Excel@FIT 2015
http://excel.fit.vutbr.cz

Systém pro pokročilé plánování

Aleš Horký*

Abstrakt
Tento článek pojednává o návrhu a implementaci softwaru pro automatické plánování rozvrhů zkoušek a přednášek. Návrh je přizpůsoben pro specifické potřeby *Fakulty informačních technologií Vysokého učení technického v Brně*, kterým nevyhovuje žádný komerční ani volně dostupný rozvrhovací produkt. Problém je řešen kombinací genetického a heuristického algoritmu. Pomocí genetického algoritmu je získáno pořadí předmětů v jakém mají být vkládány do výsledného rozvrhu heuristickým algoritmem. Výsledky na reálných vstupních datech vykazují přibližně o 40 % méně denních kolizí (případů, kdy má student naplánováno více než jednu zkoušku v jednom dni) a přibližně o 15 % větší rozestupy mezi jednotlivými termíny oproti stávajícím rozvrhům za dobu výpočtu řádově desítek minut. Nastavitelné parametry aplikace umí postihnout většinu běžných požadavků (např. časové preference, vybavení učeben, paralelní plánování do více učeben, fixní termíny, a další). V průběhu vývoje probíhá spolupráce s osobami odpovědnými za vytváření rozvrhů na fakultě tak, aby mohli tento nástroj v budoucnu využívat.

Klíčová slova: Plánování školních rozvrhů — Genetický algoritmus — Heuristický algoritmus — Kolizní matice — Multikriteriální optimalizace

Přiložené materiály: Zdrojový kód — Prezentovaná řešení rozvrhů

*xhorky17@stud.fit.vutbr.cz, *Faculty of Information Technology, Brno University of Technology*

1. Úvod

Tato práce popisuje návrh a implementaci programu pro automatizované plánování rozvrhů a zkoušek na *Fakultě informačních technologií Vysokého učení technického v Brně (FIT)*. Program má za cíl zvýšit kvalitu a zjednodušit vytváření těchto rozvrhů, které na fakultě probíhá dodnes z větší části manuálně.

Pro automatizované plánování školních rozvrhů lze nalézt mnoho komerčních i volně dostupných programů. Tyto programy se snaží nabídnout obecné rozhraní, aby pokryly požadavky co nejvíce institucí. Navíc se typicky zaměřují na plánování rozvrhů přednášek a ne (pro fakultu kritických) zkoušek. Ze studie provedené v roce 2012 v rámci bakalářské práce Moniky Kubalcové [6] však vyplynulo, že žádný z nich není bez dalších úprav pro potřeby fakulty přijatelný. Důvodem je mimo jiné nedostupnost detailních (byť i anonymních) informací o studenty zapsaných předmětech. Závislosti mezi předměty jsou dostupné pouze v sumarizované formě tzv. *kolizní matice*[1].

Tvorba rozvrhů je NP-úplný problém, jehož obtížnost spočívá především v exponenciálně rostoucím prostoru řešení, který je navíc multimodální. Pro řešení tohoto problému tak z časových důvodů nelze použít přímočaré algoritmy. Různými heuristickými postupy se však můžeme snažit nalézt tzv. *sub-optimální řešení*. Pro něj sice nedokážeme říci, zda neexistuje nějaké jiné lepší řešení, ale dosáhneme u něj takových vlastností, že jej můžeme považovat za vyhovující [3].

V samotném průběhu generování školních rozvrhů je potřeba volit takové kombinace entit (přednášejících, učeben a studentů) umístěných do časových oken, které nabídnou jejich optimální rozvržení z hlediska nákladů, času, lidského pohodlí a podobně [3]. Konkrétní kritéria, kterými je třeba se řídit, jsou popsána v kapitole 2.

Jak již bylo zmíněno, existující rozvrhovací programy nejsou schopny pokrýt specifické požadavky

[1] Kolizní matice obsahuje pro všechny dvojice předmětů počet společných studentů pro tyto dva předměty.

Fakulty informačních technologií. Uveďme například volně dostupný *Free Timetabling Software (FET)*. Tento program využívá speciálně vyvinutý algoritmus *recursive swapping (rekurzivní prohazování)*, který nejprve umístí některé předměty do rozvrhu tak, aby neporušovaly žádná ze zadaných kritérií. Poté přidává zbývající předměty a snaží se rekurzivně přepočítávat, zda by se rozvrh nezlepšil jejich záměnou. Tento výpočet se opakuje, dokud existuje záměna zvyšující kvalitu rozvrhu[2].

