Dokumentácia k projektu z SNT

University Course Timetabling Problem

Matej Marušák, Apríl 2018

1 Úvod

Plánovanie rozvrhov je problém, ktorému sa venuje mnoho prác. Jedná sa o komplexný problém, ktorý hľadá taký rozvrh, kde by boli splnené všetky nutné podmienky a zároveň vzniklo čo najmenej konfliktov. Najčastejšie narazíme na optimalizačné metódy z oblasti soft-computingu, napríklad[1][2][3][4] a mnohé ďalšie. Táto dokumentácia popisuje riešenie problému tvorby univerzitných rozvrhov za pomoci hybridného genetického algoritmu, ktorý bol publikovaný v [4]. Kapitola 2 sa venuje popisu riešenej problematiky. Kapitola 3 vysvetľuje fungovanie zvoleného prístupu, ktorý následne v kapitole 4 je implementovaný a kapitola 5 obsahuje experimentálne zhodnotenie algoritmu.

2 Tvorba univerzitných rozvrhov

Základom problému tvorby univerzitných rozvrhov je rozmiestenie kurzov do učební a častí dňa pri splnení *mäkkých* a *tvrdých* obmedzený. *Tvrdé* obmedzenia, ktoré budú ďalej v texte označované ako povinné, sú také, ktoré musia byť vždy splnené. *Mäkké*, ďalej označované ako nepovinné sú také, ktorých porušenie je povolené, ale ich porušenie je penalizované. Popis jednotlivých obmedzení je v podsekcii 2.1.

Samotný problém je 5tica P=(C,R,T,F,A) kde

- C je množina kurzov
- R je množina miestností
- T je množina časových úsekov (takmer vždy |T| = 45)
- F je množina vlastností, ktoré môže miestnosť nadobúdať
- A je množina študentov

Okrem toho existujú 4 funkcie:

- ρ: R $\rightarrow$ N
- α: A x C $\rightarrow$ 0,1
- φ: R x F $\rightarrow$ 0,1
- γ: C x F $\rightarrow$ 0,1

2.1 Obmedzenia

Povinné obmedzenia sú:

- V jednom časovom úseku môže byť v jednej miestnosti vyučovaný len jeden kurz

- Všetci študenti, ktorý majú zapísaný kurz sa musia zmestiť do miestnosti
- Miestnosť musí spĺňať všetky kritéria, ktoré požaduje kurz
- Žiaden študent nemôže mať v jednom momente viac ako jeden kurz

Nepovinné obmedzenia sú:

- Študent má viac ako dva za sebou idúce kurzy (za každý ďalší je jeden penalizačný bod, teda za 3 za sebou idúce kurzy je jeden bod)
- Študent má len jeden kurz v jeden deň
- Študent má kurz v poslednom časovom úseky pre jeden deň

Za porušenie každého nepovinného obmedzenia je jeden penalizačný bod.

3 Hybridný genetický prístup

Táto dokumentácia popisuje implementáciu založenú na [4]. Jej základom je hybridný prístup, teda využitie genetických algoritmov spolu s lokálnym prehľadávaním. Oproti genetickým algoritmom ale neobsahuje kríženie, ktoré by vo väčšine prípadov porušilo povinné obmedzenia. Z genetických algoritmov sa využívajú prístupy populácie, mutácie a výberu. Namiesto kríženia je aplikované lokálne prehľadávanie.

Algoritmus začína vytvorením generácie. Počet jedincov bol stanovený na 100 podľa [4]. Popis tvorby prvej generácie je popísaný v sekcii 3.1. Následne sa prejde do hlavného cyklu programu, ktorý v prvom kroku aplikuje mutáciu na 20% jedincov populácie. Mutácii sa venuje sekcia 3.2. Po skončení mutácie je aplikované lokálne prehľadávanie, ktoré je detailne vysvetlené v podkapitole 3.3. Posledným krokom hlavného cyklu je výber novej populácie, na ktorú je opäť aplikovaný tento cyklus. Spôsob, akým sa vyberajú nový jednotlivci, je vysvetlený v podkapitole 3.4.

