Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Desktopová aplikace pro generování školního rozvrhu malé školy

Plzeň 2020 Jakub Mikeš

ZÁPADOČESKÁ UNIVERZITA V PLZNI

Fakulta aplikovaných věd

Akademický rok: 2019/2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jakub MIKEŠ**
Osobní číslo:	**A16B0085P**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informatika**
Téma práce:	**Desktopová aplikace pro generování školního rozvrhu malé školy**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Seznamte se metodami generování školního rozvrhu malé školy.
2. Navrhněte a vytvořte desktopovou aplikaci pro generování školního rozvrhu, jejímž primárním vstupem a výstupem budou soubory. Dbejte na řádné oddělení jednotlivých vrstev/částí aplikace dle zvolené architektury.
3. Vytvořenou aplikaci důkladně otestujte tak, aby množství chyb bylo minimální.

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce: **Ing. Tomáš Potužák, Ph.D.**
Katedra informatiky a výpočetní techniky

Datum zadání bakalářské práce: **7. října 2019**
Termín odevzdání bakalářské práce: **7. května 2020**

L.S.

Doc. Dr. Ing. Vlasta Radová
děkanka

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 15. října 2019

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 6. května 2020

Jakub Mikeš

Abstract

The aim of this bachelor thesis was to create desktop application for generating school timetable of small school using external files as inputs and outputs. In theoretical part, we focus on methods that can address this issue. In practical part, the design of generation algorithms and the actual implementation of the entire appliacation are described. A substantial part of this work is focused on the testing of the implemented application and evaluation of the results.

Abstrakt

Cílem této bakalářské práce je vytvoření desktopové aplikace pro generování školního rozvrhu malé školy používající externí soubory jako vstupy a výstupy. V teoretické části se budeme věnovat metodám, kterými lze tuto problematiku řešit. Dále bude popsán návrh algoritmů ke generování a samotná implementace celé aplikace. Podstatná část práce je věnována testování implementované aplikace a zhodnocení výsledků.

Poděkování

Tímto bych chtěl velmi poděkovat mému vedoucímu bakalářské práce panu Ing. Tomáši Potužákovi, Ph.D. za věnovaný čas při konzultacích a užitečné rady při vytváření práce.

Obsah

1 Úvod

Cílem této práce bylo vytvořit funkční a důkladně otestovanou desktopovou aplikaci, která by následně mohla sloužit jako součást benchmarku pro hodnocení testovacích metod. Podle zadavatele aplikace by se neměla připojovat k databázi a neměla by mít grafické uživatelské rozhraní. Těmto požadavkům vyhovuje aplikace pro tvorbu školních rozvrhů malé školy, která bude pracovat s externími soubory a bude ovládána pomocí parametrů příkazové řádky. Vstupní soubory budou ve formátu CSV nebo XML. Výstupní pak ve formátech CSV, XML nebo PDF. Aplikace bude mít možnost výběru generování mezi dvěma algoritmy.

V práci je nejprve popsána problematika tvorby školních rozvrhů včetně zjištění jaké jsou vhodné metody ke generování. Na základě této analýzy jsou navrženy dva způsoby pro generování a je vytvořen program, který dá uživateli možnost výběru, kterou metodu pro generování použije. Vytvořený program bude důsledně otestován, aby množství chyb bylo minimální a následně budou výsledky zhodnoceny. Hodnotit se bude kvalita jednotlivých školních rozvrhů, časová náročnost při generování a možnost použití aplikace v praxi.

2 Metody pro generování školního rozvrhu

V této kapitole se budeme zabývat existujícími algoritmy, které lze použít pro vytváření školních rozvrhů za použití omezujících podmínek.

2.1 Vytváření školního rozvrhu v praxi

V dnešní době lze pořídit mnoho programů, které dokážou generovat rozvrhy automaticky. Například aplikace *aSc TimeTables*, *Bakaláři* a další [1, 5].

2.1.1 aSc TimeTables

Aplikace *aSc TimeTables* je schopná automaticky generovat školní rozvrhy a používá při práci vlastní GUI. Do aplikace se zadají příslušné požadavky, jako počty tříd a seznam jejich předmětů, seznam učebem a učitelů. Pokud při zadávání uživatel udělá chybu, je možné jen data poupravit a program dopracuje do rozvrhů příslušné změny. Dokáže vygenerovat rozvrhy pro třídy, učitele a učebny a to v libovolném formátu (PDF,Excel...), nebo je sdílet online.

Výhodou této aplikace je, že si ji lze vyzkoušet zdarma. Můžeme si tedy stáhnout demo verzi, která obsahuje veškeré funkcionality jako verze plná. Uživatel si tedy na základě vlastních požadavků může vyzkoušet vygenerovat školní rozvrhy a funkce, které aplikace umožňuje. Plná verze je však placená. [1, 5].

2.1.2 Bakaláři

Aplikace obsahuje interaktivní generátor, který je napojený na systém *Škola OnLine*. Aplikace z tohoto systému načte vstupní data (třídy, skupiny, předměty, učitelé, ...). Následně se o samotné generování stará aplikace *Bakaláři*. Vygenerované rozvrhy pak posílá opět do systému *Škola OnLine*.

Výhodou této aplikace je, že si uživatel může vybrat, zda chce rozvrh vytvářet manuálně, nebo náhodně. V případě manuálního vytváření aplikace hlídá, aby nevznikaly kolize a hledá možné změny a výměny položek. Veškerá data rozvrhů se ukládají na cloud serveru, proto je třeba, aby byl uživatel při práci s aplikací připojen k internetu.

Abychom si mohli aplikaci vyzkoušet, server vyžaduje registraci. Pokud tak učiníme, je možné aplikaci stáhnout a nainstalovat. Pro správné používání je na stránce odkaz na videoukázku, jak správně při generování rozvrhů postupovat [2, 5].

2.2 Omezující podmínky

Omezující podmínky jsou podmínky, které nám zajišťují správné řešení daného problému. Při generování školního rozvrhu mluvíme o silných a slabých omezujících podmínkách, které nám zajišťují řešitelnost a kvalitu vygenerovaného rozvrhu [8, 10–12].

2.2.1 Silné podmínky

Silné podmínky (hard) jsou takové, které musí být splněny *vždy*. Pokud by tomu tak nebylo, nemůžeme řešení považovat za splnitelné. Mohlo by se stát například, že jeden učitel vyučuje na dvou místech najednou, což samozřejmě není reálně možné. Příklad silných podmínek [8, 10–12]:

- Učitel nemůže učit více předmětů ve stejný čas
- Třída nemůže mít výuku ve stejný čas na více místech
- Třída muže mít maximálně N hodin denně.

2.2.2 Slabé podmínky

Slabé podmínky (soft) jsou takové podmínky, které neovlivňují splnitelnost řešení, ale ovlivňují jeho kvalitu. Čím více daný algoritmus těchto podmínek splní, tím je výsledný školní rozvrh kvalitnější. Například třída 1A nemá vůbec výuku v pondělí. V pátek má ale školu od rána do večera. Tento stav je samozřejmě možný, ale určitě by bylo lepší, kdyby třída měla výuku rovnoměrně každý den. Příklad slabých podmínek [8, 10–12]:

- Učitel vyučuje předměty dle jeho specializace
- Každá třída má vyučování první 4 hodiny každý den
- Třída má vyučovaný předmět maximálně 1 krát v jeden den

2.3 Rozdělení algoritmů ke generování

Pro generování školních rozvrhů se používají různé typy algoritmů. Na základě poznatků z literatury bylo zjištěno, že se při této problematice používají nejvíce heuristické a genetické algoritmy. Neznamená to však, že neexistují další algoritmy, kterými si dá tato problematika také úspěšně řešit.

Heuristické algoritmy se používají pro řešení optimalizačních úloh. Metody sice nezaručují, že se podaří nalézt optimální globální řešení, ale jsou schopné vzhledem k náročnosti řešeného problému najít optimální řešení, kterému se také říká lokální. Při prohledávání musí být co největší prohledávací stavový prostor a lokální minima v blízkém okolí výchozího bodu musí být v rovnováze. Heuristickými algoritmy, které lze při generování použít jsou například *metoda lokálního hledání*, *metoda zakázaného prohledávání (Tabu search)* nebo *horolezecký algoritmus*.

Druhým typem algoritmů, který se používá pro vytváření školních rozvrhů je genetický algoritmus, který patří mezi tzv. evoluční algoritmy. jsou založeny na principu genetiky a přirozeném výběru. Vychází z Darwinovy teorie o vývoji druhu a napodobují boj jedinců o přežití. Používají tedy procesy jako je dědičnost, mutace, křížení a přirozený výběr. Tyto operace jsou relativně jednoduché, ale při jejich opakování lze efektivně najít řešení. [3, 7, 9–13].

2.4 Genetický algoritmus

Genetické algoritmy pracují s jedinci (chromozomy), kdy každý jedinec představuje právě jedno řešení daného problému. Mají určité vlastnosti a chování. Většinou jsou reprezentováni binárně, nebo speciálními znaky. Skupina jedinců tvoří populaci.

Abychom mohli určit, jak moc je jedinec kvalitní, ohodnodnocuje se tzv. fitness hodnotou. Jedná se o číselnou hodnotu vypočtenou na základě splněný omezujících podmínek.

Algoritmus pracuje následujícím způsobem. Nejprve vygeneruje prvotní populaci jedinců. Většinou se generuje náhodně, nebo tak, aby jedinci byli co nejblíže optimálnímu řešení. Jedince lze reprezentovat více způsoby, například binárně, polem, maticí či stromem. Nyní se pomocí tzv. *fitness funkce* každému jedinci přiřadí jeho fitness hodnotu. V případě použití pro generování rozvrhu kvalitu určuje na základě silných a slabých podmínek. Nyní budeme opakovat následující tři operace [3, 7, 9–13].

2.4.1 Selekce

Selekce, neboli výběr, vybírá co nejkvalitnější jedince (s nejvyšší fitness hodnotou) z populace. K výběru se nejčastěji používají tyto tři typy - *ruleta, ruleta založená na pořadí a turnaj.*

Obyčejná *ruleta* pracuje tak, že každému jedinci se vypočte fitness hodnota. Následně se vypočte ze všech získaných hodnot procentuální pravděpodobnost a vytvoří se interval, pro výběr do nové populace. Pro samotný výběr se náhodně vygeneruje číslo od 0 do 1. Ve které části intervalu odpovídajícímu danému jedinci se číslo nachází, ten jedinec je vybrán. Silní jedinci tedy mají vyšší pravděpodobnost, že budou vybrání než jedinci, kteří jsou slabí. Výběr se opakuje do té doby, dokud není nová generace kompletní. Na obrázku 2.2 můžeme vidět ukázku vytváření intervalu při použití *rulety* za pomoci 4 jedinců. Kdyby náhodně vygenerované číslo bylo např. 0,61, byl by vybrán jedinec 3, neboť se vygenerované číslo nachází v části intervalu, který tomuto jedinci odpovídá (0,392 až 0,652).

jedinci	fitness hodnota
jedinec 1	40
jedinec 2	50
jedinec 3	60
jedinec 4	80
součet	230

procentuální zastoupení	interval
40/230 = 0,174	0,174
50/230 = 0,217	0,174 + 0,217 = 0,391
60/230 = 0,261	0,391 + 0,261 = 0,652
80/230 = 0,348	0,652 + 0,348 = 1,000

Obrázek 2.1: Vytvoření intervalu pro *ruletu.*

Ruleta založená na pořadí, je modifikací obyčejné *rulety.* V té se může stát, že nalezneme jednoho silného jedince, který znemožní výběr horším jedincům dostat se do nové populace, kteří však můžou vést ke správnému řešení. Všichni jedinci se na základě vypočtené fitness hodnoty seřadí a následně se dle jejich pořadí vypočte procentuální zastoupení. Dále se pak postupuje při tvorbě intervalu stejným způsobem, jako v obyčejné *ruletě.* Na obrázku můžeme vidět, jak tvorba intervalu při *ruletě založené na pořadí* probíhá.

jedinci	fitness hodnota	ohodnocení dle pořadí
jedinec 1	40	1
jedinec 2	50	2
jedinec 3	60	3
jedinec 4	80	4
součet		10

procentuální zastoupení	interval
1/10 = 0,1	0,1
2/10 = 0,2	0,1 + 0,2 = 0,3
3/10 = 0,3	0,3 + 0,3 = 0,6
4/10 = 0,4	0,6 + 0,4 = 1,0

Obrázek 2.2: Vytvoření intervalu pro *ruletu založenou na pořadí.*

Třetím typem je *turnaj*, který vždy náhodně vybere N jedinců z populace

a z nich vrátí nejsilnějšího. Tato operace se opakuje do té doby, dokud není nově vytvářená populace kompletní. [3, 7, 9–13].