Jiným příkladem může být komerční aplikace *ASC Timetables*, kterou využívá přibližně 150 000 škol po celém světě. Jedná se tak o jeden z nejrozšířenějších programů pro tvorbu rozvrhů. Komerční licence vhodná pro použití na fakultě pro rok 2015 stojí 23 900 Kč[3]. I když je tento program primárně určen pro základní a střední školy, byl ve studii z roku 2012 [6] zvolen jako nejvhodnější – přesto byly přínosy *ASC Timetables* pro plánování rozvrhů na fakultě vyhodnoceny diskutabilně. Program sice zjednodušuje některé úkony, ale jeho použití naopak přináší další nové problémy [6].

Mezi hlavní nedostatky obou uvedených programů patří chybějící podpora (pro *FIT* velmi typického) slučování místností při jejich nedostatečné kapacitě. *FET* navíc neumožňuje pracovat na úrovni jednotlivých studentů, ale pouze na úrovni tříd (ročníků). V prostředí *FIT* však v mnoha případech studenti jednoho ročníku navštěvují často dosti rozdílnou skladbu předmětů.

Cílem této práce je řešit uvedený problém pomocí hybridního algoritmu, který kombinuje genetické a heuristické postupy. Návrh je od svého prvopočátku vytvářen tak obecným způsobem, aby byl bez výraznějších změn vyhovující jak pro zkouškové rozvrhy tak i pro rozvrhy přednášek na *FIT*. Vlastní výpočet je řízen genetickým algoritmem, který k ohodnocení kvality jedinců v populaci využívá heuristického algoritmu. Genotyp jedince totiž obsahuje pouze postup, podle kterého heuristický algoritmus rozvrh deterministicky vygeneruje.

Takto implementovaný program vykazuje v posuzovaných aspektech lepší výsledky, než jakých bylo dosaženo při ruční tvorbě rozvrhů v minulosti. Výpočet může běžet paralelně ve více procesech a na běžném počítači trvá řádově od desítek minut po jednotky hodin. V průběhu této doby má uživatel možnost interaktivně vstupovat do procesu výpočtu, prohlížet aktuálně nejlepší rozvrhy a případně dodatečně přizpůsobit jeho parametry. Po dokončení výpočtu je zobrazeno několik vzájemně odlišných variant rozvrhů, ze kterých si uživatel pouze vybere tu pro něj nejvíce přijatelnou.

[2]Uváděné informace o programu *Free Timetabling Software* jsou čerpány z `http://www.lalescu.ro/liviu/fet/`.

[3]Uváděné informace o programu *ASC Timetables* jsou čerpány z `http://www.asctimetables.com/`.

2. Plánování rozvrhů na FIT

Plánovat rozvrh za pomocí výpočetní techniky je možné dvěma základními způsoby [4]. První z nich má za cíl asistovat u procesu manuálního plánování a co nejvíce ho usnadnit. Takové programy jsou založeny především na kvalitní vizualizaci, automaticky upozorňují na nežádoucí kolize a mohou uživateli doporučit nejvhodnější pozici pro zvolený předmět.

Druhou možností jsou komplexnější programy, jejichž součástí je autonomní plánovací algoritmus. Uživatel před spuštěním výpočtu zadá množiny zdrojů (učebny, časová okna, vyučující a podobně) a seznam běhů předmětů, navíc nastaví kritéria a parametry algoritmu. Podle zadaných požadavků je poté vygenerován hotový rozvrh.

Plánování rozvrhů spadá do kategorie *optimalizačních úloh s omezujícími podmínkami* (angl. *constraint optimization problems*) [5]. Jsou to právě specifika omezujících podmínek, které znemožňují efektivní nasazení obecných optimalizačních algoritmů a činí tak z plánování náročnou činnost. Kvalitně vytvořený rozvrh musí v praxi splňovat mnoho často protichůdných omezení – ta lze rozdělit na tzv. *měkká* a *tvrdá*. *Tvrdá (hard-constraints)* rozdělují prohledávaný prostor na množinu přípustných a nepřípustných řešení. *Měkká omezení (soft-constraints)* určují kvalitu jednotlivých řešení a vytváří tak na prostoru přípustných řešení relaci částečného uspořádání [4, 5].

