Soccer Tournament Scheduling Using Constraint Programming

Mike DiNunzio · Serge Kruk

Abstract Larger soccer tournaments of school-age children in Michigan can host as many as 600 teams divided into three age groups, all of which must play a set number of games, on a fixed number of fields, during a weekend. This leads to three scheduling problems of roughly 200 teams that must be done concurrently. With side constraints involving resting time, limited number of games in a day and playoffs, the task soon becomes unruly if done by hand. This paper summarizes the results of efforts to develop a Constraint Programming solution to this Soccer Tournament Scheduling Problem, and concludes that an appropriate model, combined with an appropriate search strategy, can handle problems of practical size.

Keywords Tournament · Constraint Prohramming

1 Introduction

1.1 Background

Based on one of the authors' multi-year experience, scheduling soccer tournaments is an exercise typically done with pencil and paper. As a result, there is an inherent unfairness in the schedules that are computed, as well as certain inefficiencies (empty fields, long hours). Overflow games, or games that do not fit neatly into the Saturday/Sunday preliminary schedule, are usually scheduled on the previous Friday night, bringing players, referees, and tournament officials to the tournament site for an additional evening. While commercial sports tournament scheduling software is available, most is of the drag and drop variety (not functionally different from using pencil and paper), or does not address the issues and requirements that are particular to soccer tournaments.

M. DiNunzio
Oakland University
MI, USA
E-mail: mrdinunz@oakland.edu

S. Kruk
Oakland University
MI, USA
E-mail: kruk@oakland.edu

This paper examines a practical solution to these issues through the use of constraint programming. A number of different approaches were considered, some of which did not work at all, and others which worked moderately well (scheduling 10-50 teams). Ultimately, tuning the search strategy, a solution technique was reached which works well with over 200 teams.

1.2 How Soccer Tournaments are Organized

The first step in scheduling a soccer tournament is organizing the divisions, small groups of three to five teams of comparable ability and skills. This aggregation is done by the coaches to offer to the players a reasonable yet challenging experience. Then divisions are paired to offer intra and inter-divisional games. Usually three preliminary games are played, followed by a playoff game where the winners of divisions face off against each other. Divisions of varying sizes are scheduled in the following manner:

(3 Teams) Each team plays the other two teams in their division, plus a team from a 2^{nd} division. A championship playoff game is scheduled pitting the winners of the two divisions.
(4 Teams) Each team plays the other 3 teams within their division. The winner of the division plays the winner of another division in a playoff game.
(5 Teams) Each team plays the other 4 teams within their division. A playoff game may or may not be scheduled.

The ability to schedule divisions of 3,4, or 5 teams gives the tournament organizers a great deal of flexibility in setting up the divisions and handling last minute team additions. A typical tournament has preliminary games all day Saturday, and the 1st half of Sunday. The 2^{nd} half of Sunday is set aside for the playoff games. If necessary, games can be played (local teams only) on the Friday night prior to the start of the tournament.

Soccer fields are one of three sizes; 6v6, 8v8, and 11v11 referring to the number of players on a side. Younger players, up to about age 9, play on the 6v6 fields. Players from about the age of 9 through 11 play on the 8v8 fields. Older players use the 11v11 fields. There is no mixing of the three groups. Scheduling a soccer tournament then becomes three completely separate problems, one for each of the three field sizes.

Based on the experience of one author, the following issues stand to be improved through better soccer tournament scheduling:

1. Games are often widely spread throughout the day. It is not uncommon for a team to play a game at 7 am, then have to return to play again at 7 pm. Most families would prefer to play at 7 am and 11 am, for example, freeing up the rest of the day to do something else. We would like to cap the length of time a team has to wait between games.
2. Fields are not always efficiently used; there are often empty fields throughout the day. Is it possible to compress the schedule for increased efficiency?
3. Related to 2, is that overflow and hard-to-schedule games are put into a Friday night time slot, prior to the official start of the tournament. This uses tournament resources as well as disrupting families Friday nights. Can we eliminate the need for Friday night games?

4. Scheduling a tournament is a drawn out process, generally taking 2-3 weeks from the time registration closes to several days before the start of the tournament. Can we reduce that 2-3 week time period to under an hour even when scheduling concurrently the three age groups?
5. When scheduling playoff games, we would like to begin playoff games for a particular pair of divisions before the end of the preliminary schedule, provided the slots are open and there is no chance of any preliminary games interfering with the playoffs for that division.
6. A large soccer tournament can have 200 or more teams to be scheduled for each of the three field sizes. Our solution has to be capable of handling that many teams.

1.3 Constraints to be Implemented

From the above observations, we formulate the problem. Our goal is to schedule all preliminary games and playoff games, given a number of teams, divisions, playing fields, and time slots. The following hard and soft constraints are under consideration.

