VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

http://fit.vut.cz

# SNT - Projekt 2019/2020

# Řešení TUP pomocí genetického algoritmu

Jakub Svoboda*

**Abstrakt**
V této práci byla implementována metoda využívající genetické algoritmy pro řešení úlohy cestujících rozhodčích. Metoda byla naimplementována dle článku [3]. Práce algoritmu byla poté experimentálně ověřena na datasetu [2].
Dále byly provedeny dva experimenty. První z nich upravuje mutaci algoritmu, druhý experiment ověřoval efektivnost lokálně optimálního křížení. Bylo zjištěno, že změna mutace nevedla ke zlepšení a že lokálně optimalizované křížení podává lepší výsledky, než rychlejší náhodné křížení.

**Klíčová slova:** Travelling Umpire Problem — TUP — genetické algoritmy — GA — simulace

*xsvobo0z@stud.fit.vut.cz, *Fakulta informačních technologií, Vysoké učení technické v Brně*

## 1. Úvod do problematiky

**[Motivace]** V této práci je prezentován genetický algoritmus využitý k optimalizaci problému cestujících rozhodčích (travelling umpire problem). Algoritmus byl poprvé představen v článku *Locally Optimized Crossover for the Traveling Umpire Problem* [3].

Problém cestujících rozhodčích je nejvíce známý ze sportovních lig. Například v nejvyšší basebalové lize v Americe, Major League Baseball, je před začátkem sezony nutné rozvrhnout 2430 her [3] a ke každé hře přiřadit tým rozhodčích. Tento problém je vysoce komplexní kvůli velkému množství možných kombinací a kvůli mnoha podmínkám, které musí být splněny. Tyto podmínky mohou zahrnovat různá omezení stanovená ligou, jako například omezení opakování rozhodčích na jednotlivých stadionech, nutnost rozhodčích navštívit všechny týmy za sezonu a podobné. Nalezení efektivních řešení je tak nejen složité, ale zároveň velmi žádoucí, jelikož lépe nastavený turnaj může vést k lépe rozvrhnuté sezoně, sníženým nákladům na cestování a podobné.

Problém cestujících rozhodčích je NP-úplný (důkaz proveden například redukcí z problému obchodního cestujícího). Více o složitosti problému lze nalézt v článku *On the complexity of the traveling umpire problem*[1].

**[Definice problému]** V problému cestujících rozhodčích se turnaj skládá ze dvou částí, kde každá část je hraná stylem každý s každým (*round-robin*). Každá dvojice týmů se tak za celou sezonu utká dvakrát, z toho jednou na svém domácím hřišti.

V celém turnaji se utká $2n$ týmů v celkem $4n-2$ herních slotech. Každý slot se skládá z $n$ her hraných zhruba ve stejný čas, na každou hru dohlíží jeden rozhodčí (celkem $n$ rozhodčích). Při řešení TUP je cílem přiřadit rozhodčí ke každé hře tak, aby rozřazení bylo co nejvíce optimální. Optimálnost řešení je definována podle následujících pěti pravidel:

1. Každá hra má přiřazeného rozhodčího.
2. Každý rozhodčí řídí právě jednu hru v každém slotu.
3. Každý rozhodčí navštíví každý domácí stadion alespoň jednou za sezonu.
4. Žádný z rozhodčích není na jednom domácím stadionu více než jednou v $n-d_1$ po sobě následujících hrách.
5. Žádný z rozhodčích neřídí hru jednoho týmu více než jednou v $(n/2)-d_2$ po sobě následujících hrách.

Celočíselné parametry $d_1$ a $d_2$ v rozsahu $(0..n-1)$ a $(0..(n/2)-1)$ určují přísnost omezení pravidel (4) a (5). Nejpřísnější omezení je pak s parametry $d_1 = d_2 = 0$, s vyššími hodnotami jsou pravidla natavená

volněji, s parametry $d_1 = n-1$, $d_2 = n/2-1$ se pak pravidla (4) a (5) neprojeví.

## 2. Dataset

Katolická univerzita v Lovani (KU Leuven) spravuje dataset sloužící jako benchmark při testování řešení problému cestujících rozhodčích. Na webových stránkách [2] lze stáhnout potřebné soubory[1] a zároveň je zde uveden žebříček nejúspěšnějších řešení.