2.4.2 Křížení

Po selekci přichází proces křížení. To znamená, že si nejméně dva jedinci vymění své genetické informace, čímž získáme nového jedince, který nese informace obou rodičů. Pro křížení se nejčastěji používají tyto tři typy křížení a to *jednobodové*, *vícebodové* a *uniformní*.

Jednobodové křížení dvou jedinců probíhá tak, že nový potomek nese polovinu každého jedince. Pokud bychom tedy měli jedince 0011 a AABB, potomek by byl jedinec 00.BB.

Při vícebodovém křížení je obvykle nový jedinec tvořen více částmi obou rodičů. Např. máme jedince 00110110 a jedince AABBABBA. Výsledný jedince může vypadat například 001.BA.1B.0.

Posledním typem je uniformní křížení, které zajišťuje, že jsou všichni potomci jednotní. Nový jedinec pomocí uniformního křížení dvou jedinců 00110110 a AABBABBA by vypadal např. 0.B.1.0.1.A.0.1. [3, 7, 9–13].

2.4.3 Mutace

Po křížení jedinců přichází další operace a tou je mutace. Obecně se jedná o malou změnu genetické informace jedince, která někdy přinese zlepšení, zhoršení, nebo změnu ani nepocítíme [3, 7, 9–13].

Po vytvoření generace se provede ohodnocení jedinců pomocí fitness funkce. Z ohodnocených jedinců se opět provede selekce, následně křížení a mutace. Tento proces se opakuje do té doby, dokud není dosaženo potřebného množství generací. Na obrázku 2.3 můžeme vidět pseudokód, jak by mohl algoritmus vypadat [3, 7, 9–13].

2.4.4 Zhodnocení genetického algoritmu

Genetický algoritmus je poměrně jednoduchý a současně dokáže vygenerovat správné řešení. Nejdůležitější je správné navržení fitness funkce a vhodná reprezentace jedince. Nevýhodou genetického algoritmu je vysoká časová náročnost při zvolení vysokého počtu jedinců v populaci, nebo vysokého počtu generací, což nám poměrně zvyšuje časovou náročnost [7, 9–13].

```
genetickyAlgoritmus(){
   vytvoreniPrvotniPopulace();
       pocetGeneraci = 1;
   while(PocetGeneraci < maxPocetGeneraci){
      vypocetFitnessHodnotyKazdehoJedince();
      selekce();
      krizeni();
      mutace();
      pocetGeneraci++;
   }
}
```

Obrázek 2.3: Pseudokód genetického algoritmu pro generování školních rozvrhů.

2.5 Brute force algoritmus

Algoritmus brure force, česky řešení hrubou silou, systematicky prochází všechny možné kombinace, dokud nenalezne správné řešení. V případě generování rozvrhů by postupně vytvářel nové kombinace rozvrhů a testoval, zda jsou všechny rozvrhy splnitelné. Pokud ne, vygeneruje novou kombinaci. Takto by postupoval do té doby, dokud by nenalezl správné řešení. Na obrázku 2.4 je znázorněn pseudokód, jak by algoritmus mohl vypadat [7, 12].

2.5.1 Zhodnocení algoritmu brute force

Výhodou algoritmu brute force je, že dokáže projít všechny možné kombinace školních rozvrhů a vybrat tu nejlepší. Bohužel však má exponenciální složitost. Pokud bychom měli vysoké množství kombinací, nebyl by schopen projít všechny možnosti v historicky krátkém čase. Pokud bychom chtěli projít všechny kombinace, kdy rozvrh jedné třídy má právě 40 hodin, bylo by možné vytvořit 40! kombinací školního rozvrhu právě jedné třídy. Celkem bychom tedy museli projít počet tříd krát 40! kombinací a to je opravdu hodně. Algoritmus brute force tedy není vhodný pro řešení této problematiky [7, 12].

```
BruteForceAlgorimus(){
    VytvoreniPrvniKombinaceRozvrhu();
    while(true){
        if(kombinaceRozvrhuSplnitelna()){
            return rozvrhy;
        }else{
            systematickeVytvoreniNoveKombinaceRozvrhu();
        }
    }
}
```

Obrázek 2.4: Pseudokód algoritmu brute force pro hledání školních rozvrhů

2.6 Backtracking

Backtracking, jinak česky řečeno metoda pokusů a oprav, se používá při řešení algoritmických problémů založených na prohledávání stavového stromu problémů a pracuje na vyhledávání do hloubky. Jedná se o vylepšený algoritmus Brute force (řešení hrubou silou). Vylepšený v tom smyslu, že dokáže vyloučit velké množství potencionálních řešeních bez přímého vyzkoušení, čímž snižuje počet možných kroků, tedy časovou náročnost. V případě generování školního rozvrhu, algoritmus postupně přidává položky do rozvrhu a kontroluje, zda jsou splněné všechny silné podmínky. Pokud nám přidaná položka nějakou silnou podmínku poruší, algoritmus se vrátí na předchozí položku a nevyhovující se pokusí nahradit novou položkou. Tímto způsobem se dopracujeme k validnímu řešení.

Na obrázku 2.5 můžeme vidět pseudokód, jak by mohl algoritmus vypadat. S rozvrhy se pracuje jako se stromem, kdy si každý uzel představující rozvrhy nese odkaz na svého rodiče [7, 12].

2.6.1 Zhodnocení algoritmu Backtracking

Pomocí algoritmu Backtracking můžeme řešit mnoho problémů. Jeho nevýhodou je, že má stejně jako brute force exponenciální složitost s tím rozdílem, že neprochází všechny možnosti. V případě generování školních rozvrhů by však časová náročnost byla nejspíše vysoká. Na druhou stranu již zmíněná aplikace *aSc TimeTables* při generování používá právě backtracking. Mohl by být tedy jeden s kandidátů na samotnou implementaci [7, 9, 12].

```
backtrackingProHledaniRozvrh(){
    vlozitPrvniPolozkuDoRozvrh()
    while(true){
        if(neporusujeSilnePodminky()){
            vlozitNovouPolozku(); //vytvoreni noveho potomka
        }else{
            vraceniNapredchoziRozvrh();  //vrace do rodice
            vlozitNovouPolozkuDoRozvrhu(); //vytvoreni noveho
                potomka
         }
         if(jeRozvrhSplnitelny()){
         return rozvrhy;
         }
    }
}
```

Obrázek 2.5: Pseudokód backtrackingu při generování školních rozvrhů

2.7 Generuj a testuj

Generuj a testuj je velice jednoduchý algoritmus, který je založený na vygenerování a následném otestování. Generování se provádí náhodně. Pokud vygenerovaná data splňují všechny důležité podmínky, nalezli jsme správně řešení. Pokud ne, opakujeme generování do té doby, dokud nenalezne validní řešení nebo dokud neproběhne určitý počet opakování.

Obrázek 2.6 obsahuje pseudokód, jak by mohlo generování fungovat [12].

2.7.1 Zhodnocení generuj a testuj

Tento algoritmus je velice rychlý a je schopný vygenerovat dobré výsledky. Nevýhodou však je, že řešení vůbec nemusí najít, i když ve skutečnosti existuje. Rozdíl mezi tímto způsobem generování a brute force algoritmem je takový, že brute force projde všechny možnosti rozvrhů, kdežto tento způsob generování funguje na náhodě. Tedy náhodně uloží všechny položky do rozvrhů a kontroluje, zda jsou splněny všechny silné podmínky. Dalším rozdílem je, že má na nalezení určitý počet možností pro nalezení správného řešení. V případě generování školního rozvrhu není příliš vhodný, neboť je velmi malá pravděpodobnost, že by vygenerovaný rozvrh splňoval všechny silné podmínky a alespoň nějaké slabé [12].

```
generujATestuj(){
   int pocetOpakovani;
   while(pocetOpakovani < maximalniPocetOpakovani){
      generovaniReseni();
      if(reseniValidni){
         return reseni;
      }
      pocetOpakovani++;
   }
}
```

Obrázek 2.6: Pseudokód při generování školních rozvrhů způsobem generuj a testuj.

2.8 Dynamické algoritmy

Pomocí dynamických algoritmů se dají efektivně řešit optimalizační úlohy. Algoritmus rozdělí problém na menší podproblémy. Každý tento problém se řeší zvlášť. Po vyřešení jednotlivých podproblémů se všechny spojí dohromady, čímž vytvoří kompletní řešení.

V případě generování rozvrhů by problém mohl být rozdělený na dva podproblémy. První zajišťuje přidělení všem předmětům tříd učitele, dle specializace. Druhý podproblém by umisťoval všechny předměty do rozvrhů. Po získání výsledků obou podproblémů by se výsledky spojily, čímž by nám daly kompletní školní rozvrhy. [12].

2.8.1 Zhodnocení dynamického algoritmu

Dynamické algoritmy jsou schopné generovat velmi dobré výsledky. Bohužel v problematice generování školních rozvrů jsou nepoužitelné. Položky se v rozvrhu navzájem ovlivňují, proto není možné problém rozdělit na podproblémy [12].

2.9 Metoda lokálního hledání

Metoda lokálního hledání je heuristický algoritmus, který většinou řeší úlohy jednoduššího rázu.

Pracuje tím způsobem, že nejprve náhodně vygeneruje řešení problému. Následně prochází sousední řešení a jejich ohodnocení. Pro další postup zvolí

to sousední řešení, které má nejlepší ohodnocení (nejvíce vyhovuje). Pokud řešení nesplňuje všechny silné podmínky, nebo není v okolí lepší řešení, opět se začne prohledávat sousední řešení a jejich ohodnocení. Tento postup se opakuje do té doby, dokud není nalezeno správné řešení [6, 13].

2.9.1 Zhodnocení metody lokálního hledání

Pomocí metody lokálního hledání jsme schopni získat lokální řešení. Bohužel však nalezené lokální řešení nemusí být zároveň globálním. I když by tato metoda pravděpodobně vedla k vygenerování školních rozvrhů, trvala by pravděpodobně dlouho, neboť možností uspořádání předmětů je opravdu mnoho, proto není vhodná pro řešení naší problematiky [6, 13].

2.10 Horolezecký algoritmus

Horolezecký algoritmus umožňuje krok směřující k horšímu prozatimnímu zadání. Stejně tak jako *metoda lokálního hledání* prochází sousední řešení a jedno vybírá. Liší se však tím, že prohledávání je omezeno počtem iterací.

Pracuje následujícím způsobem. Nejprve dojde k vygenerování náhodného řešení. Dále se začnou procházet sousední řešení a jejich ohodnocení. Vybere to s největším ohodnocením a řešení si zapamatuje. Dále se tyto operace opakují do té doby, dokud neproběhne příslušný počet iterací.

Na obrázku 2.7 je znázorněn pseudokód, jak by mohla imlementace vypadat [9, 11, 13].

2.10.1 Zhodnocení horolezeckého algoritmu

Tím, že si algoritmus ukládá jednotlivé prošlé stavy, po ukončení generování se vrátí k nejlepšímu řešení. Algoritmus má však jeden problém a tím je možnost, že se nám průběh zacyklí, kdy se algoritmus dostane k řešení, ke kterému se pak vrací. Pro generování školních rozvrhů by mohl být jedním kandidátem na implementaci [9, 11, 13].

```
horolezeckyAlgoritmus(){
    nahodneVytvoreniRozvrhu()
    pocetIteraci = 1;
    while(pocetIteraci < maxPocetiteraci){
        ohodnotitSousedniReseni();
        zvoleniNejsilnejsihoSouseda();
        zapamatovaniReseni();
        pocetIteraci++;
    }
    return nejlepsiRozvrhy;
}
```

Obrázek 2.7: Pseudokód horolezeckého algoritmu.

2.11 Metoda zakázaného prohledávání (Tabu search)

Metoda zakázaného prohledávání se používá při řešení kombinatorických optimalizačních problémů. Jedná se o vylepšený horolezecký algoritmus.

V samotném horolezeckém algoritmu může nastat situace zacyklení. Tomuto stavu však tato metoda předchází. Při použití nového nejsilnějšího souseda se nejprve přesvědčí, zda již tento stav nebyl procházen. Pokud ano, použije jiného souseda. Uchovává si tedy seznam prošlých stavů, pomocí nichž kontroluje nově vybrané.

Na obrázku 2.8 mužeme vidět pseudokód, jak by mohla implementace vypadat [9, 11, 13].

2.11.1 Zhodnocení metody zakázaného prohledávání

Výhodou oproti horolezeckému algoritmu je, že nemůže dojít k zacyklení. Nevýhodou je díky procházení již prošlých stavů vyšší časová náročnost. I přesto by mohl být jedním z kandidátů na implementaci [9, 11, 13].

```
metodaZakazanehoProhledavani(){
    nahodneVytvoreniRozvrhu();
    pocetIteraci = 1;
    while(pocetIteraci < maxPocetIteraci){
        if(aktualniStavNeniVSeznamu()){
            pridatStavDoSeznamu();
            vybratNovyStav();
            pocetIteraci++;
        }else{
            vybratNovyStav();
        }
    }
    return nejlepsiRozvrhy;
}
```

Obrázek 2.8: Pseudokód metody zakázaného prohledávání (tabu search).