1. No team plays more than 2 preliminary games in a day.
2. Each team plays the prescribed number of games against division and cross-division opponents.
3. Each team needs a rest period between games.
4. There is a maximum time gap between games, so that families do not need to spend all day at the field.
5. A playoff game can only be scheduled after the last scheduled preliminary game for the divisions involved, with an appropriate rest period between the games.
6. Each fields use is maximized so that Friday night games do not need to be scheduled (soft constraint).
7. Use the minimum number of time slots to fill the schedule, possibly allowing teams to finish earlier and/or start later, thereby increasing goodwill (soft constraint).

1.4 Objectives

The soft constraints could be seen as an objective function: minimizing the total number of time slots so that we do not schedule on Friday night. Note that minimizing more than that is neither useful not required. A second additional objective (assuming that the 'no Friday games' is achieved) is to minimize the time between the first and last game of each team in a day. Minimizing idle time provides the player and their families a better experience.

1.5 Previous work in Sports Scheduling

The definitive annotated bibliography of sport scheduling is [3]. Much research has been done in the area of starting with the pioneering work of de Werra [12, 11, 10]. Much of the focus has been on round-robin or double round-robin tournaments, where a team plays once home and once away, [1, 6], or some other variation where a certain balance between home and away is the goal [2]. Sometimes, there may be a limit on one or the

other tournament [1,8], possibly based on the strength of the team or the geographical distance. Some other restrictions are based on the sharing of home facilities between teams [13].

Much of this research has little connections with our problem since our teams are all converging on a single venues, with multiple fields, for a weekend of games. More closely related is the work of Schaerf [8] on a general Constraint Programming approach. Our secondary objective of minimizing the time between the first and last game of a day is related to the break minimization problem [9,7,4]. Our sharing of fields among all teams is related to the work of [5] though they used a simulated annealing algorithm, and the work of [2].

In addition to being a problem combining temporal as well as spacial constraints, we are looking at a very large number of teams. A large scale soccer tournament for one age group (say 200 teams) can be thought of as perhaps 30 small round robin tournaments, each of which must be scheduled around each other, and fit into the confines of a single weekend. The challenge, then, is to efficiently schedule each team for the desired number of games while maintaining constraints discussed earlier. By efficiently, we mean that, as late as on the Friday or even Saturday morning, it must be possible to add teams and produce an amended schedule in a few seconds or minutes.

2 Moving Toward a Solution

It was decided early on to use constraint programming to work toward a solution. ECLiPSe (`http://eclipse-clp.org/`) was chosen because it easily allows experimentation and prototyping. Once the implementation proved correct, we could always, if needed, re-implement using a faster library. In implementing, these steps were followed:

1. Choose an appropriate model, using only integer values.
2. Input the parameters: number of playing fields, time slots, start time, etc, in a form that is compatible with the data structure and the organizers' experience.
3. Apply as many constraints as possible to limit the search, prior to assigning the variables.
4. Label for the variables, paying special attention to the order in which these variables are chosen.
5. After scheduling the preliminary games, schedule playoff games using time/field slots not used during the preliminary games.
6. Manipulate the results as necessary to present them in a format meaningful to the organizers.

2.1 Failures

2.1.1 The Multi-Dimensional Array

One strategy that was unsuccessful was to simulate a large multi-dimensional array, with a separate dimension for: team number, opponent team number, field number, time, and day. If all the variables coincided, a 1 value would signify that the game was on. Otherwise, the value of the variable was assigned 0.

This solution worked, but not for more than 10 teams. The search space was simply too large (or pruning was too ineffective) to come up with a solution in a reasonable

time. To use this technique with 50 teams, and 15 fields, with 10 time slots over each of 2 days, would require a total of 750,000 variables! And, the objective was to be able to schedule 200+ teams, which would have involved a considerably larger search space.

2.1.2 The 2-Dimensional Array

The second approach involved simulating a 2-dimensional array, each dimension with N entries, where N is the total number of Teams. Each variable in the array would contain either a 0, or, if a game was to be played between those two teams, a game number greater than 0. The game number was encoded to contain both time slot and field information:

$$\lfloor (G-1)\ N \rfloor + 1 = T$$
$$G - (T-1) \times N = F$$

where G is the game number, N is the number of fields, T is the time slot and F is the field number of the game. For example, in the matrix

	Team1	Team 2	Team 3	Team 4
Team 1	0	3	117	230
Team 2	3	0	238	239
Team 3	117	238	0	115
Team 4	15	239	115	0

Teams 1 and 4 will meet on game number 230. Since we have 15 playing fields available and $\lfloor (230-1)/(15) \rfloor + 1 = 16$, the game will be played on time slot 16. Consequently, $230 - (16-1) * N = 5$ which means it will be played on field 5.

