Minimizing Breaks by Maximizing Cuts*

Matthias Elf[‡†] Michael Jünger[‡] Giovanni Rinaldi[⋆]

[‡]Institut für Informatik, Universität zu Köln, Cologne, Germany
[⋆]Istituto di Analisi dei Sistemi ed Informatica del CNR, Rome, Italy

Abstract

We propose to solve the break minimization problem in sports scheduling by transforming it into a maximum cut problem in an undirected graph and applying a branch-and-cut algorithm. Our approach outperforms previous approaches with constraint programming and integer programming techniques.

Keywords: sports scheduling, maximum cut, break minimization

1 Introduction

During the Seminar "Constraint Programming and Integer Programming", Schloß Dagstuhl, Germany, 17–21 January 2000, the participants tried to identify problems for which a fruitful interaction/competition of constraint programming techniques and integer/combinatorial optimization techniques appears challenging and is likely to enhance the interaction of both communities. One problem area the participants agreed upon consists of various feasibility/optimization problems occuring in scheduling sports tournaments. The break minimization problem for sports leagues was addressed by both communities during the workshop, in particular by Jean Charles Régin of the constraint programming community and by Michael Trick of the integer programming/combinatorial optimization community.

In section 2, we will define the break minimization problem and give a summary of current solution approaches. In section 3, we will show how the problem can be transformed to a maximum cut problem in an undirected graph with edge weights. In section 4, we will report on computational results on random and real world instances. It will turn out that our approach is applicable to problem instances of size beyond those that could be handled previously

*Partially supported by the Future and Emerging Technologies Programme of the European Union under contract number IST-1999-14186 (ALCOM-FT).

Work carried out as part of DONET (Discrete Optimization Network) TMR project no. ERB FMRX-CT98-0202 of the European Union.

[†]Corresponding author:
Matthias Elf, Institut für Informatik, Pohligstraße 1, D-50969 Cologne, Germany
email: elf@informatik.uni-koeln.de

by constraint programming as well as integer programming/combinatorial optimization techniques. In section 5, we will discuss our results and open questions that we hope to resolve in the future.

2 Break Minimization

We deal with the situation where in a sports league consisting of an even number n of teams each team plays each other team once in $n-1$ consecutive weeks. Such tournaments are called *round robin tournaments*. Each game is played in one of the two opponents home towns, such that the following restrictions apply to each *feasible schedule*:

[FS1] For each team, the teams played in weeks $1, \ldots, n-1$ are a permutation of all other teams.

[FS2] If in week w team i plays team j "at home" ("+") then team j plays team i in week w in i's town, i.e. "away" ("−").

We also consider feasible schedules without home-away assignments. These are schedules satisfying [FS1] and [FS2] where the property of being a home or away game has been omitted.

Figure 1 shows two possible schedules for a league of eight teams. The rows show the game plan for each team, column 1 displays a team, columns $w \in \{2, \ldots, n\}$ show the opponent in week $w-1$, "+", and "−", respectively, indicate if the game is at home or away.

In sports scheduling it is considered undesirable if any team plays two consecutive games either both at home or both away. Such a situation is called a *break*. The schedule of Figure 1(a) imposes 8 breaks whereas the schedule in Figure 1(b) imposes only 6 breaks.

Schreuder [10] has shown that, for an even number n of teams, it is always possible to construct a schedule with $n-2$ breaks and that this number is minimum. He has given an efficient algorithm to compute such a schedule. However, sport tournament schedules are subject to a number of requirements, among them restrictions such as "geographically close teams should not play at home during the same week". Often these requirements are mandatory or at least more important than having a minimum number of breaks. Therefore, optimum $(n-2)$-break schedules become unfavorable. Different strategies have been proposed to find schedules matching all requirements with a small number of breaks. Some authors (like Schreuder [10]) propose to start with an optimal schedule with $n-2$ breaks and incorporate the additional requirements at the cost of more breaks. Others (like Régin [8, 9] and Trick [11]) propose to consider a schedule without home-away assignment that obeys the various (often not formally describable) side conditions and compute a home-away assignment as to minimize the number of breaks. It is the latter attitude we take here:

Break Minimization Problem: *We are given a feasible tournament schedule without home-away assignment and our task is to find a home-away assignment that minimizes the number of breaks.*

$$
\begin{array}{lccccccc}
1: & +2 & -3 & -4 & +5 & +6 & -7 & +8 \\
2: & -1 & +7 & -5 & +4 & -3 & -8 & +6 \\
3: & -7 & +1 & +8 & -6 & +2 & +5 & -4 \\
4: & +5 & -8 & +1 & -2 & +7 & -6 & +3 \\
5: & -4 & -6 & +2 & -1 & +8 & -3 & +7 \\
6: & +8 & +5 & -7 & +3 & -1 & +4 & -2 \\
7: & +3 & -2 & +6 & -8 & -4 & +1 & -5 \\
8: & -6 & +4 & -3 & +7 & -5 & +2 & -1
\end{array}
$$

(a)

$$
\begin{array}{lccccccc}
1: & +8 & +3 & -5 & +7 & -2 & +4 & -6 \\
2: & +7 & -8 & +4 & -6 & +1 & -3 & +5 \\
3: & +6 & -1 & +8 & +5 & -7 & +2 & -4 \\
4: & -5 & +7 & -2 & -8 & +6 & -1 & +3 \\
5: & +4 & -6 & +1 & -3 & +8 & +7 & -2 \\
6: & -3 & +5 & -7 & +2 & -4 & -8 & +1 \\
7: & -2 & -4 & +6 & -1 & +3 & -5 & +8 \\
8: & -1 & +2 & -3 & +4 & -5 & +6 & -7
\end{array}
$$

(b)

Figure 1: Two feasible schedules for eight teams

The home-away assignments for the feasible schedules of Figure 1 are both optimum solutions for the corresponding instances of the break minimization problem.

Régin [8] formulated a constraint programming model with 0-1-variables and was able to solve instances up to size 20. Trick [11] introduced an integer programming formulation and was able to solve instances up to size 22.

The complexity status of the break minimization problem has not yet been determined (to the best of our knowledge), we believe it is NP-hard (see section 5).

3 From Minimizing Breaks to Maximizing Cuts

In this section we model the break minimization problem as the problem of finding a maximum capacity cut in a specific graph. The modeling is as follows: We are given a feasible schedule

$$
\begin{array}{lcccc}
1: & t_{11} & t_{12} & \dots & t_{1,n-1} \\
2: & t_{21} & t_{22} & \dots & t_{2,n-1} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
n: & t_{n1} & t_{n2} & \dots & t_{n,n-1}
\end{array}
$$

without home-away assignment in which $t_{ij} \in \{1, 2, \ldots, n\}$ is the opponent of team i in week j. Our task is to decide whether the teams t_{ij} play either at home or away. The number of breaks in the resulting home-away assignment must be minimum.

From the schedule above we construct an undirected graph $G = (V, E)$. It contains nodes $v = (i, j) \in V$ for $i \in \{1, 2, \ldots, n\}$ and $j \in \{1, 2, \ldots, n-1\}$ that correspond exactly to the opponents t_{ij} in the schedule. For every possible break, i.e., for each pair of opponents in consecutive weeks we introduce an edge. More precisely, there is an edge in E between nodes (i, j) and (k, l) in V if and only if $i = k$ and $1 \leq j = l - 1 \leq n - 1$. I.e., $G = (V, E)$ is as follows:

$$\begin{array}{ccccc}
(1,1) & \text{———} & (1,2) & \text{———} \cdots \text{———} & (1,n-1) \\
(2,1) & \text{———} & (2,2) & \text{———} \cdots \text{———} & (2,n-1) \\
\vdots & & \vdots & \ddots & \vdots \\
(n,1) & \text{———} & (n,2) & \text{———} \cdots \text{———} & (n,n-1)
\end{array}$$

