**Faculty of Electrical Engineering and Computing**

**Graduate study**

**Heuristic Optimization Methods**

**Academic year 2020/2021**

**Student:** ***Toma Petrač (0036493691)***

1. Details and results

Programming language  
\_\_\_Python 3.7.1\_\_\_

Best found results

Instance 1  
Tabu Search  
Score: \_1448\_\_\_\_  
First team lineup: 10,71,119,73,130,280,343,283,542,548,551 (Henderson, Alexander-Arnold, Baldock, Van Dijk, Lundstram, De Bruyne, Traore, Salah, Ings, Jimenez, Vardy)  
Substitutions: 70 Button, 273 Gibson, 521 Guendouzi, 520 Elneny

Simulated annealing  
Score: \_\_1427\_\_\_  
First team lineup: \_10,71,73,72,130,280,283,324,542,549,596 (Henderson, Alexander-Arnold, Van Dijk, Lundstram, Baldock, Richarlison, De Bruyne, Salah, Ayew, Ings, Vardy

Substitutions: 70 Button, 273 Gibson, 521 Guendouzi, 520 Elneny

Instance 2  
Tabu Search  
Score: \_1551\_\_\_\_  
First team lineup: 16,72,139,74,119,285,282,344,545,551,554 (Pope, Alexander-Arnold, Van Dijk, Lundstram, Baldock, Traore, Salah, De Bruyne, Ings, Jimenez, Vardy)  
Substitutions: 521,522,271,70

Simulated annealing  
Score: \_\_1546\_\_\_  
First team lineup: 16,72,73,74,139,285,291,290,545,551,554 (Pope, Alexander-Arnold, Robertson, van Dijk, Lundstram, Mahrez, Richarlison, De Bruyne, Ings, Jimenez, Vardy)  
Substitutions: 521,522,271,70

Instance 3  
Tabu Search  
Score: \_\_2050\_\_\_  
First team lineup: 10,74,76,117,75,301,308,315,578,583,584 (Pope, Alexander-Arnold, Stevens, Robertson, Willian, Martial, De Bruyne, Ings, Jimenez, Vardy)  
Substitutions: 297,573,68,572

Simulated annealing  
Score: \_2023\_\_\_\_  
First team lineup: 10,74,75,117,87,301,308,315,578,583,584 (Pope, Alexander-Arnold, Robertson, Stevens, Tarkowski, Willian, Martial, De Bruyne, Ings, Jimenez, Vardy)  
Substitutions: 297,573,68,572

1. Tabu search

Algorithm components

|  |  |
| --- | --- |
| initial solution | Solution given by the greedy algorithm. Firstly, we searched for the 4 least valuable players to be our substitutes (of course, we need to consider problem’s constraints regarding the number of possible players on different positions).  For this particular problem, we had to reduce our search space for greedy algorithm in order to get a feasible solution due to problem’s constraints. This is done by taking out every n-th player out of the search space. |
| neighborhood definition | Neighboring solution is every list of eleven players which differs from our current solution by only one player. |
| structure of the tabu list and the tabu tenure | Tabu list is a counter object (dictionary) where keys are players and values are numbers of iteration in which they cannot be chosen for our solution. Every time a player is swapped for another player, player that is removed goes into tabu list with value that is tabu tenure.  Different numbers have been tried for tabu tenure for this particular problem, and it seems that the optimal value for tenure goes around 150. |

Pseudocode

initialize tabu\_list

initialize tabu\_tenure

incumbent\_solution = best\_eleven\_from\_greedy

incumbent\_points = total\_points\_from\_greedy

no\_iter\_without\_improvement = 0

while (no\_iter\_without\_improvement < 10000):

player\_replacements = []  
 for i in range(11):  
 player\_to\_remove = best\_eleven[i]  
 position = player\_to\_remove.pos

tabu\_list[player\_to\_remove] = tabu\_tenure  
   
 for player in sorted(all\_players\_dict[position],key=pnts):  
 if constraints\_are\_met and tabu\_list[player] < 1:  
 player\_replacements.append((player\_to\_remove, player, point\_difference))  
   
 player\_replacements.sort(key=point\_difference)  
 player\_to\_remove, player\_to\_add = player\_replacements[0]  
 my\_team.remove(player\_to\_remove)  
 my\_team.add(player\_to\_add)

update tabu\_list()  
 update\_budget and total\_points()

update\_club\_constraints()

if total\_points > incumbent\_points:

incumbent\_solution = my\_team

incumbent\_points = total\_points

no\_iter\_without\_improvement = 0

else:

no\_iter\_without\_improvement++

Description

Firstly, we have to initialize the tabu list and tabu tenure. We store our solution from greedy algorithm into incumbent solution and memorize the total points. Then, we start the tabu search, and keep it running as long as we don’t reach a certain number of iterations without improvement.