A strictly triangular matrix could also be used . This approach also worked, but not for the required number of teams. It was capable of scheduling a tournament with about 50 teams in what we considered a reasonable time.

2.2 Success

It was decided that limiting the overall number of variables might yield better results, so the following model was chosen. The Game Number encoding approach was kept but the matrix was discarded in favour of lists. Note that every variable holds meaningful data (none take the value 0). We introduced two lists of lists of the same size. One for opponent and one for game. For example, the same schedule between teams $1, 2, 3$ and 4 would be encoded as

```
OpponentList = [[2, 3, 4], [1, 3, 4], [1, 2, 4], [1, 2, 3], ...]
GameList = [[3, 117, 237], [3, 238, 239], [117, 238, 115], [15, ...],...]
```

The lists GameList and OpponentList contain all the coded information necessary to represent a game Both Team Numbers, Time, Day, and Field Number. This model was the one which ultimately provided a solution meeting the objectives stated earlier in this paper. Note that there is still some redundancy that could be eliminated but this structure proved to be small enough not to cause memory problems. It could be allocated from the start, as we know exactly how many preliminary games each team

will play and the symmetry constraint (team 1 vs team 2 is identical to team 2 vs team 1) is easy to impose.

The playoff games were scheduled after all preliminaries games were scheduled. The overall structure of the approach was:

```
solve(GameList,OpponentList):-
initialize(GameList,OpponentList,NumGames),
constrain(GameList,OpponentList),
  labeling(OpponentList),!,
ourlabeling(GameList),
outputSchedule(OpponentList,GameList),
schedulePlayoffs(GameList,PlayoffGameList),
outputPlayoffs(PlayoffGameList).
```

After the appropriate model and the encoding that allowed us to have variable domains representing with one integer both the spatial component (the field) and the temporal component (the time slot), the key to success was the search strategy, i.e. the labeling variable order and the value order. Specifically,

- We label the Opponent List first. When backtracking occurs in an effort to find a solution, there is no need to swap opponents around. If 1 is scheduled to play 2, then swapping 2 for 3,is most likely not getting any closer to a solution. So we label the Opponent List first, which is easy, and then focus primarily on getting the Game List right.
- Labeling the games in order by team caused problems: too many backtracks. Instead, the variable ordering was to choose the team that has the most games to be assigned. Followed by the possible opponent that has the most games to be assigned. Assign a game number to those two teams at that point.
- The game is chosen to minimize the time to the last game assigned to the team, while maintaining the appropriate rest period. This is done by pruning domains of the games not yet assigned immediately after a game assignment, an easy constraint given our encoding.
- Note that there is no need for one labeling on time slots and one on fields, with possible ensuing conflicts. The encoding allowed, in a sense, both labellings at the same time. This was also crucial for the success of the approach.

2.2.1 Objective function

Recall that we had two objectives: first to eliminate, if possible Friday games; second to minimize the spread between the first and last game of a team. We aim at this multiple objective by minimizing the time between first and last game of each team, constraining time slots to avoid Fridays. If the time spent backtracking is too large, we restart the search allowing Friday time slots. This is clearly heuristic and it is possible to miss an optimal solution with small game spread and no Friday games. But on the instances we tried, Friday games were never used.

2.3 Efficiency of the Solution

Actual tournament data was run through the application. We had available the data for two tournaments. From these, we synthesised variations, by deleting playing fields,

deleting time slots, adding teams here and there more or less haphazardly to stretch the implementation and test its robustness.

For the largest real tournament instance, comprised of 227 teams in the 11v11 bracket, the manually produced schedule used 23 fields, Friday evening games were scheduled and some teams had to wait up to 10 hours between weekend games in a given day. We now comment on the automated solution achieved for this instance and whether the objectives were met.

1. Our ECLiPSe implementation was able to come up with a solution where the maximum time between games was 4 time slots. This is a considerable improvement over the 8,9, or 10 time slots that teams commonly have to contend with.
2. In the largest instance solved, 343 games were scheduled across 19 time slots using 19 fields (361 possible events). A clear improvement on the 23 fields required by the manual solution. Theoretically, one needs $\lfloor 343/19 \rfloor = 19$ fields. Therefore 19 fields is the absolute minimum that can be used. After scheduling the preliminary games field utilization was just over 95%. In addition, due to the value ordering of our encoding, most of the unused slots were at the end of the preliminary game schedule and were used later to schedule playoff games.
3. Playoff scheduling was efficient and made optimal use of the remaining time slots. After filling the unused preliminary slots with playoff games, only 1 slot could not be used, for an efficiency of 360/361 or over 99%.
4. It was not necessary to schedule any games on Friday evening.
5. Execution time was below 10 seconds on an average PC.
6. One unintended benefit is that each age group, since they were assigned team numbers close to each other, tended to play similar schedules. So a coach of several teams across different age groups has an excellent chance of attending all the games. It is now conceivable, through applying additional constraints, to ensure this additional constraint because the current solution technique is fast enough.