Definition *A* cut $C \subseteq E$ *in a graph* $G = (V, E)$ *is an edge set of the form* $C = \delta(V^+)$*, where* $V^+ \subseteq V$ *and* $\delta(V^+) = \{e \in E \mid v \in V^+,\ w \in V \setminus V^+\}$*. The cut partitions* V *into* $V = V^+ \cup V^-$ *(*$V^+ \cap V^- = \emptyset$*), where* V^+ *and* V^- *are called the different* shores *of the cut. If the graph* G *is weighted with edge weights* $u \in {}^E$ *then the* weight *of a cut* C *is defined as* $u(C) := \sum_{e \in C} u_e$ *.*

Any home-away assignment partitions the node set V of the model graph G into two sets V^+ and V^-, with nodes in V^+ for home games and nodes in V^- for away games. The sets V^+ and V^- are the shores of the cut $\delta(V^+)$ that contains exactly the edges of G which correspond to non-breaks. All other edges correspond to breaks. The number of breaks is $|E| - |\delta(V^+)|$ which means that we must maximize $|\delta(V^+)|$ in order to minimize the number of breaks.

Conversely, any cut in the model graph G partitions the nodes into $V = V^+ \cup V^-$. This partitioning defines a home-away assignment if and only if it satisfies property [FS2], i.e., $(i, j) \in V^+$ if and only if $(t_{ij}, j) \in V^-$.

Thus, we can solve the break minimization problem by finding a maximum cardinality cut if we can disregard cuts not satisfying condition [FS2]. We achieve this by weighting the edges of the model graph and adding additional edges with large weight that force nodes (i, j) and (t_{ij}, j) to belong to different shores of any cut. The model graph is modified in the following way.

We assign a capacity of 1 to all edges of G introduced so far. For all $\frac{n(n-1)}{2}$ pairs of nodes (i, j) and (t_{ij}, j) we create an edge with a capacity of $M \geq n(n-2)+1$. Figure 2 shows the graph G resulting from this transformation for the instance given in Figure 1(a).

Lemma *The shores of a maximum weighted cut of the model graph* G *define a home-away assignment with the minimum number of* $n(n-2) + \frac{n(n-1)}{2} M - \Delta$ *breaks, where* Δ *is the weight of the cut.*

Proof. Because each cut corresponding to a home-away assignment respects condition [FS2] it contains all edges with weight M. Therefore, its weight is at least $\frac{n(n-1)}{2} M$ and the maximum capacity cut has at least its weight. Due to

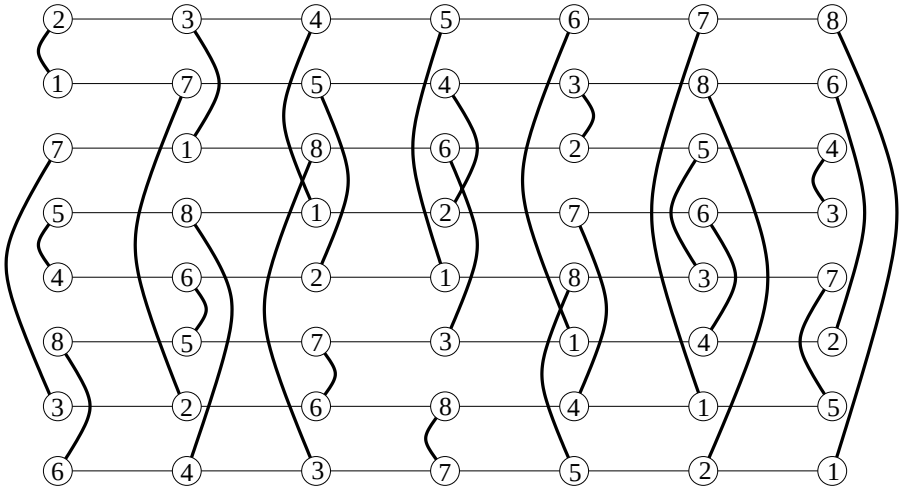

Figure 2: Result of the big-M transformation. Bold edges have weight M, thin edges have weight 1

the choice of $M \geq n(n-2)+1$ every cut with weight at least $\frac{n(n-1)}{2}M$ must contain all edges with weight M and therefore respects [FS2]. It follows that all cuts corresponding to home away assignments have larger capacity than those that do not define a home-away assignment. In particular, the maximum cut defines a home-away assignment.