Next part is very similar to local search algorithm. Main difference is, of course, the tabu list. So, whenever we remove a player from our team (our solution), we put that player into tabu list, meaning we won’t be able to put him back in team for n number of iterations, where n is tabu tenure. Also, whenever we make a replacement, besides updating our budget, total points and club constraints, we also need to update the tabu list by reducing the tabu value for each player except the one we’ve removed in this iteration.

Finally, we check if the total points accumulated in this iteration is greater than the points for incumbent solution, we update our incumbent solution.

Analysis

Solutions we’ve acquired through this tabu search algorithm are notably better than the ones given by the local search algorithm. However, due to the large number of iterations in the while loop, tabu search implemented for this problem is considerably slower than the local search algorithm.

A couple of different values have been tried for tabu tenure. Here’s a list of them with their respective results for all instances:

* **tt = 3:** instance1 points = 1427; instance2 points = 1529; instance3 points = 2050
* **tt = 30:** instance1 points = 1442; instance2 points = 1551; instance3 points = 2050
* **tt = 100:** instance1 points = 1448; instance2 points = 1551; instance3 points = 2050
* **tt = 500:** instance1 points = 1427; instance2 points = 1551; instance3 points = 2050

We can see that the best results are accomplished for the value of tabu tenure around 150. If the tenure is too low, then we get the results similar to those of local search algorithm. Same goes for too big tenure (as it can be seen for the results for instance1).

1. Simulated annealing

Algorithm components

|  |  |
| --- | --- |
| initial solution | Initial solution for simulated annealing is the same as the solution for tabu search. |
| neighborhood definition | Also the same as for tabu search. Team of eleven players where only 1 player is different than the one in our team (solution). |
| initial temperature | T = 100 |
| decrement function | I’ve tried both geometric decrement function (T = alpha\*T) with alpha=0.99 and very slow decrease decrement function (T = T / (1 + beta\*T) with beta=0.001 |
| cooling schedule | Homogeneous cooling schedule: T is decreased after each iteration by a very small amount |
| stopping criteria | Until T reaches 0.01 or less. |

Pseudocode

incumbent\_solution = best\_eleven\_from\_greedy

incumbent\_points = total\_points\_from\_greedy

alpha = 0.99

beta = 0.001

for i in range(11):

temperature = 100

while temperature >= 0.01:

player\_to\_remove = best\_eleven[i]

position = player\_to\_remove.pos

player\_to\_add = randomly\_pick(all\_players[position])

if constraints\_are\_met():

point\_diff = player\_to\_add.points – player\_to\_remove.points

prob = 1 if point\_diff > 0 else exp(point\_diff / temperature)

if prob >= random(0, 1):

best\_eleven[i] = player\_to\_add

update club\_constraints()

update\_budget()

update\_points()

if total\_points > incumbent\_points:

incumbent\_solution = best\_eleven

incumbent\_points = total\_points

temperature \*= alpha (or temperature /= (1 + beta \* temperature))

Description

We loop through all 11 of our players, trying to find the best player possible for each position. We initialize the temperature and then start randomly searching for players who play a given position. After we pick one randomly, if the constraints are met, we put him in our team If the solution gives us more points. If not, we can still put him in our team based on the probability that depends on the point difference and the current temperature.

After we have put the player in our team, we just need to update everything, and check if the new solution gives us better results than the incumbent solution.

Finally, we decrease the temperature, and start over, repeating the process until the temperature falls to 0.01 or less.  
  
Analysis

Considering the fact that this is a stochastic algorithm, results can differ from one run to another. Performance-wise, algorithm runs much faster than the tabu search. However, I couldn't get as good results as with tabu search (although it's not that big of a difference). Perhaps I missed something. Also, difference between the initial temperatures is almost nonexistent, as is for different temperature reduction functions (except that very slow decrease is much slower than the geometric reduction function, but still not as slow as tabu ☺).

I also tried accepting nonfeasible solutions, as a way of finding new solutions and escaping local optima. However, difference in results was also almost nonexistent (except maybe it was a little bit more consistent because when the budget would drop below 0, it wouldn't be so easy to return back to positive, so incumbent solution would stay at about the same value in every run).

For instance1, I got the same result as with local search algorithm. For instance2, I got better than local search, and only little worse than tabu.