V případě rozvrhů na *FIT* musíme uvažovat dále uvedená kritéria, která byla získána z osobní konzultace s *proděkanem pro vzdělávací činnost v bakalářském studiu* na *FIT* panem Ing. Bohuslavem Křenou Ph.D., studiem současných postupů vytváření rozvrhů na *FIT* od pana Ing. Miloše Eysselta CSc., Ing. Davida Martínka a Ing. Jaroslava Dytrycha a rešerší výše zmíněné studie [6].

Z významných *tvrdých kritérií* jmenujme požadavky na výskyt osob (studentů, vyučujících) pouze na jednom místě současně, přidělování místností podle jejich vybavení, časová omezení přednášejících nebo místností, možnost nastavit minimální rozestupy mezi termíny zkoušek (implicitně je požadavek na alespoň tří denní rozestup z důvodu případných reklamací výsledků, které musí proběhnout nejpozději 48 hodin[4] před následným termínem, navíc je potřeba přidat tolik dnů, aby bylo možné do reklamací opravit všechny písemky), využití více učeben pro jeden předmět podle jejich kapacit, fixní termíny (nutné pro předměty

[4]Viz Rozhodnutí děkana FIT č. 54/2012, článek 12, bod 5.

zajišťované FEKT[5]) a další.

Mezi požadovaná *měkká kritéria* patří minimalizace počtu studentů s více zkouškami za den, maximum volných dnů mezi zkouškami, dostatečný odstup jednotlivých termínů zkoušek (o kolik jsou větší než tvrdé kritérium minimálních rozestupů), časové preference přednášejících, v případě rozvrhů přednášek dále koncentrace výuky studentů jen do některých dnů, obědové pauzy a další. Protože je potřeba optimalizovat více než jedno měkké kritérium současně, je nutné pohlížet na řešení tohoto problému jako na *multikriteriální optimalizační úlohu*.

3. Systém pro plánování rozvrhů na FIT

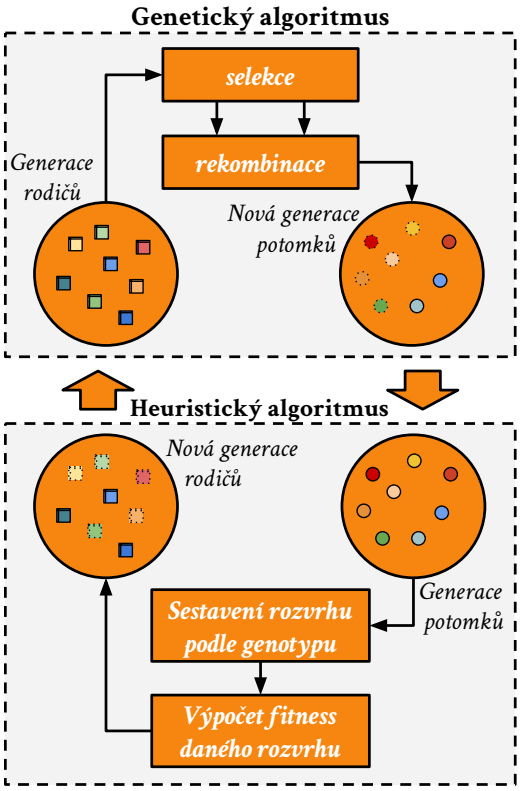
Návrh programu vychází z článku *Solving Timetabling Problem Using Genetic and Heuristic Algorithms* [8] z roku 2007. Ten navrhuje obecnou šablonu pro řešení rozvrhovacích problémů za pomoci hybridního algoritmu, který kombinuje genetické a heuristické postupy. Šablona byla využita jako obecný základ zde prezentovaného systému.

Vstupem algoritmu je množina *faktorů*, které je třeba naplánovat do dostupných *zdrojů*. Pod pojmem *faktor* se míní trojice *předmět*, množina participujících *studentů* a množina *přednášejících*. Zaveďme pojem *časový slot* jako dvojici den a hodina. *Zdroj* poté odpovídá prvku z kartézského součinu *časových slotů* a *místností* (viz obrázek 2). Chromozomy jedinců jsou pak kódovány právě jako permutace jednotlivých faktorů.