3 Návrh aplikace pro generování školního rozvrhu

3.1 Případy užití

Aplikaci je možné použít pro vytvoření školních rozvrhů malé školy nebo pro benchmarkování.

Samotné užití aplikace je znázorněno v diagramu užití na obrázku 3.1. Aplikace má pouze jednoho zadavatele, který aplikaci obsluhuje.

3.1.1 Jednotlivé případy užití

Diagram na obrázku 3.1 celkově obsahuje 5 případů užití. *Zadat vstupní soubory*, *Zvolit algoritmus ke generování*, *Nastavit umístění vygenerovaných souborů*, *Nastavit formát k uložení vygenerovaných rozvrhů* a *Generovat školní rozvrhy*. V této sekci budou tyto případy popsány.

Zadat vstupní soubory

Od zadavatele se požaduje zadání cest ke vstupním souborům. Konkrétně k souborům s daty tříd a učitelů. V případě tříd ve formátu CSV, cesta k adresáři, kde jsou třídy uloženy.

Zvolit algoritmus ke generování

Od zadavatele se očekává výběr algoritmu, kterým se budou rozvrhy generovat. Na výběr má genetický algoritmus nebo algoritmus založený na prohledávání a ukládání jen nekolizních položek.

Nastavit umístění vygenerovaných souborů

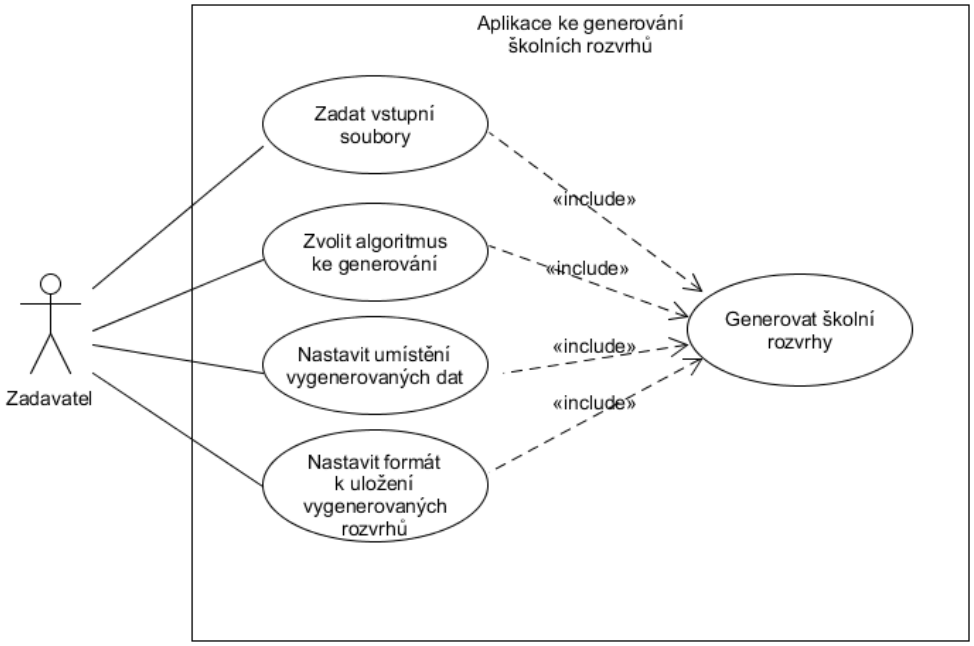
Od zadavatele se očekává cesta k adresáři, kde budou všechny vygenerované školní rozvrhy uloženy.

Nastavit formát k uložení vygenerovaných rozvrhů

Od zadavatele se očekává zadání formátu, ve kterém budou vygenerované rozvrhy uloženy (CSV, PDF nebo XML).

Generovat školní rozvrhy

Na základě získaných dat pomocí již zmíněných případů užití se spustí generování. Následně se provede uložení vygenerovaných školních rozvrhů do předem vybraného adresáře.



Obrázek 3.1: Diagram užití aplikace.

3.2 Specifikace požadavků

Aplikace bude konzolová a bude sloužit ke generování školních rozvrhů všech tříd a učitelů. Pro generování bude možnost výběru mezi dvěma algoritmy. Aplikace tedy na základě zadaných parametrů vygeneruje rozvrhy v požadovaném formátu.

3.2.1 Vstup a výstup

Při spuštění bude program používat soubory s daty ve formátu CSV nebo XML. Výsledné rozvrhy pak budeme moci uložit ve formátech CSV, XML nebo PDF.

Abychom omezili velké množství parametrů v případě souborů s daty tříd ve formátu CSV, bude vstupním souborem v případě načítání dat tříd tohoto

formátu adresář, ve kterém budou jednotlivé třídy uloženy. Každý soubor třídy v adresáři musí obsahovat příponu na konci souboru `Trida.csv`, aby program rozeznal, že k souboru může přistoupit.

3.2.2 Formát příkazové řádky/parametry

Aplikace bude spouštěna s těmito parametry, v pořadí, jak jdou po sobě v seznamu.

1. číslo algoritmu
2. cesta k souboru data tříd (v případě CSV k adresáři, kde jsou soubory uloženy)
3. cesta k souboru data učitelé
4. cesta k adresáři, kam budou uloženy výstupy
5. formát ukládaných dat

3.3 Hlavní požadavky na rozvrh

Cílem je vytvořit program pro generování školních rozvrhů malé školy. Budeme tedy pracovat s omezujícími podmínkami, které byly popsány v sekci 2.2.

Vzhledem k tomu, že generování rozvrhů je netriviální problém a úkolem je vygenerovat rozvrh malé školy, použiji omezení, že jedna třída (uskupení žáků) bude mít celý týden výuku v jediné učebně. Toto omezení usnadní řešení a zároveň se značně blíží situaci na malých školách.

3.3.1 Silné podmínky pro generování rozvrhu

Jak již bylo zmíněno v sekci 2.2.1, silné podmínky jsou takové, které musí být splněny všechny, aby úloha měla řešení. V našem případě se bude jednat o tyto podmínky:

- Učitel nemůže učit více předmětů ve stejný čas.
- Každý předmět třídy musí být v rozvrhu přesně tolikrát, kolikrát má být v týdnu vyučován.
- Třída muže mít maximálně 7 hodin denně (rozvrh má denně 8 hodin).
- Třídu na daný předmět vyučuje pouze jeden učitel.

3.3.2 Slabé podmínky pro generování rozvrhu

Stejně tak jsme v sekci 2.2.2 hovořili o slabých podmínkách. Tyto podmínky nemusejí být splněny všechny, ale čím více jich platí, tím je řešení kvalitnější. Pro naše generování budeme používat tyto podmínky:

- Předmět se v rámci jednoho dne vyučuje pouze jednou.
- První čtyři hodiny v rozvrhu jsou obsazeny.
- Učitel učí předměty dle jeho specializace.

3.4 Zdoje pro generování

Pro generování se budou používat soubory ve formátu CSV nebo XML, které budou obsahovat data pro jednotlivé třídy a učitele.

3.4.1 Třídy

Každá třída bude obsahovat název třídy a seznam předmětů k umístění do rozvrhu. Každý předmět ponese informaci o názvu předmětu, jeho zkratce a počtu hodin, kolikrát bude v týdnu vyučován.

Třídy CSV

V případě načítání tříd ve formátu CSV bude každé třídě odpovídat jeden soubor. To nám umožní lehčí modifikaci při úpravě seznamu předmětů. Soubor bude obsahovat název třídy a seznam předmětů, kdy každý předmět ponese hodnotu, kolikrát musí být za týden vyučován.

Třídy XML

Na rozdíl od načítání souborů, kdy se musí načíst každá třída zvlášť, budou všechny třídy uloženy v jednom souboru. Informace budou stejné, jako v několika CSV souborech.

3.4.2 Učitelé

Každý učitel bude obsahovat jméno a seznam specializací, které předměty může vyučovat. To znamená, že každá specializace ponese jméno daného předmětu. V případě, že vstupní data budou obsahovat dva učitele se stejným jménem, program vyhodí výjimku a oznámí, že je třeba jména těchto učitelů nějakým způsobem odlišit. Například přidáním čísla za jméno.

Učitelé CSV

Na rozdíl od tříd ve formátu CSV budou všichni učitelé uloženi v jednom souboru. Každý řádek bude představovat jednoho učitele. Ten ponese jméno učitele a názvy specializací.

Učitelé XML

Učitelé budou uloženi jako v případě CSV v jednom souboru. Každý učitel ponese jméno a jednotlivé specializace obsahující názvy předmětů.

3.5 Algoritmy pro generování

V této sekci budou důkladně popsány algoritmy, které jsem se rozhodl použít pro generování školního rozvrhu na základě provedené analýzy v sekci 2, kde jsem si přiblížil, jaké jsou možnosti při řešení této problematiky. Rozhodl jsem se pro použití Genetického algoritmus a algoritmu založeném na postupném prohledávání a ukládání jen nekolizních položek, který jsem si na základě poznatků navrhl.

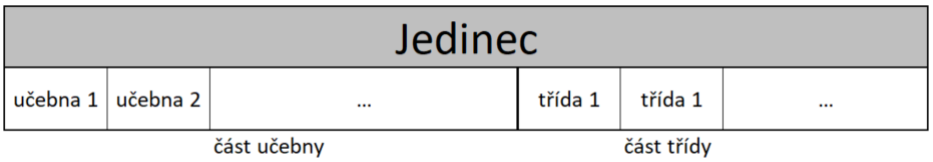
3.6 Genetický algoritmus

Pro tento algoritmus jsem se rozhodl z toho důvodu, že každým novým generováním, při použití stejných vstupních dat, nalezne jiné řešení, ale může se stát, že řešení bude i stejné, byť často může být jiné. Uživatel si tedy může vygenerovat více variant rozvrhů a následně si vybrat, které řešení mu nejvíce vyhovuje. Generování u genetických algoritmů trvá sice delší dobu než heuristické nebo rekurzivní algoritmy, zato dokážou nalézt velice kvalitní řešení [12].

Jak již bylo zmíněno v sekci 2.4, genetický algoritmus vychází z přírodních jevů, kdy dochází mezi jedinci ke křížení a mutacím. Abychom tento jev mohli použít na naší problematiku, je třeba ho příslušně poupravit.

3.6.1 Jedinec

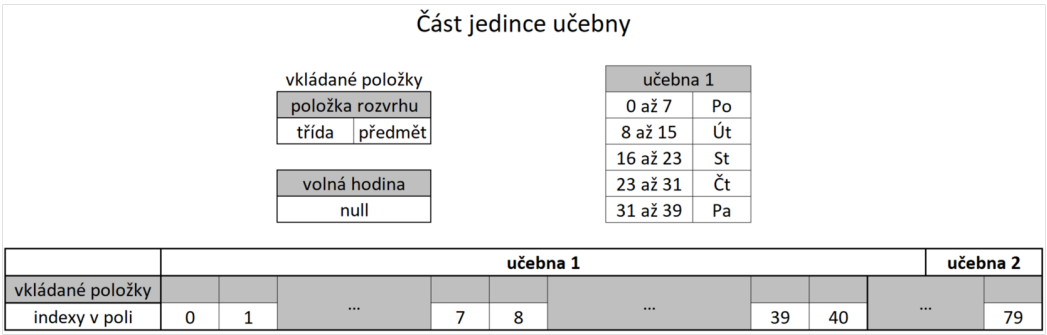
Nejdůležitější věcí je správně zvolit, jakým způsobem bude jedinec reprezentován. Vzhledem ke zvolenému omezení každá třída bude mít vyučování ve stejné učebně, bude jedinec reprezentován dvěma hlavními částmi. První část bude reprezentovat učebny a druhá třídy. Rozdělení můžeme vidět na obrázku 3.2.



Obrázek 3.2: Na obrázku je znázorněna reprezentace jedince.

