VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

GENEROVÁNÍ ŠKOLNÍCH ROZVRHŮ

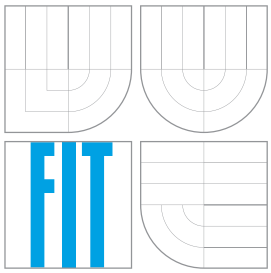
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAN FIALA

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

GENEROVÁNÍ ŠKOLNÍCH ROZVRHŮ

GENERATING OF SCHOOL SCHEDULES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JAN FIALA
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAROSLAV ROZMAN, Ph.D.
SUPERVISOR

BRNO 2012

Abstrakt

Tato práce se zabývá problémem generování školních rozvrhů. Řešení je založené na třech různých heuristických algoritmech (horolezecký algoritmus, simulované žíhání, genetický algoritmus) a je celé implementováno v jazyce Java. Přináší srovnání implementovaných heuristických algoritmů včetně popisu jejich výhod a nevýhod.

Abstract

This work deals with problem of generatich school schedules. The solution is based on three heuristic algorithms (hill-climbing, simulated annealing, genetic algorithm) and is fully implemented in JAVA. It provides a comparison of implemented heuristic algorithms including desccription of their pros and cons.

Klíčová slova

rozvrhování, školní rozvrh, omezující podmínky, heuristické metody, genetický algoritmus, JAVA, XML

Keywords

scheduling, school timetable, constraints, heuristic methods, generic algorithm, JAVA, XML

Citace

Jan Fiala: Generování školních rozvrhů, bakalářská práce, Brno, FIT VUT v Brně, 2012

Generování školních rozvrhů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Jaroslava Rozmana, Ph.D.. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jan Fiala
14. května 2012

Poděkování

Tímto bych chtěl velmi poděkovat mému vedoucímu Ing. Jaroslavu Rozmanovi, Ph.D. za ochotu při konzultacích a dobré rady. Dále bych chtěl poděkovat mé rodině a blízkým za podporu.

Obsah

Kapitola 1

Úvod

Tvorba školního rozvrhu není vůbec jednoduchá záležitost. Dříve se tento proces prováděl ručně tzv. „lístečkovou“ metodou. Nešlo o nic jiného, než o zkoušení různých kombinací do té doby, než rozvrh splňoval všechny podmínky vycházející z možností učitelů a učeben. Když byl rozvrh hotov, zpravidla nastala nečekaná změna podmínek učitelů nebo učeben a celý proces tvorby mohl začít znovu. Nástup moderní výpočetní techniky a nových heuristických metod vedl k zamyšlení, zda by nebylo možné celý proces zautomatizovat a významně tak uspořit čas a usnadnit práci lidem zodpovědným za tvorbu rozvrhu.

Dnes už se k tomuto účelu na většině škol používá speciální software. Odpadá tedy problém časové náročnosti a zároveň to umožňuje pružné reakce na změny podmínek. Počet dnešních kvalitních programů, které se zaměřují na generování školních rozvrhů, není nikterak velký. Přece jen se jedná o dosti specializovanou činnost. Firmy, které se o školní rozvrhování zajímají, se často snaží nabídnout i něco víc, než jen generování školních rozvrhů. Zaměřují se na celý školní informační systém jako takový (třídní kniha, žákovské knížky, suplování, ...).

Na vytváření školních rozvrhů neexistuje ideální algoritmus. Programy si k tomuto účelu volí většinou různé optimalizační algoritmy, které obohacují o své další vlastní heuristiky a vylepšení.

Cílem této práce je návrh a implementace aplikace, která bude moci zastoupit složitou ruční tvorbu školních rozvrhů. Generování rozvrhů bude moci probíhat pomocí tří různých optimalizačních algoritmů a v praktické části bude uvedeno jejich srovnání. Motivací je tedy úspora času osoby zodpovědné za přípravu rozvrhu, která tímto úkolem trávila i několik dnů. Práce se zabývá pouze klasickými rozvrhy pro základní a střední školy.

V druhé kapitole je vysvětlen obecný pojem rozvrhování a všechny pojmy s ním související. Nechybí zde i popis nejčastěji řešených problémů v praxi. Třetí kapitola se věnuje popisu různých optimalizačních algoritmů, přičemž jsou vyjmenovány jejich výhody i nevýhody. Čtvrtá kapitola se zaměřuje na rozvrhování školních rozvrhů. V páté kapitole je uveden popis existujících programů, které se problémem generování školních rozvrhů zabývají, jejich klady i zápory. Analýza a návrh cílové aplikace je uvedena v kapitole šesté. A samotný popis řešení a implementace aplikace se nachází v kapitole sedmé. Následuje už jen shrnutí dosažených výsledků a závěr.

Kapitola 2

Rozvrhování

V této kapitole je popsána oblast umělé inteligence zvaná rozvrhování a nejčastěji řešené problémy v ní. Nejprve je nutné zavést několik základní pojmů [9].

Zdroj je označení pro lidskou nebo strojovou pracovní sílu. Může mít různé parametry nebo omezení, např. cenu za vykonání práce, nejvyšší možnou rychlost práce, nevhodné termíny pro práci atd.

Úloha/aktivita je vykonávána na nějakém zdroji nebo jej využívá. Velmi důležitý je fakt, že každá aktivita musí trvat předem určenou dobu. Každá úloha obsahuje řadu různých parametrů. Dělíme je na dynamické a statické.

Statické parametry úlohy:

- *doba trvání* - doba provádění úlohy na daném zdroji
- *termíny dostupnosti* - seznam časových slotů, ve kterých může být aktivita vykonávána
- *termín dokončení* - čas, do kdy by mělo být provádění úlohy dokončeno
- *deadline* - čas, do kdy musí být provádění úlohy dokončeno
- *váha* - důležitost úlohy při porovnání s ostatními

Dynamické parametry úlohy:

- *čas startu úlohy* - čas zahájení provádění úlohy na daném zdroji
- *čas konce úlohy* - čas ukončení provádění úlohy na daném zdroji

Rozvrh je dán umístěním aktivit podle času na konkrétní zdroje. *Úplným* rozvrhem se myslí rozvrh, ve kterém jsou umístěny všechny úlohy ze zadaného problému. Pokud v rozvrhu nejsou umístěny úplně všechny úlohy ze zadaného problému, nazýváme ho *částečný*.

Konzistentním rozvrhem rozumíme takový rozvrh, který splňuje všechny omezující podmínky kladené na zdroje nebo umístěné úlohy. Nabízí se zde tedy otázka, zda je lepší úplný nekonzistentní nebo neúplný konzistentní rozvrh.

Ideálním nazýváme rozvrh, který je nejlepší možný vzhledem k zadanému optimalizačnímu kritériu - např. dosahuje nejvyšší využití zdrojů.

Rozvrhování je disciplína, ve které se snažíme přiřadit zdroje a čas k aktivitám takovým způsobem, abychom dosáhli splnění všech nutných podmínek kladených na výsledný rozvrh a maximalizovali zisk za daných omezení.

Rozvrhování se často plete s pojmem plánování. Plánovaní (scheduling) klade důraz na uspořádání aktivit a minimalizaci celkové ceny daných zdrojů. Kdežto při rozvrhování (timetabling) je nejdůležitější konkrétní umístění aktivit do omezených časových slotů a splnění všech omezujících podmínek.

Při rozvrhování musíme brát v potaz řadu omezujících podmínek. Omezení se mohou týkat zdrojů, času i aktivit. Můžeme je rozdělit na slabé a silné omezení (více v podkapitole 4.2.2). Dále pak musí existovat možnost rozvrhy mezi sebou porovnávat. K tomuto účelu se zavádí tzv. ohodnocení rozvrhu. Ohodnocení rozvrhu provádí ohodnocující funkce. Jde o číselnou hodnotu vyjadřující splnění či nesplnění podmínek kladených na výsledný rozvrh. Rozvrhování tedy řeší optimalizační úlohu, kde se hledá maximální (resp. minimální) hodnota ohodnocující funkce a tedy i ideální rozvrh.

Z hlediska složitosti se jedná o velmi obtížný problém. Při zvyšování počtu zdrojů, aktivit a času dochází totiž k velmi velkému zvětšování stavového prostoru. Ve výsledku jsou stavové prostory u těchto úloh tak velké, že je není možné v rozumné době celé prohledat a nalézt ideální řešení. Musíme se tedy spokojit s prohledáním pouze části celého stavového prostoru. V rámci tohoto podprostoru nalezneme nejlepší možné řešení. V praxi se tedy musí volit mezi velikostí prohledaného stavového podprostoru (dobou běhu výpočtu) a kvalitou výsledného rozvrhu. Obecné platí, že k největšímu zlepšování ohodnocení rozvrhu dochází na začátku algoritmu. Ke konci už dochází pouze k malým zlepšením.

2.1 Klasifikace rozvrhovacích problémů

Rozvrhovací problémy se formálně popisují podle tzv. Grahamovy klasifikace [4]. Zapisuje se ve tvaru $\alpha \mid \beta \mid \gamma$. α značí charakteristiky strojů a popisuje, jakým způsobem se na ně alokují úlohy. β popisuje všechny omezení aplikovaná na úlohy a γ jsou optimalizační kritéria.

Charakteristiky strojů obsahují informaci o vztazích mezi nimi. Zda jsou to stroje identické nebo pracují různou rychlostí. Zda pracují paralelně nebo jsou na sobě nezávislé. Z charakteristik úloh lze zmínit např. její trvání. Významných optimalizačním kritériem bývá zpoždění nebo cena za pracovní sílu.

2.2 Složitost

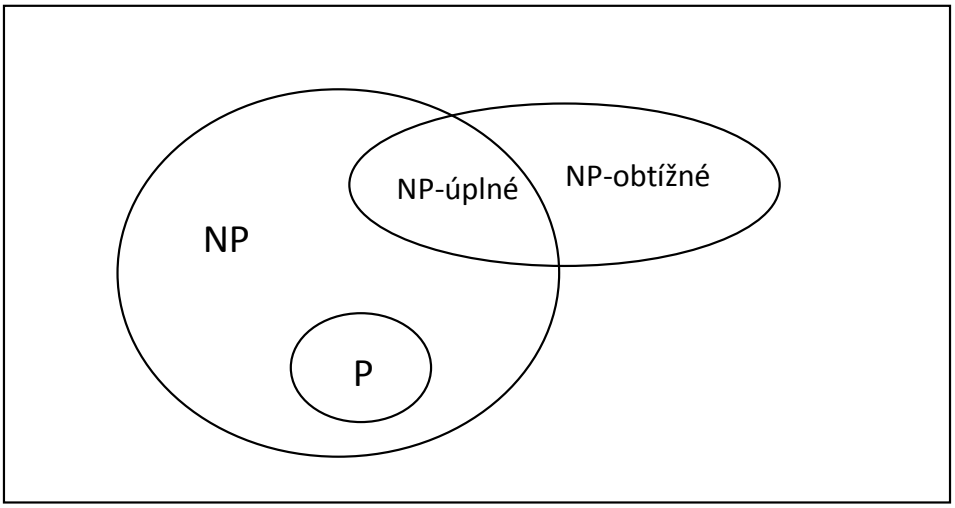
Při řešení úloh pomocí výpočetní techniky je zapotřebí mít nástroj, kterým dokážeme porovnat efektivitu a rychlost vykonávání jednotlivých algoritmů. K tomuto účelu byl zaveden pojem asymtotická složitost.

Asymptotická složitost [13] je způsob klasifikace výpočetních algoritmů. Určuje operační náročnost každého algoritmu podle způsobu, jakým se bude chování algoritmu měnit v závislosti na změně velikosti nebo počtu vstupních dat. Zapisuje se pomocí („*velké O notace*“) jako *O(f(N))* (např. *O(N)* je lineární složitost). Nejčastěji se v praxi používá asymptotická časová a prostorová složitost.

Příklady asymptotických časových složitostí:

- *O(1)* - přístup k prvku pole podle indexu
- *O(N)* - vyhledání prvku v neseřazeném poli linearním vyhledáváním
- $O(log_2 N)$ - vyhledání prvku v seřazeném poli metodou půlení intervalů
- $O(2^N)$ - přesné řešení NP-úplného problému hrubou silou (problém obchodního cestujícího)

Algoritmy se dělí do různých tříd složitosti. Do třídy P patří takové problémy, které řeší nějaký polynomiální algoritmus. Jsou tedy řešitelné v polynomiálním čase na Touringově stroji. Do třídy *NP – obtížných* patří takové problémy, které jsou řešitelné v polynomiáně omezeném čase na nedetrministickém Touringově stroji. Rozvrhování patří do třídy *NP – obtížných* problémů. Nejzajímavější a nejobtížnější problémy se nazývají *NP úplné*.