Každý vstupní soubor obsahuje tři hlavní části s informacemi. Položka `nTeams=N` obsahuje informaci o počtu týmů v turnaji. Druhou položkou je matice reprezentující vzdálenosti mezi jednotlivými hřišti. Tato matice o rozměrech $nTeams \times nTeams$ obsahuje položky, kde číslo na řádku $n$ a ve sloupci $m$ reprezentuje vzdálenost (v kilometrech) mezi domácími hřišti týmů $n$ a $m$. Třetí položkou je rozvrh turnaje, kde každý řádek matice obsahuje informace o tom, které týmy se v daném slotu utkají a na kterém stadionu se bude hrát.

## 3. Metoda řešení

Trick a Yildiz ve svém článku [3] navrhli genetický algoritmus, jehož úkolem je aproximovat ideální řešení. Genetické algoritmy využívají genetickou reprezentaci potenciálních řešení, které se nazývají chromozómy. Úspěšnost těchto řešení je zhodnocena pomocí *fitness* funkce. Nejlepší řešení jsou pak zvolena jako rodiče další generace. Tvorba nových členů populace pak probíhá obvykle zkombinováním chromozómů několika rodičů. Chromozómy mohou dále mutovat, tedy v každé generaci je každé řešení s určitou pravděpodobností (většinou náhodně) pozměněno.

**[Reprezentace řešení]** Pro tuto úlohu jsou řešení reprezentována jako přiřazení rozhodčích k dané hře v každém slotu. Ukázka řešení je zobrazena v tabulce 1. Fitness funkce řešení se pak skládá ze dvou částí. První část je celková vzdálenost, kterou musí rozhodčí urazit v daném turnaji. V této úloze je cena přímo úměrná nacestovaným kilometrům. Druhá část fitness funkce vychází z pravidel definovaných v kapitole 1, kde za každé porušení podmínek (3), (4) a (5) je pak přiřazena vysoká pokuta. Podmínky (1) a (2) nelze za použití této metody porušit, jelikož každá permutace řešení obsahuje pouze validní rozložení trenérů. V populaci jsou jedinci s nižším ohodnocením funkcí fitness nejúspěšnější (mají nejnižší cenu). Evolučně jsou pak eliminováni jedinci, kteří porušují podmínky zadání a kteří rozesílají trenéry neefektivně.

**[Lokálně optimalizované křížení]** V tradičních genetických algoritmech je potomek vytvořen jako náhodná kombinace dvou rodičů. Díky této náhodné kombinaci je běžné, že potomci mají výrazně horší ohodnocení funkce fitness, než jejich rodiče. Trick a Yildiz se tento jev pokusili eliminovat pomocí lokálně optimalizované křížení. V této metodě jsou náhodně zvoleni dva rodiče. Dále se náhodně zvolí index řezu $t$ a potomek zdědí od prvního rodiče řešení pro sloty 0 až $t-1$ a od druhého rodiče řešení pro sloty $t$ až do konce chromozómu. Metoda následně provede lokální optimalizaci, kde pro část řešení zděděnou od druhého rodiče nalezne takovou permutaci, která porušuje nejméně podmínek. Toto odpovídá situaci, kdy si trenéři mohou rozpis svých her od slotu $t$ navzájem povyměňovat. Ukázka lokálně optimalizovaného křížení je zobrazena v tabulce 2.

**[Mutace]** V této metodě je nejprve náhodně zvolen slot řešení a v tomto slotu jsou zaměněny pozice dvou náhodně zvolených her. Mutace tak do algoritmu přináší další prvek náhodnosti, který může přinést řešení, které by nebylo možné nalézt pouhým křížením. Ukázka mutace řešení je zobrazena v tabulce 3.

**[Inicializace jedinců]** U běžných genetických algoritmů je obvykle původní populace generována náhodně. Trick a Yildiz ve svém článku [3] přišli s metodou generování jedinců, kde každý jedinec je generován od prvního slotu směrem k poslednímu. Při vytvoření každého nového slotu je ověřeno, která varianta přiřazení her v tomto slotu neporušuje pravidla (4) a (5). Jakmile je nalezena vhodná varianta, slot je přiřazen a algoritmus postupuje k následujícímu slotu. Pokud není žádné řešení možné, program se vrátí o jeden slot nazpět a ověřuje další možné kombinace (*backtracking*).

## 4. Implementace

V rámci projektu byla vytvořena implementace výše zmíněného genetického algoritmu. Program byl napsám v jazyce Python3 a byl testován na serveru Merlin.