Učebny

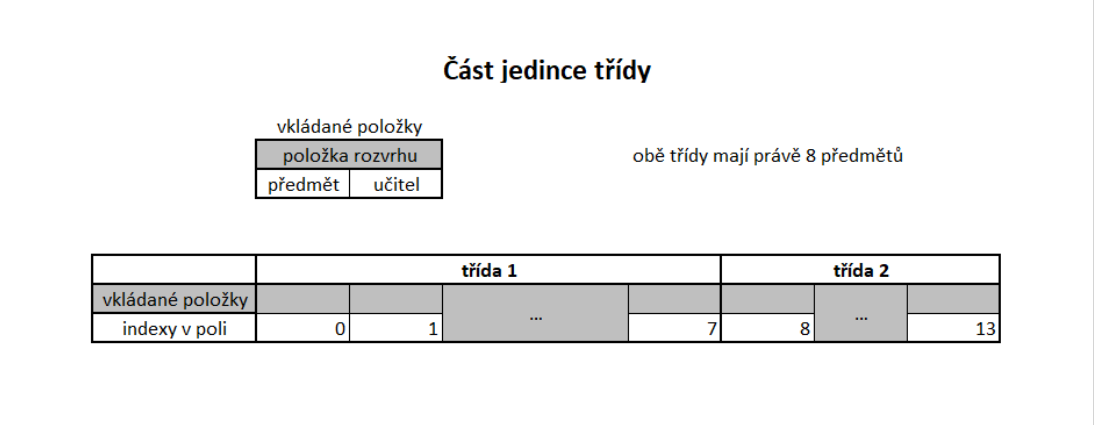
Část učebny bude reprezentována polem. Každá učebna bude reprezentovat rozvrh právě jedné třídy. Velikost bude odpovídat počtem tříd vynásobenému počtu hodin v týdnu (součin konstant počet dní v týdnu (5 dní) a počtu hodin denně (8 hodin)).Každá učebna bude reprezentována 40 položkami (učebna odpovídá rozvrhu právě jedné třídy), kdy prvních 8 položek znázorňuje pondělní rozvrh, dalších 8 úterní a tak dále až do pátku. Učebny (rozvrhy tříd) jsou tedy v poli umístěny za sebou. Jednotlivé položky v poli mohou být reprezentovány dvěma způsoby. Buďto hodnotou `null`, což znamená volnou hodinu, nebo objektem nesoucím odkaz konkrétní třídy a předmětu, který se v danou hodinu a daný den v učebně vyučuje. Na obrázku 3.3 můžeme vidět, jak by vypadala část jedince učebny tvořena dvěma učebnami.



Obrázek 3.3: Na obrázku můžeme vidět část jedince učebny, který tvoří dvě třídy.

Třídy

Část třídy bude reprezentována rovněž polem. Velikost bude rovna součtu všech předmětů každé třídy. Všechny třídy budou uložené do pole ve stejném pořadí, jako v případě tříd v části jedince učebny. Každá třída bude vlastnit takový počet položek, kolik má předmětů. Znamená to tedy, že položka bude obsahovat odkaz na předmět dané třídy a učitele, který danou třídu na tento konkrétní předmět bude vyučovat. Na obrázku 3.4 můžeme vidět, jak by tato část jedince vypadala s dvěma třídami, kdy každá třída má 8 předmětů.



Obrázek 3.4: Na obrázku můžeme vidět, jak bude vypadat část jedince třídy, kdy jedince tvoří dvě třídy o osmi předmětech.

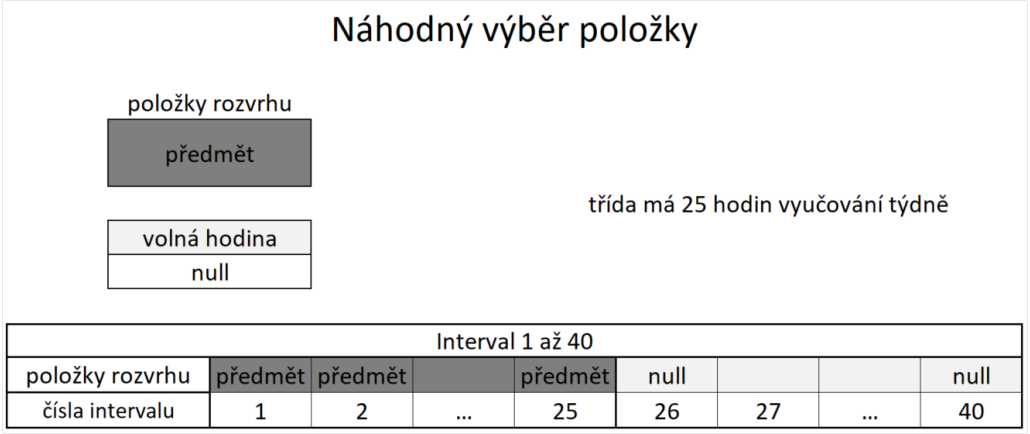
3.6.2 Vygenerování počáteční populace

Počáteční populace bude tvořena 10000 jedinci, kdy data každého jedince budou generovány náhodně. Jak již bylo řečeno, jedinec se skládá z částí učebny a třídy a v části učebny odpovídá rozvrhu každé třídy právě 40 hodnot. Při vytváření části učebny postupně procházíme všechny třídy a náhodně vybíráme předmět nebo volnou hodinu. Výběr položek se provádí za pomoci diskrétního rovnoměrného rozdělení. Všechny položky mají stejnou pravděpodobnost, že budou vybrány. Při výběru k umístění položky na místo 1. až 4. hodiny každého dne vybíráme jeden předmět ze seznamu předmětů třídy. Pokud umisťujeme 5. až 8. hodinu, může být vybraná položka volná hodina, nebo náhodně vybraný předmět.

Pravděpodobnost vložení prázdné hodiny nebo předmětu se počítá následovně. Víme, že rozvrh tvoří předměty a volné hodiny. Celkový počet hodin v rozvrhu je 40 a každá hodina má stejnou pravděpodobnost na uložení. Budeme tedy pracovat s intervalem od 1 do 40. Pokud náhodně vygenerované

číslo bude menší než je součet všech hodin dané třídy (součet hodin všech předmětů), vybereme náhodně předmět ze seznamu předmětů. Pokud bude číslo větší, vložíme hodnotu `null` (volná hodina) [4].

Na obrázku 3.5 máme znázorněný interval, jak se postupuje při rozhodování mezi uložením předmětu a volné hodiny do části jedince učebny. Pokud by v tomto případě náhodné číslo bylo od 1 do 25, vybírali bychom náhodný předmět. Pokud by bylo od 26 do 40, uložili bychom volnou hodinu.

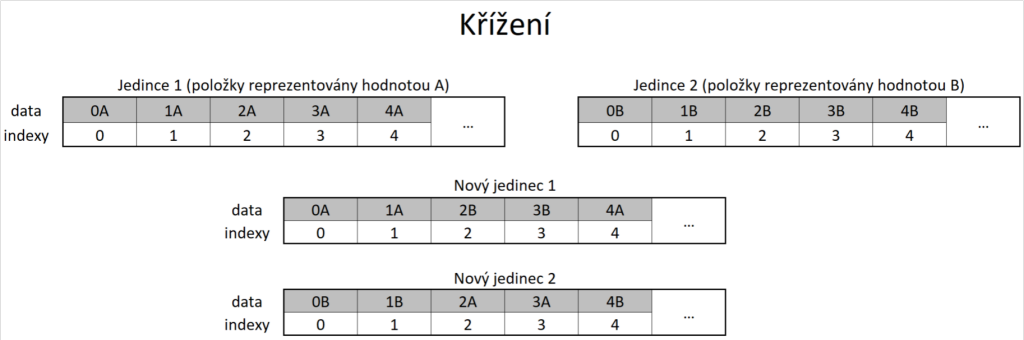


Obrázek 3.5: Obrázek náhodného výběru položky

Část jedince třídy bude tvořena všemi předměty každé třídy. Ke každému předmětu bude náhodně vybrán učitel ze seznamu učitelů.

3.6.3 Křížení

Důležité je zvolit si správnou strategii a kolik jedinců v populaci budeme křížit. Na základě předběžných experimentů jsem se rozhodl, že budeme křížit 60 % populace. Vybereme tedy z celé populace náhodně dva jedince (rodiče) a začneme provádět křížení. V obou případech, jak v části učebny, tak v části třídy budeme postupovat stejným způsobem. Vždy z křížení dvou rodičů vzniknou dva noví potomci, kteří nesou informaci obou rodičů. První potomek ponese první dvě hodiny prvního rodiče, další dvě hodiny druhého, další dvě hodiny opět prvního atd. dokud nebude nový jedinec kompletní. Druhý potomek bude vytvořen stejným způsobem, s tím rozdílem, že tentokrát nezačne informaci předávat první, ale druhý rodič. Jedná se o vícebodové křížení [3].



Obrázek 3.6: Na obrázku můžeme vidět strategii při křížení dvou jedinců a vzniku dvou nových potomků, kteří nesou informaci obou rodičů.

3.6.4 Mutace

Po operaci křížení přichází na řadu tzv. mutace, kdy dojde k náhodné změně informace v jedinci na náhodném místě. Na základě menších experimentů jsem se rozhodl, že budeme mutovat polovinu populace (5000 jedinců).

V části jedince učebny se budou mutovat dvě náhodně zvolené položky, neboť při experimentování s touto operací se kvalita jedinců zvyšovala rychleji. Nově vybraná položka může být jak volnou hodinou, tak i přiděleným předmětem. V případě mutování položky, která je volnou hodinou, nová položka musí být nějaký náhodný předmět. Pokud však mutujeme položku, která odpovídá nějakému předmětu, novou položkou je buďto volná hodina, nebo nějaký jiný předmět. Rozhodování, zda vybrat volnou hodinu nebo předmět se řeší stejným způsobem jako při ukládání 5. až 8. hodiny při vytváření prvotní populace v sekci 3.6.2.

V části jedince třídy se bude mutovat na rozdíl od části jedince učebny pouze jedna položka. Je to z toho důvodu, že i změna jedné položky se dostatečně projeví na kvalitě rozvrhu.

Vzhledem k tomu, že část učebny v jedinci nese mnohem více informací než část třídy, budeme mutovat dvě položky na náhodném indexu. Výběr nové položky budeme provádět stejným způsobem, jako při vytváření počáteční populace s tím rozdílem, že nebudeme brát v potaz, zda vybraný index odpovídá prvním čtyřem hodinám, nebo ne. Je tedy možné, že se nám v prvních čtyřech hodinách může objevit volná hodina. Aby došlo skutečně ke změně informace, bude daná hodina na indexu mutovat do té doby, dokud nezíská novou informaci. Tedy pokud budeme mutovat volnou hodinu, musíme dostat předmět. V případě mutování předmětu bude možné dostat jiný předmět nebo volnou hodinu.

3.6.5 Fitness funkce

Po operaci křížení a mutace je třeba vytvořit novou generaci, kterou by měli tvořit ti nejsilnější jedinci, u kterých je vysoká pravděpodobnost, že povedou ke správnému řešení. Kvalitu každého jedince budeme určovat pomocí tzv. fitness funkce.

Jak již bylo popsáno v sekci 2.4 fitness funkce pracuje s omezujícími podmínkami. Každému jedinci tedy vrátí hodnotu, kterou vypočte podílem splněných silných podmínek a celkového počtu silných podmínek. Následně k této hodnotě přičte počet splněných slabých podmínek. Fitness funkce kontroluje následující podmínky.

Učitel nemůže učit více předmětů ve stejný čas (silné podmínky)

Funkce bude procházet celého jedince, konkrétně část učebny. V případě, že se položka rovná `null` (žádný učitel na tomto místě v tuto dobu neučí), zvýšíme počet splněných silných podmínek. Stejně tak, pokud neučí na jiném místě ve stejnou dobu. To budeme kontrolovat průchodem celého jedince části učebny se skokem 40 (rozvrh třídy má 40 hodin týdně, tedy skočíme na další třídu na stejný den a hodinu).

Každý předmět třídy musí být v rozvrhu přesně tolikrát, kolikrát má být v týdnu vyučován (silné podmínky)

Pro kontrolu počtu předmětů v rozvrhu budeme používat hash mapu, která nám spočte počet výskytů daného předmětu každé třídy. Po vytvoření mapy projdeme každou třídu. Pokud se počet předmětů třídy shoduje s hodnotou v mapě, nebo je hodnota nižší, něž má ve skutečnosti být, zvýšíme počet silných podmínek o tuto hodnotu. Pokud však hodnota je vyšší, přičteme rozdíl správného počtu daného předmětu a hodnoty v mapě (správná hodnota - hodnota v mapě).

Třída může mít maximálně 7 hodin denně (silné podmínky)

Funkce bude procházet část jedince učebny. Bude počítat počet předmětů po 8 položkách (8 položek = 1 den). Pokud bude počet nižší než 8, zvýšíme počet silných podmínek.

Třídu na daný předmět vyučuje pouze jeden učitel (silné podmínky)

Tyto podmínky budou splněné vždy, neboť část třídy nám zajistí, že předmět bude učit pouze jeden učitel.

Předmět se v rámci jednoho dne vyučuje pouze jednou (slabé podmínky)

Pro ověření, zda se předmět dané třídy nachází v rozvrhu v rámci jednoho dne pouze jednou, si vytvoříme hash mapu, která nám ponese informace o počtu hodin předmětu konkrétní třídy v rámci dne. Nyní projdeme celou mapu a zkontrolujeme všechny položky, zda jejich hodnota se rovná 1 (vyskytuje se v daný den v rozvrhu pouze 1 krát). Pokud ano, zvýšíme slabé podmínky o 1.

První čtyři hodiny v rozvrhu jsou obsazeny (slabé podmínky)

Budeme procházet část jedince učebny. Pokud se položka nebude rovnat `null` (volné hodině) a bude se jednat o 1. až 4. hodinu v rámci jednoho dne, zvýšíme slabé podmínky o 1. Pauza na oběd by tedy při splnění všech těchto podmínek měla být nejdříve 5. hodinu.