Pseudokód hybridního algoritmu znázorňuje algoritmus 1. Na obrázku 1 je navíc zobrazeno schéma spolupráce mezi genetickým a heuristickým algoritmem. Pro každou generaci se provádí výpočet v obou z nich následovně. Rodičovská generace je v první fázi pomocí genetického algoritmu transformována do generace potomků (horní oblast obrázku, řádek 4 algoritmu). V následné druhé fázi probíhá mapování genotypu těchto potomků na odpovídající fenotypy heuristickým algoritmem (spodní oblast, řádek 5). Takto získaný fenotyp jedince již reprezentuje konkrétní rozvrh a je možné vyhodnotit jeho *fitness*.

Následně pokračuje iterace genetického algoritmu. Provede se selekce, čímž se potomci stávají novými rodiči a obě fáze se opakují. V případě, že je výpočet u konce (je dosaženo řešení požadované kvality nebo zvoleného počtu generací), jsou vybráni nejlepší jedinci, kteří budou zobrazeni uživateli.

[5] Výuku vybraných předmětů na *FIT* zajišťují některé fakulty VUT v Brně. Jedná se například o *FEKT* (Fakulta elektrotechniky a komunikačních technologií). Tyto fakulty si vytváří vlastní rozvrhy, jejichž termínům výuky je třeba se přizpůsobit.



Obrázek 1. Schéma navrženého algoritmu. Genetická část (nahoře) v každé generaci využívá heuristickou (dole), aby ohodnotila kvalitu jedinců v populaci.

Rozdíl mezi generováním zkouškového nebo přednáškového rozvrhu na *FIT* se projevuje pouze změnou kritérií (umístění předmětů pouze do jednoho týdne namísto přibližně pětí týdnů, hlídání obědových pauz, koncentrování výuky jen do některých dnů namísto snahy o rovnoměrné rozložení a další), podle kterých jsou konstruovány rozvrhy v heuristické části.

Algoritmus 1: Pseudokód prezentovaného hybridního algoritmu. Oranžově jsou zvýrazněny výpočty v rámci jeho heuristické části.

```
1  Generování počáteční populace;
2  for požadovaný počet generací do
3      if not první iterace cyklu then
4          Generování potomků (rekombinace);
5      Sestavení rozvrhů (genotyp → fenotyp);
6      Určení fitness jedinců (jejich fenotypů);
7      Selekce rodičů (NSGA-II);
8      if Nalezeno dostatečně kvalitní řešení then
9          break;
10 return množina nejlepších řešení, které si
   vzájemně nedominují
```

3.1 První fáze – genetický algoritmus

Permutace jednotlivých faktorů (chromozom jedince) reprezentuje pořadí, ve kterém budou faktory do rozvrhu postupně vkládány heuristickou částí. Předpokladem je, že nejobtížněji umístitelné předměty se budou v čase dostávat na přední pozice v permutaci a naopak. Počáteční populaci je možné generovat náhodně. Bylo však ukázáno, že po vložení několika vhodně vytvořených permutací, může algoritmus dosáhnout řešení rychleji [8].

Šablona [8] nepředepisuje konkrétní typy rekombinačních operátorů ani způsob selekce jedinců. Protože problém plánování rozvrhů na *FIT* je *multikriteriální optimalizační úloha* (viz kapitola 2), jeví se vhodné pro její řešení použít multikriteriální optimalizační algoritmus. V našem případě byl zvolen jeden ze standardních algoritmů, konkrétně *NSGA-II (Non-dominated Sorting Genetic Algorithm II)* [7].

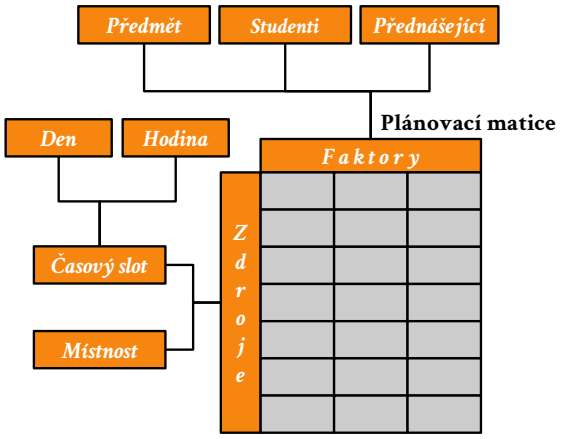
Pro větší množství měkkých kritérií byl zvažován také NSGA-III [2] z důvodu, že u vybraného NSGA-II může dojít k významnému poklesu selekčního tlaku v závislosti na narůstajícím množství kritérií. V průběhu řešení se však ukázalo, že jednodušší algoritmus je dostačující, protože bylo možné některá měkká kritéria agregovat do jednoho a jiná transformovat na tvrdá.