Program nejprve pomocí regulárních výrazů zpracuje zadaný vstupní soubor z datasetu a vytvoří instanci reprezentující daný řešený problém. Následně je vytvořena počáteční populace s 500 jedinci, což je stejný počet jako vytvářeli Trick a Yildiz ve svém článku [3].

Následně proběhne cyklus reprezentující jednu epochu algoritmu. V každé epoše je nejprve změřena úspěšnost populace pomocí funkce fitness. Výsledky

[1] Archív ke stažení na těchto stránkách neobsahuje všechny soubory, na kterých Trick a spol. algoritmus testovali. Doporučuji použít dataset přibalený k tomuto projektu v adresáři `dataset/`.

| Slot: | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Rodič1: | | | | | | | | | | | | | | |
| Rozhodčí1: | (1,5) | (2,8) | (5,6) | (3,7) | (8,1) | (4,6) | (7,5) | (2,4) | (6,8) | (4,5) | (3,1) | (8,6) | (1,3) | (4,2) |
| Rozhodčí2: | (4,8) | (5,3) | (1,4) | (8,2) | (6,5) | (2,3) | (1,8) | (3,6) | (5,7) | (2,1) | (7,6) | (3,5) | (6,4) | (8,3) |
| Rozhodčí3: | (6,2) | (4,7) | (3,8) | (5,4) | (7,2) | (8,5) | (2,6) | (5,8) | (4,1) | (6,3) | (8,4) | (1,2) | (7,8) | (5,1) |
| Rozhodčí4: | (7,3) | (1,6) | (2,7) | (6,1) | (3,4) | (7,1) | (4,3) | (1,7) | (3,2) | (8,7) | (2,5) | (7,4) | (5,2) | (6,7) |
| Rodič2: | | | | | | | | | | | | | | |
| Rozhodčí1: | (1,5) | (2,8) | (5,6) | (8,2) | (3,4) | (7,1) | (4,3) | (1,7) | (3,2) | (8,7) | (2,5) | (7,4) | (5,2) | (6,7) |
| Rozhodčí2: | (4,8) | (5,3) | (1,4) | (3,7) | (6,5) | (2,3) | (1,8) | (3,6) | (5,7) | (2,1) | (7,6) | (1,2) | (6,4) | (8,3) |
| Rozhodčí3: | (6,2) | (4,7) | (3,8) | (5,4) | (8,1) | (4,6) | (7,5) | (2,4) | (6,8) | (4,5) | (3,1) | (8,6) | (1,3) | (4,2) |
| Rozhodčí4: | (7,3) | (1,6) | (2,7) | (6,1) | (7,2) | (8,5) | (2,6) | (5,8) | (4,1) | (6,3) | (8,4) | (3,5) | (7,8) | (5,1) |

**Tabulka 1.** Dvě řešení turnaje s osmi tými, čtyřmi rozhodčími a 14 herními sloty. Data převzata z [3].

| Slot: | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Potomek: | | | | | | | | | | | | | | |
| Rozhodčí1: | (1,5) | (2,8) | (5,6) | (3,7) | (8,1) | (4,6) | (7,5) | (2,4) | (6,8) | (4,5) | (3,1) | (8,6) | (1,3) | (4,2) |
| Rozhodčí2: | (4,8) | (5,3) | (1,4) | (8,2) | (6,5) | (2,3) | (1,8) | (3,6) | (5,7) | (2,1) | (7,6) | (3,5) | (6,4) | (8,3) |
| Rozhodčí3: | (6,2) | (4,7) | (3,8) | (5,4) | (7,2) | (8,5) | (2,6) | (5,8) | (4,1) | (6,3) | (8,4) | (1,2) | (7,8) | (5,1) |
| Rozhodčí4: | (7,3) | (1,6) | (2,7) | (6,1) | (3,4) | (7,1) | (4,3) | (1,7) | (3,2) | (8,7) | (2,5) | (7,4) | (5,2) | (6,7) |

**Tabulka 2.** Lokálně optimalizované křížení. Sloty 1 až 10 pochází z rodiče1 z tabulky 1, sloty 11 až 14 pak z rodiče2. Přiřazení slotů z rodiče2 jednotlivým trenérům je nastaveno tak, aby byla minimalizována cena řešení (ohodnocení funkce fitness). Data převzata z [3].