Učitel učí předměty dle jeho specializace (slabé podmínky)

Na rozdíl od ostatních slabých podmínek, kde jsme prohledávali část jedince učebny, budeme prohledávat část jedince třídy. Vzhledem k tomu, že tato část bude obsahovat položky nesoucí předmět a učitele, zkontrolujeme pouze, zda se předmět shoduje se specializací učitele v položce. Pokud ano, zvýšíme slabé podmínky o 1.

3.6.6 Výběr nové populace

Po operacích křížení a mutace přichází na řadu selekce. Na základě zkoušení jak typu rulety, rulety s pořadím i turnaje, se nejlépe chovala modifikovaná forma turnaje. Klasická forma turnaje již byla napsána v sekci 2.4, při výběru každého jedince se vybere náhodně N jedinců. Ten jedinec z vybrané množiny, který má nejvyšší ohodnocení, je vybrán do nové populace. V našem případě však místo náhodného výběru N jedinců při každém novém ukládání jedince do nové populace vybereme 25 nejsilnějších jedinců dle ohodnocení fitness funkce z celé populace na začátku selekce. Následně za pomoci této množiny vytváříme náhodným výběrem jedinců celou novou populaci, kdy

pravděpodobnost výběru každého jedince z množiny je pro všechny jedince stejná.

Víme, že populaci tvoří 10000 jedinců. Takto vysoký počet jedinců byl zvolen záměrně, neboť při lehkém zkoušení generování při různém počtu jedinců v generaci dokázal nejrychleji najít splnitelné řešení. Stejným způsobem bylo zvoleno, že nejsilnějších jedinců se vybírá právě 25. Abychom zajistili, že se všichni vybraní nejsilnější jedinci dostanou do nové populace, i když je velmi malá pravděpodobnost, že by se nějaký jedinec při takto vysokém počtu jedinců v populaci do nové populace nedostal, uložíme je na začátek pole představujícího novou populaci. Zbytek populace doplníme náhodně vybranými jedinci ze seznamu nejsilnějších jedinců. Nová populace tedy obsahuje více stejných jedinců.

3.7 Prohledávání a ukládání jen nekolizních položek

Aplikace obsahuje další algoritmus, který bude fungovat na postupném ukládání položek do rozvrhu po jednotlivých třídách. Položku do rozvrhu uloží pouze tehdy, pokud neporušuje žádnou silnou podmínku, tedy uložením nenastane žádná kolize.

Nejprve se náhodně přiřadí každému předmětu každé třídy učitel dle specializace. Tím se zajistí, že při opakovaném spuštění generování za pomoci stejných vstupních dat nedostaneme pokaždé stejný výsledek. Po přidělení učitelů začne samotné generování.

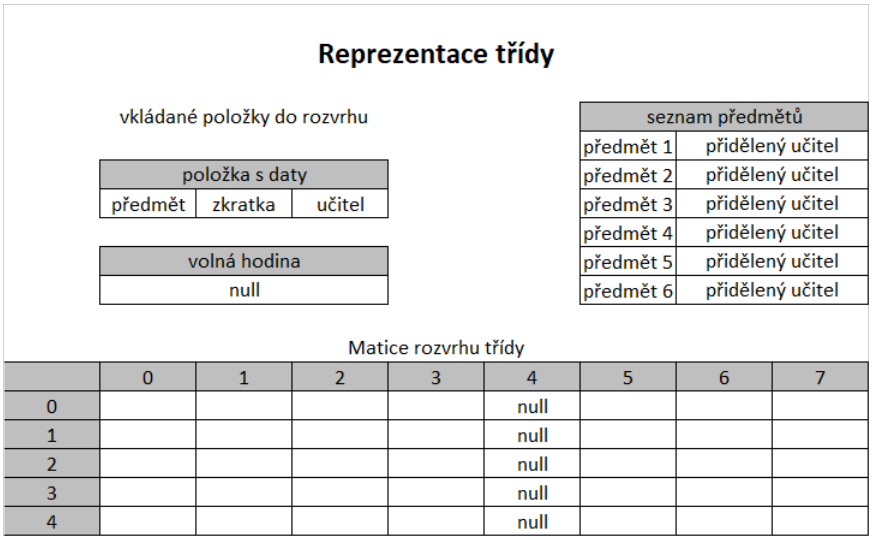
Algoritmus bude pracovat s dvěma hlavními částmi a to polem objektů nesoucím rozvrhy tříd a seznamem představujícím umístění všech učitelů v rozvrhu.

3.7.1 Pole objektů nesoucí rozvrhy tříd

Každý objekt bude obsahovat dvě důležité vlastnosti, které ponesou veškeré informace o rozvrhu dané třídy.

První bude pole všech předmětů třídy, kdy ke každému předmětu bude přidělen učitel, který má příslušnou specializaci. Položky tedy budou znázorňovat předmět a učitele, který jej vyučuje.

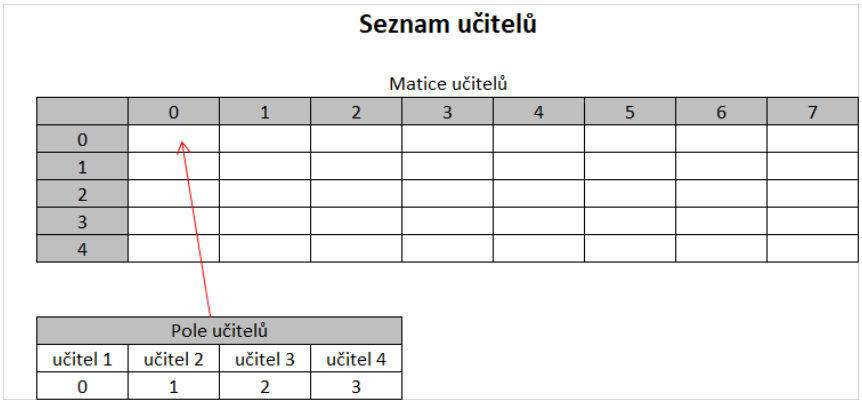
Druhou důležitou položkou bude matice představující rozvrh. Bude tedy obsahovat 5 řádek (5 dní) a 8 sloupců (8 hodin denně). Do matice budou přidávány položky, které ponesou informaci o předmětu a učiteli, který jej vyučuje. Volná hodina bude v rozvrhu reprezentována hodnotou `null`.



Obrázek 3.7: Obrázek představuje, jak bude vypadat objekt, který ponese veškerá data o rozvrhu třídy.

3.7.2 Seznam všech učitelů v rozvrhu

Pro uchovávání informací o všech učitelích, kteří již byli uloženi do rozvrhu, bude vytvořena matice o 5 řádcích (5 dní) a 8 sloupcích (8 hodin denně), která bude sloužit ke kontrole, zda učitel neučí na více místech najednou. Každá její položka bude obsahovat pole učitelů, kteří v daný den a hodinu již vyučují. Tedy při přidávání nové položky do rozvrhu nebude nutné procházet všechny rozvrhy každé třídy a kontrolovat, zda učitel již učí na jiném místě nebo ne. Pouze se podíváme do pole učitelů matice, které odpovídá hodině a dni ukládané položky, a zkontrolujeme, zda se učitel v tomto poli nenachází.



Obrázek 3.8: Na obrázku vidíme příklad reprezentace seznamu učitelů v rozvrhu. Vidíme, že v první hodinu v pondělí již vyučují učitelé 1, 2, 3 a 4.

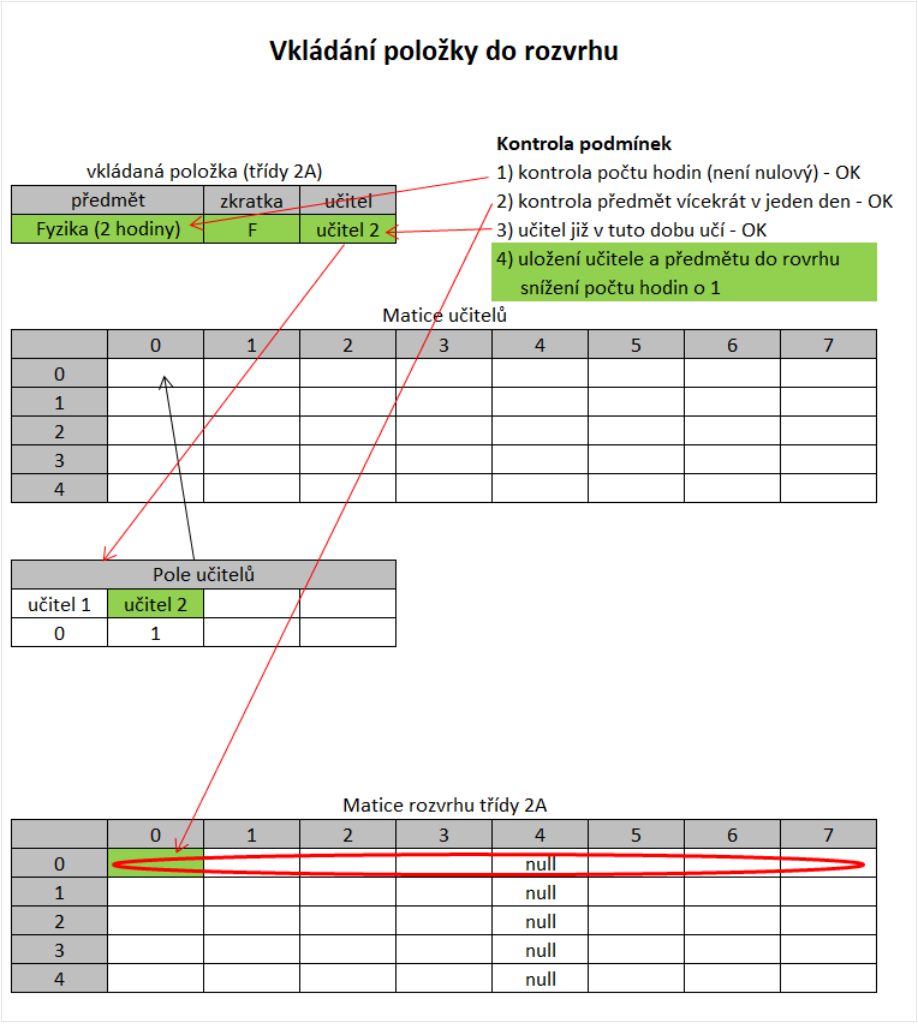
3.7.3 Postup při generování

Jak již bylo řečeno, každá třída bude reprezentována objektem, který ponese rozvrhovou matici. Položky do jednotlivých rozvrhů budeme ukládat postupně po jednotlivých třídách. Nejdříve tedy vytvoříme rozvrh první třídy, dále na základě informací z vytvořeného rozvrhu budeme vytvářet další. Takto budeme pokračovat do té doby, dokud nebudou všechny rozvrhy vytvořeny.

Ukládání položek bude probíhat následujícím způsobem. Začneme procházet seznam předmětů třídy. Nejprve zkontrolujeme, zda počet kontrolovaného předmětu již není nulový (předmět byl již plně umístěn). Pokud není, zkontrolujeme, zda již v tento den nebyl do rozvrhu umístěn (zamezení umístění jednoho předmětu vícekrát v rámci jednoho dne). Pokud i tato podmínka bude splněna, nahlédneme na stejné souřadnice, na kterých se aktuálně v matici rozvrhu nacházíme, do matice učitelů. Projdeme pole na příslušném indexu matice a zkontrolujeme, zda již učitel, který má předmět vyučovat, nevyučuje na jiném místě. Pokud jsou všechny podmínky splněny, předmět uložíme do rozvrhu a snížíme jeho počet o 1. Pokud ne, pokusíme se umístit další předmět ze seznamu předmětů. V případě, že se nepodaří uložit žádný předmět, uloží se na dané souřadnice hodnota `null` (volná hodina). Na obrázku 3.9 můžeme vidět postup, jak vkládání nové položky bude probíhat.

Abychom docílili toho, že výuka bude probíhat již od ranních hodin, budeme ukládat do rozvrhu po sloupcích. Začneme tedy první hodinou v pondělí, pokračujeme první hodinou v úterý atd. Pátá hodina bude náležet pauze na oběd a je nastavena atributem třídy, proto hodnota na těchto pozicích bude `null`. V případě genetického algoritmu není zaručeno, že pátá hodina bude právě pauza na oběd, neboť algoritmus pouze kontroluje, zda jsou první čtyři hodiny zaplněny.

Po vygenerování všech rozvrhů ještě zkontrolujeme, zda se všechny předměty podařilo umístit, nebo ne. Pokud se všechny předměty nepodařilo umístit, program vyvolá výjimku, kde uživatele upozorní, že rozvrh není kompletní. V takovém případě se uživatel může pokusit pustit generování znovu, neboť řešení záleží na náhodném přiřazení učitelů k jednotlivým předmětům tříd před spuštěním generování. Může se tedy stát, že při opětovném spuštění se vygenerování podaří.



