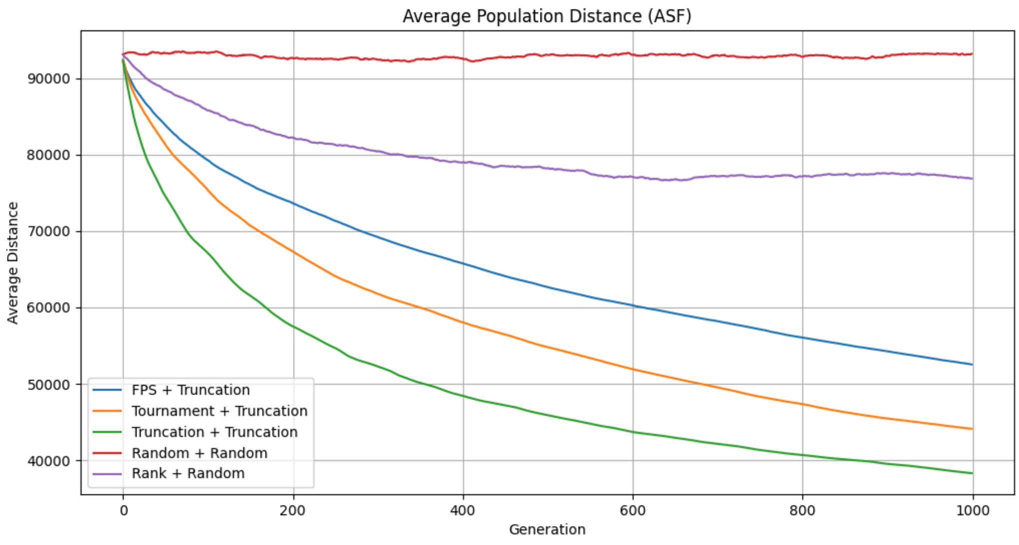
Computational Intelligence Assignment 1

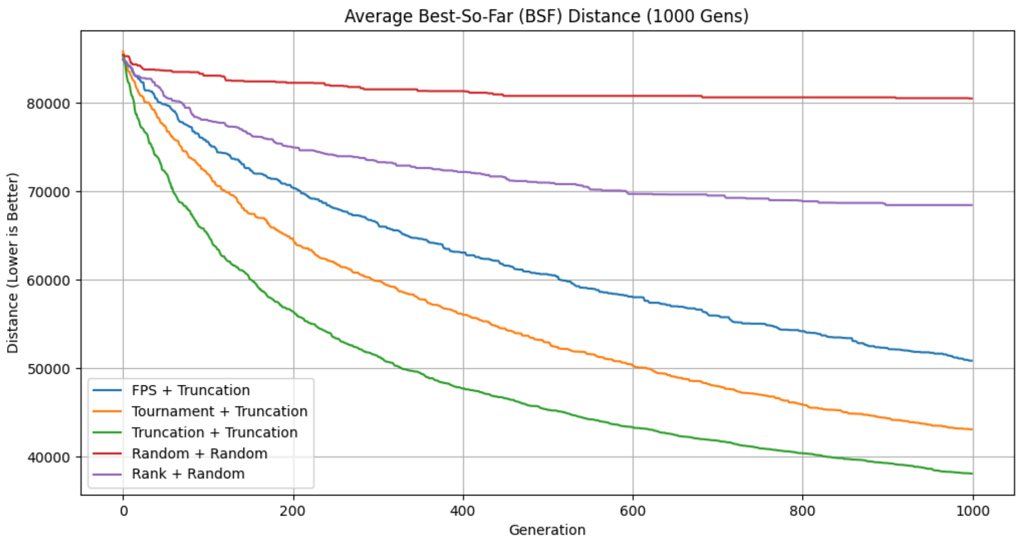
Nahyan Jawed
Muhammad Hasan Nizami

Code at: https://github.com/mn08065/Computational-Intelligence-Assignment-1.git

Q1:

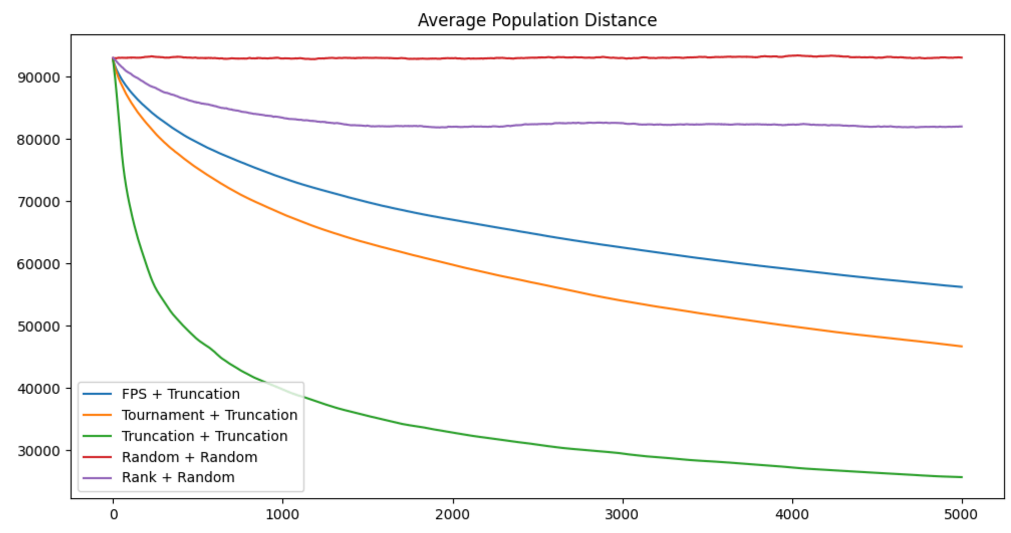
Analysis:

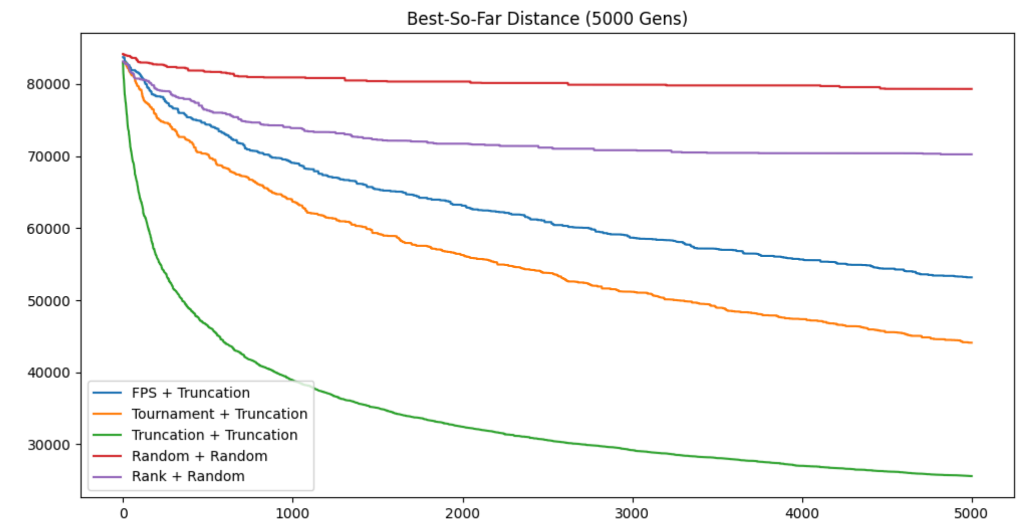




The experimental results demonstrate a clear hierarchy of effectiveness among the tested selection strategies for the TSP (Qatar 194) dataset, strongly favoring elitism and scalability. In the initial low-resource baseline (Population 30, 50 Generations), Tournament + Truncation proved most effective, achieving a best distance of 69,248, significantly outperforming the random baseline of 84,501. However, as computational resources scaled to Population 100 and 500 Generations, Truncation + Truncation emerged as the dominant strategy, improving from 71,772 in the first run to 45,294, a 37% improvement that highlights its ability to aggressively exploit larger populations. This trend continued in the high-resource Run 7, where Truncation + Truncation achieved 25,601, massively outperforming non-elitist strategies like Rank + Random (70,230) which failed to converge meaningfully. The final optimization phase, which introduced elite seeding, saw FPS + Truncation overtake the greedy strategies to achieve the global best distance of 10,683, representing an 86% total improvement over the initial baseline and confirming that while greedy selection drives early convergence, fitness-proportional selection combined with strong elitism (Truncation survivor selection) is superior for fine-tuning solutions in the later stages of optimization.

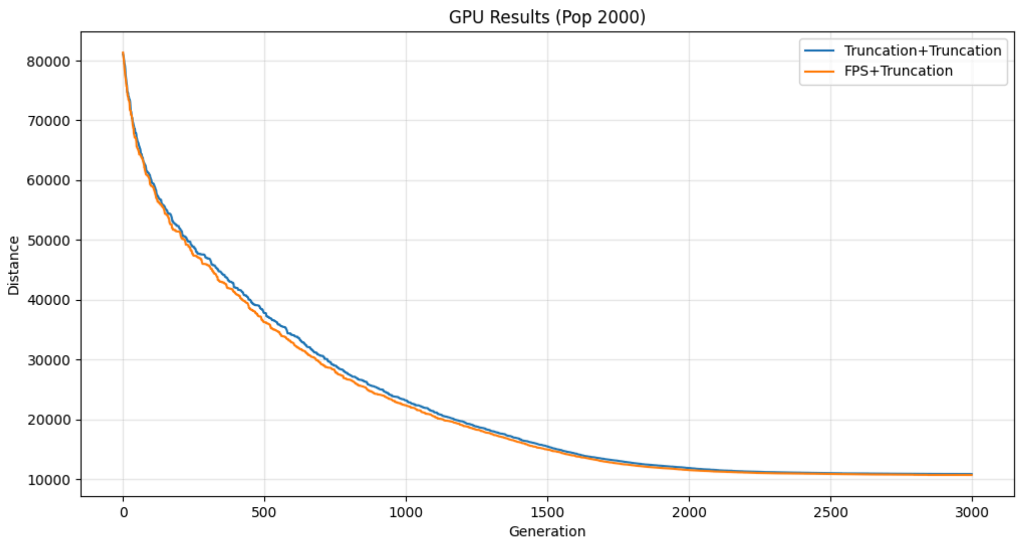
Run 7:





The experiments clearly demonstrated that Survivor Selection and Population Scaling were the decisive factors for success in the TSP. Strategies using Random survivor selection failed to converge, remaining stuck near 80,000 km, whereas those using Truncation (keeping the elite individuals) consistently improved. As we increased the population size and generations, the optimal parent strategy shifted from Tournament Selection (best for quick starts, 69,000 km) to Truncation Selection (best for scaling, reaching 25,601 km). However, the massive final leap to 10,683 km an 86% total improvement was achieved not just by parameter tuning, but by Seeding the population with previous best results, proving that hybridizing evolution with "memory" is far more effective than starting from scratch.