Suppose that a cut in G respects [FS2] and has value Δ. Then it contains $\Delta - \frac{n(n-1)}{2}M$ many edges with weight 1. Those edges define non-breaks in the corresponding home-away assignment. Therefore, finding the maximum capacity cut in G results in a cut defining a home-away assignment with the maximum number of $\Delta - \frac{n(n-1)}{2}M$ non-breaks. This is a home-away assignment with the minimum number of $n(n-2) + \frac{n(n-1)}{2}M - \Delta$ breaks.

□

From the theoretical point of view the previous lemma tells us that we can apply a maximum cut solver to our model graph in order to solve the break minimization problem. From the computational point of view we can do much better, because we can transform the model graph into an equivalent maximum cut instance without big-M edges and with only half of the nodes of the original graph. The transformation uses the following result of Barahona and Mahjoub [2].

Definition *Let $G = (V, E)$ be a graph and $v \in V$ a node of G. The set of edges star$(v) := \{(v, u) \in E \mid u \in V\}$ incident to v is called the* star *of v.*

Theorem (Barahona/Mahjoub [2]) *Let $G = (V, E)$ be a graph with edge capacities $u \in {}^{E}$ and let V^+ and V^- be the shores of a maximum capacity cut. If we take a node $v \in V^+$ and change the signs of the capacities of all edges in star(v) then $V^+ \setminus \{v\}$ and $V^- \cup \{v\}$ are the shores of a maximum capacity cut in G with the new capacities.*

We transform our model graph as follows. Take an arbitrary big-M edge $e_M \in E$ and switch the signs of the weights of the edges in the star of one of

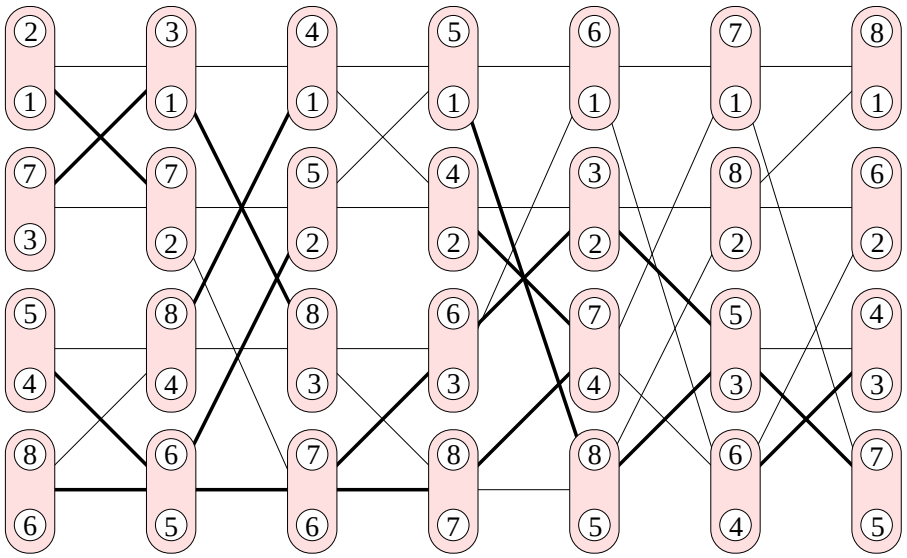

Figure 3: Result of the transformation

its end nodes. After this operation the graph contains one edge with capacity $-M$, some edges with capacity M, and $n(n-1)$ edges with capacity either 1 or -1. Due to the choice of the value M a maximum cut in the transformed graph does not contain e_M. By reversing the switching operation we get a maximum cut of the original graph from a maximum cut in the transformed graph. Since a maximum cut in the transformed graph does not contain e_M we can contract e_M, i.e., identify its end nodes as one node and delete the edge e_M. This results in a graph with $n(n-1)-1$ nodes, $n(n-2)$ edges of capacity either -1 or 1, and $\frac{n(n-1)}{2}-1$ edges with capacity M.

