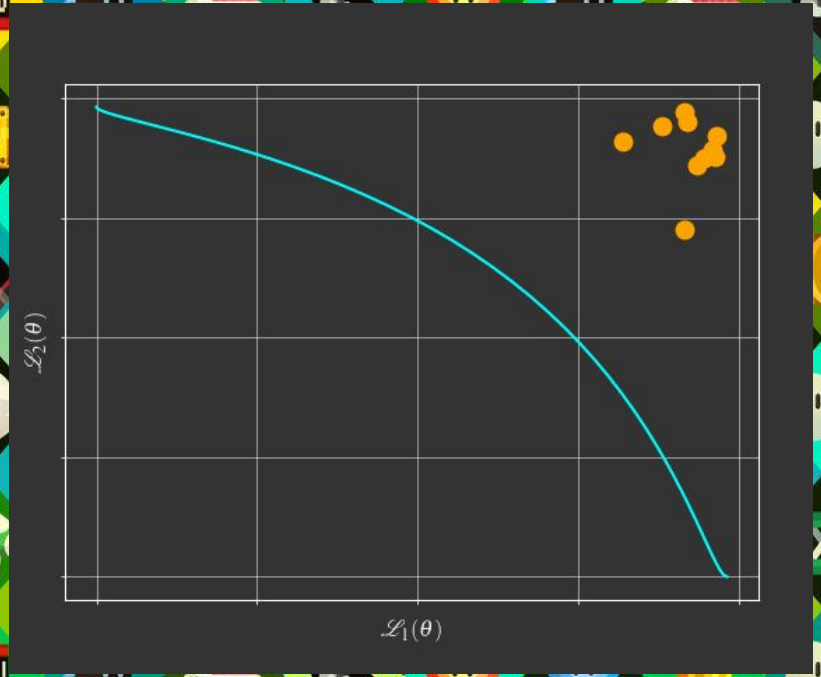
CSCI 165 Bio-Inspired Machine Learning

# Multi-Objective Optimization (MOO) Methods for Engineering

A Survey by R.T. Marler and J.S. Arora


**Brian Kurzeja**
**Nailea Ibarra**
**Miranda Hernandez**
**Sarida Ngo**

**Dr. Thanos Panagopoulos**

# Introduction to Multi-Objective Optimization (MOO)

**Definition:** Finding the best trade-offs between conflicting objectives under constraints. This leads to a **Pareto Optimal Set.**

## Foundational Uses of MOO:

- Economic Equilibrium
- Welfare Theories
- Game Theory
- Pure Mathematics

# Types of Pareto Optimality:

1. **Pareto Optimal**
2. **Weakly Pareto Optimal**
3. **Properly Pareto Optimal:**

$\mathbf{X} =$ Feasible design space
$x =$ Vector of design variables (point in the design space)
$x^* =$ Pareto Optimal Point
$\mathbf{F} =$ Vector of objective functions
$F_i =$ Objective function

**Definition 1.** *Pareto Optimal: A point,* $\mathbf{x}^* \in \mathbf{X}$, *is Pareto optimal iff there does not exist another point,* $\mathbf{x} \in \mathbf{X}$, *such that* $\mathbf{F}(\mathbf{x}) \leq \mathbf{F}(\mathbf{x}^*)$, *and* $F_i(\mathbf{x}) < F_i(\mathbf{x}^*)$ *for at least one function.*

**Definition 2.** *Weakly Pareto Optimal: A point,* $\mathbf{x}^* \in \mathbf{X}$, *is weakly Pareto optimal iff there does not exist another point,* $\mathbf{x} \in \mathbf{X}$, *such that* $\mathbf{F}(\mathbf{x}) < \mathbf{F}(\mathbf{x}^*)$.

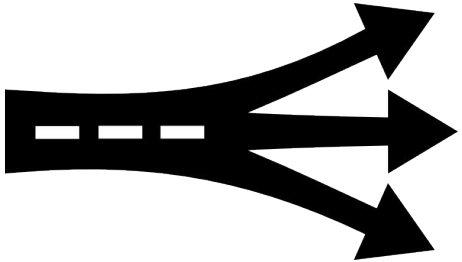
**Definition 3.** *Properly Pareto Optimal: A point,* $\mathbf{x}^* \in \mathbf{X}$, *is properly Pareto optimal (in the sense of Geoffrion) if it is Pareto optimal and there is some real number* $M > 0$ *such that for each* $F_i(\mathbf{x})$ *and each* $\mathbf{x} \in \mathbf{X}$ *satisfying* $F_i(\mathbf{x}) < F_i(\mathbf{x}^*)$, *there exists at least one* $F_j(\mathbf{x})$ *such that* $F_j(\mathbf{x}^*) < F_j(\mathbf{x})$ *and* $\frac{F_i(\mathbf{x}^*) - F_i(\mathbf{x})}{F_j(\mathbf{x}) - F_j(\mathbf{x}^*)} \leq M$. *If a Pareto optimal point is not proper, it is improper.*

# Categories of MOO Methods

## 1. Priori articulation of preferences

**Decide first, optimize later**

Preferences are set before running the optimization.



## 2. Posteriori articulation of preferences

**Explore, then decide**

Optimization finds multiple solutions first, then the user picks the best one.

## 3. No articulation of preferences

**Let the algorithm decide**

The algorithm automatically balances the objectives without human input.

# 1. Methods of Priori Articulation of Preferences

1. **Weighted Global Criterion:**
   - All objective functions combined into single function
   - Weights are assigned to scale features

$$U = \left\{ \sum_{i=1}^{k} w_i [F_i(\mathbf{x}) - F_i^{\circ}]^p \right\}^{\frac{1}{p}}$$

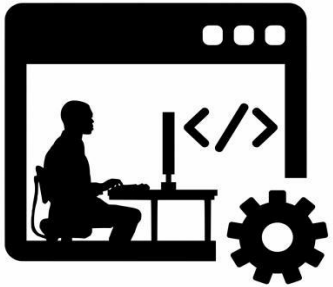
2. **Weighted Sum Method:**
   - Most common approach, requires deciding on weights ahead of time.

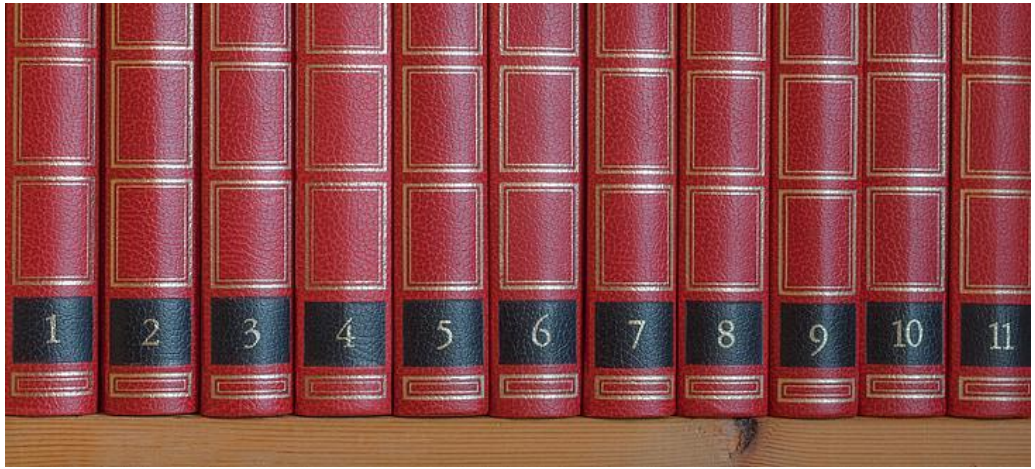
$$U = \sum_{i=1}^{k} w_i F_i(\mathbf{x})$$