Run 11:



The experimental results demonstrate that Survivor Selection and Population Scaling were the critical factors for success in solving the TSP. Strategies utilizing Random survivor selection failed to converge, remaining stuck near 80,000 km, whereas those employing Truncation (keeping the elite individuals) consistently improved performance. As the population size and number of generations increased, the optimal parent strategy shifted from Tournament Selection (best for quick starts in small populations, reaching ~69,000 km) to Truncation Selection (best for scaling in larger populations, reaching 25,601 km). However, the most significant performance leap an overall 86% improvement to 10,683 km was achieved in the final phase by Seeding the population with previous elite results, confirming that hybridizing evolution with a "memory" of past success is far more effective than starting from scratch.

Q2:
There were two approaches employed for this question by the team.

The main difference was the fitness function, in the first approach simply the number of student conflicts were observed as an evaluation criteria.

Tournament+Tournament	Conflicts:	1600
Tournament+Tournament	Conflicts:	1575
Tournament+Tournament	Conflicts:	1631
Tournament+Tournament	Conflicts:	1618
Tournament+Tournament	Conflicts:	1552
Tournament+Truncation	Conflicts:	1511
Tournament+Truncation	Conflicts:	1456
Tournament+Truncation	Conflicts:	1530
Tournament+Truncation	Conflicts:	1522
Tournament+Truncation	Conflicts:	1553

The best approach that lead to 1456 conflicts was Tournament for parent selection and truncation for survival strategy.

The second approach involved a much more complex fitness function. Each constraint was assigned a weight higher weights for hard constraints and lower weights for soft constraints.

1. Decision Variables

- **Exams (E):** A set of **2,167** exams that need to be scheduled.
- **Periods (P):** A set of **29** available time slots.
- **Assignment (X_i):** For every exam i, we must assign a period j

2. Objective Function

The goal is to **minimize** the total penalty score (Fitness). Our fitness function focuses on student centric optimization by penalizing every instance where a student is scheduled for two exams at once.

$$Fitness = \sum_{p=1}^{29} \sum_{i,j \in E_p, i<j} (\text{SharedStudents}_{i,j} \times W_{direct})$$

1. **E_p:** The set of exams assigned to period p.
2. **SharedStudents[i,j]:** The count of students enrolled in both exam i and exam j
3. **W_direct:** The penalty weight, set to **1,000** (as per the "Production" dataset parameters).

3. Constraints

- **Hard Constraint (Conflict Avoidance):** Ideally, no student should have two exams in the same period. We enforce this through the high penalty weight ($W_{direct} = 1000$).
- **Capacity Constraint (Implicit):** Exams must be distributed across periods such that the total student count in any given period does not exceed the university's total room capacity.
- **Domain Constraint:** Every exam must be assigned exactly **one** period and **one** or more rooms.

EA Process

To implement the Evolutionary Algorithm for the Purdue Exam Timetabling problem, we represented each potential solution as a fixed-length integer chromosome of size 2,167. In this encoding, each index in the array corresponds to a unique Exam ID, and the integer value at that index (ranging from 1 to 29) represents the specific time period assigned to that exam.

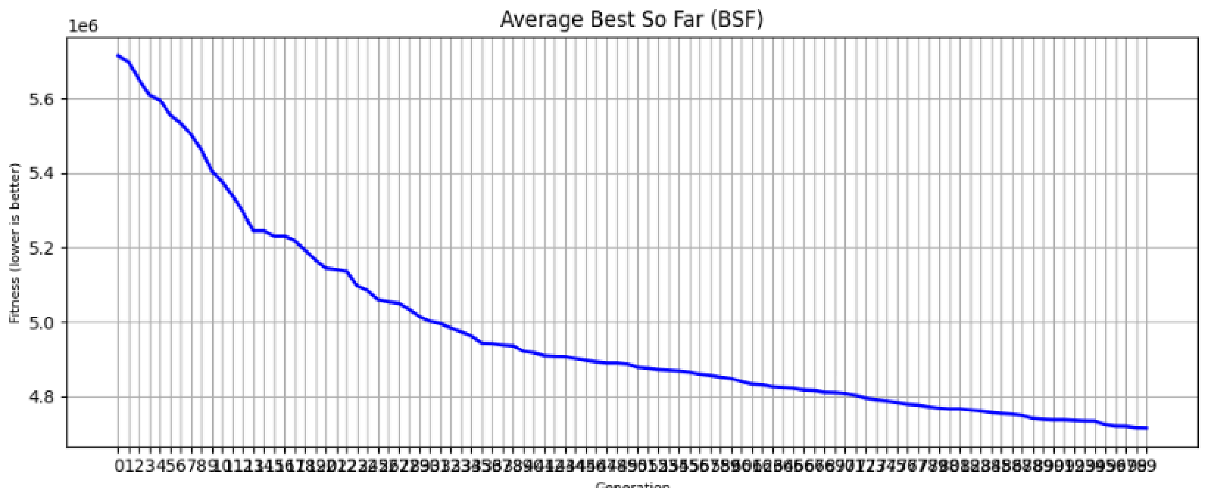
To produce new candidate solutions, we utilized One-Point Crossover, a genetic operator where a random "split point" is chosen along the length of two parent chromosomes. The offspring is created by taking all period assignments before the split point from the first parent and all assignments after the point from the second parent, allowing the algorithm to combine successful clusters of exam timings.