The transformation described above can repeatedly be applied $\frac{n(n-1)}{2}$ times to the big-M edges in the resulting graphs until there are no big-M edges. We obtain a maximum cut instance with $\frac{n(n-1)}{2}$ nodes and $n(n-2)$ edges with capacities either 1 or -1. The result for our example is shown in Figure 3, in which we have (arbitrarily) chosen to switch the cut of the lower indexed vertex each time.

The same graph is displayed in Figure 4 along with a cut of maximum capacity 22 that is indicated by white and grey node colors.

By backward transformation, this cut corresponds to the home-away assignment displayed in Figure 1(b) with 8 breaks, which proves our claim in section 1 that 8 breaks is minimum for this instance.

4 Computational Results

For the computation of maximum capacity cuts we have used an implementation of the algorithm described in [3] that was reimplemented by Martin Diehl using the ABACUS software [6], version 2.3 using CPLEX 6.5 as an LP-Solver. The

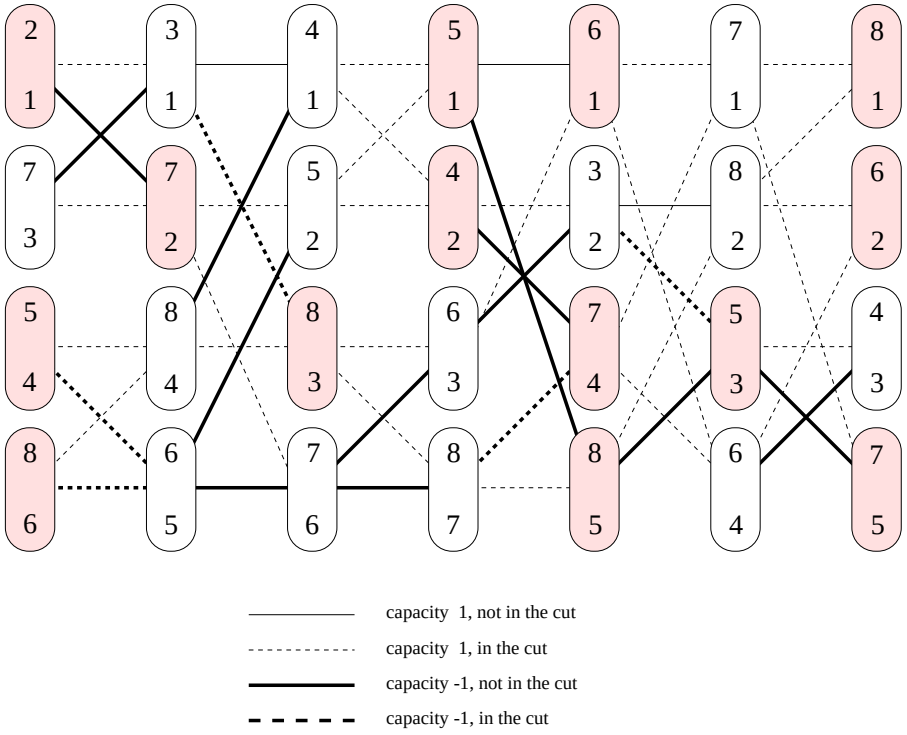

Figure 4: A cut of maximum capacity 22

same implementation was used successfully in, e.g., [4] for computing ground states of Ising spin glasses.

The instances were created by computing optimal ($(n-2)$-breaks schedules) by Schreuder's procedure and permuting the columns randomly. For each size, we created five random schedules and applied the branch-and-cut algorithm. The results displayed in Table 1 were obtained on a 296MHz Sun Ultra SPARC machine. Times are given in CPU seconds where 0.0 means that the execution time was below the system's accuracy for process times.

We also solved one real world instance, namely the Bundesliga 1999/2000 (first national German soccer league) instance. There are 18 teams and we found that the schedule is already optimal at 16 breaks. The Bundesliga is played in two rounds. Each round is a round-robin tournament. For our experiments we took the first round. The second round contains the same schedule where the properties of playing at home or away are switched.