3. **Lexicographic**

4. **Physical Programming**

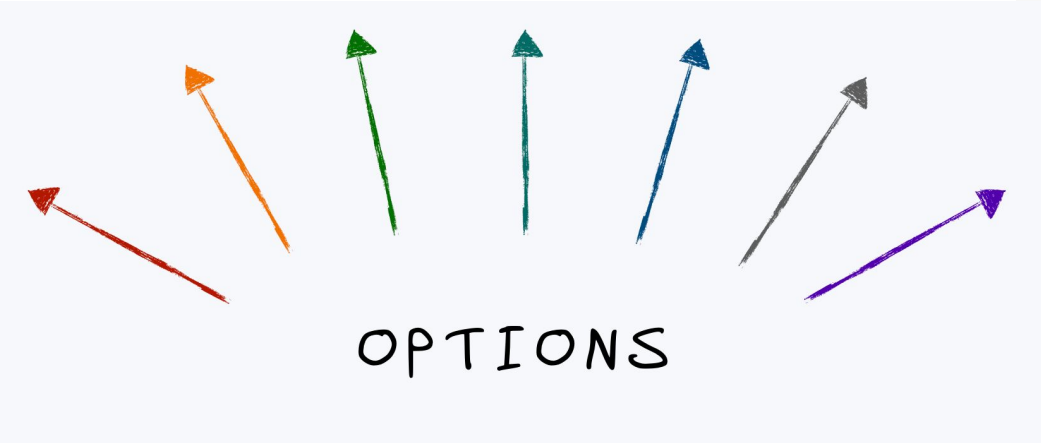
5. **And more…**



# 2. Methods of Posteriori Articulation of Preferences

Optimization finds multiple solutions first, then the user picks the best one.




**Comparison Table**

| | Product A | Product B | Product C |
|---|---|---|---|
| Factor 1 | ✓ | ✗ | ✗ |
| Factor 2 | — | — | ✓ |
| Factor 3 | — | ✓ | — |

NNGROUP.COM NN/g

1. Physical Programming
2. Normal Boundary Intersection (NBI) Method
3. Normal Constraint (NC) Method

# 3. Methods with No Articulation of Preferences

**Does not specify which objectives are more important before optimizing.**

**min-max method**: minimizes the worst-performing objective to find a balance between objectives

🚫 prevents large bias to one objective (ensuring no single stat is too weak)

⚖️ more balanced optimization

$$\min \max_i \left( \frac{f_i(x) - f_i^*}{f_i^{\max} - f_i^{\min}} \right)$$

# 3. Methods with No Articulation of Preferences

**Genetic algorithms (GAs):** An optimization method inspired by natural selection and genetics used to find optimal solutions to complex problems

Advantages:

- Explores *all* possible trade-offs between multiple objectives
- Balance between *exploration* and *exploitation* to maintain diverse and refined solutions

# Genetic Algorithms (GAs) for Multi-Objective Problems

Mario Kart application:

- Finds all best setups among <u>different play-styles</u> depending on preference for speed, acceleration, etc.
- Can dynamically adapt to different tracks

| | | Survey Section | Scalar Method | Possible Pareto Opt. | Necessary for Pareto Opt. | Sufficient for Pareto Opt. | Utopia Point | PC (0 to 5) | SUC (0 to 5) | CC (0 to 5) | PAP (0 to 5) |
|---|---|---|---|---|---|---|---|---|---|---|---|
| A Priori Articulation of Preferences | Wtd. Global Critn. I | 3.1 | x | | F.W. | x | x | 0 | 1 | 1 | 2 |
| | Wtd. Global Critn. II | 3.1 | x | | x* | x | x | 0 | 1 | 1 | 2 |
| | Weighted Sum | 3.2 | x | | | x | | 0 | 1 | 0 | 1 |
| | Lexicographic | 3.3 | | | | x | | 2 | 1 | 2 | 1 |
| | Weighted Min-Max | 3.4 | | | x | x-weak | x | 1 | 1 | 2 | 1 |
| | Exponential Weighted | 3.5 | x | | x | x | | 0 | 1 | 0 | 1 |
| | Weighted Product | 3.6 | x | | F.W. | x | | 0 | 1 | 1 | 1 |
| | Goal Programming | 3.7 | | x | | | | 1 | 1 | 2 | 1 |
| | Bounded Obj. Fnc. | 3.8 | | x | | | | 1 | 1 | 1 | 1 |
| | Physical Programming | 3.9 | | | x | x | x | 3 | 3 | 1 | 4 |
| A Posteriori Articulation | Physical Programming | 4.1 | | | x | x | x | 4 | 1 | – | 5* |
| | NBI | 4.2 | | | | x | x | 2 | 1 | – | 5* |
| | Normal Constraint | 4.3 | | | x | x | x | 2 | 1 | – | 5* |
| | Genetic | 6.2 | | x | | | | 5 | 1 | – | 5* |
| No Articulation of Preferences | Global Criterion | 5.1 | | | | | | | | | |
| | Achievement Fnc. | 5.1 | x | | x | x | | 0 | 1 | 0 | 0 |
| | Compromise Fnc. | 5.1 | x | | | x | x | 0 | 1 | 1 | 0 |
| | Objective Sum | 5.1 | x | | | x | | 0 | 0 | 0 | 0 |
| | Min-Max | 5.1 | | x | | x-weak | x | 1 | 0 | 2 | 0 |
| | Nash Arbitration | 5.2 | x | | F.W. | x | | 0 | 1 | 1 | 0 |
| | Objective Product | 5.2 | x | | | x | x | 0 | 0 | 1 | 0 |

# Limitations and Challenges

- Computational complexity in high-dimensional problems
- More objectives -> more possible solutions
- Choosing the right method depends on problem characteristics:
  a. A Priori Methods: When prioritized objectives are known
  b. A Posteriori Methods: When exploring complex-tradeoffs
  c. No Preference Methods: When trade-offs are unknown
- MOO provides multiple solutions, making optimization complex
- Using Pareto Optimization as a Posteriori method allows us to explore different trade-offs to choose the best solution

# Conclusion

- MOO provides solutions by optimizing multiple conflicting objectives at the same time

> “It is found that no single approach is superior.
> Rather, the selection of a specific method depends on
> the type of information that is provided in the problem,
> the user’s preferences, the solution requirements, and the
> availability of software.“

# MARIOKART

## Pareto-Optimal Ka[illegible]s for Different Track Types

# Mario Kart: Multiplayer

Race against 11 other players to win 1st place!

In competitive Mario Kart, we want to choose a combination of a **driver**, **kart**, **wheels**, and **glider** maximize our stats, such as speed, acceleration, etc.