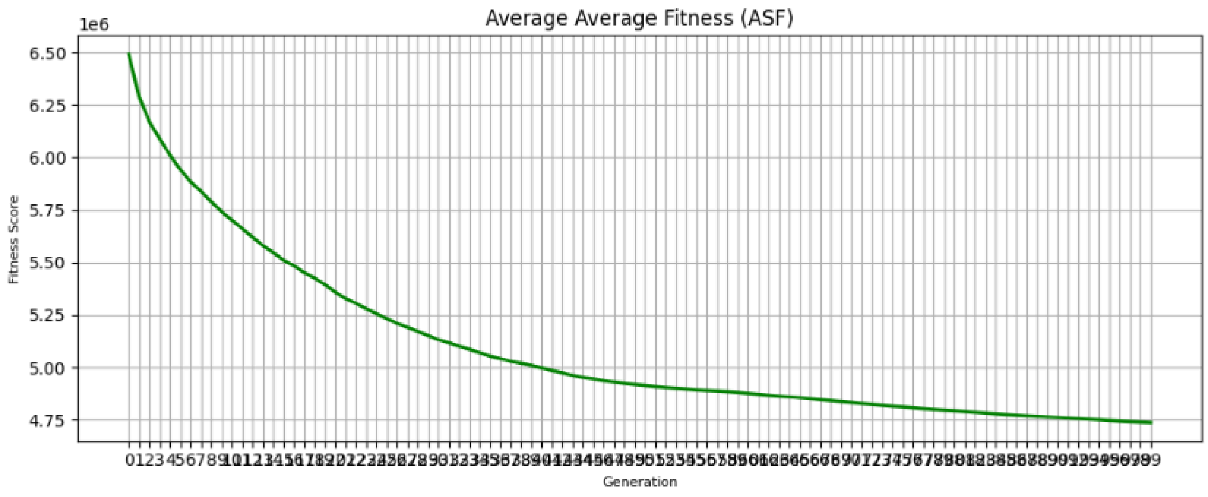
To maintain genetic diversity and prevent the population from converging too quickly on a sub-optimal "local trap," we applied a Random Reset Mutation. With a set probability (the mutation rate), a specific gene (exam) is selected at random and its value is changed to a different period, effectively "testing" a new slot for that exam.

Strategy Comparison BSF ASF

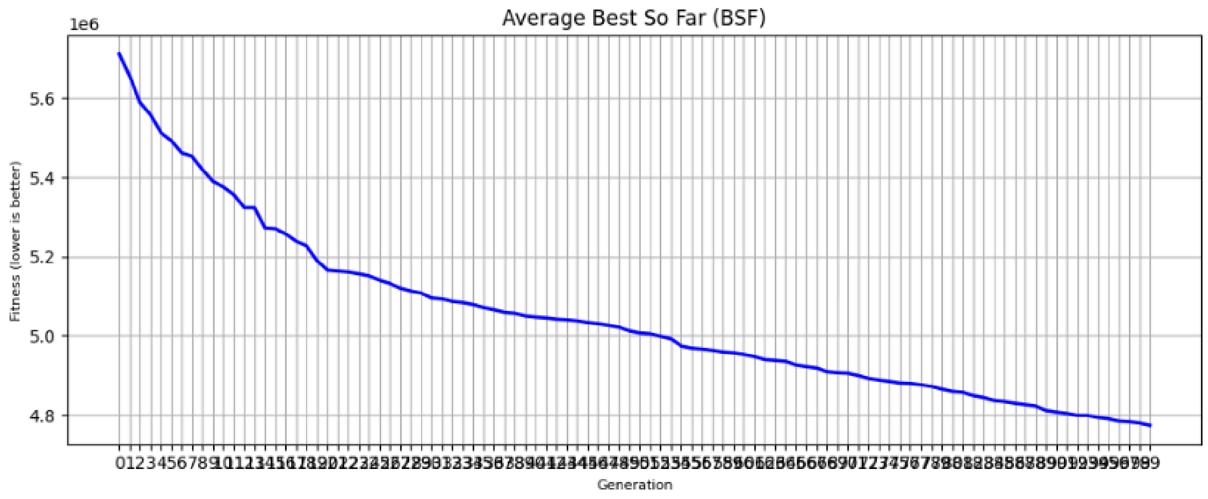
(The following plots are averages of 10 runs of 100 generations on each combination of strategies)