Obrázek 2.1: Třídy složitosti [3]

2.3 Řešené problémy

Rozvrhování je velmi rozsáhlá oblast, ve které řešíme velkou škálu reálných problémů. Stručně popíšeme alespoň některé z nich. Celá tato práce je zaměřena na problém školních rozvrhů. Tento problém je detailně popsán v kapitole 4.

2.3.1 Rozvrhování směn zaměstnanců

V tomto případě jsou tedy zdroje zaměstnanci a aktivity jsou směny. Jde o přiřazení zaměstnanců ke směnám takovým způsobem, aby bylo dosaženo splnění všech podmínek, které plynou z daného druhu práce. Každý zaměstnanec má určitou klasifikaci, může tedy vykonávat pouze některé směny. Hlavním cílem je zajištění dostatečného množství personálu v průběhu celého pracovního dne. Variabilita různých podmínek je velmi vysoká a vždy závisí na konkrétním druhu vykonávané práce. Častým požadavkem je např. minimalizace počtu přesčasů.

2.3.2 Rozvrhování výroby

Zde je úkolem maximalizace efektivity využití jednotlivých strojů a dosažení výrobních cílů. Škála omezujících podmínek u tohoto problému je velmi vysoká. Např. podmínky související s kapacitou stroje, dobou přesunu produktu k dalšímu stroji, dobou změny konfigurace stroje, vedlejšími produkty atd.

Kapitola 3

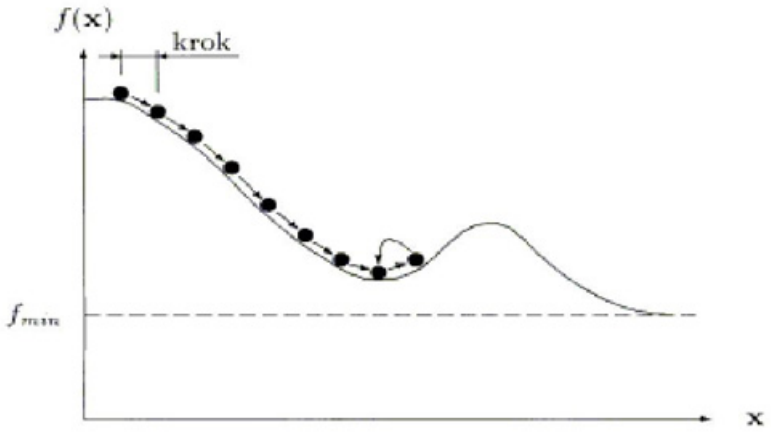
Optimalizační algoritmy

Jedná se o skupinu algoritmů, které se snaží najít řešení náhodným prohledáváním stavového prostoru. Neprocházejí úplně celý stavový prostor, jsou tak schopné nalézt řešení (často neúplné) ve velmi krátké době. Na druhou stranu se však může stát, že model ideální řešení má, nicméně algoritmus ho nenalezne. Všechny tyto algoritmy vychází ze stejného principu a tím je neustálé zlepšování stávajícího řešení [12].

3.1 Hill Climbing (Horolezecký algoritmus)

Tato metoda se řadí mezi to nejjednodušší, co lze k optimalizaci řešení použít. Používá tzv. gradientní techniku. Před startem algoritmu se zvolí počáteční rozvrh, který má nějaké ohodnocení. Následně se prohledávají sousední rozvrhy, a pokud se nalezne nějaký s lepším ohodnocení, vybere se.

Výrazným negativem je možnost uváznutí v lokálním minimu. To znamená, že algoritmus často nenajde ideální rozvrh. Tento jev lze však částečně eliminovat opakovaným spouštěním algoritmu s různými počátečními rozvrhy.

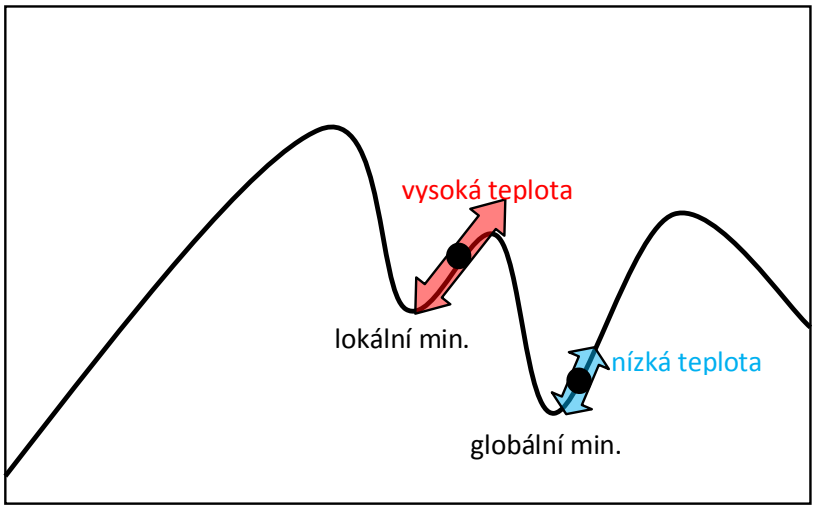


Obrázek 3.1: Uváznutí v lokálním minimu [11]

3.2 Simulated Annealing (Simulované žíhání)

Metoda simulovaného žíhání je inspirována praktickým procesem žíhání oceli, při kterém dochází k zahřátí ocele na velmi vysokou teplotu a následně se teplota pomalu snižuje. Tento princip se využívá také v optimalizačním algoritmu. Algoritmus prochází postupně sousední rozvrhy. Na začátku se nastaví nějaká teplota, která určuje pravděpodobnost, že může být vybrán i rozvrh s horším ohodnocením. Následným snižování teploty tato pravděpodobnost klesá a postupem času už je teplota tak nízká, že v podstatě dochází k přijímání jen těch kvalitnějších rozvrhů.

Tato metoda tedy zabraňuje uváznutí v lokálním minimu tím, že se teplota pomalu snižuje a je tedy možné z něho včas uniknout. Tímto se zvyšuje pravděpodobnost nalezení ideálního rozvrhu.



Obrázek 3.2: Vliv teploty na uváznutí v lokálním minimu

3.3 Tabu Search

Další metodou, která se snaží vyhnout uváznutí v lokálním minimu je Tabu search. Jak již název algoritmu napovídá, bude zde hrát důležitou roli zakazování. Konkrétně jde o zapamatování zakázaných přesunů v rozvrhu. Pokud se pak při procházení narazí na rozvrh, který obsahuje zakázaný přesun, dál se sním již nepracuje. Tímto je vyřešeno jednak zacyklení při prohledávání rozvrhů, ale i uváznutí v lokálním minimu.

Důležitým parametrem je zde paměť. Musí se vhodně zvolit její velikost a způsob jejího chování. Zvolit ideálně parametry paměti tak, aby byl nalezen ideální rozvrh, je velmi obtížné.

3.4 Genetické algoritmy

Tak jako metody z předchozí kapitoly patří Genetické algoritmy k základním optimalizačním metodám [7]. Používají se zejména k řešení složitých problémů, kde neexistuje přesný algoritmus pro nalezené ideálního řešení. Princip těchto algoritmů je založen na Darwinově

teorii evoluce. Snaží se tedy napodobit evoluční procesy jako je dědičnost, přirozený výběr, mutace, křížení a jiné.

V průběhu generování rozvrhu se udržuje tzv. populace. Členy populace jsou jednotlivé rozvrhy s určitým ohodnocením. V každém cyklu algoritmu se podle určitého kritéria vybere několik rozvrhů. Následně se z těchto vybraných rozvrhů pomocí operace křížení, mutace nebo reprodukce, vloží do populace nové rozvrhy.

3.4.1 Princip genetického algoritmu

1. Inicializace - nová populace je obvykle složena z náhodně vygenerovaných rozvrhů
2. Pomocí určité výběrové metody jsou z populace vybrání jedinci podle určitého kritéria, většinou na základě jejich ohodnocení
3. Z vybraných rozvrhů jsou generováni rozvrhy nové pomocí následujících metod:
   (a) *křížení* - prohození částí rozvrhů mezi sebou
   (b) *mutace* - náhodná změna části rozvrhu
   (c) *reprodukce* - kopíruj rozvrh do další populace beze změny
4. Výpočet zdatnosti jednotlivých nově vytvořených rozvrhů
5. Opakuj od bodu dva, pokud není splněna některá ukončovací podmínka
6. Pokud je splněna ukončovací podmínka, vezmi rozvrh s nejlepším ohodnocením a pošli ho jako výstup metody

3.4.2 Křížení

Pomocí operace křížení [10] vznikají v populaci nové rozvrhy. Výběr dvojic rozvrhů, které se budou křížit, by měl být proveden na základě jejich ohodnocení. Tím se dosáhne toho, že budou v populaci vznikat úplně nové kombinace rozvrhů s lepším ohodnocením. Křížení dělíme podle toho, kolik používá významných bodů(crosspoints), na:

1. *Jednobodové* - Máme vybránu dvojici jedinců, kteří se mezi sebou kříží podle jednoho významného bodu. Náhodně se vygeneruje pozice bodu, musí to být celé číslo v intervalu *(0 ; délka chromozomu)*. Proces křížení je znázorněn v tabulce níže. Pro názornost jsou na příkladech použity binární chromozomy.

Původní chromozómy (rodiče)		Nové chromozómy (potomci)
(0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1)	→	(0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1)
(0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1)	→	(0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1)

Tabulka 3.1: Proces křížení pro bod 3

2. *Dvoubodové* - Probíhá stejným způsobem jako u jednobodového. Proces zde využívá dva významné body, takže nový jedinec má vždy dvě částí převzaty od jednoho rodiče a část od druhého. Toto je možné vidět na tabulce níže.
3. *Vícebodové* - Podobným způsobem probíhá i křížení s více významnými body. Při tvoření nových jedinců se musí dávat pozor na to, zda je bod lichý nebo sudý.

Původní chromozómy (rodiče)		Nové chromozómy (potomci)
(0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1)	→	(0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1)
(0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1)	→	(0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0)

Tabulka 3.2: Proces křížení pro body 4 a 7

3.4.3 Mutace

Mutace přináší do genetického algoritmu významnou prvek náhodné změny. Chromozom náhle změní část své informace. Díky tomu se může dosáhnout takových jedinců, které by pouze křížením nikdy nevznikly.

Četnost mutace se musí velmi citlivě nastavovat. U jedinců s nejlepším ohodnocením by mutace nastat neměla. Při inicializaci algoritmu se četnost mutace většinou nastavuje na výšší hodnotu a postupně se snižuje. Ze začátku se tedy bude prohledávat spíše do šířky, později už se snažíme vylepšit nadějná řešení.

Příklad mutace je uveden na tabulce níže.

Chromozóm před mutací:	(1, 1, 0, 0, 1, 0, 1, 1, 1, 1)
Chromozóm po mutaci:	(1, 1, 0, 0, 1, 0, 1, 0, 1, 1)

Tabulka 3.3: Proces mutace na pozici 8

Kapitola 4

Problém školního rozvrhování

Rozvrhování je disciplína, ve které se snažíme přiřadit zdroje a čas k aktivitám takovým způsobem, abychom dosáhli splnění všech nutných podmínek kladených na výsledný rozvrh.

V případě školních rozvrhů jsou zdroje učitelé. Aktivity jsou vyučování předmětů danými učiteli [8].

4.1 Vlastnosti školních rozvrhů

Školní rozvrh pro žáky základních a středních škol je většinou po celý školní rok stejný. Stejně tak většinou platí, že všichni žáci dané třídy mají rozvrh stejný (toto je hlavní rozdíl od rozvrhů pro studenty vysokých škol). Studijní týden začíná vždy v pondělí a končí pátkem. Vyučovací hodiny jdou za sebou s přestávkami. Může však nastat situace, kdy mají žáci mezi některými hodinami volno. Také se může stát, že předmět nebude vyučován každý týden, ale pouze v týdny sudé či liché. Vybavení jednotlivých učeben je také zásadní kritérium ovlivňující výslednou podobu školního rozvrhu.