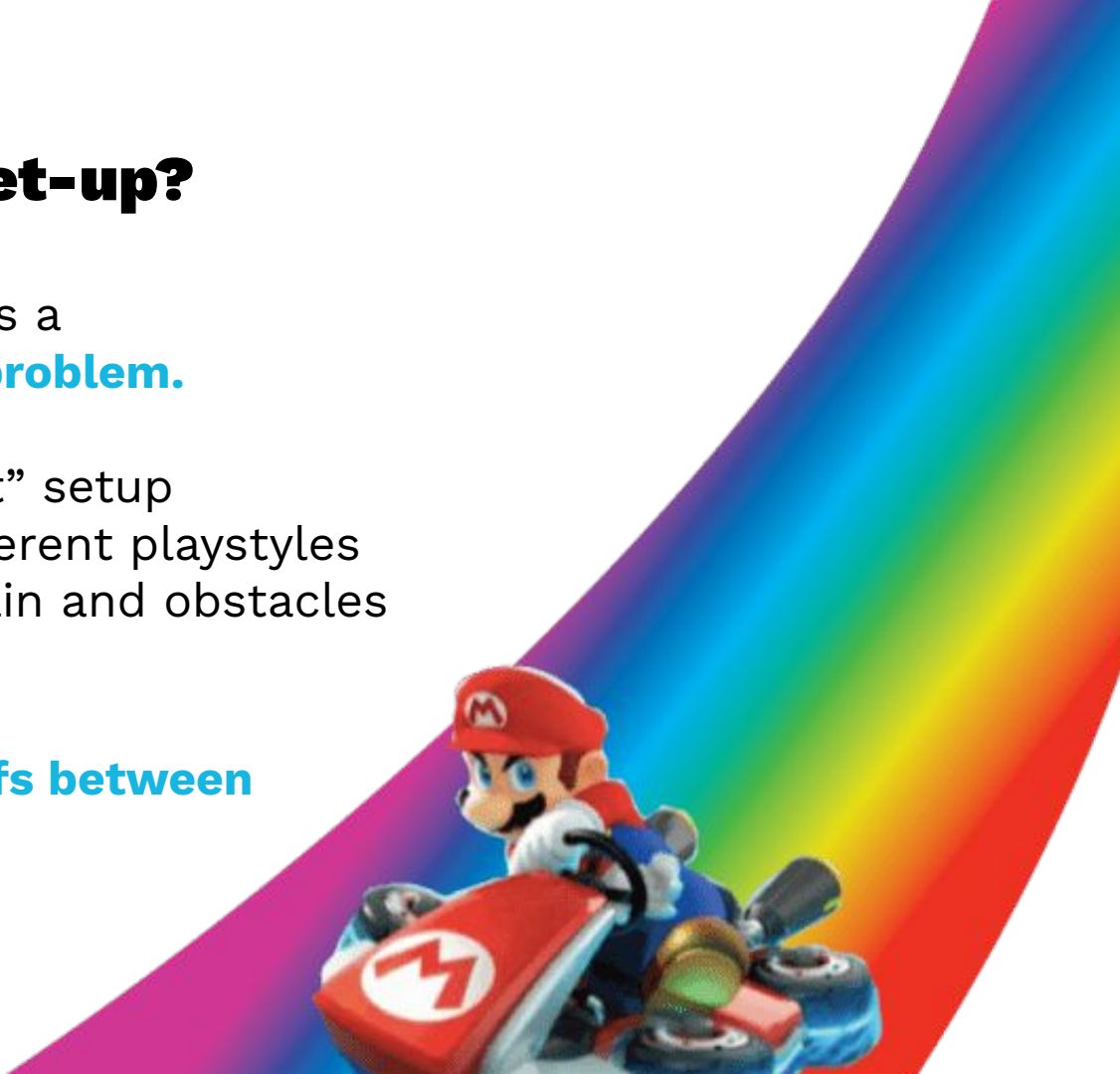
# Motivation:
## How to find the best set-up?

Choosing the best kart setup is a
**multi-objective optimization problem.**

- There is no universal “best” setup
- Different players have different playstyles
- Tracks have different terrain and obstacles

**Goal:**
**Find a balance in the trade-offs between different stats**

- ex: speed vs acceleration

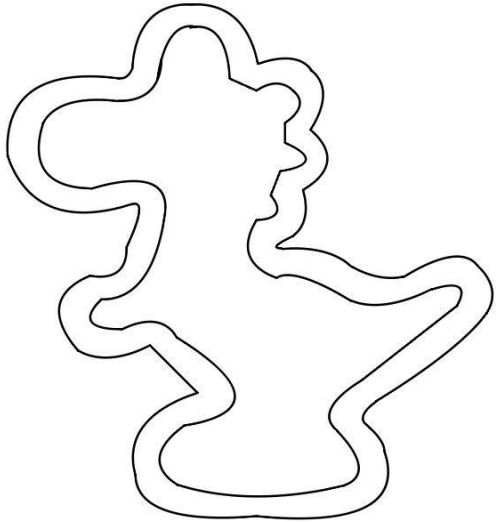


# Tracks

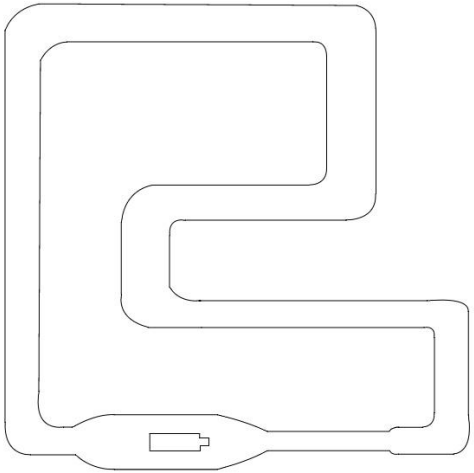
- Different track categories
- We consider 3 track types:
  - “curvy”: handling + miniturbo
  - long straights: speed
  - off-roading: acceleration + traction
- Different kart stats are ideal for maximizing performance in certain tracks
- Is it possible to find a track combo that is optimal for multiple tracks?

# Track Types

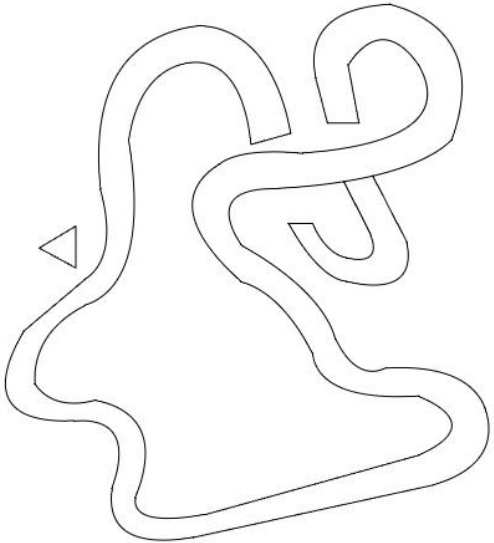
## Curvy: Yoshi Circuit



## Long Straights: Rainbow Road (SNES)



## Off-Roading: Cheese Land (GBA)



# Related Work

- MOO (Marler & Arora): Multi-Objective Optimization

  -Provide insight into method selection based on problem constraints, user preferences, and solution requirements

- Mario Meets Pareto (Mayerowitz): Kart Optimization

  -Applies pareto optimization to real kart combinations, illustrating tradeoffs between different kart stats

- NSGA II (Deb et. al) - Elitist Multi-Objective Genetic Algorithm

  -Introduces an efficient algorithm for generating diverse pareto optimal solutions

  -Keeps best solutions of each generation

# Approach

**Research Question:** How can we use **MOO** to determine the best Mario Kart setups based on different performance metrics?