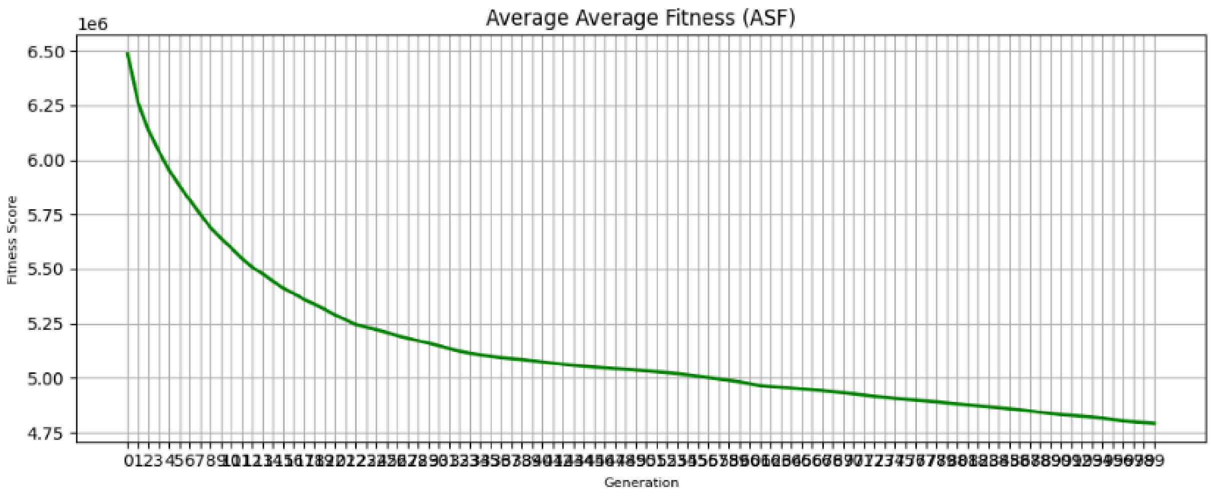
Combination:
FPS & Truncation



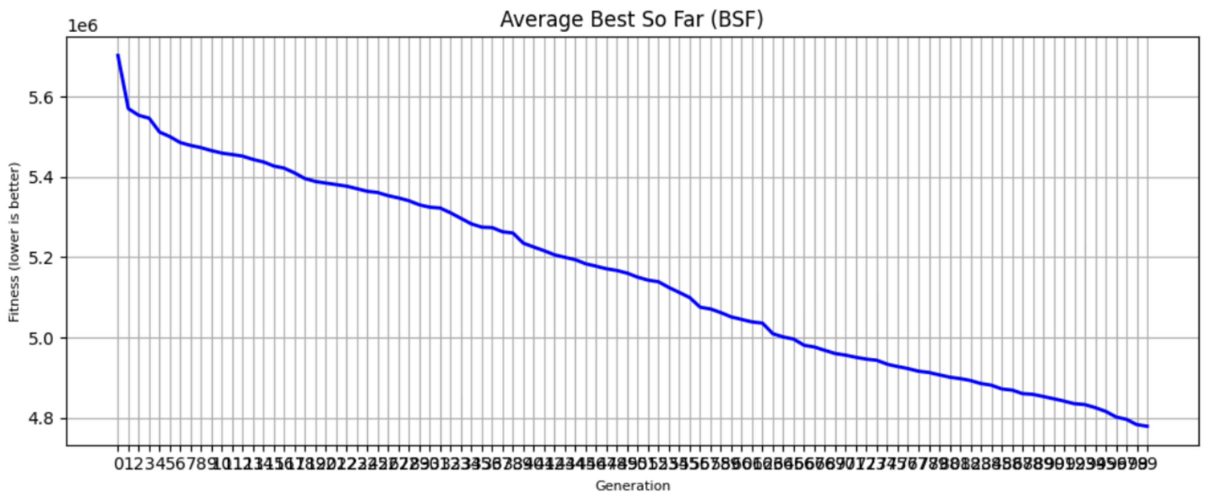


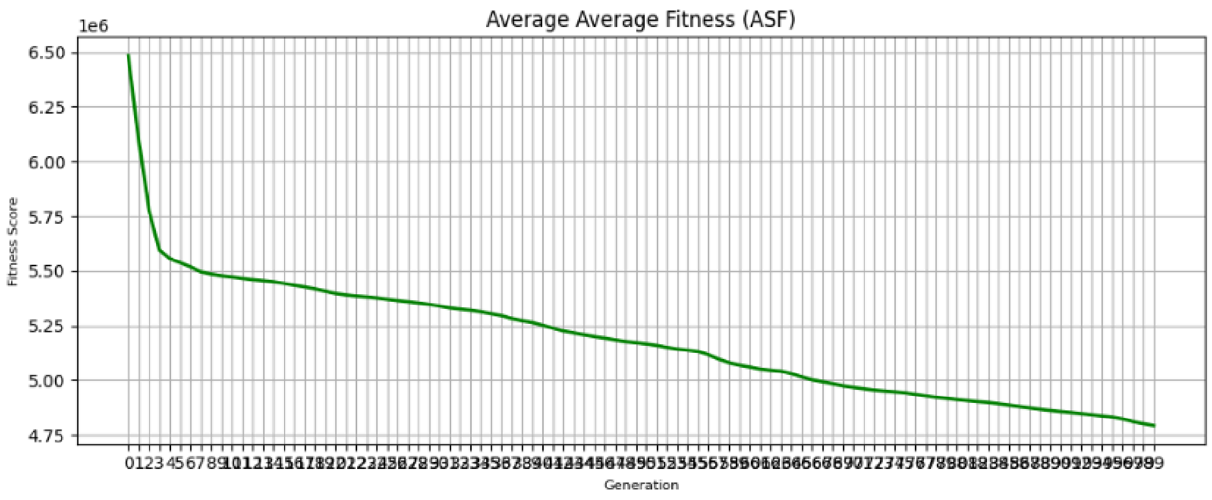
Combination:
Binary Tournament & Truncation



