Turnajové rozlosování s rozhodčími

Kombinatorická optimalizace - A4M35KO
Michal Vlček <vlcekmi3@fel.cvut.cz>
ČVUT FEL - 2014

Zadání a kategorizace problému

Mějme následující problém: Chcete uspořádat turnaj pro více týmů, které se utkají mezi sebou. Naším úkolem je vytvořit rozlosování, popřípadě týmy ještě rozdělit do několika skupin a pro ně vytvořit rozlosování.

Jako téma pro semestrální práci jsem si tedy vybral téma ze skupiny rozvrhovacích („timetable“) problémů. Přesněji se jedná o problém sportovního rozlosování. V odborné literatuře nalezitelný pod názvem *sport league scheduling problem*. V něm se typicky snažíme najít optimální rozlosování pro určitý počet týmů splňující určité podmínky, neboli omezení (níže).

Co se mého zadání týká, tak cílem bylo udělat klasický turnajový rozvrh a do něho navíc zahrnout i rozlosování rozhodčích pro jednotlivé zápasy. Pod rozhodčím si představte některého hráče z účastněných týmů (u turnajů podobného formátu bývá přátelská atmosféra a tak toto opatření není problémem). Čili namísto shánění externích rozhodčích bude zápas pískat vždy někdo z týmů, které aktuálně nehrají.

Omezující podmínky

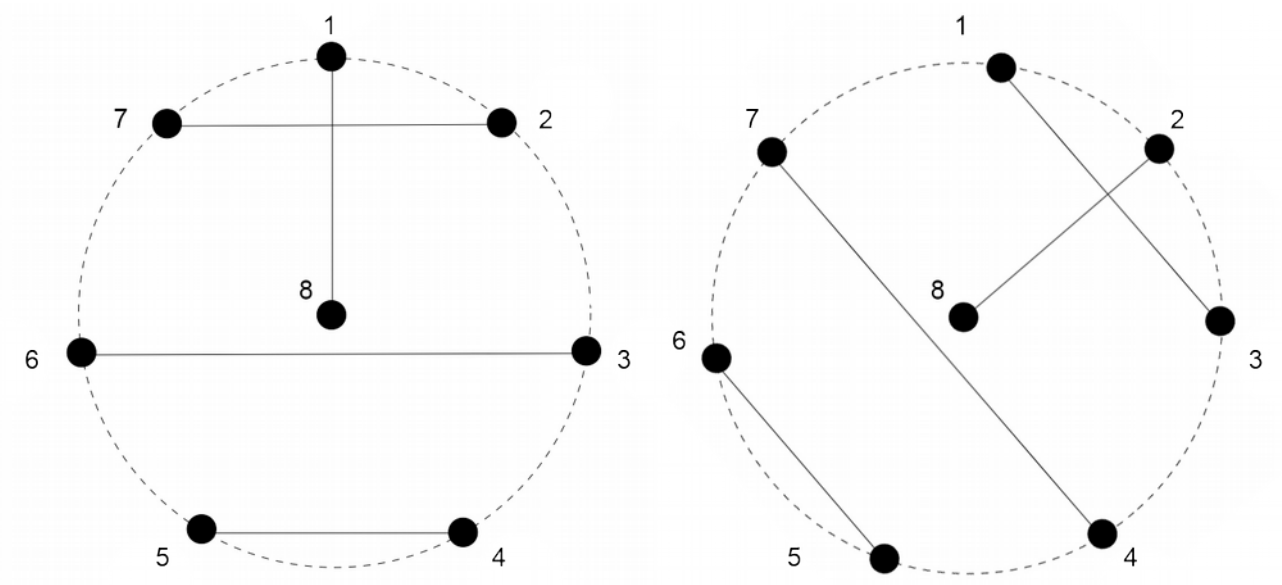
- Sudý počet týmů
- Každý tým hraje s každým týmem přesně a pouze jednou
- Počet zápasů, který tým hraje tzv. „v řadě“ je roven maximálně 2 zápasům
- Pro každý zápas je přiřazen rozhodčí, který je z týmu který právě nehraje
- Počet odpískaných zápasů pro týmy musí být vybalancovaný (tzn. Nesmí se stát že by jeden tým odpískal 5 zápasů a druhý ani jeden)

Související práce

Rozlosování sportovní ligy je obecně rozšířený problém, který řeší kdejaká sportovní organizace. Při rešeršním vyhledávání souvisejících prací jsem narazil na několik klíčových slov, která jsou s mou problematikou nějak spjata: *sports league scheduling, travelling tournament problem, timetable problem, simple round robin schedule, double round robin tournament, single team problem*, aj.