| Slot: | 1 | 2 | 3 | 4 | **5** | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Potomek: | | | | | | | | | | | | | | |
| Rozhodčí1: | (1,5) | (2,8) | (5,6) | (3,7) | **(7,2)** | (4,6) | (7,5) | (2,4) | (6,8) | (4,5) | (3,1) | (8,6) | (1,3) | (4,2) |
| Rozhodčí2: | (4,8) | (5,3) | (1,4) | (8,2) | (6,5) | (2,3) | (1,8) | (3,6) | (5,7) | (2,1) | (7,6) | (3,5) | (6,4) | (8,3) |
| Rozhodčí3: | (6,2) | (4,7) | (3,8) | (5,4) | **(8,1)** | (8,5) | (2,6) | (5,8) | (4,1) | (6,3) | (8,4) | (1,2) | (7,8) | (5,1) |
| Rozhodčí4: | (7,3) | (1,6) | (2,7) | (6,1) | (3,4) | (7,1) | (4,3) | (1,7) | (3,2) | (8,7) | (2,5) | (7,4) | (5,2) | (6,7) |

**Tabulka 3.** Ukázka mutace řešení z tabulky 2. V náhodně zvoleném slotu (5) jsou náhodně zvoleny dvě hry (první a třetí) a v rámci slotu jsou tyto dvě hry vyměněny mezi trenéry. [3].

nejlepšího jedince jsou vytisknuty, spolu s informacemi o počtu porušení pravidel (3), (4) a (5). Za každé porušení pravidla je jedinci přiřazena penalta ve výši 100 000. V původní práci není uvedeno, kolik rodičů je zvoleno k vytvoření nové generace a kolik jedinců je odstraněno z generace. V této implementaci je počet jedinců vždy 500, z toho v každé epoše je méně úspěšná polovina populace eliminována a jsou nahrazeni novými jedinci. Konkrétně tedy z 500 jedinců je 250 eliminováno a zbylých 250 rodičů je náhodně rozděleno do dvojic, kde každý pár jedinců vygeneruje dva potomky.

Po vytvoření nové generace je pro každého jedince rozhodnuto, zda pro něj proběhne mutace. Šance na zmutování je 5 %, stejně jako v původním článku.

## 5. Měření a výsledky

Pro značnou část datasetu (instance s menším počtem týmů) jsou optimální řešení předem známá. Správnost implementace algoritmu lze tady ověřit tím, že pro vyřešené úlohy metoda nalezne známé optimální řešení. Výsledky měření jsou shrnuty v tabulce 4. Pro všechny turnaje s počtem týmů menší než 10 bylo vždy pomocí této implementace nalezeno optimální řešení a implementaci lze tak považovat za korektní.

Nedostatek implementace se projevuje především v její rychlosti. Pro instance s desíti a více týmy je algoritmus značně pomalý. Při měření sice je schopen řešení optimalizovat, korektně odstraňuje chybná řešení a postupně snižuje ohodnocení funkcí fitness, nicméně pro malou výpočetní rychlost nenalezne optimální řešení v rozumném čase (na referenčním serveru dojde k překročení maximálního času využití CPU).

## 6. Vlastní experimenty

**[Mutace]** V původním článku je mutace aplikována pouze na dvě hry v rámci jednoho slotu, kde jsou tyto dvě hry zaměněny. V rámci experimentování byla vyzkoušena i mutace, kdy jsou jednoduše všechny hry v jednom slotu zamíchány a s náhodným pořadím přiřazeny rozhodčím. Tato metoda obecně zvýší počet mutací v populaci.