	0	1	2	3	4	5	6	7
Po		čj Ham	Aj Pav / Got	Čjs Bar	M Ham	čj Ham		
Út		Uak H Po	Tv Tv Ham	M Ham	čj Ham	čj Ham		Čjs Bar
St		Čjs Bar	čj Ham	M Ham	Aj Pav / Got	Uak Ham		
Čt		čj Ham	Inf Vt Pra	Čjs Bar	M Ham	Pč Ham		
Pá		Aj Pav / Got	čj Ham	M Ham	Tv Ham	čj Ham		

Obrázek 4.1: Školní rozvrh pro třídu základní školy

4.2 Návrh školního rozvrhu

Návrh školního rozvrhu je zadání všech potřebných údajů, podmínek a vztahů mezi nimi tak, aby z nich mohl speciální software vytvořit výsledný rozvrh. Ne vždy je však možné nalézt ideální řešení. V takovém případě má zadavatel dvě možnosti. Buď rozvrh dokončí sám ručně (bude při tom muset porušit některé zadané podmínky), nebo lehce poupraví vstupní podmínky a nechá program znovu vytvořit rozvrh.

4.2.1 Zadání údajů

Jde vlastně o naplnění vnitřní databáze programu údaji, potřebnými k vytvoření rozvrhu. V první řadě se musí vyplnit seznamy učitelů, předmětů, tříd a učeben. Následně se naváží mezi těmito údaji vztahy. Např. které předměty učitel učí, jaké předměty bude mít určitý ročník (resp. třída) atd.

4.2.2 Zadání podmínek

Pestrost vstupních podmínek je velmi velká. Dělí se na slabé (soft conditions) a silné (hard conditions).

Silné podmínky musí být ve výsledném rozvrhu vždy splněny. Pokud je ve výsledném rozvrhu porušena byť jen jedna tato podmínka, rozvrh je chybný. Typickou silnou podmínkou ve školním rozvrhu je, že učitel nemůže učit více hodin ve stejném čase, třída nemůže mít více hodin ve stejném čase atd.

Slabé podmínky nemusí být na rozdíl od těch silných splněny. Snažíme se jich však splnit co nejvíce a získat tak ideální řešení. V praxi je na školní rozvrh kladeno velké množství slabých podmínek, nicméně většinou je nemožné je všechny splnit. Do těchto typů podmínek tedy zadáváme spíše přání, při jejichž nesplnění se toho zase tolik nestane.

Obecně platí, že na každé škole mohou být tyto podmínky různé. Níže jsou vypsány příklady některých podmínek:

- Kolik hodin může učit denně, případně za celý týden
- V které dny chce/může učit, případně v které nemůže
- Jestli chce mít ve svém rozvrhu mezery
- V kterou hodinu učit nemůže

4.3 Tvorba školního rozvrhu

Po dokončení návrhu rozvrhu přichází na řadu samotná tvorba. Tu už provádí program, a tak by se mohlo zdát, že pro uživatele je celá věc dokončena. V některých případech tomu tak skutečně je, avšak ve složitějších rozvrzích nikoliv. V nich je zadání podmínek tak rozsáhlé, že není možno vygenerovat ideální rozvrh, který by je všechny splňoval. V takovém případě program zahlásí konflikty a je jen na uživateli, jak bude postupovat dál.

Pokud se jedná jen o jednoduchý konflikt, může se pokusit vzít nedokončený rozvrh a dokončit ho ručně. Zde je však jasné, že se nevyhne porušení některých podmínek. Může se ale vrátit k návrhu rozvrhu a poměnit intuitivně podmínky, které s konflikty souvisely. U obou možností musí uživatel sáhnout k určitému kompromisu a změnit/zrušit podmínky, u kterých si to může dovolit.

Jak je vidět úspěšný proces tvorby rozvrhu silně závisí na nekonfliktním návrhu. Samozřejmě platí, čím je větší počet podmínek a tím složitější rozvrh, tím je větší pravděpodobnost, že nastanou konflikty.

Kapitola 5

Rešerše existujících programů

Ke tvorbě školních rozvrhů je možno dnes použít několik programů. Většina jich je komerčních a jsou součástí celých školních systémů. Najdou se však i jednoduché bezplatné aplikace, které na tvorbu jednoduchých školních rozvrhů stačí.

5.1 Tvůrce rozvrhů

5.1.1 Popis aplikace

Program Tvůrce rozvrhů [6] umožňuje plně automatickou tvorbu školních rozvrhů s využitím techniky genetických algoritmů. Generuje rozvrhy ve dvou verzích, pro třídy a pro jednotlivé učitele. Výsledné rozvrhy umožňuje uložit ve formátu TXT i HTML. Zadávání údajů do databáze je uživatelsky přívětivé.

5.1.2 Použité algoritmy

Jak již bylo řečeno výše, program využívá ke tvorbě rozvrhů genetické algoritmy. Proces generování rozvrhu umožňuje libovolně ovládat. Umožňuje také nastavit některé parametry genetického algoritmu (např. počet generací, pravděpodobnost křížení atd.).

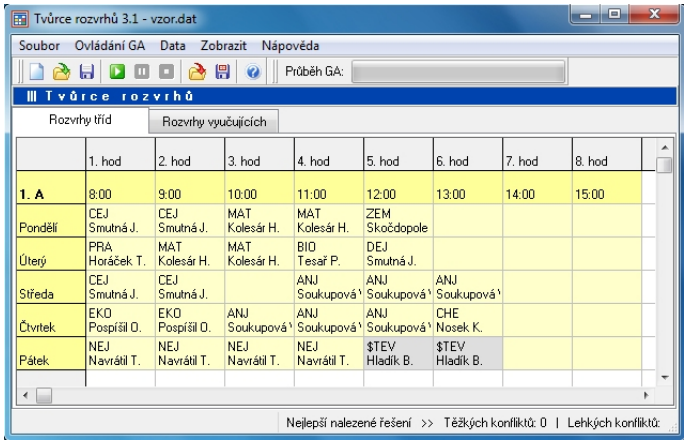
Rozlišuje dva druhy podmínek, slabé (volitelné) a silné (povinné).

Silné podmínky (povinné):

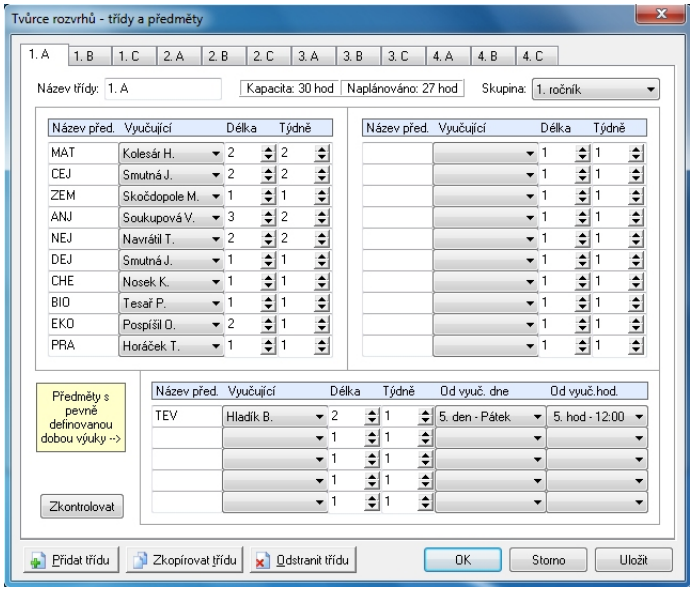
- žádný učitel nesmí v daném čase (vyučovací hodině) učit více než jeden předmět.

Slabé podmínky (nepovinné):

- maximální přípustný počet výskytů stejného předmětu v daném dnu,
- maximální přípustná mezera v rozvrhu,
- maximální přípustná mezera na začátku dne,
- maximální přípustný počet hodin denně,
- minimální přípustný počet hodin denně.



Obrázek 5.1: Tvůrce rozvrhů - základní rozhraní



Obrázek 5.2: Tvůrce rozvrhů - přiřazení učitelů

5.1.3 Zhodnocení

Jedná se o velmi jednoduchý a intuitivní program. Jeho hlavní výhodou je fakt, že je poskytován zcela zdarma. Nabízí mnoho různých nastavení, dokonce i samotného genetického algoritmu. Bohužel v sobě nemá zakomponovanou podporu pro místnosti, pracuje tedy jen s třídami, předměty a učiteli. Tento program se hodí zejména pro základní a malé střední školy.

5.2 aSc rozvrhy

5.2.1 Popis aplikace

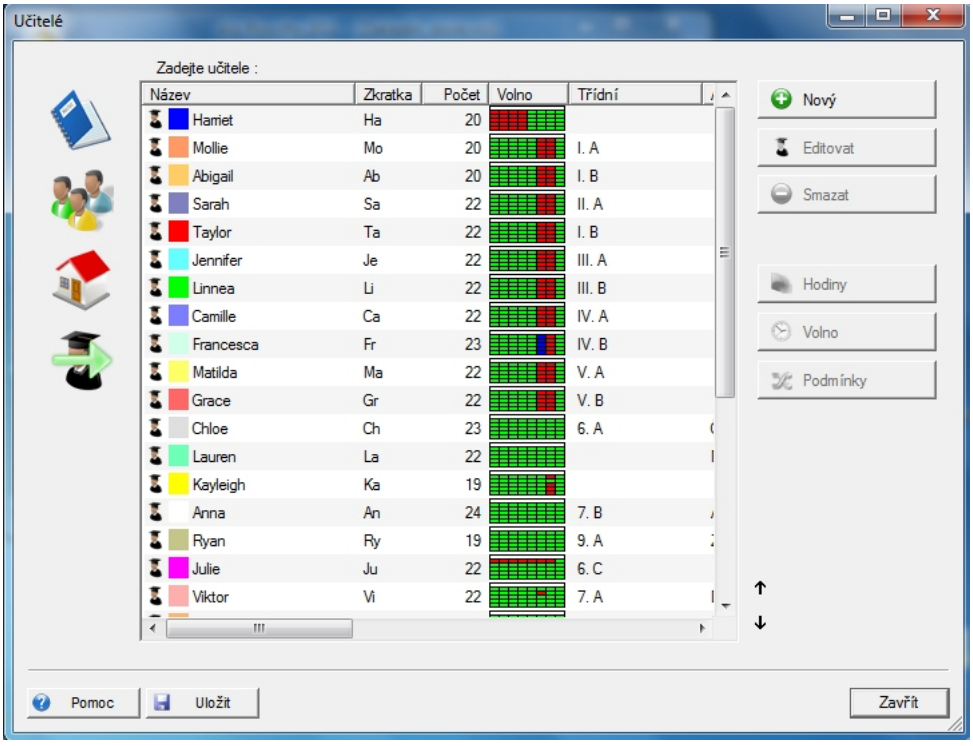
Program aSc rozvrhy [1] se na automatickou tvorbu školních rozvrhů zaměřuje poněkud komplexněji. Jedná se již o profesionální software, který podporuje složitější nastavení (např. skupiny v rámci třídy, více-týdenní cykly atd.). K dispozici je zde proto textový průvodce, který má za úkol uživatele s jednotlivými nastaveními seznámit.

Vygenerované rozvrhy lze exportovat do MS Excel, XML, HTML. Tiskové sestavy je možno velmi dobře upravovat. Součástí tohoto programu je také modul aSc suplování, který řeší problém suplování v již vygenerovaném rozvrhu.

5.2.2 Použité algoritmy

Tato aplikace používá ke generování rozvrhů svůj vlastní originální algoritmus, který je založen na metodě backtracking. Využívá také různé heuristiky a speciální datové struktury optimalizované pro maximální výkon.

Obrázek 5.3: aSc - základní rozhraní



Obrázek 5.4: aSc - přiřazení učitelů

5.2.3 Zhodnocení

Jedná se o profesionální software, který umožňuje velké množství nastavení. Je poskytován ve formě neomezené demoverze, která však neumožňuje tisk rozvrhů.