Součástí *NSGA-II* je již způsob selekce jedinců pro rekombinaci. Operátory pro mutaci a křížení však nejsou striktně určeny. V případě permutačního kódování probíhá běžně mutace záměnou dvou genů chromozomu. K tomuto postupu byl navíc ještě přidán inverzní operátor – jeho aplikace obrátí pořadí permutace [5].

V případě operátorů křížení lze jen těžko předpovídat jejich vliv na průběh evoluce daného problému. Byly proto vybrány tři možné permutační operátory křížení, které se jeví jako vhodné. Jedná se o *křížení s částečným přiřazením (PMX – partially matched crossover)* [5], *křížení se zachováním pořadí (OX – order crossover)* [1] a *křížení založené na zachování pozice (PBX – position based crossover)* [1].

3.2 Druhá fáze – heuristický algoritmus

Heuristická část algoritmu sestavuje výsledný rozvrh ve formě dvoudimenzionální matice nazývané *plánovací matice (target matrix)*, viz obrázek 2. Její prvky odpovídají kartézskému součinu všech *faktorů* (sloupce) a *zdrojů* (řádky). Každá její buňka tak reprezentuje vztah mezi jedním faktorem a právě jedním zdrojem. V průběhu generování rozvrhu jsou do nich zanášeny informace tak, aby se dalo vždy jednoduše zjistit, jak přiřazení faktoru ke zdroji ovlivní jednotlivá kritéria. Buňky matice z toho důvodu nabývají právě jednu ze tří hodnot:



Obrázek 2. Hierarchie prvků plánovací matice.

Prázdná: Zdroj je dostupný pro přiřazení faktoru (výchozí hodnota).

Nedostupná: Zdroj nemůže být přiřazen faktoru, protože by takové přiřazení způsobilo tvrdou kolizi.

Přiřazená: Zdroj je přiřazen faktoru.

Pro faktory, jejichž doba trvání překračuje jeden časový slot (v případě *FIT* se jedná o naprostou většinu faktorů, tj. přednáška nebo zkouška trvá dvě a více hodin), je nutné obsadit po sobě jdoucí zdroje. To jsou zdroje, které obsahují stejnou místnost a jejichž časové sloty na sebe navazují v průběhu jednoho dne. Aby se takovéto zdroje v plánovací matici dobře vyhledávaly, je vhodné, aby v ní po sobě vždy následovaly. Toho dosáhneme seřazením zdrojů primárně podle místností a sekundárně podle časových slotů.

Před spuštěním jsou na příslušná místa plánovací matice zanesena ta kritéria, která apriori nedovolují přiřadit některé zdroje k některým faktorům – dané buňky se označí jako „nedostupné“. V dalším kroku jsou do plánovací matice vloženy faktory, jejichž pozici zvolil napevno uživatel (např. předměty organizované FEKT).

Heuristický algoritmus poté vybírá faktory v pořadí podle hodnot genotypu daného jedince. Pro každý z nich prochází příslušný sloupec matice a hledá posloupnost zdrojů v jednom dni s hodnotou „prázdné“. Velikost hledané posloupnosti odpovídá počtu časových slotů, které faktor vyžaduje.

V okamžiku kdy posloupnost nalezne, označí buňky odpovídající faktoru a nalezeným zdrojům hodnotou „přiřazené“. Všem ostatním buňkám, které jsou ve stejných řádcích, přiřadí hodnotu „nedostupné“ (zdroje jsou tímto blokovány pro ostatní faktory). Stejně tak označí hodnotou „nedostupné“ i všechny buňky, ve kterých se vyskytuje zdroj se stejným časovým slotem a faktor se stejným přednášejícím. Pokud by k tomuto nedošlo, mohlo by být porušeno kritérium, kdy jeden přednášející nemůže přednášet dva různé předměty

současně.

Ve všech případech, kdy algoritmus označuje buňku hodnotou „nedostupná“, má tato buňka hodnotu „prázdná“ nebo hodnotu „nedostupná“ již nabývá. Buňka označovaná za nedostupnou nemůže nikdy mít hodnotu „přiřazená“, protože by to znamenalo, že poslední přiřazení porušilo tvrdé kritérium, a tedy že plánovací matice byla před tímto přiřazením v nekonzistentním stavu.