3.1 Tvorba populácie

Prvý krok, ktorý je potrebné vykonať je tvorba náhodnej populácie. Každý člen populácie musí byť riešenie, v ktorom nie sú porušené žiadne povinné obmedzenia. Tvorba takýchto riešení je problematická. Odlišnosť jednotlivcov v populácia je základom rýchlej konvergencie. Preto nie je možné použiť bežne používané algoritmy na hľadanie takýchto riešení, ktoré vyberajú kurzy s najväčšími požiadavkami prvé. Je použitá dvojkroková metóda, ktorá najskôr za pomoci farbenia grafu rozdelí kurzy do jednotlivých časových úsekov pri zachovaní podmienky, že žiaden študent nemá v jednom časovom okne dva kurzy. Následne sa pre každé časové okno skontroluje, či je pokryteľné miestnosťami. V prípade, že nie je, sú náhodné kurzy preradené do iného náhodného okna až kým všetky časové okná nie sú splnené.

3.2 Mutácia

Každému vybranému jednotlivcovi pre mutáciu je vybraných 20% kurzov, ktoré sú presunuté do iného náhodného časového úseku. Je dôležité, aby sa neporušili povinné obmedzenia. Pseudokód mutácie je nasledovný:

```
for (i = 0; i < courses_count/5; i++) {
    tries = 0;
    victim = random_course(); // course to be moved
    while (tries < timeslots_count){
        timeslot = random_timeslot();
        if (timeslot == victim->timeslot)
            continue;
        if (keep_hard_constraints(victim, timeslot) {
            set_timeslot(victim, timeslot);
            break;
        }
        tries++;
    }
}
```

3.3 Lokálne prehľadávanie

Ako lokálne prehľadávanie je použité aplikovanie 11 funkcií na každého jednotlivca, ktoré sú popísané v sekcii 3.3.1 Ich aplikáciou sa vytvorí nový potomok, ktorý je vložení do zásobníku nových potomkov. Takýto jedinec neskôr môže byť vybratý do novej populácie.

3.3.1 Lokálne funkcie

Žiadna operácia nemôže porušiť povinné obmedzenia a teda *náhodne* v nasledujúcom popise znamená *náhodne zo všetkých, ktoré neporušia povinné obmedzenia.*

- N1 : Vyber dva náhodné kurzy a vymeň ich časové okná
- N2 : Presuň náhodný kurz do náhodného časového okna
- N3 : Vyber dve náhodné časové okná a vymeň všetky kurzy v nich
- N4 : Vyber dve náhodné časové okna a vykonaj rotáciu o jedna doľava pre všetky prvky medzi nimi (vrátane zvolených)
- N5 : Presuň kurz s najväčšou penaltou z 10% náhodne zvolených kurzov do náhodne zvoleného časového okna
- N6 : N5 ale s 20% kurzov
- N7 : N5 ale presuň do časového okna, ktoré generuje najmenšiu penaltu
- N8 : N7 ale s 20% kurzov
- N9 : Vyber náhodný kurz a náhodné časové okno a vykonaj kempe chain
- N10 : N9 ale vyber kurz s najvyššou penaltou z 5% náhodných kurzov
- N11 : N9 ale s 20% kurzov

3.4 Výber novej generácie

Po predchádzajúcich krokoch vznikne viac členov populácie, ako bola pôvodná. Aby sa zabránilo nárastu populácie, ruleta je použitá pre výber pôvodného počtu členov. Princíp ruletového výberu je založený na pravdepodobnosti, že jednotlivec s väčším skóre bude skôr zasiahnutý pri výbere náhodného čísla. V tejto aplikácií je ale dôležité aby sa preferovane vyberali jednotlivci so skóre čo najnižším.