5.3 Bakaláři

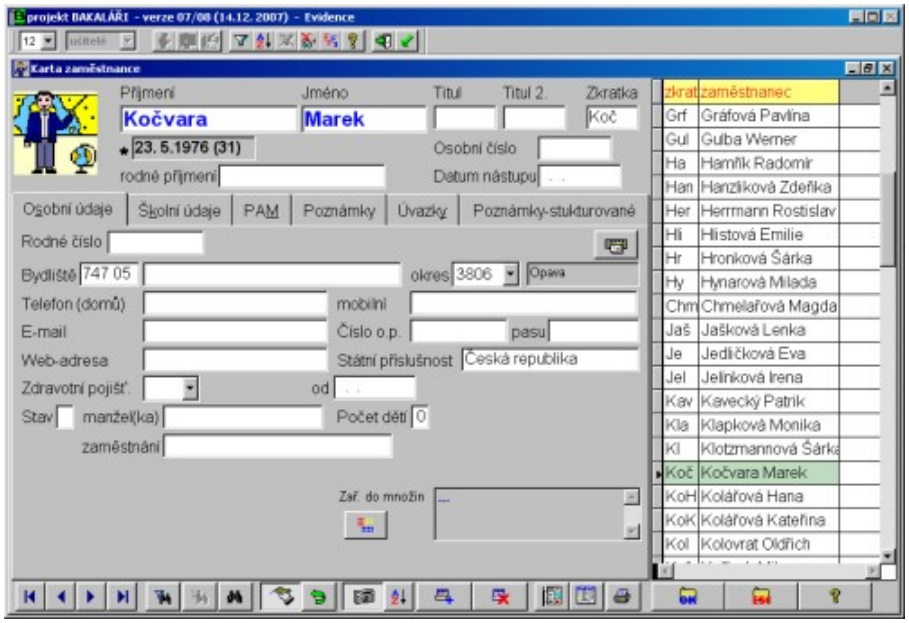
5.3.1 Popis aplikace

Program Bakaláři [2] se zaměřuje na celkovou školní administrativu. Systém zahrnuje evidenci žáků a učitelů, zadávání známek, třídní knihu, generování rozvrhu hodin, suplování a mnoho dalších modulů. Moduly však nelze spouštět samostatně. Vyžadují společné datové prostředí, které obsahuje všechny zadané údaje. Pokud škola ale používá více modulů, data jsou uložena jen na jednom místě. Jednotný systém dat umožňuje dokonalé využití všech údajů. Učitelé mohou například zapisovat známky třídám a skupinám podle úvazků, rozvrhy lze tisknout s odlišnostmi pro jednotlivé žáky atd.

5.3.2 Použité algoritmy

Modul pro generování rozvrhu hodin pracuje na bázi tzv. „lístečků“, které se snaží úspěšně rozmístit do rozvrhu. Při tvorbě rozvrhu se modul zaměřuje na vyhledávání hodin, u nichž by pozdější nasazení způsobilo problémy. To je zásadní odlišnost od jiných generátorů, které obvykle rychle nasadí všechny bezproblémové hodiny, nechají však stranou několik procent neřešitelných. Nasazení může probíhat automaticky (počítač vyhledá nejvhodnější místo v

rozvrhu podle nastavených kritérií), tvorbu rozvrhu však můžeme (s nastavitelnou časovou prodlevou) sledovat a jednoduše do ní vstupovat.



Obrázek 5.5: Bakaláři - zadávání učitelů

5.3.3 Zhodnocení

Bakaláři je plně profesionální, placený školní administrativní systém. Je používán na mnoha českých školách pro svou robustnost a propojenost datového systému.

Kapitola 6

Analýza a návrh

Fáze analýzy a návrhu aplikace je často velmi podceňována. Přitom se jedná o nejdůležitější část projektu. Důkladně promyšlená a správná rozhodnutí v těchto fázích bývají odměnována v průběhu celého řešení projektu. Naopak, chybná rozhodnutí vedou k významnému prodloužení doby vývoje, v některých případech dokonce i k jeho nucenému zrušení.

6.1 Použité technologie

Na začátku samotné analýzy je vhodné zvolit programovací jazyk, ve kterém bude cílová aplikace naimplementována. Výběr programovacího jazyka má zásadní vliv na dobu vývoje. Při výběru se musí brát v potaz požadavky aplikace a charakteristiky programovacího jazyka.

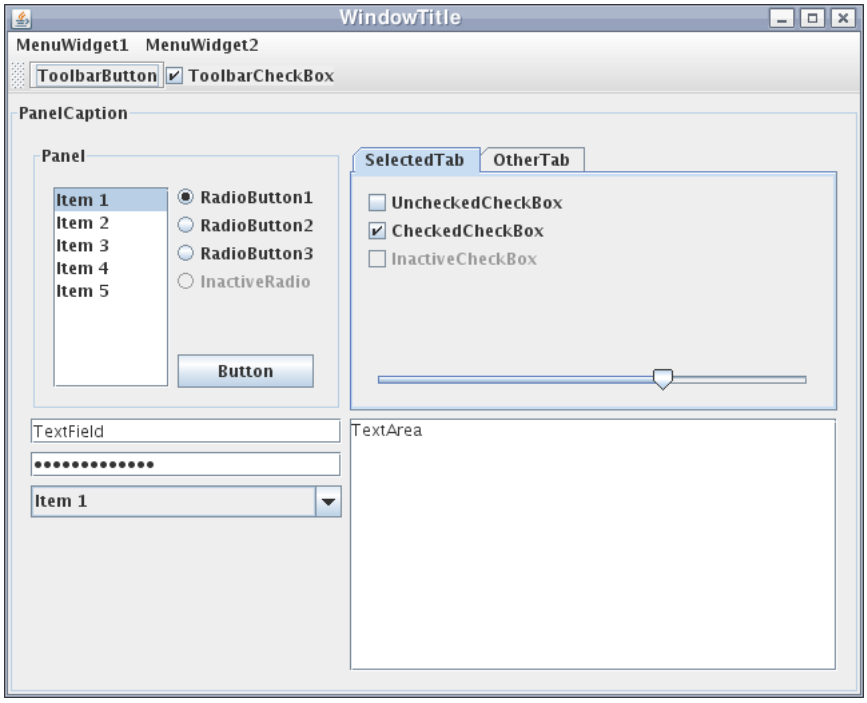
Cílem bylo vytvořit grafickou aplikaci, která bude přenositelná mezi operačními systémy Windows a Linux. Proto byl zvolen programovací jazyk Java. Na zobrazení grafického uživatelského rozhraní byla zvolena knihovna Swing.

6.1.1 Java

Java [5] je populární, univerzální, objektově-orientovaný programovací jazyk vyvinutý firmou Sun Microsystems. Zásadní výhoda Javy je přenositelnost aplikací, které jsou v ní naimplementovány. Namísto klasického přeložení do strojového kódu je každá aplikace za pomocí Java Development Kit (JDK) přeložena do tzv. byte-kódu, který je poté vykonáván v Java Runtime Environment (JRE). JDK i JRE jsou tedy platformě závislé a bežně dostupné pro většinu dnešních platforem. Javě bývá často vytýkán pomalejší start programů a paměťová náročnost, proto byla při provádění programu použita technika překladu Just In Time (JIT).

6.1.2 Swing

Swing je hlavní knihovna uživatelských prvků na platformě Java. Poskytuje aplikační rozhraní pro tvorbu a obsluhu klasického grafického uživatelského rozhraní. Slouží tedy k vytváření např. oken, tlačítek, dialogů atd. Snaží se podporovat přenositelnost Javy tím, že jeho prvky vypadají na všech platformách stejně, což je pro návrh uživatelského rozhraní klíčová věc. Je to modernější alternativa původního nástroje pro tvorbu grafického uživatelského rozhraní Abstract Window Toolkit (AWT), vyvíjeného firmou Sun Microsystems jako součást Javy.



Obrázek 6.1: Swing - ukázka GUI

6.1.3 XML

Extensible Markup Language (zkráceně XML) je obecný značkovací jazyk, který byl vyvinut a standardizován konsorciem W3C. Jedná se o zjednodušenou formu staršího jazyka SGML. Nejčastěji se používá pro serializaci dat v aplikaci, v čemž soupeří např. s JSON či YAML. Zpracování XML je podporováno celou řadou nástrojů a knihoven v mnoha programovacích jazycích.

6.2 Funkcionalita aplikace

Aplikace má sloužit k vygenerování školního rozvrhu, který musí bezpodmínečně splňovat všechny tvrdé omezení a splnit co nejvíce omezení měkkých.

Funkcionalita aplikace se bude skládat ze dvou hlavních fází. Návrhu a samotného generování rozvrhu.

6.2.1 Návrh rozvrhu

Návrh rozvrhu je velmi důležitá deklarativní fáze generování. Uživatel ovlivňuje výslednou podobu rozvrhu výhradně jeho návrhem.

Zadání objektu do databáze:

- Učitelé - zadávat se bude pouze jméno učitele
- Třídy - bude zde uvedeno také pouze jméno třídy
- Hodiny - tímto se myslí vyučovací hodiny. U každé vyučovací hodiny musí být zadán název vyučovaného předmětu, učitel, který tuto hodinu bude učit, a třída. Dále bude

možno nastavit, jak dlouho bude vyučovací hodina trvat (např. “dvouhodinovka”) a kolikrát za týden se má vyučovat.

Zadání omezujících podmínek:

- Nevhodné termíny pro učitele
- Maximální mezera mezi vyučovacími hodinami
- Upřednostňovaný začátek a konec vyučování pro každou třídu
- Zadání max./min. počtu hodin, které muže mít třída během dne
- Nastavení max. počtu opakování hodiny během dne

6.2.2 Generování rozvrhu

Generování rozvrhu již uživatel nechává zcela na aplikaci. Nicméně muže si vybrat, pomocí jakého optimalizačního algoritmu se bude generovat. Každý optimalizační algoritmus obsahuje několik významných parametrů, které muže uživatel libovolně měnit.

Aplikace uživatele informuje o průběhu tvorby v podobě nejlepšího ohodnocení rozvrhu a počtu již proběhlých iterací algoritmu. Proces generování může uživatel kdykoliv zastavit nebo si jen nechat vykreslit dosud nejkvalitnější rozvrh. Průběžné vykreslování rozvrhů bylo zavrženo, kvůli velké výpočetní a časové náročnosti.

Po každém vykreslení rozvrhu si může uživatel zvolit ze dvou pohledů. Rozvrh z pohledu tříd nebo z pohledu učitelů. Pokud nebyl proveden kvalitní návrh rozvrhu nebo byl optimalizační algoritmus předčasně ukončen, v rozvrhu se můžou objevit hodiny porušující nějakou tvrdou podmínku. V takovém případě jsou tyto hodiny vykresleny červenou barvou.

Vykreslený rozvrh si může uživatel uložit. Později tak může načíst již hotový rozvrh bez časově náročného generování. Uživatel si tedy může nechat vygenerovat několik variant rozvrhů a nabídnout je k diskusi. Vykreslené rozvrhy lze rovněž exportovat do formátu HTML.

6.3 Grafické uživatelské rozhraní

Aplikace by mohla být implementována jako konzolová. Implementace by byla poměrně rychlá, jelikož by odpadly veškeré starosti s grafickými prvky a soustředilo by se výhradně na funkční stránku aplikace.

Vzhledem k povaze aplikace bylo však zvoleno grafické řešení, tedy implementace s grafickým uživatelským rozhraním. Generování rozvrhů vyžaduje zadání vstupní dat a vztahů mezi nimi. Toto by muselo být v konzolové aplikaci řešeno pomocí konfiguračních souborů. Pro uživatele by to bylo velmi matoucí, zdlouhavé a pracné. Stejně tak by muselo být řešena konfigurace jednotlivých algoritmů.

Výhoda grafické aplikace spočívá tedy hlavně ve zjednodušení a zpřehlednění zadávání dat potřebných pro vygenerování rozvrhu. Dalším kladem je větší interaktivita. Uživatel nemusí po každém vygenerování hledat vyexportovaný soubor s rozvrhem, ale výsledný rozvrh se mu zobrazí hned přímo v aplikaci. Díky grafickým ukazatelům bude mít také větší přehled o průběhu generování a bude do něj moci aktivně zasahovat.

Rozvržení grafického prostředí bude následovné:

- *Menu* - obsahuje standardní souborovou část (načíst/uložit/export rozvrhu) a část nastavení, ve které se zadávají vstupní data a podmínky pro generování a umožňuje nastavení parametrů algoritmů.
- *Vykreslovací část* - vykreslení rozvrhů bude možno provést buď z pohledu tříd, nebo z pohledu učitelů.
- *Ovládací panel* - umožňuje spouštět/zastavovat generování, sledovat průběh generování a přepínat pohled pro vykreslení rozvrhu.

