

# Evolving Bids for a Fantasy-Football Auction

## Metaheuristics and Inverse Optimization in a Multi-Manager Setting

Marco De Rito

University of Trieste

2024/25

# Agenda

- 1 Problem Statement
- 2 Software Architecture
- 3 Representation
- 4 Algorithmics
- 5 Fitness
- 6 Conflict Resolution
- 7 Example
- 8 Hyper-parameter Tuning
- 9 Inverse multi-tuning

# Problem Statement

## Goal

Optimise a multi-manager fantasy-football auction by evolving a vector of **bids** that maximises total team score while satisfying *all* hard constraints.

### Notice legend

$B$	current budget of <i>one</i> manager
$b_i$	bid placed by that manager for player $i$
$N_{\max}$	squad size
$N_{\text{rem}}$	empty slots still to be filled
$m_r, M_r$	min / max players for role $r$
$r$	role index: P, D, C, A

### Constraints

- Bid domain:  $b_i \in \{0\} \cup \{1, 2, \dots, B_{\max}\}$
- Turn budget  $\sum_i b_i \leq B$
- Per-player cap  $b_i \leq 0.4B$
- Reserve credits  $B - \sum b_i \geq N_{\text{rem}}$
- Squad size  $|\text{team}| = N_{\max}$
- Role quotas enforced every turn

### Inputs

- Initial budget per manager  $B$  (CLI)
- Number of managers (CLI)
- Algorithm for each manager  $r \in \{PSO, DE, ES\}$  (CLI)
- squad size  $N_{\max}$  (CLI)
- Role quotas  $(m_r, M_r)$  for  $r \in \{P, D, C, A\}$  (CLI)
- Data set: 600+ Serie A players (23/24 stats)

# Project Files

Logical separation between data, logic, optimization and output.

## Main Modules

- `main.py` – CLI input & full pipeline control
- `data_loader.py` – loads & cleans player data
- `utils.py` – defines `Player`, `Manager`, `score_player`
- `optimization.py` – implements PSO, DE, ES + auction logic
- `report_generator.py` – generates PDF report

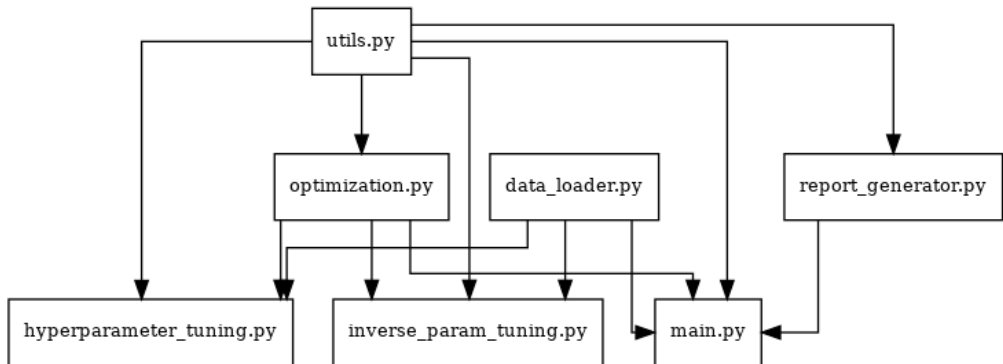
## Tuning Scripts

- `hyperparameter_tuning.py`
- `inverse_param_tuning.py`

## Supporting Files

- `Fantacalcio_stat.csv`, `requirements.txt`, `.git/`, `venv/`

# Architecture



# Genotype vs Phenotype

## Genotype

- Continuous bid vector:  
 $\mathbf{b} = (b_1, b_2, \dots, b_n)$
- Index  $i$  is bound to a fixed player
- Crossover / mutation touch numerical values only
- *Example:*  $(30.5, 0, 7.8, \dots, 0)$

## Phenotype

- Simulated squad obtained from  $\mathbf{b}$
- Includes roles, spent budget, expected score
- Graded via fantasy-football scoring rules

# Algorithmic Formulas & Parameters

## Particle Swarm Optimization (PSO)

$$\mathbf{v}_i^{t+1} = \omega \mathbf{v}_i^t + c_1 r_1 (\mathbf{p}_i - \mathbf{x}_i^t) + c_2 r_2 (\mathbf{g} - \mathbf{x}_i^t)$$

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^{t+1}$$

## Differential Evolution (DE)

$$\mathbf{y} = \mathbf{x}_a + F(\mathbf{x}_b - \mathbf{x}_c)$$

$$z_j = \begin{cases} y_j & \text{if } r_j < CR \text{ or } j = j_{\text{rand}} \\ x_{i,j} & \text{otherwise} \end{cases}$$

## Evolution Strategies (ES)

$$\mathbf{x}_{\text{child}} = \mathbf{x}_{\text{parent}} + \sigma \mathcal{N}(0, I)$$

(best  $\mu$  of parents + offspring survive)

## Algorithm Parameters

Alg.	Param.	Desc.	Val.
PSO	$\omega$	inertia	0.7
PSO	$c_1$	cog.	1.8
PSO	$c_2$	soc.	1.8
DE	$F$	diff. w.	0.5–1.0
DE	$CR$	cross.	0.7
ES	$\mu$	parents	40
ES	$\lambda$	offspr.	80