Kart setups involve trade-offs (e.g. speed vs acceleration)

We use **NSGA-II** (Non-Dominated Sorting Genetic Algorithm II) to find **Pareto optimal** kart setups**.**

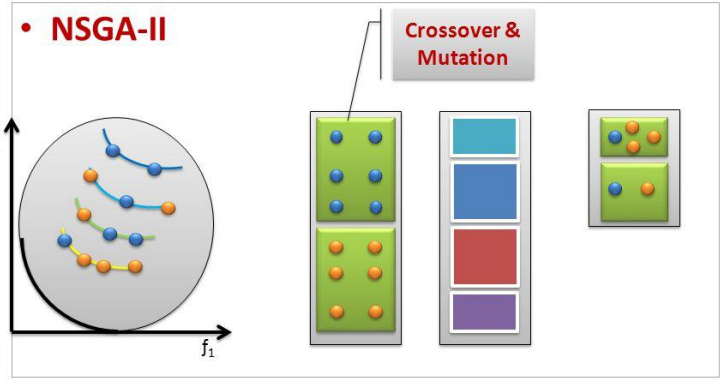
# NSGA-II

NSGA-II is an evolutionary algorithm designed for multi-objective optimization based on natural selection and evolution.

**How it works:**

1. Initialize population by creating a set of random setups
2. Evaluate each setup's performance
3. Select best setups
4. Perform crossover and mutation
5. Select survivors (best individuals) for the next generation
6. Repeat for X iterations

-> <u>Result</u>: Set of non-dominated optimal solutions (Pareto front)

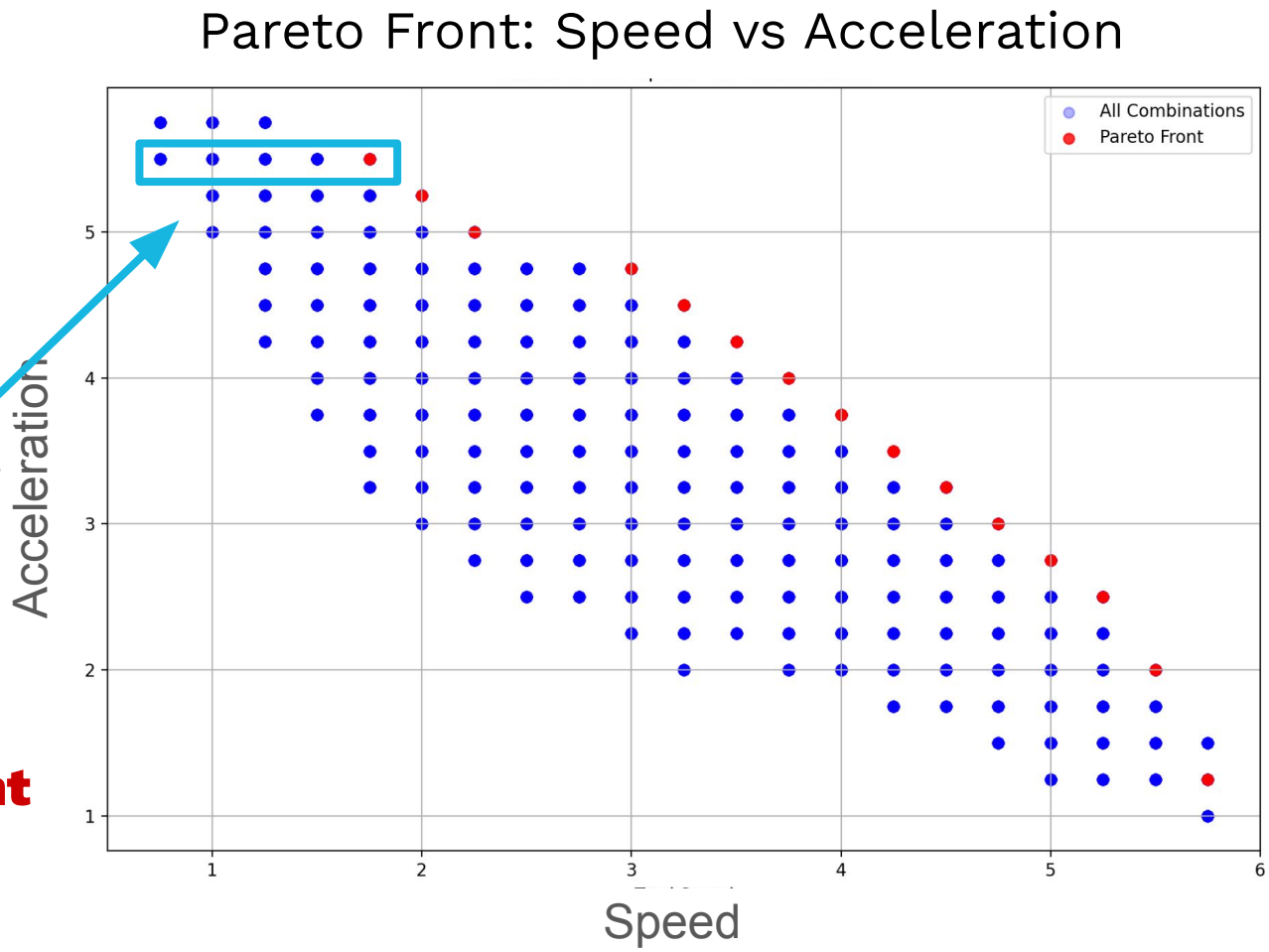


# Pareto Front

- 2 competing objectives
- Pareto front setups **dominate** other setups

For all possible setups at Acceleration = 5.5, there is one setup that has the highest speed

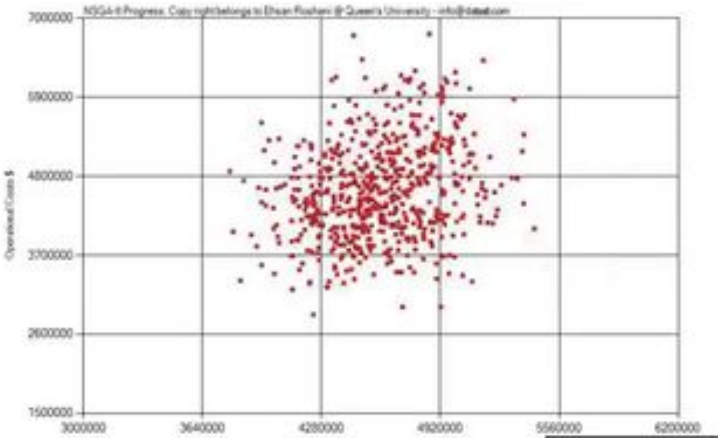
This is the **Pareto front**



# Why we chose NSGA-II

**Strengths:**

- Finds <u>multiple optimal setups</u>, allowing players to choose according to their playstyle
- <u>Non-dominated sorting:</u> Prioritizes setups that outperform others across multiple stats instead of just maximizing one
- <u>Diversity in solutions:</u> Prevents getting stuck in local optima



# Experiment Set-up: Data Collection

4 **Decision variables**:

We consider the following **objectives**:

- **Speed** (Ground, Water, Air, AntiGravity)
- **Handling** (Ground, Water, Air, AntiGravity)
- **Acceleration** (How quickly kart reaches top speed)
- **Traction** (How kart grips to track)
- **Mini-Turbo** (How strong the boost is after drifting)

