Faculty of Electrical Engineering and Computing
Graduate study

Heuristic Optimization Methods
Academic year 2020/2021

Student: Miljenko Šuflaj (*0036501442*)

1. Details and results

Programming language
Python

Best found results

Instance 1
Tabu Search
Score: 1419
First team lineup: 10, 71 73, 130, 542, 280, 283, 546, 119, 74, 434
Substitutions: 512, 622, 534, 70

Simulated annealing
Score: ____
First team lineup: ____
Substitutions: ____

Instance 2
Tabu Search
Score: 1520
First team lineup: 16, 72, 73, 74, 545, 285, 554, 291, 286, 75, 139
Substitutions: 533, 529, 626, 70

Simulated annealing
Score: _____
First team lineup: _____
Substitutions: _____

Instance 3
Tabu Search
Score: 1964
First team lineup: 10, 74, 75, 76, 578, 301, 300, 308, 77, 117, 635
Substitutions: 665, 572, 573, 71

Simulated annealing
Score: _____
First team lineup: _____
Substitutions: _____

2. Tabu search

Algorithm components

initial solution	Obtained by running the greedy algorithm from the previous task.
neighborhood definition	All swaps for each player after which the new solution would be feasible.
structure of the tabu list and the tabu tenure	Two sets – one containing an item paired with the "time" it was inserted, one containing only the item. Although in a separate run a tabu tenure of 2000 was tested, the results are from tabu tenures 3, 10 and 100.

Pseudocode

```
initial := greedy_algorithm(players)

current = initial
best = initial

while True:
   neighbourhood := get_neighbours(current)

   if neighbourhood is empty:
      break

   next := best(neighbourhood)

   if next is best seen solution:
      best = next

   current = next

   if there is no better solution in n iterations:
      break

return best
```

Description

Classic algorithm – gets neighbourhoods as described in the table, and fetches the best one as the next solution until the neighbourhood is empty or until n interations have passed without acquiring a new best solution. In our case, n was 100 most of the time, 1000 for tenure 2000.

Analysis

Finding neighbourhoods was very slow. It took around 1.5 seconds to find one. Changing tenures didn't change a lot. This is due to a very stable local optimum the greedy solution found. This was observed in the previous task as well – the optimum found was good (enough), but it was almost

impossible to get out of. After testing a tenure of 2000 and 1000 iterations, it seems unlikely better results could be obtained without significantly higher tenures. This could be the case because the local optimum found by the greedy algorithm is a fairly bad one – one where the neighbourhoods are more-or-less identically good or bad, and where the optimum is a lot better than its neighbourhood. When comparing results to better ones, it is apparent that the hardest circumstance our algorithm faces is a non-optimal distribution of position. In other words, if our greedy algorithm obtained a solution where the position distribution is closer to the real optimal solution, tabu search would have a better chance at finding the real optimum. Our neighbourhood definition allowed for this to happen, but not immediately, suggesting a larger tabu tenure might be able to fix its shortcomings. However, given that we needed almost half an hour to run 1004 iterations, for our solution it is infeasible. Blame me and my subpar implementation.

3. Simulated annealing

Algorithm components

initial solution	
neighborhood definition	
initial temperature	
decrement function	
cooling schedule	
stopping criteria	

Pseudocode

_____

Description

Sorry, I really lost motivation after the first part so I didn't implement this. I'll do better on the project.

Analysis

_____