Obrázek 3.9: Na obrázku je zobrazen příklad vkládání nové položky do rozvrhu třídy 2A. Jsou zde označeny veškeré operace, které je nutné při ukládání ověřit, aby nevznikla kolize, což by vedlo k chybnému řešení.

4 Implementace

Dle návrhu byla aplikace naprogramována objektově v programovacím jazyce java pomocí vývojového prostředí Eclipse [1], kde byl také program důkladně otestován.

4.1 Vstupní data

V této sekci bude popsán formát vstupních souborů.

4.1.1 Data ve formátu CSV

Dle návrhu zdrojů v sekci 3.4 byla vytvořena data ve formátu CSV. Tedy soubory s informacemi jednotlivých tříd a soubor s daty všech učitelů.

Data třídy

Data tříd se načítají z adresáře, kde jednotlivé soubory obsahují na konci názvu řetězec `Trida.csv`. Tento způsob pojmenování byl zvolen záměrně, abychom soubory při načítání z adresáře mohli lehce identifikovat od jiných souborů. Soubor s daty třídy může vypadat například `1ATrida.csv`.

Data třídy jsou reprezentovány následujícím způsobem. První řádek souboru nese název třídy. Každý další řádek obsahuje právě jeden předmět.

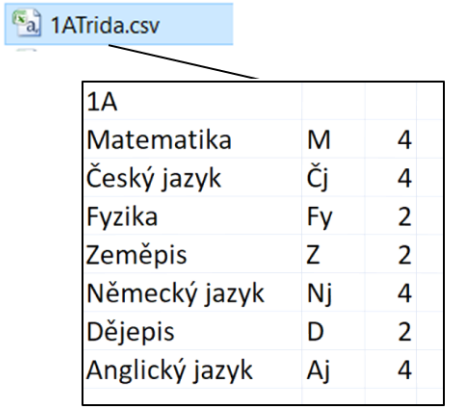
Informace o předmětu jsou zadány v pořadí `název předmětu;zkratka předmětu;počet hodin týdně`. Ke správné funkčnosti programu je třeba, aby se názvy předmětů shodovaly s názvy specializací učitelů.

Na obrázku 4.1 je ukázán soubor obsahující data třídy.

Data učitelé

Všichni učitelé jsou uloženi v jednom souboru. Každý řádek odpovídá právě jednomu učiteli. Informace jsem ukládány v pořadí `jméno učitele;názvy všech specializací (specializace jsou odděleny středníkem)`.

[1] https://www.eclipse.org/



Obrázek 4.1: Na obrázku jsou data souboru `1ATrida.csv`, který obsahuje veškeré informace třídy.

Obrázek 4.2: Na obrázku jsou data souboru `Ucitele.csv`, který obsahuje veškeré informace o každém učiteli.

4.1.2 Data ve formátu XML

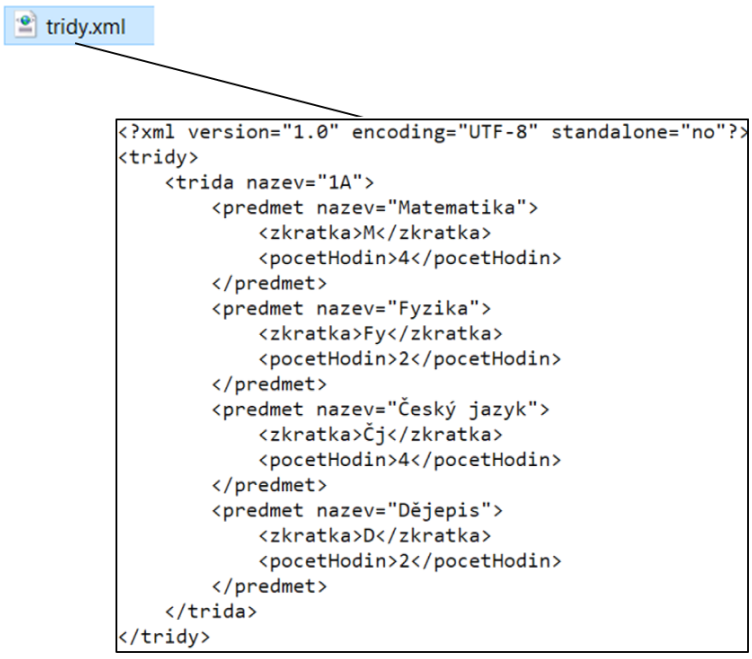
I v případě vytvoření zdrojových dat ve formátu XML bylo postupováno dle návrhu v sekci 3.4. Byly tedy vytvořeny dva XML soubory. Jeden pro třídy a druhý pro učitele.

Třídy

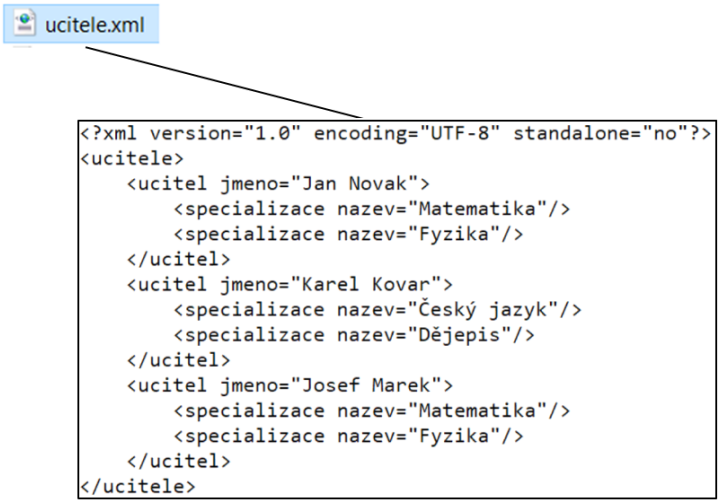
Soubor obsahuje všechny informace každé třídy. Tedy název a jednotlivé předměty. Na obrázku 4.3 je ukázán obsah vstupního XML souboru tříd.

Učitelé

Stejně tak jako v případě tříd jsou učitelé uložení v jedno souboru. Na obrázku je ukázán obsah vstupního XML souboru učitelů.



Obrázek 4.3: Obrázek znázorňuje strukturu souboru tridy.xml, který obsahuje všechny třídy. V tomto případě je v souboru pouze třída 1A.



Obrázek 4.4: Obrázek znázorňuje strukturu souboru ucitele.xml, který obsahuje data všech učitelů.

4.2 Výstupní data

Program dokáže vygenerovat rozvrhy ve formátu CSV, XML a PDF. Uživatel si může posledním zadaným parametrem při spuštění vybrat požadovaný formát dat.

4.2.1 Výstupy ve formátu CSV

V případě vygenerování CSV souborů program vytvoří adresář, do kterého uloží rozvrh každé třídy a každého učitele. Jeden soubor odpovídá jedné třídě nebo učiteli.

Každý soubor s daty třídy obsahuje vygenerovaný školní rozvrh a seznam učitelů, kteří jednotlivé předměty třídy vyučují. Název souboru odpovídá názvu třídy.

Soubor s daty učitele obsahuje školní rozvrh a seznam třída předmět, aby bylo zřejmé, který předmět a třídu vyučuje. Název souboru odpovídá jménu učitele.

1A.csv

třída: 1A							
1	2	3	4	5	6	7	8
M	Fy	D	Aj		Z		
M	Čj	D	Nj		Z		
M	Čj	Aj	Nj				
M	Čj	Aj	Nj				
Fy	Čj	Aj	Nj				
Fy	František Křešnička						
D	Jana Kovářová						
Aj	Jan Lukeš						
Z	Jakub Mikeš						
Nj	Kateřina Mladá						
M	Roman Hájek						
Čj	Ivan Křešnička						

Obrázek 4.5: Na obrázku můžeme vidět obsah vygenerovaného souboru obsahující data třídy.

Jakub Mikeš.csv

Učitel: Jakub Mikeš							
1	2	3	4	5	6	7	8
M(3A)					Z(1A)		
M(3A)					Z(1A)		
M(3A)					Z(4A)		
M(3A)					Z(4A)		
Třídy:							
Matematika: 3A							
Zeměpis: 4A							
Zeměpis: 1A							

Obrázek 4.6: Na obrázku můžeme vidět obsah vygenerovaného souboru obsahujícího rozvrh třídy.

4.2.2 Výstupy ve formátu XML

Při generování rozvrhů ve formátu XML program vytvoří adresář, do kterého uloží dva soubory. Jeden soubor s názvem `rozvrhyTrid.xml`, který obsahuje data o rozvrhu každé třídy a druhý s názvem `rozvrhyUcitel.xml` obsahující rozvrhy všech učitelů.

Soubor `rozvrhyTrid.xml` obsahuje veškerá data o rozvrhu každé třídy. Každá třída obsahuje název a každý den v týdnu. Každý element dne obsahuje předměty. Každý předmět obsahuje informaci o čísle dne, čísle vyučované hodiny, názvu předmětu, zkratky a jména učitele.

Soubor `rozvrhyUcitel.xml` obsahuje veškerá data o rozvrhu každého učitele. Tím je jméno a element každého dne v týdnu, kde jsou uloženy předměty. Každý předmět obsahuje informaci o čísle dne, čísle vyučované hodiny, názvu předmětu, zkratky a třídy, kterou učitel na daný předmět vyučuje.

rozvrhyTrid.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<rozvrhy>
  <trida nazev="1A">
    <den id="1" nazev="Po">
      <predmet den="1" hodina="1" nazev="Matematika" ucitel="Jakub Mikeš" zkratka="M"/>
      <predmet den="1" hodina="2" nazev="Český jazyk" ucitel="František Křešnička" zkratka="Čj"/>
      <predmet den="1" hodina="3" nazev="Zeměpis" ucitel="Zdeněk Kovář" zkratka="Z"/>
      <predmet den="1" hodina="4" nazev="Německý jazyk" ucitel="Jan Novák" zkratka="Nj"/>
      <predmet den="1" hodina="6" nazev="Anglický jazyk" ucitel="Jan Lukeš" zkratka="Aj"/>
    </den>
    <den id="2" nazev="Ut">
      <predmet den="2" hodina="1" nazev="Matematika" ucitel="Jakub Mikeš" zkratka="M"/>
      <predmet den="2" hodina="2" nazev="Český jazyk" ucitel="František Křešnička" zkratka="Čj"/>
      <predmet den="2" hodina="3" nazev="Zeměpis" ucitel="Zdeněk Kovář" zkratka="Z"/>
      <predmet den="2" hodina="4" nazev="Dějepis" ucitel="Jana Kovářová" zkratka="D"/>
      <predmet den="2" hodina="6" nazev="Anglický jazyk" ucitel="Jan Lukeš" zkratka="Aj"/>
    </den>
    <den id="3" nazev="St">
      <predmet den="3" hodina="1" nazev="Matematika" ucitel="Jakub Mikeš" zkratka="M"/>
      <predmet den="3" hodina="2" nazev="Český jazyk" ucitel="František Křešnička" zkratka="Čj"/>
      <predmet den="3" hodina="3" nazev="Německý jazyk" ucitel="Jan Novák" zkratka="Nj"/>
      <predmet den="3" hodina="4" nazev="Dějepis" ucitel="Jana Kovářová" zkratka="D"/>
```

Obrázek 4.7: Obrázek obsahuje část vygenerovaného souboru obsahujícího rozvrhy tříd.

Soubor `rozvrhyUcitel.xml` obsahuje veškerá data o rozvrhu každého učitele. Tím je jméno a element každého dne v týdnu, kde jsou uloženy předměty. každý předmět obsahuje informaci o čísle dne, čísle vyučované hodiny, názvu předmětu, zkratky a třídy, kterou učitel na daný předmět vyučuje.

rozvrhyUcitelu.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<rozvrhy>
  <ucitel jmeno="Jakub Mikeš">
  <den id="1" nazev="Po">
  <predmet den="1" hodina="1" predmet="Matematika" trida="1A" zkratka="M"/>
<predmet den="1" hodina="3" predmet="Zeměpis" trida="2A" zkratka="Z"/>
</den>
<den id="2" nazev="Ut">
  <predmet den="2" hodina="1" predmet="Matematika" trida="1A" zkratka="M"/>
<predmet den="2" hodina="3" predmet="Zeměpis" trida="2A" zkratka="Z"/>
</den>
<den id="3" nazev="St">
  <predmet den="3" hodina="1" predmet="Matematika" trida="1A" zkratka="M"/>
</den>
<den id="4" nazev="Ct">
  <predmet den="4" hodina="1" predmet="Matematika" trida="1A" zkratka="M"/>
<predmet den="4" hodina="2" predmet="Zeměpis" trida="4A" zkratka="Z"/>
</den>
<den id="5" nazev="Pa">
  <predmet den="5" hodina="2" predmet="Zeměpis" trida="4A" zkratka="Z"/>
</den>
```

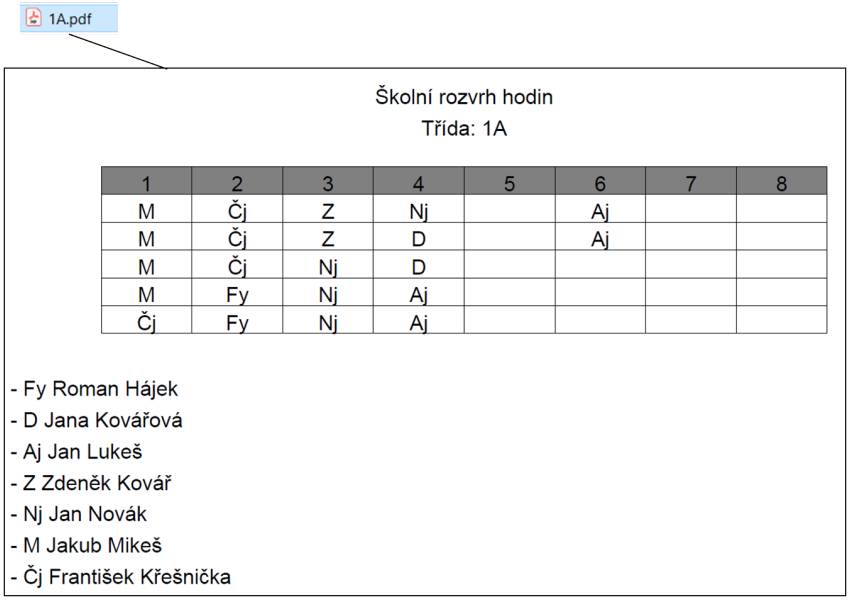
Obrázek 4.8: Obrázek obsahuje část vygenerovaného souboru obsahujícího rozvrhy učitelů.