Analogickým způsobem lze do plánovací matice značit i měkká kritéria. V případě jejich porušení však nedochází k ukončení výpočtu. Značení měkkých kritérií slouží pouze k tomu, aby algoritmus v případě nalezení více přípustných posloupností zdrojů vybral tu, která poruší měkká kritéria nejméně.

4. Implementace v jazyce Python 3

Podle uvedeného návrhu byla implementována objektově orientovaná konzolová aplikace a nad ní bylo postaveno odpovídající grafické uživatelské rozhraní. Jako implementační jazyk byl, především z důvodu možnosti rychlého prototypování pro ověření navrženého konceptu, zvolen Python 3. Díky němu je mimo jiné aplikace přenositelná mezi operačními systémy typu Unix nebo Windows.

V případě, že by se výpočetní náročnost algoritmu ukázala jako kritická, umožňuje navíc Python 3 vytvoření optimalizovaného modulu v jazyce C/C++. Jako vhodný kandidát pro tuto optimalizaci se jeví třída pro obsluhu plánovací matice (modul `target_matrix`), která tvoří jádro aplikace a spotřebovává asi 90 % výpočetních prostředků.

Sestavování rozvrhů je však na sobě v rámci jedné generace nezávislé a bylo je tak možné jednoduše paralelizovat do uživatelem zvoleného počtu procesů. V případě použití interpretu `PyPy` (namísto klasického `CPythonu`), lze navíc celý výpočet urychlit přibližně 2,5krát. Díky dalším optimalizacím jako například recyklaci obsahu plánovací matice nebo vhodnému pořadí při jejím prohledávání bylo dosaženo celkového přibližně 10násobného zrychlení bez ohledu na stupeň paralelizace. Takovéto zrychlení je dostatečné k tomu, aby byl na běžném počítači s dostatečnou rezervou získán rozvrh do fakultou požadovaných 24 hodin – a to již za desítky minut.

5. Výsledné rozvrhy

Implementovaná aplikace byla otestována na reálných datech z *Fakulty informačních technologií* – konkrétně se jednalo o zkouškové rozvrhy pro zimní a letní semestr z akademického roku 2014/2015.

Protože každý v praxi nasaditelný rozvrh musí splňovat všechna tvrdá kritéria, nemá dále smysl porovnávat míru jejich splnění – všechny výsledné rozvrhy je totiž bez vyjímky splňují. Kvalitu rozvrhů je však možné porovnat na základě splnění měkkých kritérií, konkrétně:

Denní kolize (C): Celkový součet počtu studentů majících více než jednu zkoušku za den (požadována minimalizace).

Rozestupy termínů (S): Celkový součet denních vzdáleností jednotlivých termínů zkoušek vážený počtem studentů (požadována maximalizace).

Nechť je nad množinou faktorů definováno libovolné lineární uspořádání. Exaktní výpočet počtu kolizí je pak uveden v rovnici 1. Pro každý den d a v něm naplánované faktory f_1 a f_2 je sečtena hodnota z kolizní matice M v případě, že f_1 *předchází* f_2.

$$C = \sum_{d \in days} \sum_{f_1, f_2 \in d} \begin{cases} M(f_1, f_2) & f_1 < f_2 \\ 0 & f_1 \geq f_2 \end{cases} \quad (1)$$

Nechť je nad množinou faktorů definována silně antisymetrická a antitranzitivní relace $\prec$. Dva faktory jsou v této relaci právě a jen tehdy když číslo termínu prvního je o jedna nižší než druhého (všechny první termíny jsou tedy v relaci s druhými atp.). Přesný výpočet *rozestupu termínů* je pak uveden v rovnici 2. Pro každý předmět s z množiny všech předmětů S a jeho naplánované faktory f_1 a f_2, které jsou v relaci $\prec$, sečti jejich hodnotu funkce $space(f_1, f_2)$. Ta je definována jako velikost rozestupu dnů, kdy jsou naplánovány, váženou počtem studentů druhého faktoru.

$$space(f_1, f_2) : \left| f_2.day - f_1.day \right| \cdot f_2.students$$

$$S = \sum_{s \in S} \sum_{f_1, f_2 \in s} \begin{cases} space(f_1, f_2) & f_1 \prec f_2 \\ 0 & f_1 \not\prec f_2 \end{cases} \quad (2)$$

Ve výčtu není uvedeno kritérium pro *časové preference přednášejících* z důvodu, že jsou v maximální míře splněna u všech uváděných řešení – jejich kvalita je tedy v tomto ohledu vyrovnaná.