Existuje několik metod typicky používaných v těchto úlohách: Integer Linear Programming (ILP), Constraint Programming (CP) a hladové algoritmy. ILP a CP patří do tzv. úplných metod, kdy je prohledáván celý prostor řešení. Tyto metody vracejí lepší výsledky avšak pro větší instance jsou výpočetní časy veliké, a někdy se ani nedopátráme výsledku (např. pro instanci o 8 týmech při omezeních týkající se americké baseballové ligy [1]). Pro jednodušší instance (obyčejný turnaj) se dají použít kombinatorické přístupy jako např. metoda kruhu („*circle method*") [2].

V přednášce [3] je ukázáno, že existuje „round robin" (klasický turnaj) rozlosování pro každou instanci o *2n* týmech. Lze to ukázat právě na metodě kruhu (obr. 1), kde si očíslujeme týmy *1* až *2n*. A pro každé „kolo" turnaje získáme rozlosování následovně: pro kolo značené *i dostaneme zápasy i vs 2n*, a zbytek A *vs B pro A+B = 2i (mod 2n-1).* V obrázku si to lze lépe představit. Prvně si zvolíme hrany jako na obrázku, a pak je jen otáčíme. Každé otočení je další kolo turnaje.



Obrázek 1: Circle method

Článek [4] se částečně věnuje mé problematice, obsahuje matematický popis pro základní omezení, která budu využívat, a tak jsem z něho vycházel při tvorbě modelu.

Popis řešení

K řešení využiji *ILP* metodu za použití prostředí *MATLAB* s *glpk* solverem. Budu uvažovat rozdělení pouze do jedné skupiny, kde budou všechny týmy. Rozlosování pro více skupin by se dělalo téměř identicky, vytvořil by se rozvrh pro jednu skupinu a ten by se kopíroval do ostatních. Co se týče rozhodčích, tak kdybychom chtěli, aby pískali jinou skupinu než tu, v které hrají, tak bychom k zápasu, který se hraje ve skupině G, přiřadili rozhodčího ze skupiny G-1 (modulo počtu skupin).

Výsledek řešení si je možno představit jako tabulku 1, kde zápasy ***n*** týmů rozdělím do tzv. kol. Kolo obsahuje zápasy kde se týmy vyskytují jedenkrát, zápasů v kole je ***n/2*** (utvoří dvojice) a kol je ***n - 1*** (je n-1 soupeřů). Pomocí této konstrukce si poté pomůžu při aplikování dalších omezení, o tom až níže.

Kolo 1	**Kolo 2**	**Kolo 3**	**Kolo 4**	**Kolo 5**	**Kolo 6**	**Kolo 7**
1 vs 2	1vs 3	5 vs 8	4 vs 7	4 vs 8	2 vs 6	3 vs 5
3 vs 4	2 vs 8	1 vs 4	6 vs 8	2 vs 5	1 vs 7	6 vs 7
5 vs 6	4 vs 6	2 vs 7	1 vs 5	3 vs 7	3 vs 8	1 vs 8
7 vs 8	5 vs 7	3 vs 6	2 vs 3	1 vs 6	4 vs 5	2 vs 4

Tabulka 1: Příklad rozlosování pro počet týmů n = 8

Proměnné

Jako proměnnou jsem definoval X_{rijkm}, která nabývá hodnot 1 nebo 0. Proměnná se rovná 1 pokud tým *i* hraje s týmem *j* v tabulce na pozici *k* a *m* a píská je rozhodčí z týmu *r*, jinak se proměnná rovná 0.

Kriteriální funkci zde neuvažuji, nebo je rovna nule. Úlohu jsem specifikoval jako rozhodovací, buď rozlosování naleznu, nebo ne.