| Instance: | Vzdálenost: | | | | | Čas: | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Projekt | GA | IP | CP | SA | Projekt | GA | IP | CP | SA |
| 4 | 5176 | 5176 | 5176 | 5176 | 5176 | 0s | 0s | 0s | 0s | 0s |
| 6 | 14 077 | 14 077 | 14 077 | 14 077 | 14 077 | 6s | 1s | 0s | 0s | 0s |
| 6A | 15 457 | 15 457 | 15 457 | 15 457 | - | 3s | 1s | 0s | 0s | - |
| 6B | 16 716 | 16 716 | 16 716 | 16 716 | - | 10s | 1s | 0s | 0s | - |
| 6C | 14 396 | 14 396 | 14 396 | 14 396 | - | 5s | 1s | 0s | 0s | - |
| 8 | 34 311 | 34 311 | 34 311 | 34 311 | 34 311 | 12s | 5s | 2s | 2s | 1m |
| 8A | 31 490 | 31 490 | 31 490 | 31 490 | - | 0s | 12s | 1s | 0s | - |
| 8B | 32 731 | 32 731 | 32 731 | 32 731 | - | 4s | 5s | 1s | 0s | - |
| 8C | 29 879 | 29 879 | 29 879 | 29 879 | - | 8s | 4s | 1s | 0s | - |
| 10 | 50 607* | 48 595 | 48 942 | 49 595 | 50 196 | 30m* | 3m | 5m | 3h | 4m |
| 10A | 47 296* | 46 632 | 46 551 | 46 927 | - | 30m* | 2m | 23m | 3h | - |
| 10B | 47 921* | 45 609 | 45 609 | 47 840 | - | 30m* | 2m | 1m | 3h | - |
| 10C | 45 262* | 43 149 | 43 149 | 43 149 | - | 30m* | 20m | 2h | 3h | - |

**Tabulka 4.** Naměřené výsledky. Sloupec instance značí vstupní soubor z datasetu. Sloupce se vzdáleností reprezentují nejlepší dosažený výsledek pro daný algoritmus, kde slupec *Projekt* je implementace popsaná v kapitole 4, *GA* je genetický algoritmus z původního článku. Další sloupce jsou výsledky získané pomocí metod z jiných článků: *IP* – Integer Programming, *CP* – Constraint Programming a *SA* – Simulated Annealing. Hodnoty s hvězdičkou označují timeout serveru. Vynechané výsledky značí, že autoři danou instanci netestovali. Data převzata z [3].

Testování této mutace bylo provedeno na instanci `upms10`. Po 30 minutách bylo nalezeno nejlepší řešení 50 875, což je mírně horší, než 50 607 s původní mutací. Obecně tak tento experiment nepřinesl zlepšení.

**[Operátor křížení]** Trick a Yildiz ve svém článku [3] uvádí lokálně optimalizované křížení jako hlavní přínos článku při řešení této úlohy. Výpočetně je však jimi navržená lokální optimalizace velmi náročná, protože v každé epoše, pro každé dva rodiče, pro všechny kombinace jejich křížení je nutné zjistit porušení pravidel (3), (4) a (5). V rámci experimentování bylo vyzkoušeno tradiční křížení známé z genetických algoritmů, kdy je jako potomek zvolena náhodná kombinace rodičů. Snahou bylo zjistit, jestli výrazná časová úspora při výpočtu křížení nepřeváží zisk z lokálně optimalizovaného křížení.

Při testování bylo zjištěno, že náhodné křížení je přibližně 15krát rychlejší, než lokálně optimalizované křížení. Na instanci `upms10` byla po půl hodině nalezena nejlepší vzdálenost 53 035, což je horší, než 50 607 s lokálně optimalizovaným křížením. Lze tedy konstatovat, že pro tuto úlohu je lokální křížení výhodnější, než metoda náhodného křížení.

## 7. Závěr

V rámci projektu byla implementována metoda využívající genetické algoritmy pro řešení úlohy cestujících rozhodčích, dle článku [3]. Pro menší turnaje tato implementace vždy nalezla optimální řešení, pro turnaje s desíti a více hráči se projevuje nízká optimalizace implementace.

Dále byly provedeny dva experimenty. První z nich při mutaci namísto prohození dvou her zamíchal všechny hry v daném slotu. Výsledky této změny se na výsledku téměř neprojevili. Druhý experiment ověřoval efektivnost lokálně optimálního křížení. Bylo zjištěno, že lokálně optimalizované křížení podává po uběhnutí stejně dlouhého času lepší výsledky, než rychlejší náhodné křížení.

## Literatura


[1] OLIVEIRA, L. de, SOUZA, C. C. de a YUNES, T. On the complexity of the traveling umpire problem. *Theoretical Computer Science*. Elsevier. 2015, roč. 562, s. 101–111.

[2] TOFFOLO, T. A. M., WAUTERS, T. a TRICK, M. An automated benchmark website for the Traveling Umpire Problem. červen 2015. Dostupné z: `http://gent.cs.kuleuven.be/tup`.

[3] TRICK, M. A. a YILDIZ, H. Locally optimized crossover for the traveling umpire problem. *European Journal of Operational Research*. Elsevier. 2012, roč. 216, č. 2, s. 286–292.