# Running NSGA-II Algorithm

# Initialization

1. Create a multi-objective fitness class and an individual representation (list) that holds fitness values
2. Register functions to randomly select parts (driver, vehicle, tires, glider) by their index

```
# Setup NSGA-II
creator.create("FitnessMulti", base.Fitness, weights=(1.0,) * len(stat_cols))
creator.create("Individual", list, fitness=creator.FitnessMulti)

toolbox = base.Toolbox()

toolbox.register("driver_id", random.randint, 0, len(drivers) - 1)
toolbox.register("vehicle_id", random.randint, 0, len(vehicles) - 1)
toolbox.register("tire_id", random.randint, 0, len(tires) - 1)
toolbox.register("glider_id", random.randint, 0, len(gliders) - 1)

toolbox.register("individual", tools.initCycle, creator.Individual,
                 (toolbox.driver_id, toolbox.vehicle_id, toolbox.tire_id, toolbox.glider_id), n=1)

# Register the population
toolbox.register("population", tools.initRepeat, list, toolbox.individual)
```

# Crossover & Mutation

```
def evaluate(individual):
    return combine_stats(individual)

toolbox.register("evaluate", evaluate)
toolbox.register("mate", tools.cxTwoPoint)
def custom_mutate(individual, indpb):
    if random.random() < indpb:
        individual[0] = random.randint(0, len(drivers) - 1)
    if random.random() < indpb:
        individual[1] = random.randint(0, len(vehicles) - 1)
    if random.random() < indpb:
        individual[2] = random.randint(0, len(tires) - 1)
    if random.random() < indpb:
        individual[3] = random.randint(0, len(gliders) - 1)
    return individual,
toolbox.register("mutate", custom_mutate, indpb=0.2)
toolbox.register("select", tools.selNSGA2)
```

3. Crossover and mutation: Combines attributes from two parents and randomly alters a part if probability condition is met

# Running the Optimization Algorithm

```python
# Run NSGA-II optimization
pop = toolbox.population(n=100)
hof = tools.ParetoFront()
algorithms.eaMuPlusLambda(pop, toolbox, mu=100, lambda_=200, cxpb=0.6, mutpb=0.3, ngen=40, halloffame=hof, verbose=True)

# Convert to DataFrame
records = []
for ind in hof:
    d, v, t, g = ind
    stats = combine_stats(ind)
    record = {
        **dict(zip(stat_cols, stats)),
        "Driver": drivers.iloc[d]["Name"],
        "Vehicle": vehicles.iloc[v]["Name"],
        "Tires": tires.iloc[t]["Name"],
        "Glider": gliders.iloc[g]["Name"]
    }
    records.append(record)

optimized_df = pd.DataFrame(records)
```

4. Run the optimization algorithm:
   - Population size: 100
   - Offspring per generation: 200
   - Crossover probability: 0.6
   - Mutation probability: 0.3
   - Number of generations: 40

5. Convert Pareto Front results to dataframe

# Score Setups for Specific Track Types

```python
# Score setups based on track type
def score_setups(df, weights):
    scores = []
    for _, row in df.iterrows():
        score = 0
        for stat, weight in weights.items():
            score += weight * row[stat]
        scores.append(score)
    return scores
```

7. Calculate **weighted scores** based on track characteristics

```python
# Define stat weights for each track type
track_weights = {
    "long_straights": {
        "GroundSpeed": 1.5,
        "AirSpeed": 1.2,
        "Weight": 1.0,
        "Acceleration": 0.5,
        "MiniTurbo": 0.3,
        "Traction": 0.2
    },
    "curvy": {
        "Acceleration": 1.2,
        "MiniTurbo": 1.5,
        "Traction": 1.2,
        "GroundHandling": 1.0,
        "AirHandling": 1.0,
        "AntiGravityHandling": 1.0,
        "Weight": 0.5
    },
    "offroading": {
        "Traction": 1.5,
        "GroundHandling": 1.2,
        "WaterHandling": 1.0,
        "AirHandling": 1.0,
        "AntiGravityHandling": 1.0,
        "Acceleration": 1.5,
        "Weight": 1.0,
        "MiniTurbo": 1.0
    }
```

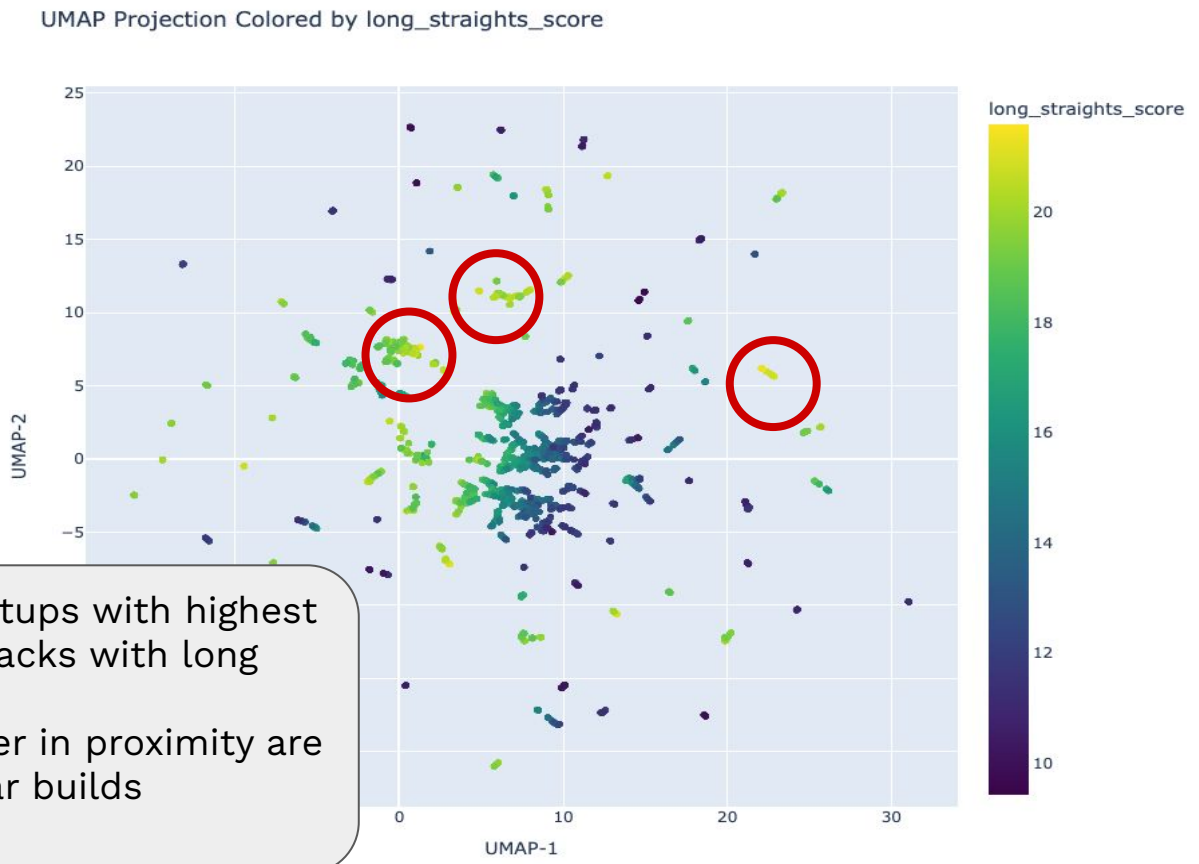
# Adding Weights for Different Track Types