Obrázek 6.2: Schéma uživatelského rozhraní

6.4 Ukládání dat

Ukládání dat do databáze je velmi výhodné zejména pro práci s velkým množství dat. Navíc je pak získávání těchto dat pomocí dotazů velmi efektivní.

V tomto případě bylo však zvoleno ukládání dat do XML souborů. Aplikace bude totiž pracovat s omezeným množstvím dat (třídy, učitelé, hodiny) a nebude nutné tato data filtrovat pomocí dotazů. Navíc je XML ukládání dat velmi jednoduché na implementaci pomocí zabudovaných knihoven v jazyce Java a pro uživatele komfortní, jelikož nevyžaduje mít nainstalován žádné aplikace třetích stran.

Kapitola 7

Popis řešení

Aplikace generátor rozvrhů se logicky skládá ze dvou částí. Část, ve které se požadovaný rozvrh navrhuje pomocí nejrůznějších nastavení, a část, ve které aplikace výsledný rozvrh generuje. Generování může probíhat pomocí tří různých optimalizačních algoritmů. Celá implementace byla provedena v jazyce Java (viz. podkapitola 6.1.1).

V této kapitole bude uveden popis realizace výše uvedeného programu. Důraz bude kladen zejména na datový model a použité optimalizační algoritmy.

7.1 Datový model aplikace

Program obsahuje několik tříd, z nichž každá má přesně své využití. Nejprve je zde několik tříd souvisejících se školským systémem. Jsou jimi *Ucitel, Trída, Hodina a Rozvrh.* Následují třídy *Algoritmus, Nastaveni*, které provádějí vlastní generování rozvrhu. Jelikož se jedná o aplikaci s grafickým uživatelským prostředím, přítomny jsou i další třídy, z nichž každá reprezentuje určité okno. Celkový výčet tříd s popisem je uveden v tabulce níže.

Třída	**Popis**
Algoritmus	obsahuje tři optimalizační algoritmy, provádí generování
Generatorrozvrhu	vstupní třída aplikace
Hodina	reprezentuje vyučovací hodinu, mapuje se do rozvrhu
Nastaveni	obsahuje všechny potřebné data pro generování
Rozvrh	reprezentuje rozvrh, více v následující podkapitole
Trida	reprezentuje školní třídu
Ucitel	reprezentuje učitele, obsahuje i jeho omezení
AlgoritmyF, HlavniOknoF, OAplikaciF, OmezeniF, UcOmezeniF, UciteleF, VyucovaniF	všechy tyto třídy reprezentují reprezentují okna v grafickém uživ. rozhraní

Tabulka 7.1: Popis všech tříd v aplikaci

7.2 Datová reprezentace rozvrhu

Rozvrh je nejdůležitější datová struktura v aplikaci. Probíhá nad ním většina datových operací, proto jeho datová reprezentace významně ovlivňuje rychlost aplikace a celkovou efektivnost implementace optimalizačních algoritmů.

Jeho nejdůležitější součástí je objekt typu *HashMap*, který mapuje každou vyučovací hodinu na určitou časovou pozici v rozvrhu. Zde bylo nutné zajistit validitu rozvrhu při různých přesunech hodin. Vyučovací hodina nemusí totiž trvat vždy pouze jednu „hodinu“, proto musí být zabráněno situaci, kdy bude např. začínat na posledním časovém slotu jednoho dne a končit na časovém slotu dne druhého. Stejně tak nesmí nastat případ, kdy vyučovací „dvouhodinovka“ bude začínat v posledním pátečním časovém slotu. Z této mapy lze tedy již sestavit určitý rozvrh a hlavně je vhodná pro operace optimalizačních algoritmů (snadno se v ní přemisťují vyučovací hodiny).

Nicméně v tomto objektu se obtížně ověřují podmínky kladené na výsledný rozvrh a hůře se vykresluje. Proto byl přidán ještě objekt typu *List*, který obsahuje dvě dimenze. První určuje časový slot v rámci určitého dne. Druhá dimenze určuje, kolik vyučovacích hodin je na tento časový slot namapováno, a umožňuje jimi iterovat. Jedná se tedy o strukturálně velmi blízkou reprezentaci reálného školního rozvrhu, ve které se snadno ověří vstupní podmínky a jednoduše se vykreslí.

Oba dva objekty mají přesně svůj účel. Při každé operaci s rozvrhem, nejčastěji přesunu hodin, je nutné tyto objekty bezpodmínečně synchronizovat, aby byla zaručena celková konzistence rozvrhu.

Dále pak každý rozvrh obsahuje atribut *fitness*, který udává jeho ohodnocení. Může dosahovat hodnot od 0 do 1, kdy ohodnocení 1 značí ideální rozvrh. Ohodnocení každému rozvrhu nastavuje metoda *setFitness()*.

7.3 Podmínky kladené na výsledný rozvrh

V následující kapitole jsou uvedeny všechny konkrétní omezující podmínky, které byly v aplikaci naimplementovány. Pokud se při ověřování zjistí porušení některé zadané podmínky, sníží se ohodnocení rozvrhu. Hodnota, o kterou se snižuje ohodnocení, není vždy stejně velká, ale závisí na závažnosti porušené podmínky. Velmi důležité je také zjistit počet porušených podmínek, aby bylo možno rozeznat přesnou kvalitu rozvrhu.

7.3.1 Tvrdé podmínky

Samozřejmostí jsou základní podmínky, které musí být splněny, aby dával výsledný rozvrh smysl:

- Učitel nesmí učit více vyučovacích hodin ve stejný čas
- Třída nesmí mít více vyučovacích hodin ve stejný čas

Tvrdé podmínky platí automaticky pro každý rozvrh a uživatel je nezadává. Případné porušení těchto podmínek vede k vysoké penalizaci ohodnocení rozvrhu. Pokud se vykresluje rozvrh, který nesplňuje některé tvrdé podmínky, pak se vyučovací hodiny, které jsou tomu příčinou, vykreslí červeně.

7.3.2 Měkké podmínky

Váha měkkých podmínek je menší, a tudíž jejich porušení snižuje ohodnocení rozvrhu méně, než porušení podmínek tvrdých. Jejich parametry jsou plně závislé na uživateli.

Časové omezení pro učitele

Pro každého učitele lze nastavit individuálně, kdy se jim učit z nějakého důvodu nehodí. Takové nastavování by bylo po chvíli nepřehledné, proto se vždy vykreslí učitelův grafický rozvrh, ve kterém je možné přehledně zaškrtnout nehodící se časové sloty. Pokud je u učitele použito nějaké omezení, objeví se v seznamu za jeho jménem příznak [O].

Časové omezení pro třídy

Třídě lze nastavit požadovaný začátek i konec vyučování. Zejména nastavení konce vyučování se uplatní např. pro nižší ročníky základních škol, kdy délka vyučovacího dne bývá zpravidla kratší.

Maximální/minimální počet hodin za den

I toto nastavení je velmi užitečné. Při určitých konfiguracích rozvrhu a nízkém počtu vyučovacích hodin, se některé třídě může stát, že bude mít některé dny doslova „nabité“ hodinami a v některé dny hodin málo nebo dokonce žádné. Proto je dobré vždy podle třídy nastavit maximální a minimální počet vyučovacích hodin za den.

Maximální velikost mezer mezi hodinami

Mezerou je zde myšleno nevyužité místo mezi vyučovacími hodinami v třídním rozvrhu. Obecně nejsou mezery ve školních rozvrzích příliš oblíbeny, nicméně pro některé konfigurace rozvrhu jsou nezbytné.

Opakování hodiny během dne

Při mapování více stejných vyučovacích hodin do rozvrhu se může stát, že se budou vyskytovat v rámci dne vícekrát. Tento jev také není příliš žádoucí. Pokud je požadováno pro třídu více stejných hodin denně, lze toho dosáhnout nastavením doby trvání hodiny.

7.4 Počáteční rozvrh

Pro každý optimalizační algoritmus je nutno nejprve vygenerovat počáteční rozvrh, který se bude postupně zkvalitňovat. Vygenerování takového rozvrhu (namapování všech vyučovacích hodin do rozvrhu) musí probíhat velmi rychle, proto bylo zvoleno náhodné vygenerování. Určitou daní je poměrně špatné ohodnocení takto vygenerovaného rozvrhu, nicméně po použití optimalizačního algoritmu se to ve výsledku nijak výrazně neprojeví.

7.5 Použité rozvrhovací algoritmy

V aplikaci byly naimplementovány dva základní a jeden pokročilý optimalizační algoritmus, z nichž každý má své výhody i nevýhody. Způsob jejich implementace byl proveden s ohledem na možnost snadného přidání dalších algoritmů.

7.5.1 Horolezecký algoritmus

Obecný popis horolezeckého algoritmu je uveden v podkapitole 3.1. Nyní však bude popsána jeho implementace v aplikaci.

Algoritmus pracuje na základě prohledávání sousedství aktuálního rozvrhu. Sousedstvím jsou myšleny rozvrhy, které se liší od toho aktuálního přemístěním jedné vyučovací hodiny.

Možnost uváznutí v lokálním minimu se při rozvrhování ukázala jako docela zásadní. Při generování složitějších rozvrhů nalezne algoritmus v takto obrovském stavovém prostoru jen zřídkakdy ideální rozvrh. Nicméně určitého zlepšení se podařilo dosáhnout pomocí několika vylepšení. Byla přidána možnost nastavit počet restartů algoritmu, z nichž každý vychází z jiného počátečního rozvrhu. Tím se částečně eliminuje brzké uváznutí v lokálním minimu.

```
rozvrh = vytvor_pocatecni_rozvrh();

do {
  prohledej_sousedstvi_rozvrhu();
  if(objevilo_se_lepsi_reseni())
    rozvrh = lepsi_rozvrh;
  else if(objevilo_se_stejne_kvalitni_reseni && random() < pst)
    rozvrh = stejne_kvalitni_rozvrh;
  else
    break;
}while(rozvrh_neni_idealni());

return rozvrh;
```

Pseudokód 7.1: Horolezecký algoritmus

7.5.2 Simulované žíhání

Obecný popis tohoto algoritmu je uveden v podkapitole 3.2. Oproti horolezeckému algoritmu má simulované žíhání velkou výhodu – nehrozí zde tak lehce uváznutí v lokálním minimu.

Základním kamenem je opět prohledávání sousedství rozvrhu. Pro implementaci podmínky, která určuje, kdy je přijímáno i horší řešení, bylo použito Metropolisovo kritérium. Pravděpodobnost, že lze přijmout i horší řešení je tedy vyjádřená vzorcem

$$P = e^{\frac{-f}{T}},$$

kde f značí ohodnocení daného rozvrhu a T je teplota. V důsledku postupného snižování teploty v průběhu algoritmu tak dochází ke snižování pravděpodobnosti přijmutí horšího rozvrhu.

Hlavní parametry algoritmu, jako jsou počáteční teplota či koeficient chládnutí, je možné před jeho spuštěním měnit.

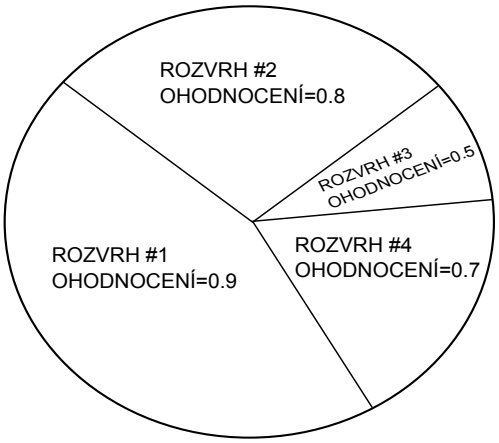
```
rozvrh = vytvor_pocatecni_rozvrh();

do {
  prohledej_sousedstvi_rozvrhu();
  if(objevilo_se_lepsi_reseni())
    rozvrh = lepsi_rozvrh;
  else if(random() < e^(-ohodnoceni/teplota))
    rozvrh = horsi_rozvrh;
}while(rozvrh_neni_idealni());

return rozvrh;
```

Pseudokód 7.2: Simulované žíhání

7.5.3 Genetický algoritmus

Obecný popis tohoto algoritmu je uveden v kapitole 3.4. Genetický algoritmus je již pokročilý optimalizační algoritmus. Jeho princip už není tedy tak jednoduchý, jako u předchozích dvou.