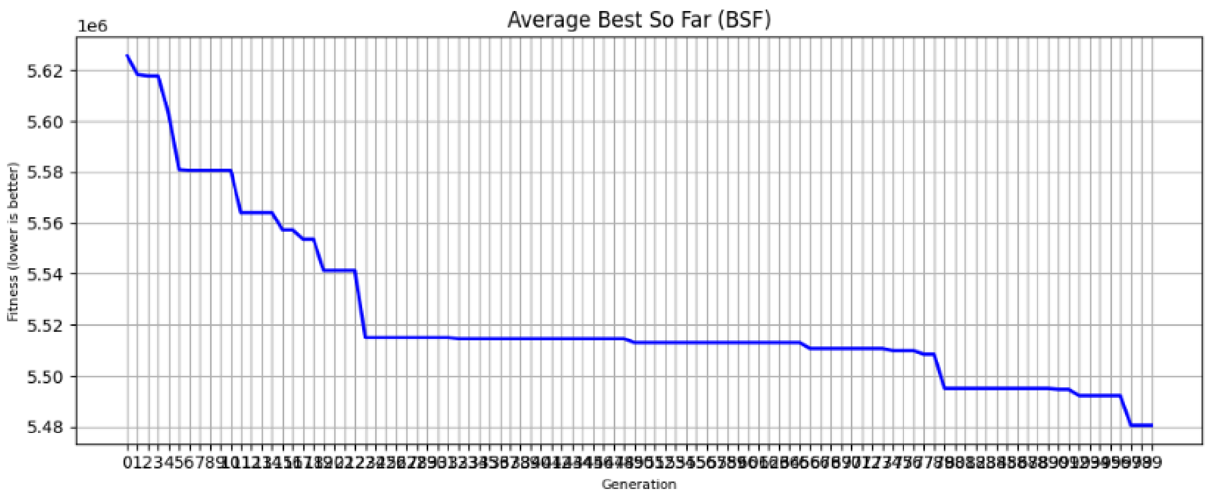


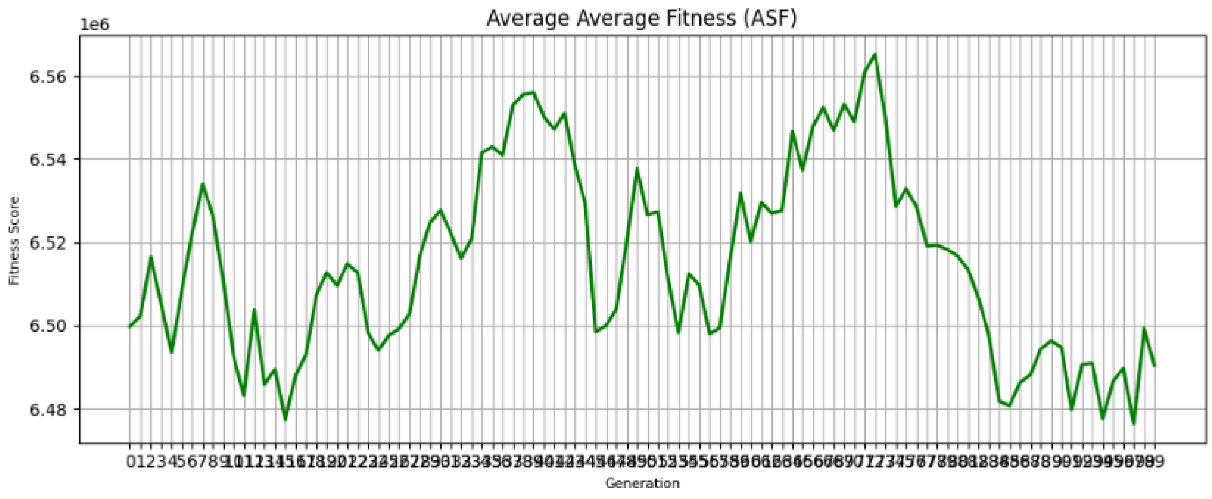
Combination:
Truncation & Truncation



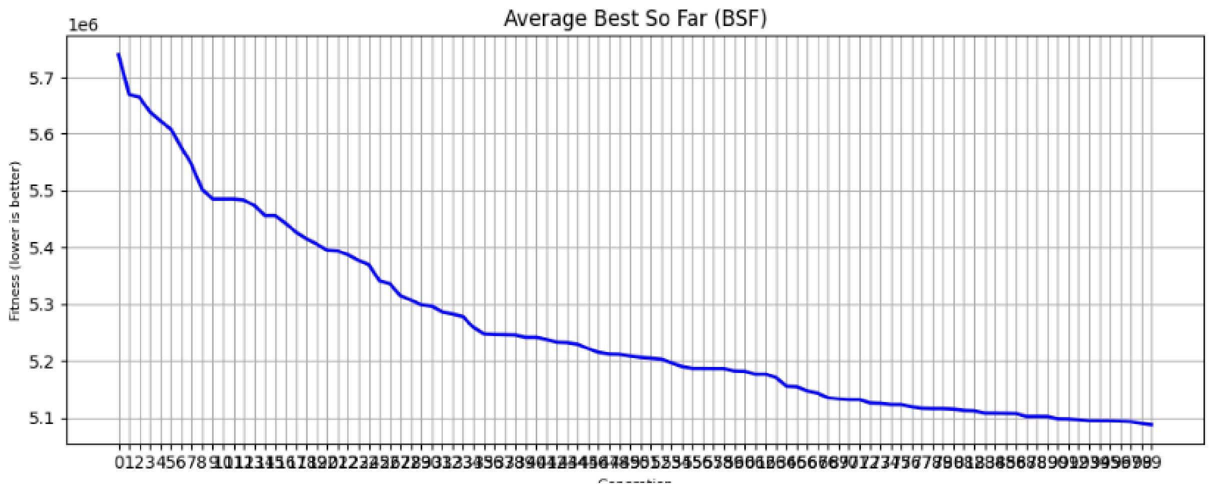


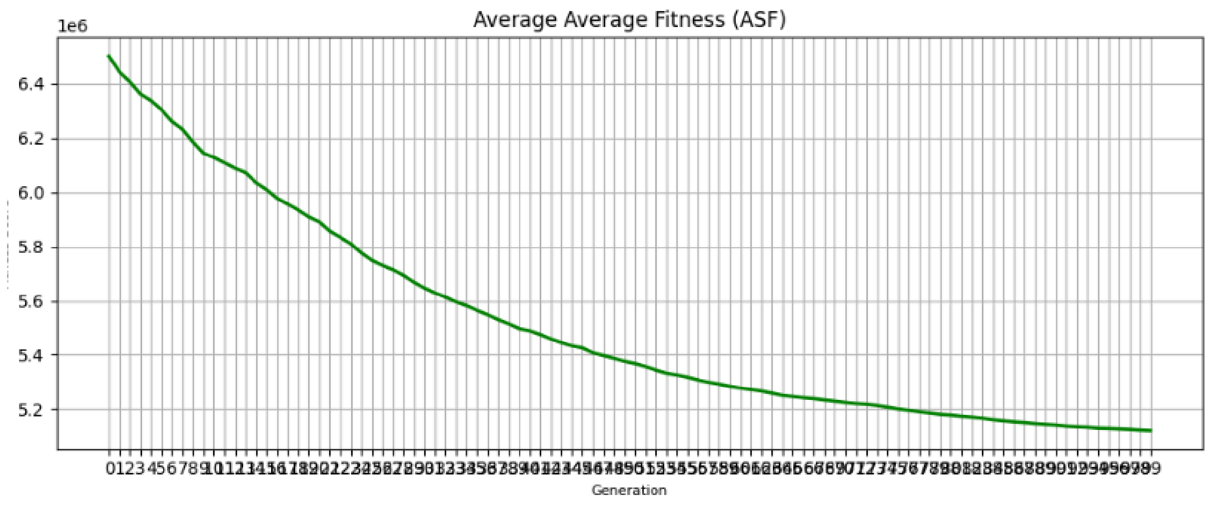
Combination:
Random & Random

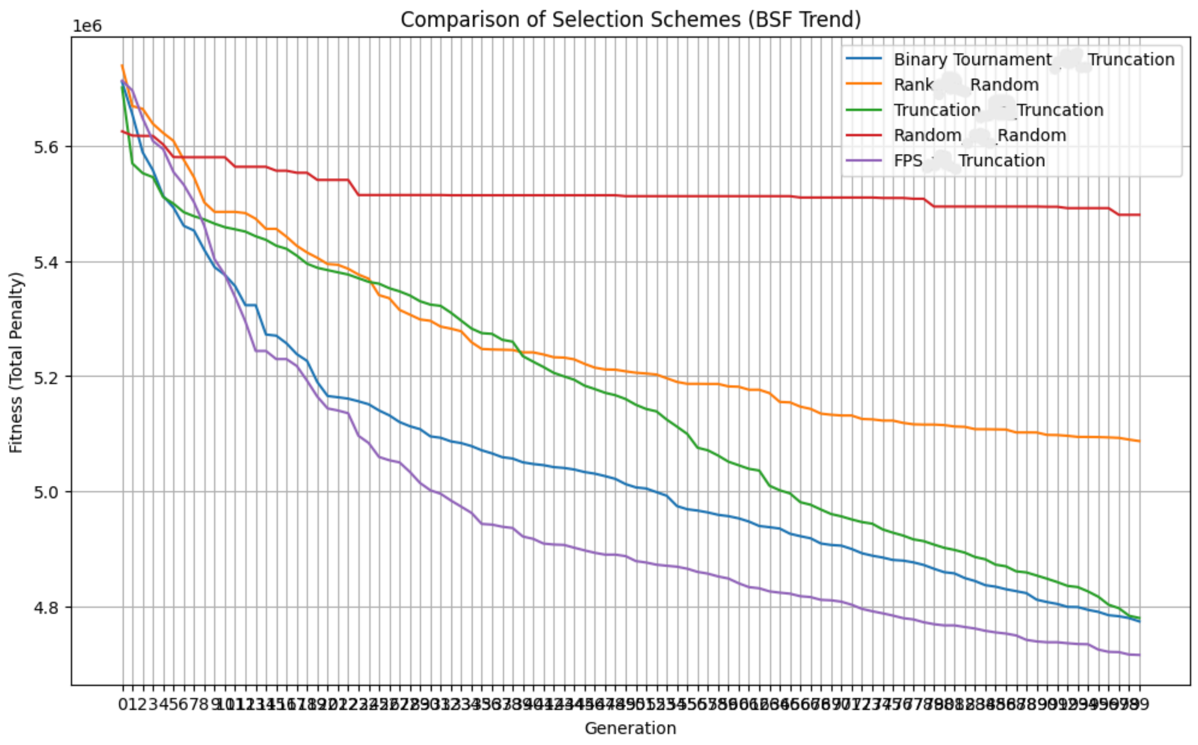




Combination:
Rank & Random







Scheme	Initial BSF (Avg)	Final BSF (Avg)	Improvement %	Consistency (Std)
FPS,Truncation	5713800	4715400	17.473485	115147.4417
Binary Tournament,Truncation	5711400	4773600	16.419792	127139.6433
Truncation,Truncation	5702000	4779600	16.17678	124699.8173
Rank,Random	5740000	5087200	11.372822	191521.8583
Random,Random	5625700	5480600	2.579235	124946.3885