4.2.3 Výstupy ve formátu PDF

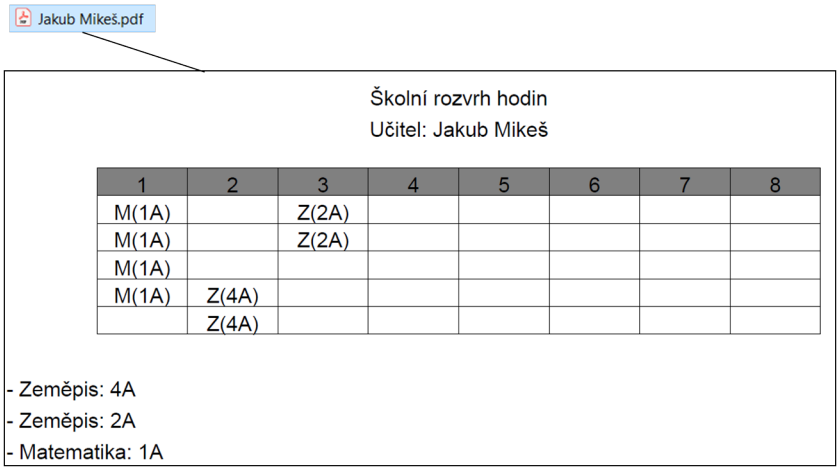
V případě generování rozvrhů ve formátu PDF program vytvoří dva adresáře. `RozvrhyTridPDF` a `RozvrhyUcitelePDF`. Do těchto souborů následně uloží vygenerované rozvrhy. Každému rozvrhu odpovídá vlastní soubor.

V případě rozvrhu třídy vygenerovaný soubor obsahuje tabulku představující rozvrh a seznam předmět učitel. Tento seznam nese informaci o všech předmětech třídy a to učitele, který jej vyučuje.

Soubor s rozvrhem učitele je reprezentován stejným způsobem, jako v případě souboru s rozvrhem třídy. Obsahuje tabulku rozvrhu, označení, že se jedná o rozvrh učitele a seznam předmětů tříd, které v rozvrhu vyučuje.



Obrázek 4.9: Na obrázku je vidět obsah souboru s rozvrhem třídy 1A.



Obrázek 4.10: Na obrázku je vidět obsah souboru s rozvrhem učitele Jakub Mikeš.

4.3 Popis struktury celé aplikace

Celá aplikace je uspořádána do stromové struktury. V následující části bude popsána kořenová struktura celého programu.

4.3.1 Struktura adresářů

V kořenovém adresáři se nachází několik položek, které zde budou popsány.

`src/`
Složka obsahuje balíček `bp`, ve kterém jsou uložené veškeré zdrojové soubory aplikace. Hlavní třídou je třída `Main`.

`test/`
Složka obsahuje balíček `bp`, ve kterém jsou uložené veškeré zdrojové soubory testovacích tříd aplikace.

`bin/`
Tato složka uchovává aplikaci po jejím zkompilování.

`testingData/`
Složka obsahuje soubory určené k testování.

`dataCSV/`
Složka osahuje předdefinované vstupní CSV soubory.

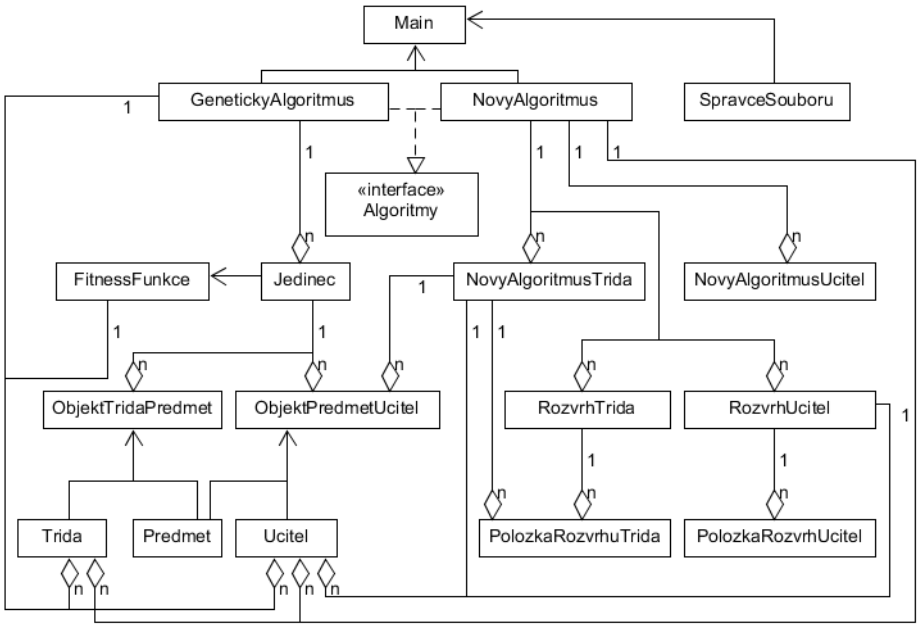
`dataXML/`
Složka osahuje předdefinované vstupní XML soubory.

`references/`
Složka obsahuje font *Arial*, aby vytvořené PDF soubory měly správnou diakritiku.

`lib/`
složka obsahuje knihovnu *itextpdf-5.4.0.jar*, kterou aplikace používá při zapisování rozvrhů do souborů ve formátu PDF.

4.3.2 Diagram tříd

Všechny třídy aplikace jsou uloženy v balíčku `bp`, který je v adresáři `src/`. Na obrázku 4.11 můžeme vidět zjednodušený diagram tříd celé aplikace. Úplný diagram tříd pak nalezneme v příloze B na obrázku B.1. Testovací třídy jsou uloženy ve stejnojmenném balíku v adresáři `test/`.



Obrázek 4.11: Diagram tříd celé aplikace.

4.4 Třídy programu

V této části budou popsány všechny třídy, které byly k implementaci použity. Každá třída nám zajišťuje vytvoření konkrétního objektu s důležitými atributy, které nesou příslušné informace. Pro přístup k atributům každá třída obsahuje gettery a settery.

4.4.1 Dokumentace

K programu byla vytvořena uživatelská dokumentace, která se nachází ve složce `doc/`. Samotné části jednotlivých metod jsou pak důkladně okomentovány v samotném kódu aplikace.

4.4.2 Spouštěcí třída celé aplikace

Spouštěcí třídou celého programu je třída `Main`, která obsahuje metodu `main()`. Tato metoda zpracovává parametry zadané uživatelem a zajišťuje běh celé aplikace. Dále obsahuje metody, které zpracovávají parametry z příkazové řádky.

Zpracovávání parametrů probíhá v následovném pořadí. Pro lepší orientaci budeme parametry číslovat od 1.

Nejprve se zpracuje parametr 2, který obsahuje cestu k souboru s daty tříd. V případě CSV vstupů k adresáři, kde jsou jednotlivé soubory uloženy.

Dále se zpracuje parametr 3, který obsahuje cestu k souboru s daty všech učitelů.

Pokud se podařilo úspěšně načíst vstupní data, zpracuje se parametr 1, který spustí generování vybraným algoritmem.

Poté, co se podaří vygenerovat příslušná data rozvrhů, program zpracuje parametr 4 a 5. Dle parametru 4 vytvoří adresář z názvu cesty a následně do něm uloží data v požadovaném formátu dle parametru 5.

4.4.3 Práce se soubory

Veškerou práci spojenou se soubory provádí třída `SpravceSouboru`.

Třída zajišťuje jak čtení dat ze souborů, tak jejich zapisování do nově vytvořených souborů. Dokáže načítat data o třídách a učitelích ve formátu CSV a XML. Vygenerované rozvrhy pak zapisovat ve formátu CSV, XML nebo PDF.

V případě zápisu dat ve formátu PDF byla použita externí knihovna `itexpdf.jar`, která umožňuje zápis v tomto formátu. Aby vytvořený PDF soubor zobrazoval znaky s diakritikou jako jsou např. ČčĚě..., bylo nutné použít externí font, proto byl použit `arial.ttf`, který je uložený ve složce `references/`.

4.4.4 Třídy reprezentující třídu, předmět a učitele

Jedná se o třídy vytvářející objekty, které používá jak genetický algoritmus, tak algoritmus pracující na postupném prohledávání a ukládání jen nekoliz-ních položek. Jsou hlavními prvky při vytváření rozvrhů.

Třída `Predmet` představuje objekt předmětu třídy. Vlastní tři atributy a to název předmětu, zkratku a počet hodin týdně.

Třída `Trida` představuje objekt načtené třídy ze vstupního souboru. Vlastní dva atributy. Jedním je název třídy a druhým pole předmětů (objektů `Predmet`).

Třída `Ucitel` představuje objekt učitele. Vlastni dva atributy. Jméno učitele a pole specializací (názvy předmětů).

4.4.5 Třídy reprezentující rozvrh

Abychom mohli stejným způsobem zapisovat vygenerované výsledky obou algoritmů do souborů, byly vytvořeny dva hlavní objekty, které nám repre-

zentují rozvrh třídy a rozvrh učitele. Oba dva algoritmy nám tedy vrací pole těchto objektů.

Prvním hlavním objektem je `RozvrhTrida`, který vytváří stejnojmenná třída. Objekt představuje rozvrh jedné třídy, kdy stavebním prvkem rozvrhu jsou objekty `PolozkaRozvrhTrida` a hodnota `null` (volná hodina). Také obsahuje atribut název třídy. V případě genetického algoritmu počáteční index v poli části jedince třídy (kde začínají data této třídy). Dále pak index třídy v poli všech načtených tříd a celkový počet předmětů dané třídy.

Samotný objekt `PolozkaRozvrhTrida` je tvořen třemi hlavními atributy. Název předmětu, jeho zkratka a jméno učitele, který ho vyučuje.

Druhým hlavním objektem je `RozvrhUcitel`, který vytváří také stejnojmenná třída. Představuje rozvrh jednoho učitele. Stavebním prvkem rozvrhu jsou objekty `PolozkaRozvrhUcitel` a hodnoty `null` (volná hodina). Dále obsahuje hlavní atributy a to jméno učitele a index v poli všech načtených učitelů.

Objekt `PolozkaRozvrhUcitel` vlastní tři hlavní atributy. Název předmětu, jeho zkratku a název třídy, kterou na tento předmět vyučuje.

4.4.6 Třídy spojené s genetickým algoritmem

Samotný genetický algoritmus používá k implementaci několik tříd.

První důležitou třídou pro generování je třída `Jedinec`, která vytváří stejnojmenný objekt. Dle návrhu zadání je jedinec reprezentován dvěma částmi. Část jedince učebny tvoří pole objektů `ObjektTridaPredmet`. Část jedince třídy tvoří pole objektů `ObjektPredmetUcitel`.

`ObjektTridaPredmet` je objekt, který reprtezentuje položku části jedince třídy. Má dva důležité atributy a tím jsou objekty `Trida` a `Predmet`.

Objekt `ObjektPredmetUcitel` reprezentuje položku v části jedince třídy. Vlastní dva hlavní atributy a tím jsou objekty `Predmet` a `Ucitel`.

