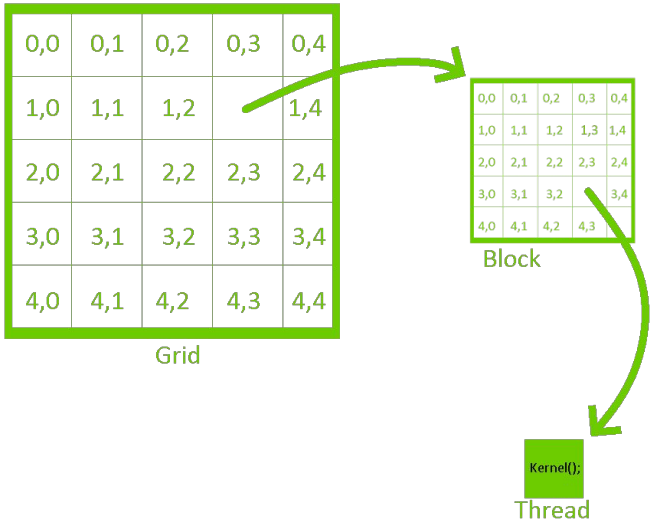






*https://developer.nvidia.com/cuda-toolkit

CUDA Architecture



- A GPU is the Graphical Processing Unit of a computer
  - Since a GPU's job is exclusively to render high-quality images, they are built to focus on concurrency with a high number of cores
- **Kernel:** A function to be executed on the GPU
- **Thread:** A single GPU thread - can execute a single task
- **Block:** Comprised of multiple threads - can execute all threads in a block in parallel
- **Grid**: Comprised of blocks - we only use one grid

The GPU used was an **NVIDIA GeForce RTX 3070 Ti**