3 Conclusion

We defined a scheduling problem modeling school-age children soccer tournaments in Michigan. We suspect the problem is almost identical elsewhere in the country. The problem has temporal and spatial constraints, both soft and hard, and is fairly large scale, as far as sport schedules are concerned. The first key element of the implementation is an encoding that allows scheduling time and place of a game with one instantiation, allows efficient pruning of variable domains for hard constraints and also allows simple minimization of violation of soft constraints. The second key element is a search based on a dynamic choice of the variables ordered according to the number of games yet to be scheduled.

Experiments with real data from two tournaments shows that our implementation in ECLiPSe, can do an excellent job of scheduling large scale soccer tournaments; much better, for the instances tested, than the manual schedules that were used. We are now in a position to add a number of secondary constraints and further improve the tournament experience for all participants, players, coaches, families and friends.

References

1. Easton, K., Nemhauser, G., Trick, M.: CP based branch-and-price. In: Constraint and integer programming, *Oper. Res./Comput. Sci. Interfaces Ser.*, vol. 27, pp. 207–231. Kluwer Acad. Publ., Boston, MA (2004)
2. Hamiez, J.P., Hao, J.K.: Using solution properties within an enumerative search to solve a sports league scheduling problem. Discrete Appl. Math. **156**(10), 1683–1693 (2008). DOI 10.1016/j.dam.2007.08.019. URL http://dx.doi.org/10.1016/j.dam.2007.08.019
3. Kendall, G., Knust, S., Ribeiro, C.C., Urrutia, S.: Scheduling in sports: an annotated bibliography. Comput. Oper. Res. **37**(1), 1–19 (2010). DOI 10.1016/j.cor.2009.05.013. URL http://dx.doi.org/10.1016/j.cor.2009.05.013
4. Knust, S.: Scheduling sports tournaments on a single court minimizing waiting times. Oper. Res. Lett. **36**(4), 471–476 (2008). DOI 10.1016/j.orl.2007.11.006. URL http://dx.doi.org/10.1016/j.orl.2007.11.006
5. Lim, A., Rodrigues, B., Zhang, X.: Scheduling sports competitions at multiple venues—revisited. European J. Oper. Res. **175**(1), 171–186 (2006). DOI 10.1016/j.ejor.2005.03.029. URL http://dx.doi.org/10.1016/j.ejor.2005.03.029
6. Nemhauser, G., Trick, M.: Scheduling a major college basketball conference. Operations Research **46**, 1–8 (1997)
7. Régin, J.C.: Minimization of the number of breaks in sports scheduling problems using constraint programming. In: Constraint programming and large scale discrete optimization (Piscataway, NJ, 1998), *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.*, vol. 57, pp. 115–130. Amer. Math. Soc., Providence, RI (2001)
8. Schaerf, A.: Scheduling sport tournaments using constraint logic programming. Constraints **4**(1), 43–65 (1999). DOI 10.1023/A:1009845710839. URL http://dx.doi.org/10.1023/A:1009845710839
9. Suzuka, A., Miyashiro, R., Yoshise, A., Matsui, T.: The home-away assignment problems and break minimization/maximization problems in sports scheduling. Pac. J. Optim. **3**(1), 113–133 (2007)
10. de Werra, D.: Scheduling in sports. In: Studies on graphs and discrete programming (Brussels, 1979), *Ann. Discrete Math.*, vol. 11, pp. 381–395. North-Holland, Amsterdam (1981)
11. de Werra, D.: On the multiplication of divisions: the use of graphs for sports scheduling. Networks **15**(1), 125–136 (1985). DOI 10.1002/net.3230150110. URL http://dx.doi.org/10.1002/net.3230150110
12. de Werra, D.: Some models of graphs for scheduling sports competitions. Discrete Appl. Math. **21**(1), 47–65 (1988). DOI 10.1016/0166-218X(88)90033-9. URL http://dx.doi.org/10.1016/0166-218X(88)90033-9
13. de Werra, D., Jacot-Descombes, L., Masson, P.: A constrained sports scheduling problem. Discrete Appl. Math. **26**(1), 41–49 (1990). DOI 10.1016/0166-218X(90)90019-9. URL http://dx.doi.org/10.1016/0166-218X(90)90019-9