Jednou ze základních operací u tohoto algoritmu je výběr jedinců z populace, s nimiž se budou provádět jednotlivé genetické operace. Pro tuto činnost byla zvolena výběrová metoda „ruletové kolo“. „Ruletové kolo“ zaručuje, že šance na výběr rozvrhu je proporciálně závislá na jeho ohodnocení. Kvalitní rozvrhy mají tedy zaručenu vysokou pravděpodobnost reprodukce do další generace.



Obrázek 7.1: Příklad ruletového výběru

Po úspěšně provedené selekci, přichází na řadu genetická operace křížení. Je nutno uvést, že tato operace závisí na parametru pravděpodobnosti křížení a nemusí vůbec proběhnout. Existuje totiž i pravděpodobnost, že se rodičovské rozrvhy rovnou zkopírují do další populace, aniž by se křížili.

Po krátkém experimentování s různými variantami křížení bylo zvoleno to nejjednodušší jednobodové křížení. Ukázalo se totiž, že v tomto případě je to dostačující. Operace křížení tedy vytvoří potomka dvou rodičovských rozvrhů, náhodně zvolí bod křížení a podle něj mapuje hodiny potomka vždy podle jednoho z rodičů.

Následně je dána možnost k provedení mutace potomka, která stejně jako křížení plně závisí na zadaném parametru pravděpodobnosti mutace.

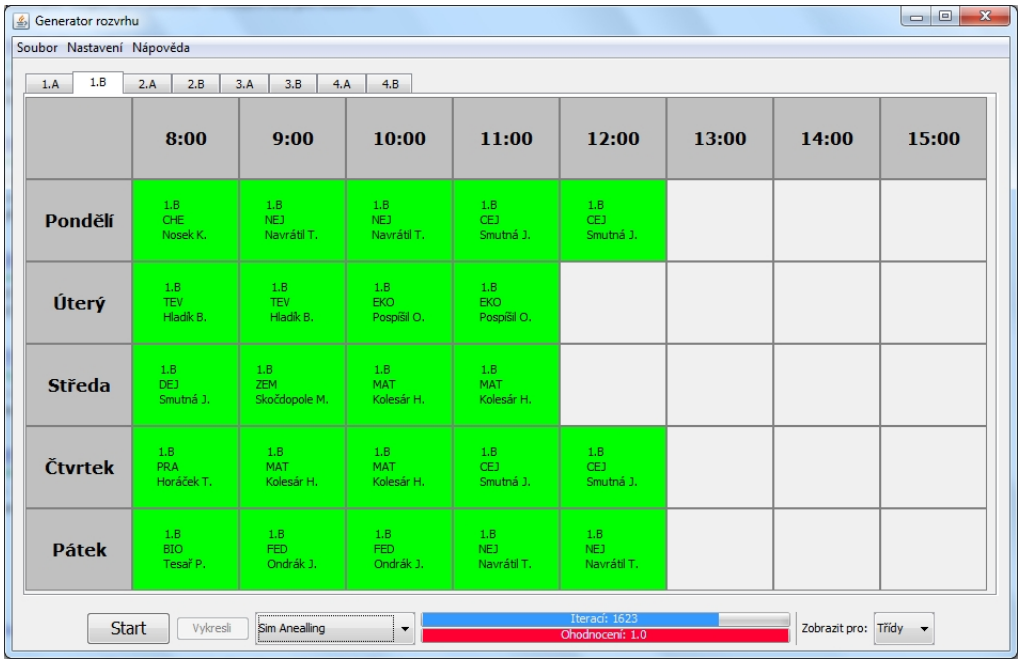
```
populace = vytvor_pocatecni_populaci();

do {
  rozvrh1 = proved_ruletovy_vyber(populace);
  rozvrh2 = proved_ruletovy_vyber(populace);

  if(random() < pst_krizeni)
    novy_rozvrh = krizeni(rozvrh1, rozvrh2);
    if(random() < psat_mutace)
      mutace(novy_rozvrh);
    pridej_do_populace(novy_rozvrh);
  else
    pridej_do_populace(rozvrh1, rozvrh2);

  odstran_z_populace_nejslabsi_rozvrhy();
  najdi_v_populaci_nejkvalitnejsi_rozvrh();
}while(nejkvalitnejsi_rozvrh_neni_idealni());

return rozvrh;
```

Pseudokód 7.3: Genetický algoritmus

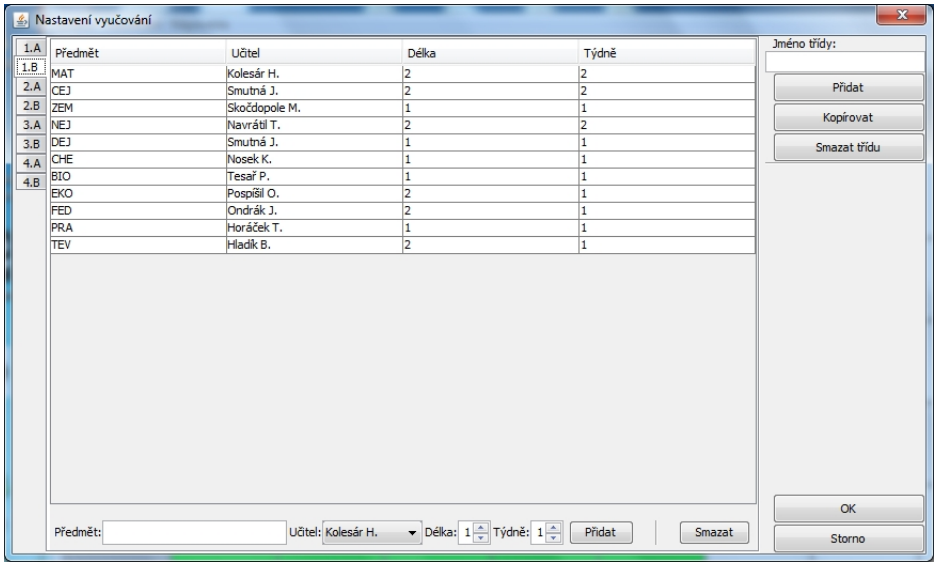
7.6 Ukázka aplikace

Po spuštění se zobrazí hlavní okno aplikace. Je v něm vyhrazen největší prostor na to nejdůležitější – rozvrh. Ve spodní části se nachází ovládací panel pro generování rozvrhu. Neméně důležitá fáze návrhu rozvrhu se provádí v položkách menu.

Nejdůležitější částí návrhu rozvrhu je sestavení vyučování. V tomto okně je možné přidávat/odebírat třídy a přidělovat jim vyučovací hodiny.



Obrázek 7.2: Hlavní okno aplikace



Obrázek 7.3: Nastavení vyučování pro jednotlivé třídy

Kapitola 8

Dosažené výsledky

V této kapitole jsou uvedeny grafy, které srovnávají naimplementované optimalizační algoritmy. Jsou srovnávány obecně, v závislosti na nastavení jejich parametrů a v závislosti na zadaných omezujících podmínkách. Jako ohodnocení rozvrhu se v grafech udává vždy jeho průměrná hodnota.

8.1 Referenční příklad

Jako referenční byl vybrán návrh rozvrhu pro 4 ročníky (8 tříd) střední školy. Na škole působí 17 učitelů s různými úvazky, které jsou rozprostřeny přes více tříd. Většina učitelů vyučuje více různých předmětů.

Ročník / třídy	Počet vyuč. hodin za týden
1. ročník / 1.A, 1.B	23
2. ročník / 2.A, 2.B	27
3. ročník / 3.A, 3.B	28
4. ročník / 4.A, 4.B	23

Tabulka 8.1: Navržená kapacita hodin pro jednotlivé třídy

Na první pohled se rozvrhy nemusí zdát příliš složité. V důsledku různých závislostí hodin, jejich trvání a zadaných omezujících podmínek se však z generování ideálního rozvrhu stává velmi komplikovaný problém.

Omezující podmínka	Nastavená hodnota
Preferováný start výuky	1
Preferováný konec výuky	7
Max. mezera mezi hodinami	0
Max. počet opakování předmětu za den	0
Max. počet hodin za den	7
Min. počet hodin za den	4

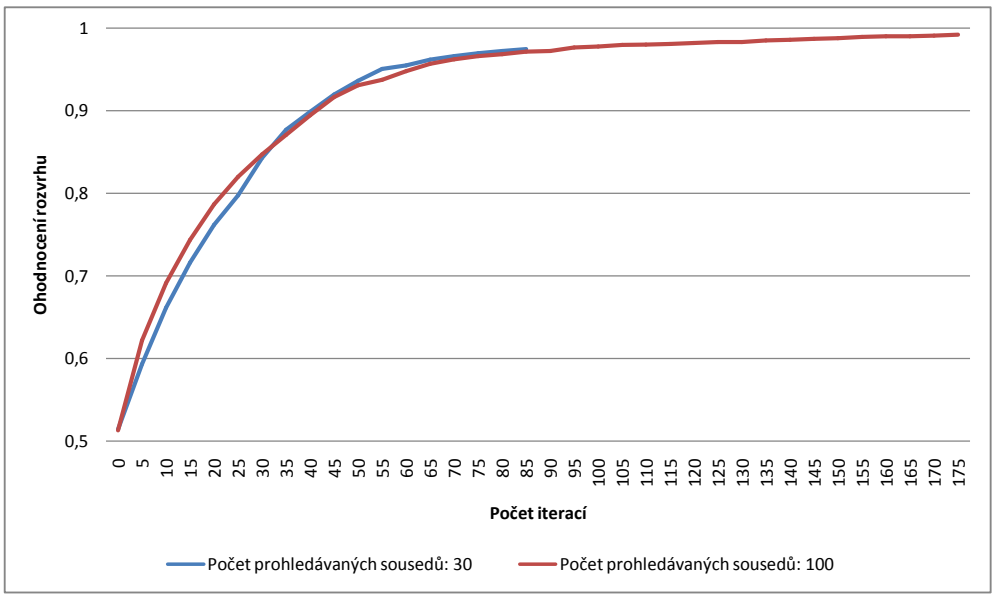
Tabulka 8.2: Zadání omezujících podmínek pro všechny třídy

Zadání omezujících podmínek bylo zvoleno dosti striktní, dá se tedy předpokládat delší doba výpočtu. Na druhou stranu pokud se podaří dosáhnout ideální rozvrh, byl by pro

studenty velmi komfortní. Aplikace umožňuje zadat omezení pro každou třídu jiné, při testování byly však pro jednoduchost voleny pro každou třídu stejné.

Definice referenčního příkladu je tímto hotová. V následujících podkapitolách bude uvedeno, jak si s takhle navrženým rozvrhem poradí optimalizační algoritmy.

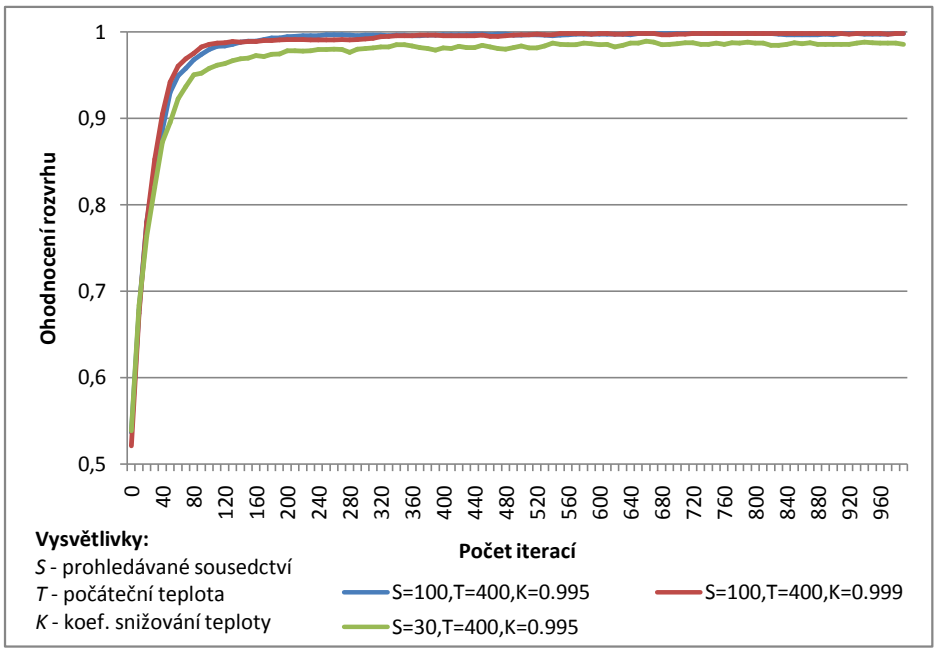
8.2 Horolezecký algoritmus



Obrázek 8.1: Závislost ohodnocení rozvrhu na počtu prohledávaných sousedů

Na grafu je jasně vidět největší nevýhoda tohoto algoritmu - uváznutí v lokálním minimu (modrá barva). Prokázalo se, že velikost prohledávaného sousedství rozvrhu významně ovlivňuje dobu, po kterou se uvázne v lokálním minimu. Docela překvapivé bylo zjištění, že tento parametr nijak neovlivňuje míru zvyšování ohodnocení rozvrhu (křivku jsou přibližně stejně šikmé).