V případě zimního semestru bylo možné k evaluaci využít hotový rozvrh, podle kterého se rozpis zkouškového období reálně řídil. U něj byla určena hodnota *denních kolizí* na 1 677 a *rozestup termínů* na 63 552. Prezentovaná aplikace byla u tohoto problému schopna dosáhnout hodnot 1 068 pro *denní kolize*

(o 36,3 % lepší výsledek) a 75 366 pro *rozestupy* (což je o 18,6 % lepší výsledek)[6].

Zkouškový rozvrh pro letní semestr nemá ještě v době psaní tohoto článku finální podobu, lze se však porovnat s jeho předběžnou variantou. Pro ni zjištěná hodnota *denních kolizí* je 1 027 a *rozestup termínů* je 74 531. Prezentovaná aplikace byla u tohoto problému schopna dosáhnout hodnot 530 pro *denní kolize* (o 48,4 % lepší) a 81 911 pro *rozestupy* (o 9,9 % lepší).

6. Závěr

V této práci byl popsán systém pro automatizované generování rozvrhů přizpůsobený pro specifické potřeby *Fakulty informačních technologií VUT v Brně*. Program je navržen dostatečně obecně, aby umožnil generovat rozrhy zkoušek i přednášek, a zároveň implementován natolik efektivně, že je možné získat výsledné rozvrhy již za 20 minut běhu na fakultním serveru.

Na základě reálných dat fakulty byly vygenerovány rozvrhy, které obsahovaly přibližně o 40 % denních kolizí méně a přibližně o 15 % větší rozestupy mezi jednotlivými zkouškovými termíny než u reálně použitých rozvrhů.

Hlavním účelem tohoto systému je zjednodušit práci lidem, kteří mají tvorbu rozvrhů na *FIT* na starost, a umožnit jim vytvářet co nejvýhodnější rozvrhy s ohledem na zadaná kritéria tak, aby maximálně vyhovovaly vyučujicím (časové preference, doba potřebná k opravení písemných prací, apod.) a zároveň studentům (minimum denních kolizí či zkoušek z povinných předmětů v jednom týdnu, apod.).

Poděkování

Chtěl bych poděkovat Ing. Michaele Šikulové za vřelý přístup, cenné rady, připomínky a podněty k této práci.

Tato práce byla podporována projektem Vysokého učení technického v Brně FIT-S-14-2297.

Literatura

[1] Aguado, F.; Doncel, J.; Molinelli, J.; aj.: Certified Genetic Algorithms: Crossover Operators for Permutations. In *Computer Aided Systems Theory – EUROCAST 2007*, Springer Berlin / Heidelberg, 2007, s. 282-289, ISBN 978-3-540-75866-2.

[2] Deb, K.; Jain, H.: An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints. *IEEE Trans. Evolutionary Computation*, ročník 18, č. 4, 2014, s. 577-601, DOI:10.1109/TEVC.2013.2281535.

[3] Fang, H.-L.: *Genetic Algorithms in Timetabling and Scheduling*. Dizertační práce, Department of Artificial Intelligence, University of Edinburgh, 1994, 227 s.

[4] Green, C. D.: The Generalisation and Solving of Timetable Scheduling Problems. In *Practical Handbook of Genetic Algorithms: Complex Coding Systems*, CRC Press, 1998, s. 17-64, ISBN 0-8493-2539-0.

[5] Hynek, J.: *Genetické algoritmy a genetické programování*. Grada Publishing, a.s., 2008, 182 s., ISBN 978-80-247-2695-3.

[6] Kubalcová, M.: *Porovnání programů pro plánování rozvrhů a zkoušek*. Bakalářská práce, Vysoké učení technické v Brně, 2012, 50 s.

[7] Shi, C.; Chen, M.; Shi, Z.: A Fast Nondominated Sorting Algorithm. In *ICNN B '05. International Conference on Neural Networks and Brain, 2005.*, 2005, s. 1605-1610, ISBN 0-7803-9422-4.

[8] Thanh, N. D.: Solving Timetabling Problem Using Genetic and Heuristic Algorithms. In *Eigth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2007. SNPD 2007.*, 2007, s. 472-477, ISBN 978-0-7695-2909-7.

[6] Vzhledem k tomu, že výstupem algoritmu NSGA-II je celá *Paretova fronta* jedinců, existují k uvedeným řešením i další, která si vzájemně nedominují.