4 Implementácia

Aplikácia bola vytvorená v jazyku C, hlavne z dôvodu rýchlosti behu. Program bol rozložený do viacerých modulov, kde každý modul implementuje určitú ucelenú časť problému. Konkrétne sa jedná o moduly:

- problem - Rozhranie pre spracovanie, uloženie a jednoduchý prístup k vstupnému problému
- timetable - Rozhranie implementujúce jeden rozvrh a prácu s ním
- solution_finder - Generovanie riešení, ktoré spĺňajú všetky povinné obmedzenia
- generation - Vytváranie nových potomkov za pomoci úpravy rodičov

Okrem toho sa vyskytujú aj ďalšie pomocné moduly a to:

- error - Spracovanie chýb
- dll - Implementácia obojstranne viazaného zoznamu
- utils - Sada pomocný funkcií

Všetky tieto moduly sú v zložke `src`. Okrem toho existuje hlavný modul `uctps.c`, ktorý implementuje celkový algoritmus priebehu optimalizácie.

4.1 Preklad

Program sa prekladá za pomoci príkazu `make`. Všetky vytvorené súbory (okrem hlavného spustiteľného) sú v zložke `build`. Na rovnakej úrovni ako je Makefile sa vytvorí spustiteľný súbor `uctps`.

4.2 Spúšťanie

Program berie jeden argument a to názov súboru, v ktorom je zadanie problému.

Teda napríklad :

```
./uctps tests/small1.tim
```

Po skončení behu, ktorý môže trvať aj pár hodín, sa na prvom riadku výstupu objaví dosiahnuté skóre a vypíše sa riešenie. Riešenie pozostáva z n riadkov, kde každý riadok reprezentuje jeden kurz. Každý riadok obsahuje dve čísla oddelené medzerou. Prvé číslo označuje časové okno, do ktorého táto udalosť bola priradená. Druhé číslo je číslo miestnosti.

5 Experimentálne výsledky

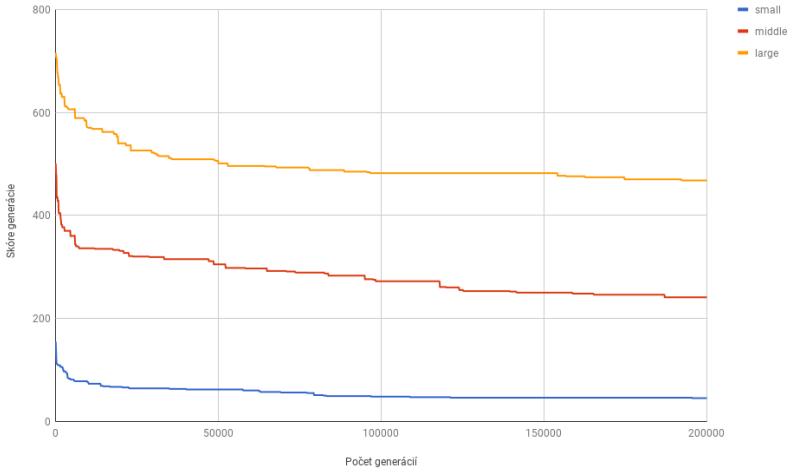
Článok [4] využíval štandardnú sadu testov, ktorú dnes už nie je možné zohnať. S ostatnými riešiteľmi problému rozvrhov v predmete SNT v roku 2017/18 sme sa pokúsili vytvoriť sadu, ktorá by približne odpovedala spomínanej sade. Následne každé zadanie bolo riešené 10 krát a najlepšie riešenie je zhrnuté v tabuľke 1. Daná tabuľka obsahuje zároveň aj riešenie ostatných riešiteľov.

dataset	implementácia z [4]	táto implementácia	xvales02	xvales03
small1	0	33	47	31.2
small2	0	32	21	34.5
small3	0	26	38	28.0
small4	0	42	53	41.7
small5	0	30	37	32.6
medium1	221	234	237	255.9
medium2	147	132	62	126.2
medium3	246	269	205	289.8
medium4	165	170	66	173.1
medium5	130	145	76	168.2
large	529	468	342	408.4