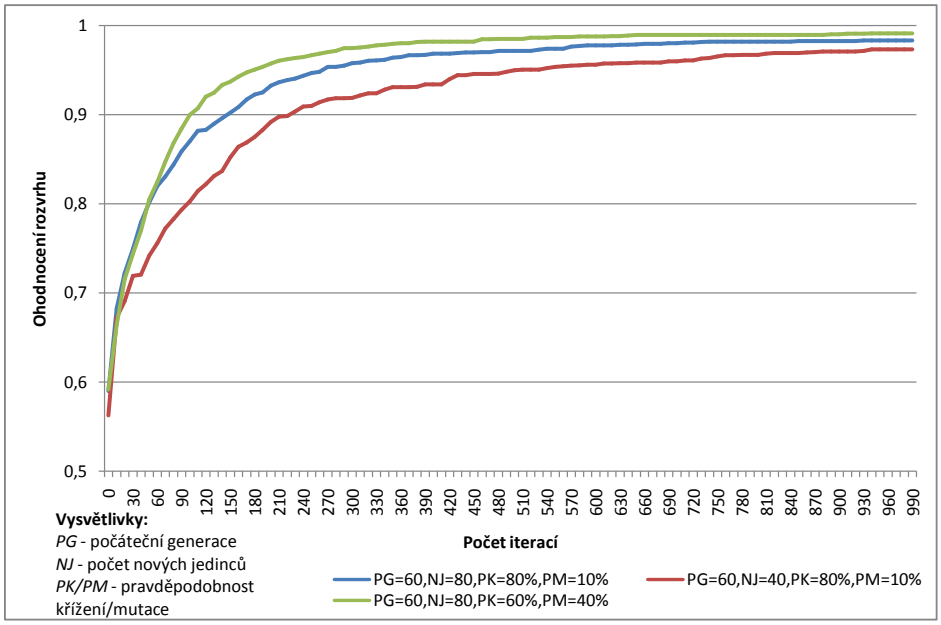
8.3 Simulované žíhání



Obrázek 8.2: Závislost ohodnocení rozvrhu na počtu prohledávaných sousedů

U simulovaného žíhání byly provedeny experimenty s následujícími parametry: velikost prohledávaného sousedství, počáteční teplota a koeficient snižování teploty. Díky klíčovému parametru teploty by mělo docházet i přijímaní rozvrhu s nižším ohodnocením, než má aktuální. Tento jev je v grafu zcela zřejmý a projevu se „zubatým“ průběhem funkce. Bylo také dokázáno, že velikosti prohledávaného sousedství ovlivňuje míru zvyšování ohodnocení rozvrhu.

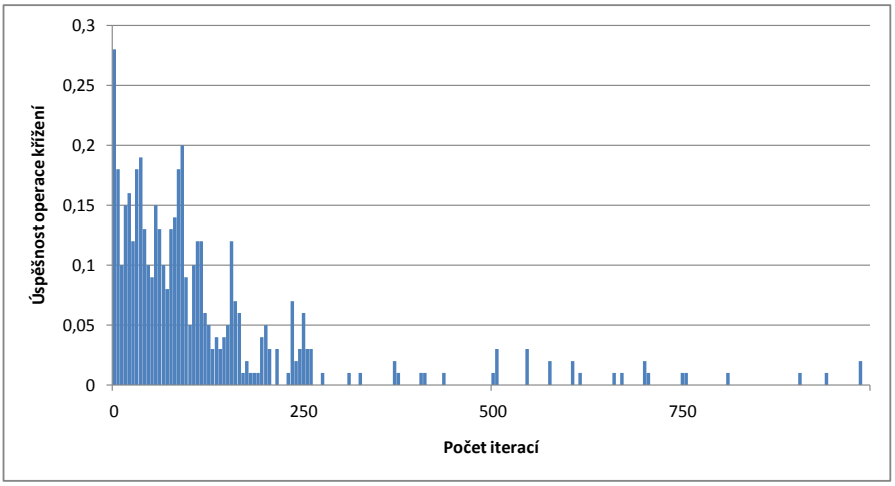
8.4 Genetický algoritmus



Obrázek 8.3: Závislost ohodnocení rozvrhu na počtu prohledávaných sousedů

U genetického algoritmu se zdá být míra ovlivňování průběhu generování pomocí parametrů nejvyšší. Počet nových jedinců, kteří jsou při každé iteraci přidávány do nové populace, zásadně ovlivňuje míru zvyšování ohodnocení rozvrhu. Pro tento parametr se ukázala býti ideální hodnota 70-80. Při nižších hodnotách je algoritmus velmi neefektivní, při vyšších se zase naráží na velkou výpočetní a časovou náročnost.

Pravděpodobnost křížení a mutace má také velký vliv. Obecně se doporučuje nastavit pravděpodobnost mutace velmi nízkou (0-10%) a pravděpodobnost křížení asi 60-80%. V grafu je však zřejmé, že v případě této aplikace dochází při zvýšení pravděpodobnosti mutace ke zlepšení průběhu generování.



Obrázek 8.4: Závislost úspěšnosti křížení rozvrhů na počtu iterací

V aplikaci bylo použito jednoduché jednobodové křížení, kdy v každé iteraci algoritmu vznikalo křížením 100 nových rozvrhů. V grafu není zahrunuta mutace rozvrhů. Úspěšností křížení rozvhů je myšlen podíl

$$U_k = \frac{U_j}{V_j},$$

kde U_j je počet potomků křížení (nových rozvrhů) mající lepší ohodnocení, než oba rodičovské rozvrhy. V_j je potom počet všech potomků křížení vzniklých při jedné iteraci algoritmu.

V grafu je vidět vysoká počáteční úspěšnost křížení rozvrhů, která je způsobena tím, že se v populaci na začátku algoritmu nachází rozvrhy s horším ohodnocením. Díky křížení tak v této fázi algoritmu dochází k velkému nárustu ohodnocení rozvrhů v populaci. Nicméně postupně úspěšnost křížení klesá a v pozdější fázi algoritmu, kdy jsou již v populaci obsaženy rozvrhy s vysokým ohodnocením, už dochází k úspěšnému křížení jen zřídka.

8.5 Časová náročnost

Měření probíhalo na tomto stroji:

CPU typ	**CPU [GHz]**	**RAM [GB]**	**OS verze**
Intel Core 2 Duo T5550	1,83	2	Windows 7 32bit

Tabulka 8.3: HW konfigurace použitého stroje

Parametry optimalizačních algoritmů byly následující:

Genetický algoritmus:

- velikost počáteční generace: 80
- počet nových jedinců: 80
- pravděpodobnost křížení: 77 %
- pravděpodobnost mutace: 20 %

Simulované žíhání:

- počet prohledávaných sousedů: 100
- počáteční teplota: 400
- koeficient snižování teploty: 0.999

Horolezecký algoritmus:

- počet prohledávaných sousedů: 100

Při použití referenčního příkladu byly naměřeny tyto hodnoty:

Počet iterací	**Čas [s] - GA**	**Čas [s] - SŽ**	**Čas [s] - HA**
100	4,1	2,9	3,1
500	17,3	14,7	uváznutí
1000	34,8	28,9	uváznutí
2000	68,4	55,4	uváznutí

Tabulka 8.4: Časová náročnost algoritmů

Kapitola 9

Závěr

V rámci této práce byla vytvořena aplikace pro generování školních rozvrhů pro střední a základní školy. Umožňuje generovat rozvrhy pomocí tří různých optimalizačních algoritmů, z nichž každý má své výhody i nevýhody, což bylo ověřeno při experimentování. Jejich implementace byla provedena s ohledem na snadné rozšíření o další algoritmy. Aplikace umožňuje zadat celou řadu omezujících podmínek a je multiplatformní. Grafické uživatelské rozhraní bylo navrženo tak, aby bylo co nejvíce přehledné a jednoduché.

Prostor pro zlepšení je určitě ve větší interaktivitě aplikace s uživatelem. Nyní může uživatel ovlivnit výsledný rozvrh pouze pomocí jeho návrhu. Pak už jen čeká, jaký rozvrh aplikace vygeneruje, případně si může nechat vygenerovat více variant. Větší interaktivita by spočívala v tom, že by mohl uživatel ručně přesouvat vyučovací hodiny v rámci rozvrhu (třeba i během generování) a aplikace by se těmto změnám musela přizpůsobit.

Také by bylo pro uživatele přínosné, aby mohlo docházet k průběžnému vykreslování rozvrhů během generování. Nyní má uživatel možnost si nechat rozvrh v průběhu generování pouze na požádání vykreslit. Je to z důvodu velké náročnosti vykreslování komponenty *JTabbedPane*, která byla použita za účelem přehlednosti. Pro průběžné vykreslování by se tedy musel použít jiný způsob zobrazení rozvrhů.

Při dalším rozvíjení této aplikace by bylo vhodné se také zaměřit na další školní problémy, které s rozvrhováním logicky souvisí. Jedná se o další činnosti učitelů na základních školách. Nejčastěji se jedná o hlídání žáků během přestávek nebo u oběda. Zejména rozvrh hlídání žáků během přestávek by mohl využít již implementované mechanismy pro učitele, musela by se však přizpůsobit datová reprezentace rozvrhu. Další oblastí, která je úzce spjatá s problematikou školních rozvrhů je suplování.

Literatura

[1] aSc Rozvrhy. 2008, [online], navštíveno 12.3.2012.
URL http://www.asctimetables.com/timetables_cz.html

[2] Bakaláři. 2008, [online], navštíveno 12.3.2012.
URL http://www.bakalari.cz

[3] Bílek, I.: *Aproximativní a heuristické metody řešení NP-těžkých problémů*. Diplomová práce.

[4] Brucker, P.: *Scheduling Algorithms*. Springer, 2007, ISBN 978-3540695158.

[5] Herout, P.: *Učebnice jazyka Java*. Kopp, 2010, ISBN 9788072323982.

[6] Jaroš, P.: Tvůrce rozvrhů. 2011, [online], navštíveno 12.3.2012.
URL http://rozvrhy.jaros.in/index.html

[7] Luner, P.: Jemný úvod do genetických algoritmu. 2008, [online], navštíveno 15.4.2012.
URL http://cgg.mff.cuni.cz/~pepca/prg022/luner.html

[8] Petrovický, L.: *Vizualizace tvorby a optimalizace rozvrhu*. Bakalářská práce, MUNI-FI, Brno, 2010.
URL http://is.muni.cz/th/173098/fi_b_b1/thesis.pdf

[9] Rudová, H.: Úvod do rozvrhování. 2012, [online], navštíveno 4.5.2012.
URL http://www.fi.muni.cz/~hanka/rozvrhovani/prusvitky/prvni.pdf

[10] Škrabal, O.: *Genetické algoritmy a rozvrhování*. Diplomová práce, FSI VUT, Brno, 2011.
URL http://autnt.fme.vutbr.cz/szz/2010/DP_Skrabal.pdf

[11] Sledujplanuj: Stochastické heuristické optimalizační metody. 2011, [online], navštíveno 13.4.2012.
URL http://www.sledujplanuj.cz/Clanek/8/Stochasticke-heuristicke-optimalizacni-metody

[12] Václavík, R.: *Algoritmy pro rozvrhování směn*. Diplomová práce, FIT ČVUT, Praha, 2011.
URL http://dce.fel.cvut.cz/Files/spoluprace/Akcekontakt/DP2011/DP-vaclavik.pdf

[13] Wikipedie: Složitost. 2011, [online], navštíveno 6.5.2012.
URL http://cs.wikipedia.org/wiki/Asymptotick%C3%A1_slo%C5%BEitost

Příloha A

Obsah CD

Přiložené CD obsahuje následující soubory a adresáře:

`app`	adresář aplikace (zdrojové kódy, knihovny)
`readme.txt`	soubor se základními informacemi o aplikaci
`install.txt`	soubor s popisem instalace aplikace
`thesis.pdf`	elektronická verze textu bakalářské práce

Příloha B

Referenční rozvrh

Příklad rozvrhu vygenerovaného na základě referenčního příkladu v kapitole 8. Obsahuje rozvrh z pohledu tříd i z pohledu učitelů.