5 Discussion

We do not have access to the instances used in the computational studies by Régin and Trick, however, it seems that Trick's approach is slightly ahead in solving 22 teams instances in about 1 hour on a 266 MHz machine (see [11]). Apparently, our approach is able to handle larger instances easily. Trick [12] reports that our random instances feature many more breaks than those he used in his computations. Additionally, he reports that our instances are much

size	time in CPU seconds			number of breaks		
n	minimum	average	maximum	minimum	average	maximum
4	0.0	0.0	0.0	2	2.0	2
6	0.0	0.0	0.0	4	4.0	4
8	0.0	0.0	0.0	8	8.0	8
10	0.0	0.0	0.0	10	12.2	12
12	0.1	0.1	0.1	14	16.8	18
14	0.1	0.3	0.5	18	23.6	26
16	0.4	1.0	2.8	28	31.2	32
18	2.8	6.1	19.0	36	40.8	44
20	1.1	8.7	18.5	44	51.2	54
22	6.8	36.7	94.0	58	61.2	64
24	23.5	72.8	140.6	68	72.4	74
26	22.9	339.0	1215.9	80	85.2	90

Table 1: Computational Results

harder to solve using his approach. We also recognized an increase of running time if the number of breaks is large.

There remain three open questions we would like to answer in the future:

1. What is the complexity of the break minimization problem? (We conjecture it is NP-hard.)

2. When we give $(n-2)$-break instances to our algorithm, the answer $n-2$ is produced very quickly. We would like to establish why this is the case. (We conjecture this is because the corresponding maximum cut instances feature graphs that are not contractible to K_5, a class of graphs for which Barahona [1] showed that the maximum cut problem is solvable in polynomial time.)

3. What is a random schedule? Is our approach to the random instance generation justified?

Acknowledgement

We would like to thank Michael Trick for teaching us the break minimization problem and sharing his insights with us.

References

[1] F. Barahona (1983), *The max-cut problem on graphs not contractible to* K_5, Operations Research Letters **2**, 107–111.

[2] F. Barahona and A. R. Mahjoub (1986), *On the cut polytope*, Mathematical Programming **36**, 157–173.

[3] F. Barahona, M. Grötschel, M. Jünger, and G. Reinelt (1988), *An application of combinatorial optimization to statistical physics and circuit layout design*, Operations Research **36**, 493–513.

[4] C. De Simone, M. Diehl, M. Jünger, P. Mutzel, G. Reinelt, and G. Rinaldi (1986), *Exact ground states of 2-dimensional $\pm J$ Ising spin glasses*, Journal of Statistical Physics **84**, 1363–1371.

[5] M. Henz (1999), *Scheduling a major college basketball conference–revisited*, Technical Note, School of Computing, National University of Singapore.

[6] M. Jünger and S. Thienel (2000), *The ABACUS System for Branch and Cut and Price Algorithms in Integer Programming and Combinatorial Optimization*, Software Practice and Experience **30**, 1325–1352, 2000.

[7] G. L. Nemhauser and M. A. Trick (1998), *Scheduling a major college basketball conference*, Operations Research **46**, 1–8.

[8] J.-C. Régin (1998) *Minimization of the number of breaks in sports scheduling problems using constraint programming*, Talk presented at DIMACS Workshop on Constraint Programming and Large Scale Discrete Optimization, Sep. 14–19, 1998.

[9] J.-C. Régin (2000), *Modelling with constraint programming*, Talk presented at Dagstuhl Seminar on Constraint Programming and Integer Programming, Jan. 17–21, 2000.

[10] J. A. M. Schreuder (1992) *Combinatorial aspects of construction of competition Dutch Professional Football Leagues*, Discrete Applied Mathematics **35**, 301–312.

[11] M. A. Trick (2000) *A schedule-then-break approach to sports timetabling*, Proceedings of the third PATAT Conference 2000, Lecture Notes in Computer Science **2079**, 242–253, Springer, 2000

[12] M. A. Trick (2000) *Personal communication*, 2000