THE WINNING STRATEGY: FPS,Truncation

The success of the **FPS (Fitness Proportional Selection) with Truncation Survival** strategy boils down to a balance between exploration and exploitation. In a massive search space like the Purdue dataset (with 29^2167 possible configurations), "greedier" methods like Tournament selection often fail because they cause the population to converge too quickly on the first "decent" schedule it finds, effectively trapping the algorithm in a local optimum.

By using **FPS**, you allow a "soft" selection pressure where even sub-optimal schedules have a chance to pass on their genes; this maintains the **genetic diversity** needed to keep searching new areas of the timetable. However, once those diverse parents produce a high-quality offspring, **Truncation Survival** acts as a strict "quality filter," immediately discarding the weakest individuals and locking in the best-found improvements. Essentially, FPS acted as the curious explorer finding new possibilities, while Truncation acted as the ruthless manager ensuring only the top tier results made the cut.

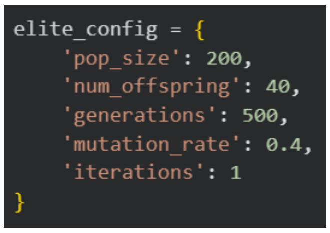
Fine tuning the Winner

To further optimize the results, the mutation strategy was refined rather than simply increasing the computational runtime. A Swap Mutation was introduced to replace the standard random reset approach; this operator selects two exams and interchanges their assigned periods, thereby preserving the relative structural integrity of the rest of the schedule while resolving specific local conflicts.

Additionally, Elitism was implemented to ensure that the highest-performing solution discovered in any generation was automatically carried over to the next. This modification prevented the accidental loss of optimal configurations during the selection and survival phases, ensuring a monotonic improvement in fitness over time.

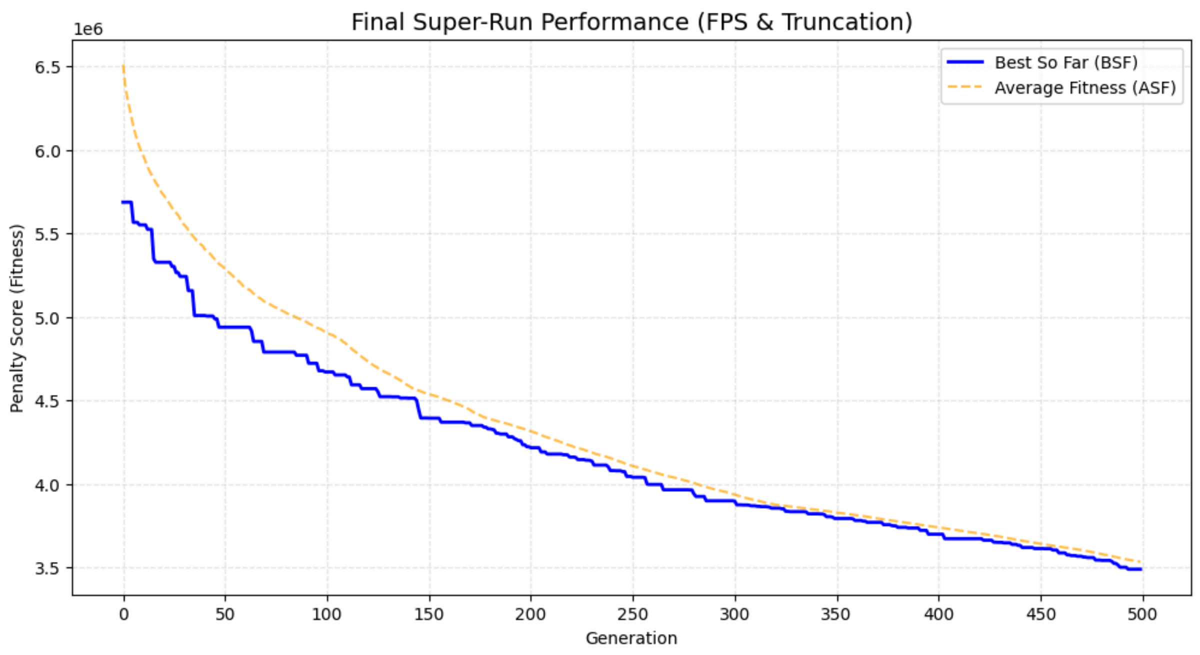
We also ran it for larger numbers to better understand and let evolution work its magic.

Configuration:



```
elite_config = {
    'pop_size': 200,
    'num_offspring': 40,
    'generations': 500,
    'mutation_rate': 0.4,
    'iterations': 1
}
```

Results:



Generations recorded: 500

Starting Fitness: 5,687,000

Ending Fitness: 3,488,000

Total Drop: 2,199,000