Omezující podmínky

- Každý slot (prvek v tabulce) obsahuje přesně jeden zápas.

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{r=1}^{n} X_{rijkm} = 1 \quad \forall k \forall m$$

- Každý tým hraje s ostatními týmy přesně jeden zápas.

$$\sum_{k=1}^{n/2} \sum_{m=1}^{n-1} \sum_{r=1}^{n} X_{rijkm} = 1 \quad \forall i \forall j, i<j$$

- Omezení na „kola“ - každý tým hraje v kole (sloupci) *m* jeden zápas. Tímto omezením zaručíme, že tým bude hrát maximálně 2 zápasy v řadě. V rámci kola se totiž utká pouze jednou. Může ale nastat situace, kdy hraje poslední zápas předchozího kola a poté hned první zápas následujícího kola. V tuto chvíli hraje tým 2 zápasy v řadě, což je nejhorší možná situace, která může vzniknout.

$$\sum_{k=1}^{n/2} \sum_{j=i+1}^{n} \sum_{r=1}^{n} X_{rijkm} = 1 \quad \forall i \forall m$$

- Přiřazení rozhodčích k zápasům

$$\sum_{k=1}^{n/2} \sum_{m=1}^{n-1} \sum_{r=1}^{n} X_{rijkm} = 1 \quad \forall i \forall j, r \neq i, r \neq j$$

- Vybalancování přiřazených rozhodčích. Balancovaný počet odpískaných zápasů lze dopočítat. Pro **n** týmů, bude polovina týmů pískat ***n/2***–krát a druhá polovina bude pískat o jeden zápas méně – ***n/2-1***-krát. (Př. n = 8, 7*4 = 28 zápasů, 4x 4 pískání + 4x 3 pískání = 28 odpískaných zápasů)

$$\sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{k=1}^{n/2} \sum_{m=1}^{n-1} X_{rijkm} \leq \frac{n}{2} \quad \forall r \quad , \quad \sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{k=1}^{n/2} \sum_{m=1}^{n-1} X_{rijkm} \geq \frac{n}{2} - 1 \quad \forall r$$

Výsledný rozpis z tabulky pak stačí přečíst po sloupcích a dostaneme seznam všech po sobě jdoucích zápasů.

V mém řešení ILP modelu vypadala struktura sloupců matice *A* následovně: Příklad ukáži pro instanci *n = 4*. Spočetl jsem si počet hracích míst (slotů), což jsou rozměry té výsledné tabulky *3 x 2 = 6 slotů*, poté jsem si vytvořil všechny možné dvojice – zápasy - (4 *nad 2* možností = 6 zápasů), a ty označil čísly. V každém slotu tyto zápasy budou figurovat a z nich budeme vybírat. A u každého zápasu chceme vybrat rozhodčího ze 4 různých týmů. Struktura je vyobrazena v tabulce 2 pod odstavcem (prvně jsou kolonky rozhodčích prázdné – nedostatek místa, v detailu je již vše znázorněno korektně).

Sloty:	1						2						3						4						5						6					
Zápasy:	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6	1	2	3	4	5	6
Rozhodčí:																																				

Sloty:	1																							
Zápasy:	1				2				3				4				5				6			
Rozhodčí:	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4

Tabulka 2: Struktura sloupců matice A + detail 1. slotu (dole)

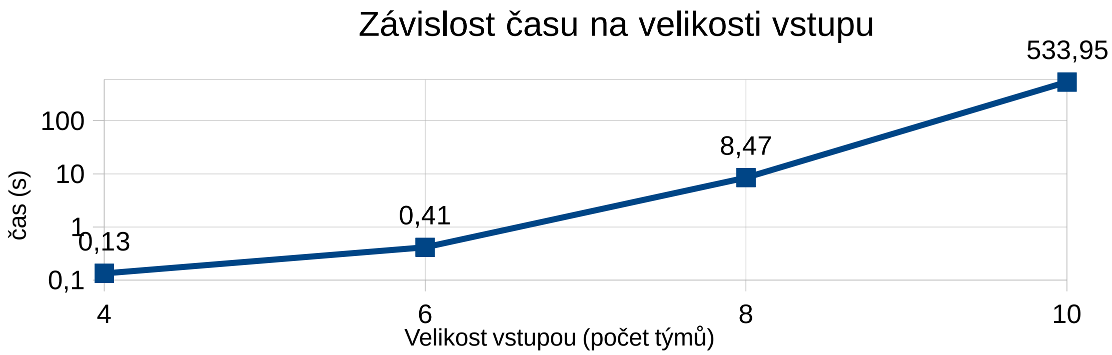
Sloupce matice ukazuji ve 3 řádcích, to slouží pouze pro lepší ilustraci indexů. Skutečný rozměr je v posledním řádku, obsahuje tedy 6x6x4 = 144 hodnot. První index značí, že se hraje na prvním slotu, zápas označený jako 1 (např. 1 vs 2) a píská rozhodčí z týmu 1.