Higher ground speed, airspeed, and weight for **long straight tracks**

Higher acceleration and mini-turbo for better drifting on **curvy tracks**

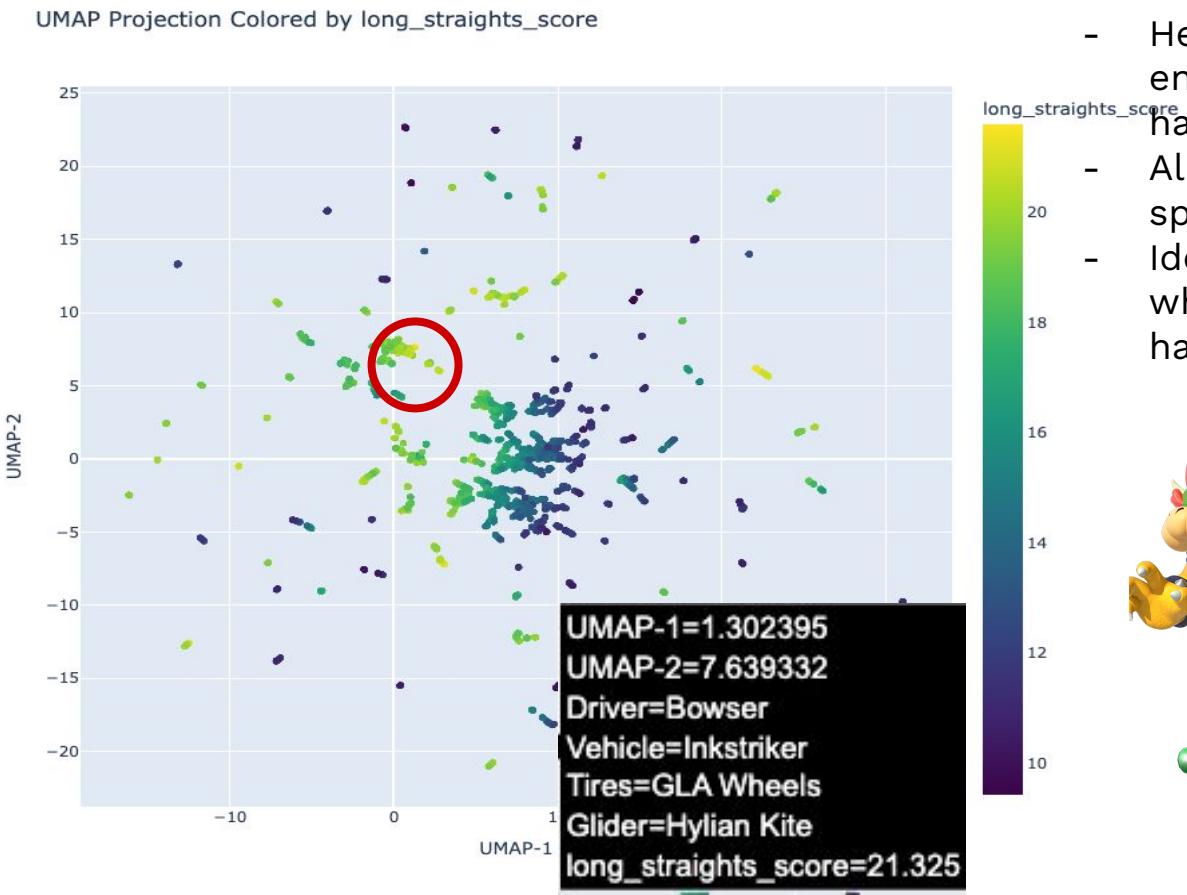
Higher traction, handling, and acceleration on **off-roading tracks**

# Results: UMAPs to Find the Best Setups



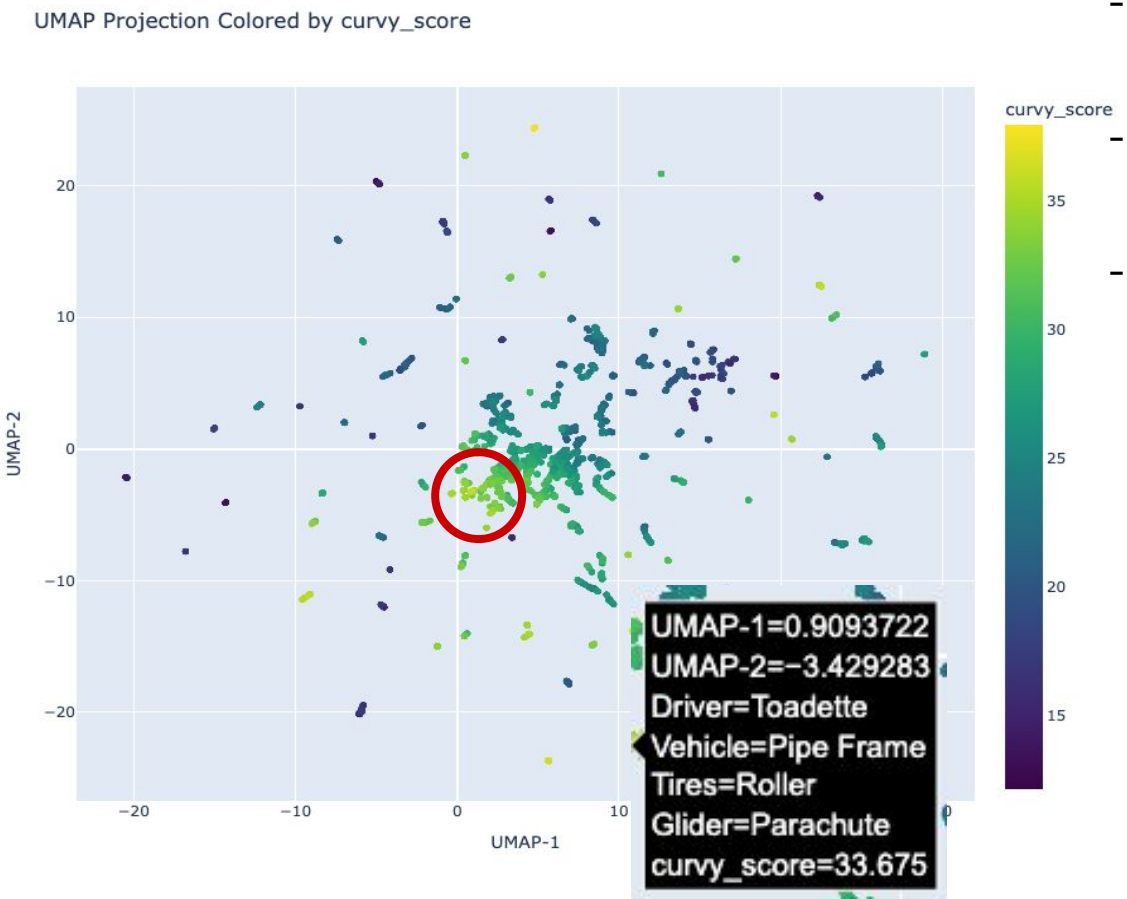

- Yellow = Setups with highest score for tracks with long straights
- Points closer in proximity are more similar builds

