Good morning, everyone. I’ll be presenting about the travelling tournament problem and use the simulated annealing approach to solve it. (next)

The Traveling Tournament Problem represents the fundamental issues involved in creating a schedule for sports leagues where the amount of team travel is an issue. There are two basic requirements. The first is a feasibility issue in that the home and away pattern must be sufficiently varied to avoid long home stands and road trips as well as back-to-back games. The second is the goal of preventing excessive travel, which means the objective is to minimize total travel distance. (next)

To formally define the problem, a set of n teams with n being even and a symmetric n by n integer distance matrix with its (i,j)th entry being the distance from the home of team i to the home of team j are given, find a double-round robin tournament on the teams in T such that the length of every homestand and road trip is less or equal to 3, no back-to-back games, and the total distance travelled by the teams is minimized. (next)

While each issue has been addressed by integer programming and constraint programming respectively, their combination is a relatively new problem for both groups. It might seem those insights from these solution methodologies would make the TTP relatively easy to solve. However, even very small instances of the TTP have proven difficult for traditional methods. Thus, the TTP seems to be a good problem for a metaheuristic optimization approach, like simulated annealing.

We all know how simulated annealing works and its advantages, so I’ll just skip this part. However, it might be worth noticing that the feature of accepting worse solutions allows for a more extensive search for the global optimal solution might come in extra handy here as even with a small number of teams, like 6, there is a huge search space and an objective function with a lot of parameters. This might lead to a relatively large number of local optima which will hinder the searching process. (next)

With that in mind, let’s focus on the problem itself. The required output is just a schedule, so directly using discrete encoding is good enough for the problem. For each solution, it will be a n by 2n-2 matrix, with its rows representing the schedule of a certain team and its columns representing the schedule on a certain round. If there is no ‘@’ in front, it means that it’s a home game, otherwise it is an away game. (next)

For example, team 1 plays an away game against team 0, then plays a home game against team 2. (next)

There are 5 different search operators for this problem. Besides changing the arrangement of the schedule, they also preserve the property of the double round robin schedule. It’s hard enough to just follow the double round robin schedule, so we will just ignore the other two constraints. (next)

The first one is swap homes. Basically, it switches the order of the home and away games against a certain team. Like team 0 plays against team 1 at home first, after the swap, team 1 will play against team 0 at home first. (next)

The second one is swap rounds. It simply just switches the schedule of the two rounds. (next)

The third one is swap teams. It completely swaps the schedule of two teams besides the game against one another. For teams other than these two, they have to change accordingly. For example, team 0 and team 1 switches their schedule, as a result, team 2 and team 3 will have to replace their game with team 0 with team 1 and vice versa. (next)

The fourth one is swap partial rounds. Instead of switching the schedule of two rounds entirely, this only swaps the schedule of two rounds from the same team. For example, we will swap round 2 and round 5 of team 1. But after that, we found a contradiction involving team 0, so we need to perform swap partial rounds on team 0 as well. Luckily, this time there are no more contradictions. Otherwise, we will keep swapping. In the worst-case scenario, it just becomes a swap rounds operation. (next)

The last one is swap partial teams, switching the schedule of two teams from the same round. Similar to swap partial rounds, if there are contradictions, like the two ‘@1’ we have here, we’ll just keep swapping on the problematic part until it is a double round robin schedule. (next)

Next is how the entire program goes. It will first generate a double round robin schedule. One of the 5 operations will be randomly performed on it. Then we’ll evaluate the score and see whether we will adopt the new solution or not. You know, the common simulated annealing procedure.

There is one thing to note is that the evaluation function is adopting a penalization strategy for the violation of constraints, no back-to-back games, and no more than three consecutive home or away games. Because I found it is very likely for randomly generated schedules to violate these two constraints and many of them cannot be fixed with only one operation mentioned above. Therefore, in order to keep the iterative process going, a penalization based on the number of constraints violated is added to the objective value instead of just the rejection. (next)

This is the final result. To further avoid the convergence to local optimum, the multi-start technique is used as well, as I found the variation of the result is quite high so this would give a steadier performance even though it is more time-consuming. There might still be some improvement to the result if we increase the number of starts but the time-performance trade is not that promising as we can see it is already quite close to the lower bound. (next)

That’s all, thank you for listening. (next)