# Fitness Function

$$\mathcal{F}(\mathbf{b}) = \text{Penalty}(\mathbf{b}) - \sum_{i \in \mathcal{A}(\mathbf{b})} w_i \text{Score}_i, \quad \mathcal{A}(\mathbf{b}) = \{i \mid b_i \geq \text{thr}\}$$

- Penalty mixes *budget leftover*, missing roles, squad size errors
- $w_i$  doubles if the role is currently under-represented
- Minimisation problem: lower  $\mathcal{F} \leftrightarrow$  stronger squad, fewer violations
- $\text{thr} = 1$

```
def score_player(player):  
    goals = getattr(player, 'goals_scored', 0)  
    ....  
    matches = getattr(player, 'matches_played', 0)  
    return (0.5 * goals + 0.2 * assists - 0.05 * yellow - 0.1 * red + 0.2 * rating + 0.2  
            * pens - 0.5 * conceded + 0.5 * saved + 0.5 * matches)
```

Listing: Function score



# Auction Conflict Heuristic

- ① Gather all bids ( $mgr, player, b$ )
- ② Group by player
- ③ Single bidder  $\Rightarrow$  immediate assignment
- ④ Otherwise:
  - Sort bids  $b_1 \geq b_2 \geq \dots$
  - If  $b_1 - b_2 > g_{\text{trigger}} \Rightarrow$  highest wins
  - Else launch up to 5 dynamic rebids

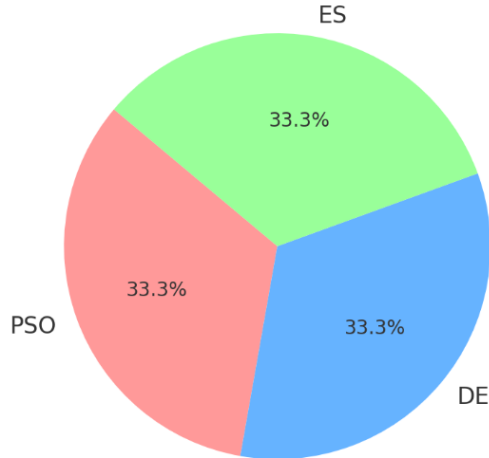
*(re bids = recompute bids with small noise or fallback threshold)*

```
# Inputs: b1 = top bid, b2 = second bid, B = second manager's remaining budget, n = number
          of players still needed
ratio = B / n
gap = b1 - b2
#Compute dynamic rebid increment
dynamic_inc = max(1, int(round(gap / 2 * ratio))) + 1
# Apply rebid
b2 += dynamic_inc
```

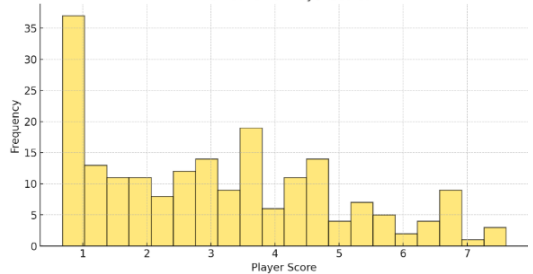
Listing: Simplified Dynamic rebid heuristic

# Example: Manager and Player Distributions

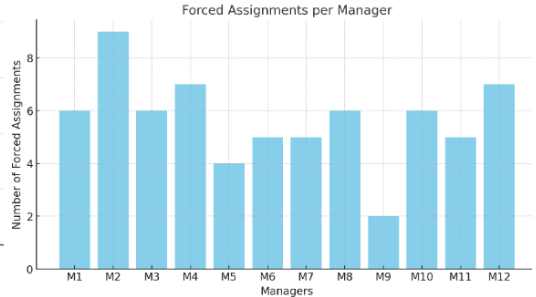
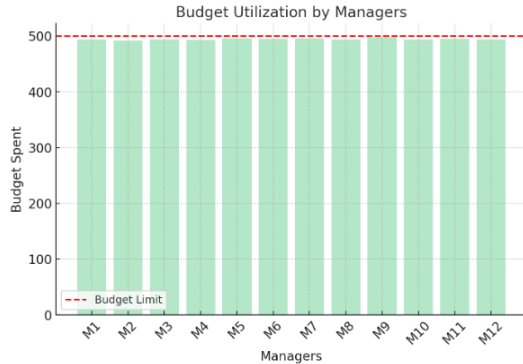
Manager Distribution by Strategy



Distribution of Player Scores



# Example: Budget and Forced Assignments



# Example: Tabular Analyses

Performance by Strategy

Str.	Mgr	Avg Score	Avg Forced
PSO	4	49.9	6
DE	4	51.8	7
ES	4	44.8	4

Manager Recap

Mgr	Forced	Spent	Score
1	6	494	52.9
2	9	492	51.5
3	6	494	44.5
4	7	493	53.6
5	4	496	49.2
6	5	495	46.0
7	5	496	47.0
8	6	494	55.3
9	2	498	44.2
10	6	494	46.6
11	5	495	40.0
12	7	494	53.2