# UMAP by Long Straight Tracks Score



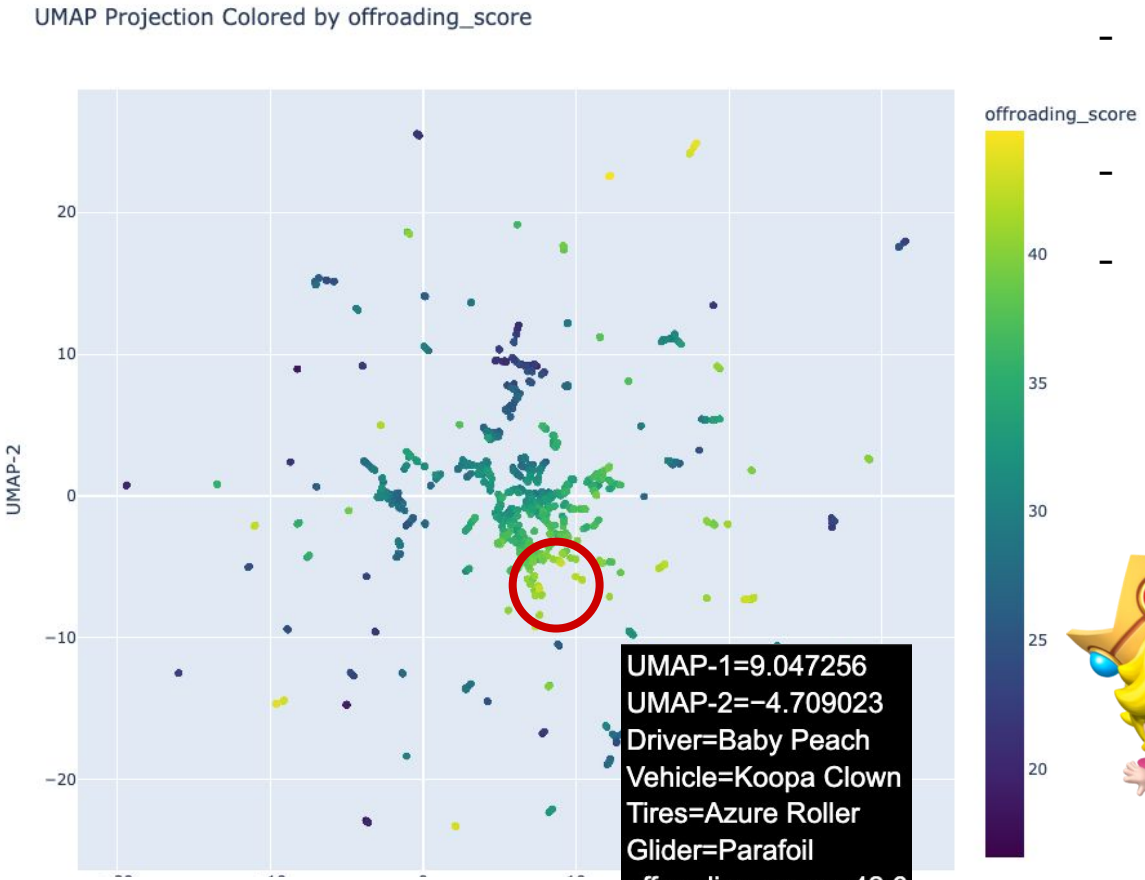

- Heavyweight driver and wheels enhancing speed over acceleration or handling
- All-terrain vehicle for high ground speed
- Ideal for maintaining a high top speed while trading off acceleration and handling

# UMAP by Curvy Tracks Score



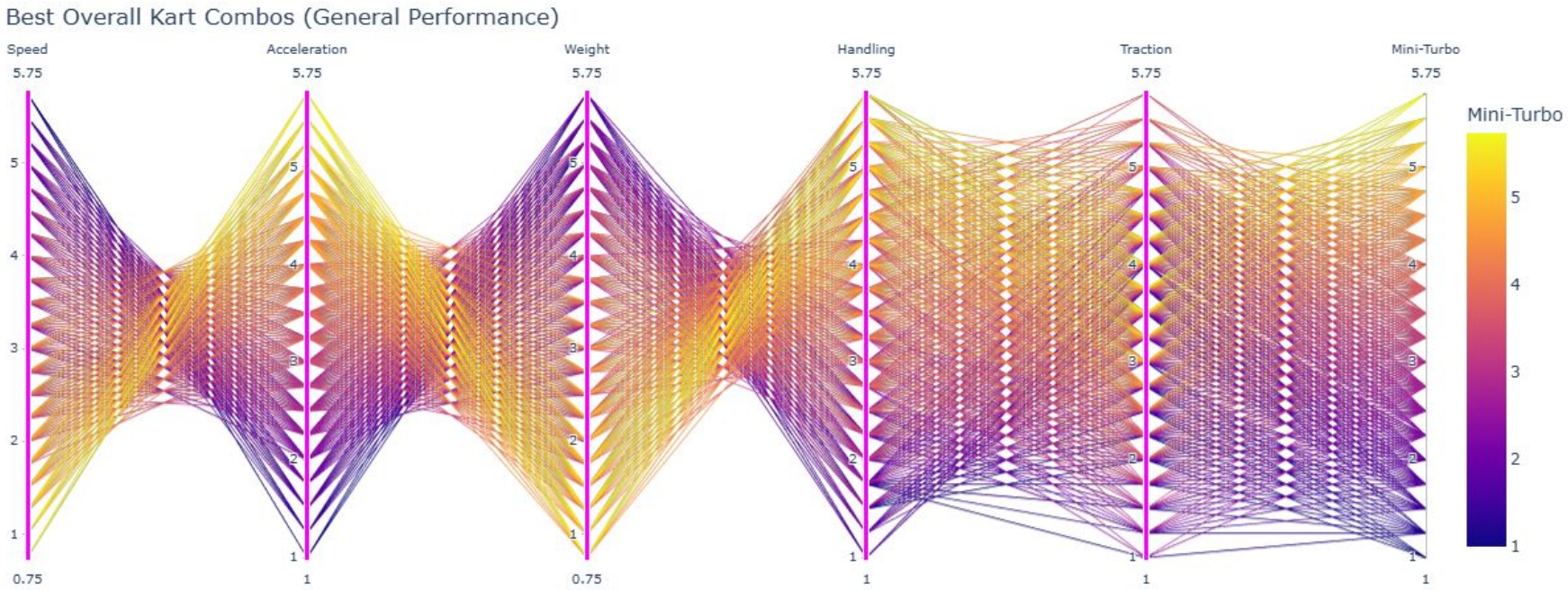

- Lightweight driver and wheels for higher acceleration and handling, trading off speed and weight
- Pipe frame for a balance between all stats, slightly favoring acceleration and handling
- Ideal for sharp turns, maintaining control and recovering speed quickly

