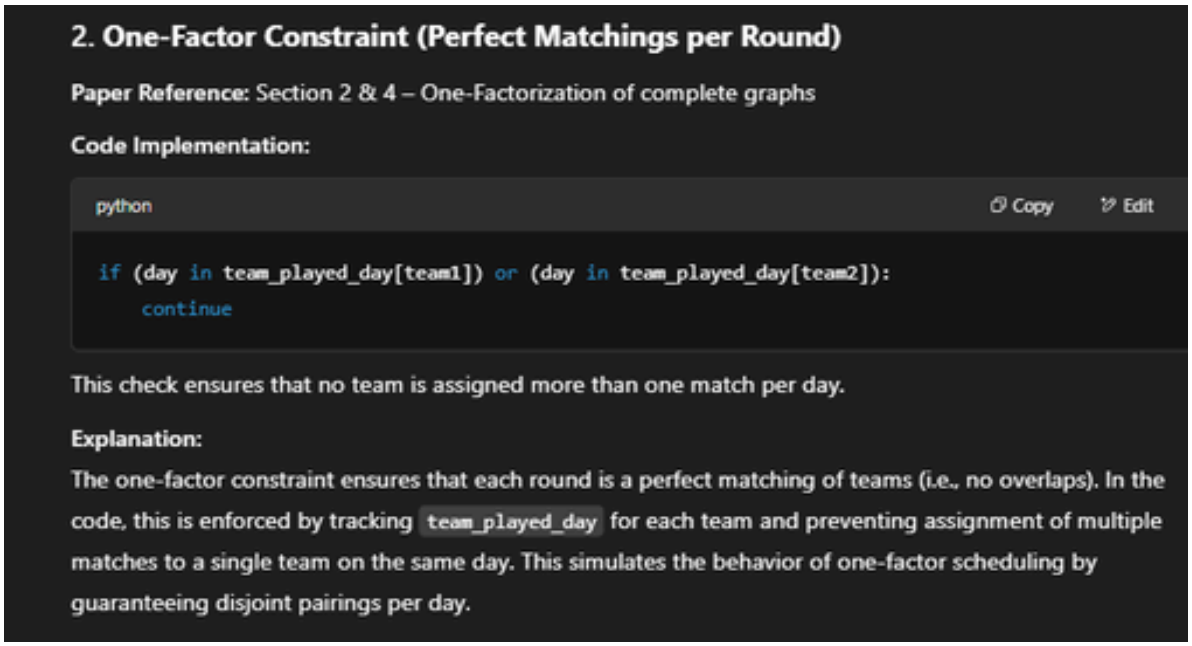
📝 Constraint Implementation Documentation

This section documents how constraint handling in the scheduling algorithm aligns with the global constraints described in the research paper:
“Global Constraints for Round Robin Tournament Scheduling” by Henz, Müller, and Thiel (2004)



3. Venue-Time Conflict Avoidance

Paper Reference: Section 6.5 – Intermural tournaments, resource conflict prevention

Code Implementation:

python Copy Edit

```python
if len(matches_at_slot) > 1:
    penalty += 100 * (len(matches_at_slot) - 1)
```

Explanation:

This constraint ensures that no two matches are scheduled at the same time in the same venue, reflecting the physical constraint that a venue can host only one match at a time. The penalty discourages such conflicts during the evolutionary search.

4. Spacing Between Matches (Breaks and Carry-Over Approximation)

Paper Reference: Section 6.3 – Carry-over effect & Section 6.4 – Break minimization

Code Implementation:

python Copy Edit

```python
for i in range(1, len(played_days)):
    if played_days[i] - played_days[i - 1] == 1:
        penalty += 20
```

Explanation:

The paper discusses minimizing the carry-over effect (e.g., teams playing consecutively without sufficient rest). In the code, a penalty is applied when a team plays on consecutive days. Although not a full carry-over matrix calculation, this approximates the idea by encouraging rest days between matches.

5. Schedule Diversity and Balance

Paper Reference: Section 3 – Constraint Propagation and search tree balance

Code Implementation:

python Copy Edit

```python
var = statistics.variance(played_days)
bonus += var * 0.5
```

Explanation:

While the paper emphasizes propagation techniques for reducing search tree depth, this implementation indirectly encourages well-balanced schedules by rewarding higher variance in played days per team—thus distributing matches more evenly across the schedule.