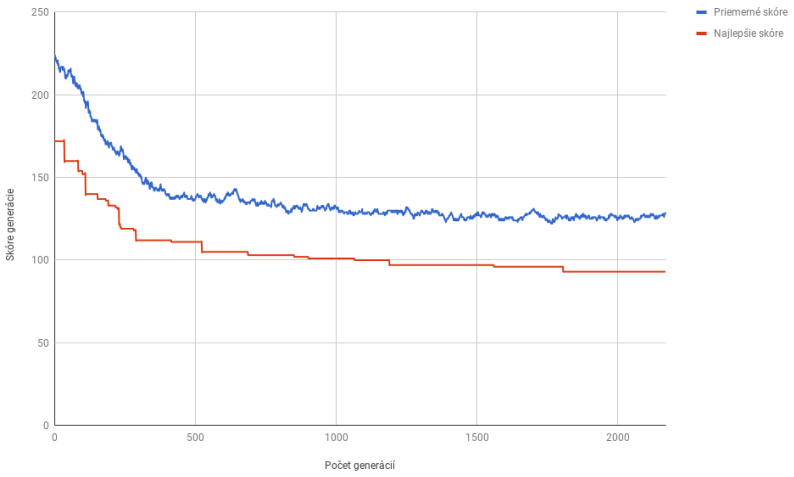
Tabuľka 1: Výsledky behu programu v porovnaní s inými riešeniami

Je vidieť značný rozdiel, hlavne pri malých zadaniach, kde referenčný článok našiel ideálne riešenie. Tento rozdiel bude pravdepodobne zapríčinený faktom, že nami vytvorená sada nemala ideálne riešenie.

Priebeh optimalizácie je znázornený na obrázkoch 1 a 2. Ako je možné vidieť, v prvých pár tisíc generácií sa prudko vylepšuje skóre, ktoré sa ustáli a lepšie riešenie je nájdené len po dlhej dobe. Obrázok 1 zobrazuje priebeh pre rôzne veľké sady. Obrázok 2 zobrazuje skóre celej generácie.



Obr. 1: Priebeh skóre pre rôzne veľké sady



Obr. 2: Priebeh fitness a priemerného fitness populácie pri zvyšujúcom sa počte generácií

6 Záver

Správa prezentovala experimentálne overenie prístupu prezentovaného v [4]. Vytvorená aplikácia dosahovalo podobné výsledky v porovnaní s inými algoritmami (nakoľko pôvodná testovacia sada nebola získaná). Daným prístupom je možné získať kvalitné riešenia univerzitných rozvrhov. Priemerne dobré výsledky sú nájdené veľmi rýchlo, na kvalitné je treba uvažovať stovky tisícov generácií. Slabinou prístupu je nutnosť vytvoriť 100 riešení, ktoré neporušujú žiadne povinné obmedzenia, čo môže trvať neprimerane dlho k celkovému behu programu.

7 Literatúra

[1] E. Aycan, T. Ayav, "Solving the Course Scheduling Problem Using Simulate Annealing", *International Advance Computing Conference* (IACC 2009) Patiala, pp. 6-7, March 2009.

[2] S. F. H. Irene, S. Deris, S. Z. M. Hashim, "A study on PSO-based university course timetabling problem", *Proc. Int. Conf. Adv. Comput. Control*, pp. 648-651, 2009.

[3] Oner, Adalet Ozcan, Sel; Dengi, Derya "Optimization of university course scheduling problem with a hybrid artificial bee colony algorithm" *2011 IEEE Congress of Evolutionary Computation*, CEC 2011, p 339-346, 2011.

[4] S. Abdullah, E. K. Burke, B. McCollum, H. Turabieh, "A hybrid evolutionary approach to the university course timetabling problem", *Proc. 2007 Congr. Evol. Comput.*, pp. 1764-1768, 2007.