1.A

	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00
Pondělí	EKO Pospíšil O.	EKO Pospíšil O.	MAT Kolesár H.	MAT Kolesár H.				
Úterý	DEJ Smutná J.	CEJ Smutná J.	CEJ Smutná J.	ZEM Skočdopole M.				
Středa	MAT Kolesár H.	MAT Kolesár H.	CEJ Smutná J.	CEJ Smutná J.	NEJ Navrátil T.	NEJ Navrátil T.		
Čtvrtek	BIO Tesař P.	TEV Hladík B.	TEV Hladík B.	CHE Nosek K.				
Pátek	PRA Horáček T.	NEJ Navrátil T.	NEJ Navrátil T.	FED Ondrák J.	FED Ondrák J.			

1.B

	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00
Pondělí	NEJ Navrátil T.	NEJ Navrátil T.	EKO Pospíšil O.	EKO Pospíšil O.				
Úterý	MAT Kolesár H.	MAT Kolesár H.	MAT Kolesár H.	MAT Kolesár H.	TEV Hladík B.	TEV Hladík B.		
Středa	ZEM Skočdopole M.	NEJ Navrátil T.	NEJ Navrátil T.	CHE Nosek K.				
Čtvrtek	FED Ondrák J.	FED Ondrák J.	BIO Tesař P.	CEJ Smutná J.	CEJ Smutná J.			
Pátek	CEJ Smutná J.	CEJ Smutná J.	PRA Horáček T.	DEJ Smutná J.				

2.A

	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00
Pondělí	CEJ Smutná J.	BIO Tesař P.	BIO Tesař P.	BIO Tesař P.	NEJ Prachařová J.	NEJ Prachařová J.		
Úterý	EKO Pospíšil O.	EKO Pospíšil O.	MAR Nosek K.	NEJ Prachařová J.	NEJ Prachařová J.			
Středa	SOC Pospíšil O.	MAT Ondrák J.	DEJ Martinová A.	DEJ Martinová A.				
Čtvrtek	SOC Pospíšil O.	CEJ Smutná J.	ZEM Martinová A.	ZEM Martinová A.	RET Černá R.			
Pátek	MAT Ondrák J.	RET Černá R.	CEJ Smutná J.	ANJ Soukalová S.	ANJ Soukalová S.	ANJ Soukalová S.	PRA Ondrák J.	

2.B

	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00
Pondělí	PRA Ondrák J.	CEJ Smutná J.	RET Černá R.	ZEM Martinová A.	ZEM Martinová A.			
Úterý	MAT Ondrák J.	NEJ Prachařová J.	NEJ Prachařová J.	DEJ Martinová A.	DEJ Martinová A.	NEJ Prachařová J.	NEJ Prachařová J.	
Středa	CEJ Smutná J.	CEJ Smutná J.	EKO Pospíšil O.	EKO Pospíšil O.	MAR Nosek K.			
Čtvrtek	ANJ Soukalová S.	ANJ Soukalová S.	ANJ Soukalová S.	BIO Tesař P.	BIO Tesař P.	BIO Tesař P.		
Pátek	SOC Pospíšil O.	MAT Ondrák J.	SOC Pospíšil O.	RET Černá R.				

3.A

	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00
Pondělí	ANJ Nováková T.	ANJ Nováková T.	CHE Nosek K.	CHE Nosek K.				
Úterý	BIO Tesař P.	FRJ Horáček T.	TEV Hladík B.	TEV Hladík B.	ANJ Nováková T.	ANJ Nováková T.		
Středa	DEJ Procházka I.	MAT Skočdopole M.	MAT Skočdopole M.	FRJ Horáček T.	MAT Skočdopole M.	MAT Skočdopole M.		
Čtvrtek	EKO Černá R.	ZEM Černá R.	MAT Skočdopole M.	MAT Skočdopole M.	PRA Procházka I.	FRJ Horáček T.		
Pátek	BIO Tesař P.	CEJ Nováková T.	CEJ Nováková T.	DEJ Procházka I.	TEV Hladík B.	TEV Hladík B.		

3.B

	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00
Pondělí	MAT Skočdopole M.	MAT Skočdopole M.	ANJ Navrátil T.	ANJ Navrátil T.	CHE Nosek K.	CHE Nosek K.		
Úterý	DEJ Procházka I.	BIO Tesař P.	BIO Tesař P.	FRJ Horáček T.				
Středa	CEJ Nováková T.	TEV Hladík B.	TEV Hladík B.	TEV Hladík B.	TEV Hladík B.	DEJ Procházka I.		
Čtvrtek	FRJ Horáček T.	ANJ Navrátil T.	ANJ Navrátil T.	ZEM Černá R.	MAT Skočdopole M.	MAT Skočdopole M.		
Pátek	EKO Černá R.	FRJ Horáček T.	PRA Procházka I.	CEJ Nováková T.	MAT Skočdopole M.	MAT Skočdopole M.		

4.A

	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00
Pondělí	TEV Hladík B.	TEV Hladík B.	PRA Ondrák J.	ZEM Černá R.				
Úterý	EKO Černá R.	MAR Martinová A.	ANJ Procházka I.	DEJ Procházka I.				
Středa	MAT Nosek K.	ANK Soukalová S.	NEK Prachařová J.	PRA Ondrák J.	ALG Kolesár H.			
Čtvrtek	FRJ Martinová A.	MAT Nosek K.	NEJ Horáček T.	NEJ Horáček T.				
Pátek	ALG Kolesár H.	ANJ Procházka I.	FRJ Martinová A.	CHE Nosek K.	BIO Černá R.	BIO Černá R.		

4.B

	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00
Pondělí	FRJ Martinová A.	ANJ Procházka I.	NEJ Horáček T.	NEJ Horáček T.				
Úterý	FRJ Martinová A.	MAT Nosek K.	BIO Černá R.	BIO Černá R.				
Středa	EKO Černá R.	ZEM Černá R.	PRA Ondrák J.	ANK Soukalová S.	PRA Ondrák J.			
Čtvrtek	MAT Nosek K.	NEK Prachařová J.	CHE Nosek K.	DEJ Procházka I.				
Pátek	MAR Martinová A.	ALG Kolesár H.	TEV Hladík B.	TEV Hladík B.	ALG Kolesár H.	ANJ Procházka I.		

Kolesár H.

	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00
Pondělí			1.A MAT	1.A MAT				
Úterý	1.B MAT	1.B MAT	1.B MAT	1.B MAT				
Středa	1.A MAT	1.A MAT			4.A ALG			
Čtvrtek								
Pátek	4.A ALG	4.B ALG			4.B ALG			

Smutná J.

	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00
Pondělí	2.A CEJ	2.B CEJ						
Úterý	1.A DEJ	1.A CEJ	1.A CEJ					
Středa	2.B CEJ	2.B CEJ	1.A CEJ	1.A CEJ				
Čtvrtek		2.A CEJ		1.B CEJ	1.B CEJ			
Pátek	1.B CEJ	1.B CEJ	2.A CEJ	1.B DEJ				

Skočdopole M.

	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00
Pondělí	3.B MAT	3.B MAT						
Úterý				1.A ZEM				
Středa	1.B ZEM	3.A MAT	3.A MAT		3.A MAT	3.A MAT		
Čtvrtek			3.A MAT	3.A MAT	3.B MAT	3.B MAT		
Pátek					3.B MAT	3.B MAT		

Navrátil T.

	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00
Pondělí	1.B NEJ	1.B NEJ	3.B ANJ	3.B ANJ				
Úterý								
Středa		1.B NEJ	1.B NEJ		1.A NEJ	1.A NEJ		
Čtvrtek		3.B ANJ	3.B ANJ					
Pátek		1.A NEJ	1.A NEJ					

Nosek K.

	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00
Pondělí			3.A CHE	3.A CHE	3.B CHE	3.B CHE		
Úterý		4.B MAT	2.A MAR					
Středa	4.A MAT			1.B CHE	2.B MAR			
Čtvrtek	4.B MAT	4.A MAT	4.B CHE	1.A CHE				
Pátek				4.A CHE				

Tesař P.

	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00
Pondělí		2.A BIO	2.A BIO	2.A BIO				
Úterý	3.A BIO	3.B BIO	3.B BIO					
Středa								
Čtvrtek	1.A BIO		1.B BIO	2.B BIO	2.B BIO	2.B BIO		
Pátek	3.A BIO							

Pospíšil O.

	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00
Pondělí	1.A EKO	1.A EKO	1.B EKO	1.B EKO				
Úterý	2.A EKO	2.A EKO						
Středa	2.A SOC		2.B EKO	2.B EKO				
Čtvrtek	2.A SOC							
Pátek	2.B SOC		2.B SOC					

Horáček T.

	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00
Pondělí			4.B NEJ	4.B NEJ				
Úterý		3.A FRJ		3.B FRJ				
Středa				3.A FRJ				
Čtvrtek	3.B FRJ		4.A NEJ	4.A NEJ		3.A FRJ		
Pátek	1.A PRA	3.B FRJ	1.B PRA					

Hladík B.

	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00
Pondělí	4.A TEV	4.A TEV						
Úterý			3.A TEV	3.A TEV	1.B TEV	1.B TEV		
Středa		3.B TEV	3.B TEV	3.B TEV	3.B TEV			
Čtvrtek		1.A TEV	1.A TEV					
Pátek			4.B TEV	4.B TEV	3.A TEV	3.A TEV		

Soukalová S.

	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00
Pondělí								
Úterý								
Středa		4.A ANK		4.B ANK				
Čtvrtek	2.B ANJ	2.B ANJ	2.B ANJ					
Pátek				2.A ANJ	2.A ANJ	2.A ANJ		

Prachařová J.

	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00
Pondělí					2.A NEJ	2.A NEJ		
Úterý		2.B NEJ	2.B NEJ	2.A NEJ	2.A NEJ	2.B NEJ	2.B NEJ	
Středa			4.A NEK					
Čtvrtek		4.B NEK						
Pátek								

Ondrák J.

	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00
Pondělí	2.B PRA		4.A PRA					
Úterý	2.B MAT							
Středa		2.A MAT	4.B PRA	4.A PRA	4.B PRA			
Čtvrtek	1.B FED	1.B FED						
Pátek	2.A MAT	2.B MAT		1.A FED	1.A FED		2.A PRA	

Martinová A.

	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00
Pondělí	4.B FRJ			2.B ZEM	2.B ZEM			
Úterý	4.B FRJ	4.A MAR		2.B DEJ	2.B DEJ			
Středa			2.A DEJ	2.A DEJ				
Čtvrtek	4.A FRJ		2.A ZEM	2.A ZEM				
Pátek	4.B MAR		4.A FRJ					

Nováková T.

	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00
Pondělí	3.A ANJ	3.A ANJ						
Úterý					3.A ANJ	3.A ANJ		
Středa	3.B CEJ							
Čtvrtek								
Pátek		3.A CEJ	3.A CEJ	3.B CEJ				

Procházka I.

	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00
Pondělí		4.B ANJ						
Úterý	3.B DEJ		4.A ANJ	4.A DEJ				
Středa	3.A DEJ					3.B DEJ		
Čtvrtek				4.B DEJ	3.A PRA			
Pátek		4.A ANJ	3.B PRA	3.A DEJ		4.B ANJ		

Tesař P.

	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00
Pondělí		2.A BIO	2.A BIO	2.A BIO				
Úterý	3.A BIO	3.B BIO	3.B BIO					
Středa								
Čtvrtek	1.A BIO		1.B BIO	2.B BIO	2.B BIO	2.B BIO		
Pátek	3.A BIO							

Černá R.

	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00
Pondělí			2.B RET	4.A ZEM				
Úterý	4.A EKO		4.B BIO	4.B BIO				
Středa	4.B EKO	4.B ZEM						
Čtvrtek	3.A EKO	3.A ZEM		3.B ZEM	2.A RET			
Pátek	3.B EKO	2.A RET		2.B RET	4.A BIO	4.A BIO		