Player Score Summary

	Best	Worst	Avg
Score	12.9	0.68	2.84

## Methodology

- Same 25-player pool, fixed seed, 60 auction turns
- Test manager + 1 random rival (guarantees bidding pressure)

## Cartesian products $\Rightarrow$ 24 runs

- PSO = 2 inertia weights  $\times$  2 swarm sizes = **4** runs
- DE = 3 population sizes  $\times$  2  $F$  ranges  $\times$  2  $CR$  = **12** runs
- ES = 4  $(\mu, \lambda)$  pairs  $\times$  2 generation counts = **8** runs

## *Why these ranges?*

- Values are the *standard defaults* most cited in the literature (Clerc and Kennedy for PSO, Storn and Price for DE,  $\lambda > \mu$  rule for ES)

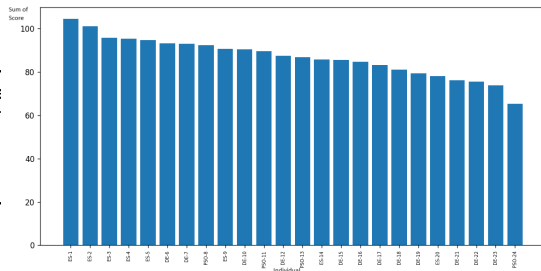
# Hyper-parameter Tuning – Results

## Best configuration for algorithm

Alg.	Best h-p	Sum Score
ES	$\mu = 20, \lambda = 40, ngen = 50$	<b>104.7</b>
DE	pop= 15, $F = (0.5, 1.0), CR = 0.7$	93.3
PSO	swarm= 60, $w = 0.9, c_1 = c_2 = 1.49445$	92.4

### Parameter grid

PSO	$w \in \{0.9, 0.5\}, c_1 = c_2 = 1.49445, \text{swarm} \in \{30, 60\}$
DE	pop $\in \{10, 15, 20\}, F \in \{(0.5, 1.0), (0.7, 1.2)\}, CR \in \{0.7, 0.9\}$
ES	$(\mu + \lambda) \in \{(15 + 30), (15 + 40), (20 + 30), (20 + 40)\}, ngen \in \{50, 80\}$



Sum of score for every configuration (higher = better).

## Goal

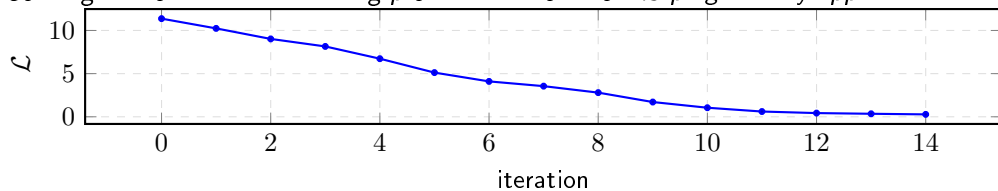
Fit the hyper-parameters of **PSO, DE & ES** so that each auctioned roster reproduces a target triple (score, forced picks, leftover) = (100, 4, 0).

$$\mathcal{L}(\theta) = |\text{score} - 100| + |\text{forced} - 4| + |\text{leftover} - 0| \quad \longrightarrow \quad \min_{\theta}$$

- **Outer optimiser:** 30-particle PSO (40 iterations,  $\omega=0.7$ ,  $c_1 = c_2 = 1.5$ ).
- **Search space:**  $algo\_id \in \{0:\text{PSO}, 1:\text{DE}, 2:\text{ES}\} + 4$  real h-params.
- **Best configuration found:**  
DE (pop = 10,  $F = [0.7, 1.2]$ ,  $CR = 0.7$ )  $\Rightarrow \mathcal{L}_{\min} = 0.278$ .

## Inverse multi-tuning/2

*Convergence of the inverse tuning process: the total loss  $\mathcal{L}$  progressively approaches zero.*

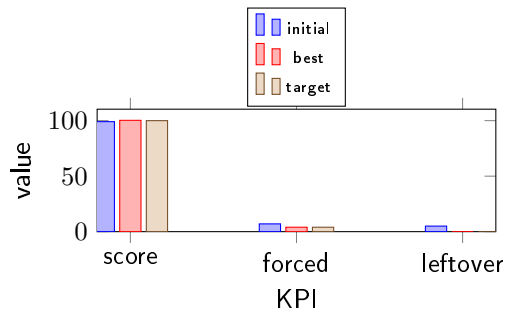




## Top-5 trials

Alg.	Pop./Swarm / $(\mu, \lambda)$	$F/(\omega)$	$CR/c_{1,2}$	$\mathcal{L}$
DE	10	0.7–1.2	0.7	0.278
ES	(20,30)	–	–	1.86
ES	(20,40)	–	–	1.98
PSO	60	0.5	1.49	4.08
DE	20	0.7–1.2	0.7	4.20

*Best configurations found during inverse tuning: DE and ES performed best in reproducing the target profile.*



*KPI comparison: the best configuration closely matches the target (score 100, 4 forced picks, 0 leftover credits).*