Experimenty

Při experimentování jsem měnil velikost vstupního parametru, tedy počet týmů z kterých se má rozlosování udělat a sledoval jak se s tím vypořádá solver. Všechny experimentální instance jsou uvedeny v tabulce 3 pod odstavcem. Sloupec *„Čas“* obsahuje časový údaj o řešení problému solverem. Sloupec „Paměť“ udává paměťovou náročnost běhu solveru, tzn. součet velikostí použitých proměnných/datových struktur. V grafu 1 jsou znázorněny časové údaje v závislosti na velikosti vstupu (časová osa je v logaritmickém měřítku).

Počet týmů	Velikost výsl. vektoru	Čas	Paměť
4	144	0.1335 s	51.39 kb
6	1 350	0.4142 s	966.91 kb
8	6 272	8.4688 s	7.67 Mb
10	20 250	533.9472 s	38.51 Mb

Tabulka 3: Paměťové a časové složitosti experimentálních instancí



Graf 1: Závislost času na velikosti vstupu

Na závěr uvedu ukázku rozlosování pro instanci o velikosti šesti týmů.

1 vs 4 - (rozhodčí: 3)
2 vs 5 - (rozhodčí: 3)
3 vs 6 - (rozhodčí: 4)
1 vs 6 - (rozhodčí: 2)
2 vs 3 - (rozhodčí: 6)
4 vs 5 - (rozhodčí: 2)
1 vs 5 - (rozhodčí: 4)
3 vs 4 - (rozhodčí: 5)
2 vs 6 - (rozhodčí: 4)
1 vs 3 - (rozhodčí: 6)
2 vs 4 - (rozhodčí: 1)
5 vs 6 - (rozhodčí: 1)
1 vs 2 - (rozhodčí: 5)
3 vs 5 - (rozhodčí: 6)
4 vs 6 - (rozhodčí: 5)

Závěr

K této semestrální práci jsem si zvolil vlastní téma, které jsem chtěl prozkoumat a dozvědět se o něm něco více. To se mi nakonec povedlo, našel jsem několik skvělých článků od kterých jsem se mohl odrazit a využít je k sestavení základního modelu. S dalšími specifičtějšími požadavky jsem se již popasoval sám, a tak vzniklo řešení, které by se dalo po mírných úpravách reálně využít.

U tohoto problému lze vidět tzv. „kombinatorickou explozi“, kde se při malé změně vstupu rapidně zvýší výpočetní složitost, to si troufám tvrdit není takový problém, protože v praxi si dokáži představit instanci pro 4 – 8 týmů v jedné skupině a pro tu je výpočet vcelku rychlý. Po implementaci jsem objevil pár míst ke zlepšení, několik omezení by šlo skloubit do jediného a zadat najednou do jednoho řádku matice, tím by se nám snížila její výška a výpočet by mohl být něco optimalizovanější a rychlejší.

Seznam použité literatury

[1] Kelly Easton, George Nemhauser, Michael Trick, Solving the Traveling Tournament Problem: A Combined Integer Programming and Constraint Programming Approach, 2003
[2] Jeffrey H. Dinitz, Designing Schedules for Leagues and Tournaments,
[3] Michael Trick, Tutorial on scheduling sports tournaments, 2004
[4] Ken McAloon, Carol Tretkoff, Gerhrad Wetzel, Sports League Scheduling,