# UMAP by OffRoading Tracks Score

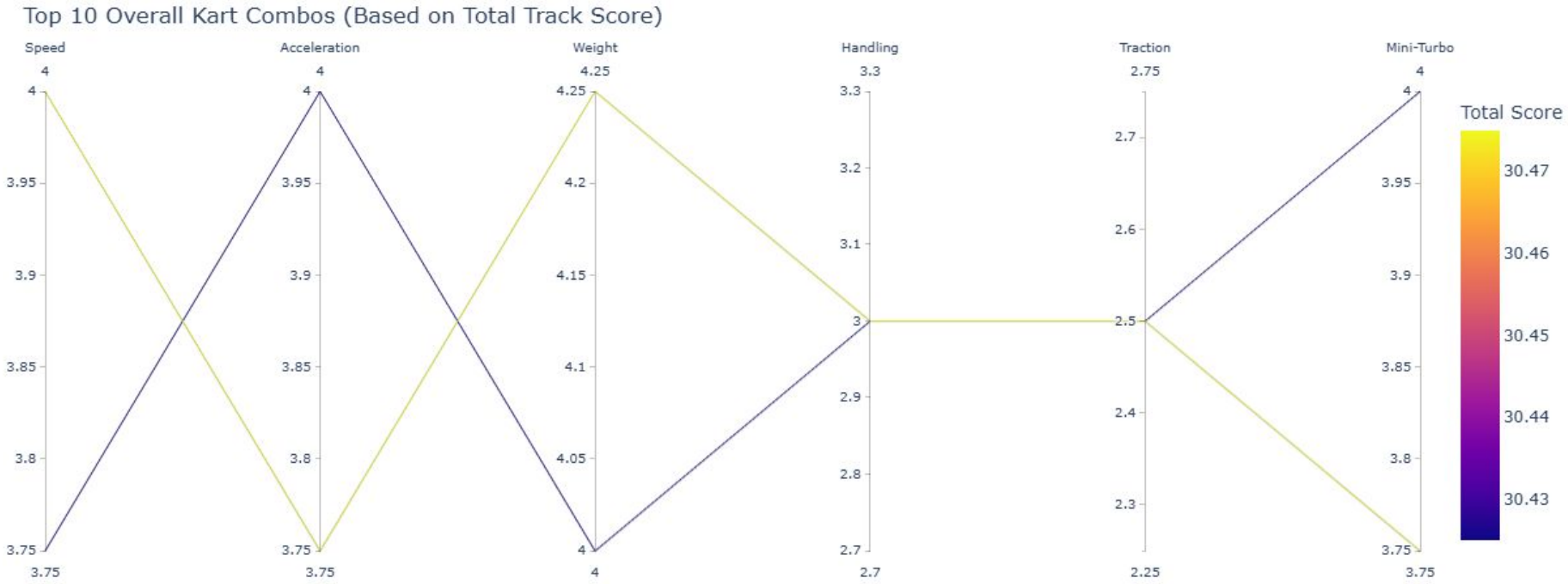



- Lightweight characters and small wheels for higher acceleration and better handling
- Koopa clown for handling and mini-turbo
- Ideal for getting back up to speed on tracks with many off-road sections

# Best Kart Combos



# Best Kart Combos (TOP 10)



# Best Kart Combos (TOP 10) Table

| Driver | Vehicle | Tires | Glider | Groundspeed | Acceleration | Weight | Groundhandling | Traction | Miniturbo | Curvy Score | Long Straights Score | Total Score |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bowser | Comet+ | Roller | Gold Glider | 4 | 3.75 | 4.25 | 3 | 2.5 | 3.75 | 10.125 | 20.35 | 30.475 |
| Morton | Comet+ | Roller | Wario Wing | 4 | 3.75 | 4.25 | 3 | 2.5 | 3.75 | 10.125 | 20.35 | 30.475 |
| Bowser | Cat Cruiser | Roller | Plane Glider | 4 | 3.75 | 4.25 | 3 | 2.5 | 3.75 | 10.125 | 20.35 | 30.475 |
| Bowser | Comet+ | Roller | Peach Parasol | 3.75 | 4 | 4 | 3 | 2.5 | 4 | 10.8 | 19.625 | 30.425 |
| Bowser | Comet+ | Roller | Parafoil | 3.75 | 4 | 4 | 3 | 2.5 | 4 | 10.8 | 19.625 | 30.425 |
| Morton | Comet+ | Roller | Hylian Kite | 4 | 3.75 | 4 | 3 | 2.75 | 3.75 | 10.125 | 20.15 | 30.275 |
| Bowser | Comet+ | Roller | Super Glider | 4 | 3.75 | 4 | 3 | 2.75 | 3.75 | 10.125 | 20.15 | 30.275 |
| Bowser | Comet+ | Roller | Waddle Wing | 4 | 3.75 | 4 | 3 | 2.75 | 3.75 | 10.125 | 20.15 | 30.275 |
| Morton | Comet+ | Roller | Cloud Glider | 3.75 | 4 | 3.75 | 3 | 2.75 | 4 | 10.8 | 19.425 | 30.225 |
| Bowser | Cat Cruiser | Roller | Flower Glider | 3.75 | 4 | 3.75 | 3 | 2.75 | 4 | 10.8 | 19.425 | 30.225 |

# Pro Player META

Medium weight; balanced stats with slight emphasis on acceleration and handling

Higher acceleration and handling, moderate speed and good traction

Lightweight wheels; high acceleration and handling, low speed

Slightly improves acceleration and handling

Overall:

- *Recovery focus*: higher acceleration = faster recovery after getting hit with items
- *Good handling and control* for complex track layouts
- *Versatile*: Effective on wide range of tracks

# Evaluation

What Worked Well

- Generated Pareto-optimal setups using NSGA-II
- Balanced competing objectives - Speed, Acceleration, Mini-Turbo, Speed, Traction

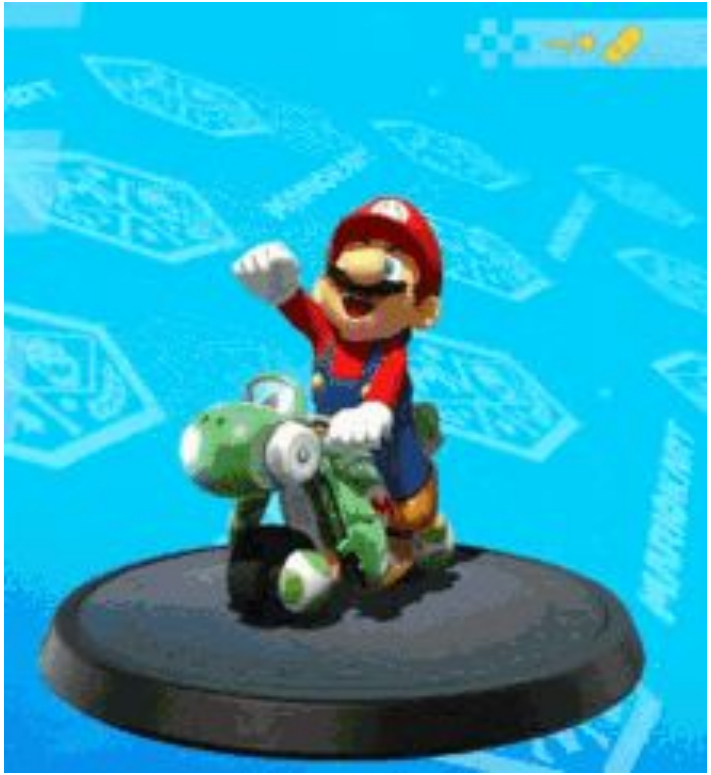
Limitations and Challenges

- Individual play styles were not considered - Aggressive vs. Defensive racers
- We simplified track features - Curvy, Straight, Off-road
- Did not consider surface types that affect performance - Mud, Sand, Grass or Water

These limitations allow for future improvement.

# Future Directions

- Empirical data: Running test races with different kart combos
- Data from real races: Running simulations with real people accounting for player preferences
- Track specific tuning: Sharp turns, different terrains.
- With more time and computing power, we'd get a greater evolution time for deeper optimization.



# Contributions

- Our project offers novel contributions to optimizing gameplay for Mario Kart 8 Deluxe players by searching for the most optimized kart combos taking into account different track types.

# References

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182–197. https://doi.org/10.1109/4235.996017College of Science and Engineering+3SCIRP+3ResearchGate+3

Marler, R. T., & Arora, J. S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, *26*(6), 369–395. https://doi.org/10.1007/s00158-003-0368-6SciSpace+6SpringerLink+6SpringerLink+6

Mayerowitz, J. (2025). Mario meets Pareto. *Mayerowitz.io*. https://www.mayerowitz.io/blog/mario-meets-pareto

jfmario. (2023). *Mario Kart 8 Deluxe Build Stats* [Data set]. GitHub. https://github.com/jfmario/mario-kart-8-deluxe-builds

# Thank you!

Questions?