Další důležitou třídou gnetického algoritmu je třída `FitnessFunkce`. Tato třída nám zajišťuje ohodnocení kvality jedince. Nejprve se třídě nastaví testovaný jedinec. Následně pomocí ověřovacích metod se provede kontrola splněných silných a slabých podmínek. Po získání počtu všech splněných podmínek vypočte dle vzorce zmíněného v sekci 3.6.5 kvalitu jedince, kterou vrátí genetickému algoritmu.

Samotné generování zajišťuje třída `GenetickýAlgoritmus`, která implementuje interface `Algoritmy`. Třída obsahuje několik důležitých atributů a to počet jedinců v populaci, maximální počet generací, počet nejsilnějších jedinců, počet dní v týdnu, počet hodin denně, objekt třídy `FitnessFunkce` pro zjišťování kvality jedinců a pole objektů `Jedinec`, které představuje

jednu populaci.

Pro generování byly vytvořeny metody, které zajišťují veškeré operace spojené s genetickým algoritmem. Nejprve dojde k náhodnému vytvoření prvotní populace stejným způsobem, jako je v návrhu v sekci 3.6.2. Dále začnou probíhat operace v pořadí křížení, mutace a výběr nové populace (selekce), které se cyklicky opakují do té doby, dokud nedojde k vytvoření maximálního počtu generací. Na obrázku 4.12 můžeme vidět pseudokód celé metody, která zajišťuje generování.

```
generovat(){
    //vytvoreni prvotni populace
    vygenerovatPrvotniPopulaci();
    //pocet prubehu = poctu generaci
    while(pocetGeneraci){
        krizeni(); //krizeni jedincu
        mutace(); //mutace jendu
        //ohodnoceni kvality vsech jedincu nasledne
        //vybrani 25 nejsilnejsich a vytvoreni z nich
        //nove populace
        getNovaPopulace();
    }
}
```

Obrázek 4.12: Pseudokód metody pro generování školních rozvrhů.

`GenetickyAlgoritmus` vrací třídě `Main` pole objektů `RozvrhTrida` a `RozvrhUcitel`, které jsou následně zapisovány do souborů.

4.4.7 Třídy spojené s algoritmem pracujícím na postupném prohledávání a ukládání jen nekolizních položek

Pro řešení tohoto algoritmu byla vytvořena třída s názvem `NovyAlgoritmus`, která zajišťuje celé generování. Algoritmus pracuje s dvěma důležitými objekty.

Prvním je objekt `NovyAlgoritmusTrida`, který nese informace o rozvrhu třídy. Vlastní dva hlavní atributy. Prvním atributem je pole objektů `ObjektPredmetUcitel`, který představuje seznam předmětů a učitelů, kteří třídu na daný předmět vyučují. Pro naplnění seznamu je vytvořena metoda, která vždy vybere všechny učitele, kteří daný předmět mohou vyučo-

vat a jednoho z nich náhodně vybere. Druhým atributem je matice objektů `PolozkaRozvrhuTrid`, která představuje školní rozvrh třídy. Do matice se budou vkládat buďto konkrétní položky rozvrhu, nebo hodnota `null`, která představuje volnou hodinu.

Druhým důležitým objektem je objekt `NovyAlgoritmusUcitel`, který algoritmus používá pro tvorbu matice, která přestavuje umístěné učitele v rozvrhu. Objekt vlastní jen jeden atribut a tím je pole učitelů. Třída vlastní metodu, která je důležitá pro samotné generování a to na základě vloženého učitele parametrem zjistí, zda se v poli učitel nachází, nebo ne.

Nyní k samotné třídě `NovyAlgoritmus`. Třída implementuje interface `Algoritmy`. Hlavními atributy třídy je pole objektů `NovyAlgoritmusTrida`, který přestavuje rozvrh každé třídy a matic objektů `NovyAlgoritmusUcitel`, která představuje seznam učitelů umístěných v rozvrhu. Samotným vytvořením algoritmu dojde k nainicializování pole tříd a náhodnému přidělení učitelů ke každému předmětu třídy. Stejně tak při spuštění dojde k naincializování matice učitelů v rozvrhu.

Důležitou funkci zde hraje metoda při vkládání položky do rozvrhu, která volá metody, které ověřují, že uložením položky do rozvrhu nenastane žádná kolize. Nejprve dojde ke kontrole, zda všechny hodiny předmětu nebyly umístěny do rozvrhu. Pokud je podmínka splněna, zavolá se další metoda, která ověří, zda již předmět v daný den nebyl vyučován. Pokud i tato vlastnost je splněna, zavolá se poslední metoda, která zkontroluje z dat matice, představující seznam učitelů, jesti již učitel v tuto dobu neučí. Pokud jsou všechny podmínky splněny, uloží položku do rozvrhu.

Na obrázku 4.13 můžeme vidět pseudokód metody, která zajišťuje vkládání položky do rozvrhu. Parametr `indexTridy` odpovídá indexu třídy v poli tříd. Souřadnice `X,Y` znázorňují den a hodinu, na jaké místo v rozvrhu se snažíme položku umístit.

```
pridatPolozkuDoRozvrhu(indexTridy, x, y){
   for(vsechnyPredmetyTridy){
        //pocet hodin roven 0
        if(!nulovaHodnotaPredmetu){
            //uz je v rozvrhu pro tento den
            if(!vicekratVRozvrhuVJedenDen){
                //ucitel jiz vyucuje
                if(!ucitelJizUci){
                    //vsechny podminky splneny
                    //ulozeni do matice rozvrhu
                    //na souradnice x y
                    pridaniPredmetuDoRozvrhu();
                }
            }
        }
    }
}
```

Obrázek 4.13: Pseudokód metody pro vkládní položky do rozvrhu.

Všechny dosud zmíněné operace se provádí v jedné metoda, která zajišťuje celé vygenerování rozvrhů. Po ukončení generování se ještě zavolá metoda na ověření, zda se podařilo všechny umístit (hodnota všech předmětů je 0). Pokud tomu tak není, program na to upozorní uživatele vyhozením příslušné výjimky. Obrázek 4.14 obsahuje pseudokód, jak celé generování v metodě probíhá.

```
generovat(){
    for(VsechnyTridy){          //pole vsech trid
        for(pocetHodinDenne){   //8 hodin
            for(pocetDni){      //5 dni
                if(obed){       //4 hodina dne
                    //volna hodina
                    pridaniNullDoRozvrhu();
                }else{
                    pridaniPolozkyDoRozvrhu();
                }
            }
        }
    }
    //pocet hodin vsech predmetu je roven 0
    kontrolaVsechnyPolozkyUmisteny();
}
```

Obrázek 4.14: Pseudokód metody pro generování školních rozvrhů.

`NovyAlgoritmus`, stejně tak jako v případě třídy s genetickým algoritmem, vrací třídě `Main` pole objektů `RozvrhTrida` a `RozvrhUcitel`, které jsou následně zapisovány do souborů.

5 Testování

Poté, co byla aplikace naprogramována, byla také důkladně otestována. Ke každé třídě byla vytvořena testovací třída tvořená JUnit testy, které důkladně testují všechny metody každé třídy. Byla zkontrolována správná funkčnost při různých vstupech. Tedy jestli program v případě špatných vstupních dat vyhodí příslušné výjimky.

5.1 Testovací soubory

Aby bylo možné ověřit správnost aplikace na větším počtu dat, bylo vytvořeno několik testovacích souborů, které se nacházejí ve složce `testingData/`. Soubory jsou načítané pomocí vytvořených JUnit testů příslušných tříd.

Soubory obsahují jak validní data, tak nevalidní, aby bylo možné řádně ověřit, že se aplikace chová správně. Nejedná se však jen o data tříd a učitelů, ale i připravená data celých objektů jako jsou v případě genetického algoritmu části jedince třídy a učebny. Díky nim vytvoříme jedince nebo celé již vygenerované rozvrhy.

Všechny testy, které ověřují správný zápis do souborů jsou cestami odkazovány do stejného adresáře, kde jsou uloženy vstupní soubory určené pro testy. Tedy v adresáři `testingData/`. Tím je zajištěno, že veškeré soubory, které jsou spojeny s testy, jsou uloženy ve stejném adresáři.

5.2 Testovací třídy

Celkem bylo vytvořeno 17 testovacích tříd, které dohromady obsahují 277 JUnit testu a jsou uloženy ve složce `test/` v balíčku `bp`. Testy pracují jak s lokálně vytvořenými daty, tak se soubory, které obsahují jak validní, tak nevalidní data. Testují správnou funkcionalitu jak jednotlivých metod celé aplikace, zda vracejí správné výsledky a dokáží odhalit špatně zadané parametry, tak různé scénáře při spuštění. V tabulce 5.1 můžeme vidět, kolik JUnit testů jednotlivé třídy obsahují.

testovací třídy	počet JUnit testů
FitnessFunkceTest	40
GenetickýAlgoritmusTest	44
JedinecTest	8
MainTest	17
NovyAlgoritmusTest	23
NovyAlgoritmusTridaTest	13
NovyAlgoritmusUcitelTest	5
ObjektPredmetUcitelTest	4
ObjektTridaPedmetTest	4
PolozkaRozvrhUcitelTest	3
PolozkaRozvrhTridaTest	3
PredmetTest	8
RozvrhTridaTest	12
RozvrhUcitelTest	9
SpravceSouboruTest	64
TridaTest	11
UcitelTest	9

Tabulka 5.1: Tabulka počtu JUnit testů každé testovací třídy.

5.2.1 Testování správného spuštění a průběhu generování aplikace

Správné spuštění aplikace se testuje ve třídě `MainTest`, která obsahuje dle tabulky na obrázku 5.1 celkem 17 JUnit testů.

Testy důkladně prověřují správnost zadaných parametrů na jejichž základě závisí spuštění generování rozvrhů. Byla tedy testována správnost zadaných cest ke vstupním souborům, správně vybraný algoritmus, zda existuje cesta k adresáři, kam budou uloženy vygenerované rozvrhy a správný výběr formátu výstupních souborů. Stejně tak byl otestován celý průběh generování za použití obou algoritmů.

5.2.2 Testování vstupů a výstupů

Veškerou práci se soubory provádí třída `SpravceSouboru`. Pro její otestování byla vytvořena testovací třída s názvem `SpravceSouboruTest`, která dle tabulky na obrázku 5.1 obsahuje celkem 64 JUnit testů.

Je důležité před samotným spuštěním generování zajistit, že vstupní sou-

bory existují a neobsahují nevalidní data. Bylo tedy vytvořeno několik vstupních souborů, které záměrně obsahují chybná data. Dle názvu vstupního souboru je patrné, jaká chyba se v něm nachází. Platí to jak pro vstupní data ve formátu CSV, tak XML. Na základě těchto chybných souborů byly napsány testy, které dokazují, že se aplikace při nalezení chybných dat zachová správně. Vyhodí příslušnou výjimku.

Pro testování zápisu rozvrhů do souborů byli vytvořené testovací soubory, které obsahují data rozvrhů tříd a učitelů. Pomocí načtených dat testy ověřují správný zápis ve všech možných formátech (CSV, XML a PDF).

5.2.3 Testování genetického algoritmu

Genetický algoritmus je testován pomocí třídy `GenetickyAlgoritmusTest`. Jak vidíme v tabulce na obrázku 5.1 obsahuje 44 JUnit testů, které testují veškeré funkcionality tohoto algoritmu a to jak jednotlivých částí algoritmu, tak celku.

Jak již víme, první fází genetického algoritmu je vygenerování náhodné populace. Bylo tedy třeba ověřit, zda je jedinec vygenerovaný správně. Připravil jsem dva testovací soubory s daty tříd a daty učitelů. Následně jsem provedl vygenerování nové populace a ověřil jsem, zda velikost obou polí, jak části učebny, tak části třídy odpovídá správným velikostem, které odpovídají počtu tříd.

Další operací genetického algoritmu je křížení. Aby bylo možné ověřit, že křížení probíhá správně, byly vytvořeni za pomoci připravených testovacích souborů dva jedinci k otestování. Pomocí těchto jedinců bylo ověřeno, že se aplikace chová správně.

Stejným způsobem jako u operace křížení byla ověřena správná funkčnost mutace. Opět na základě připravených dat v externích souborech se vytvořili dva testovací jedinci. Za pomoci těchto jedinců byla mutace důkladně otestována.

Po řádném otestování jedné iterace genetického algoritmu, tedy vytvoření nové populace, křížení a mutace, bylo třeba otestovat správné vytvoření nové populace. Tedy jestli počet jedinců nově vytvořené populace odpovídá správné velikosti populace dané tímto atributem. Bylo třeba otestovat správnou funkčnost ohodnocování fitness funkce, pro zjištění kvality všech jedinců v populaci a na jejím základě získaných hodnot se vytváří nová populace.

Správné ohodnocování za pomoci fitness funkce bylo testováno ve třídě `FitnessFunkceTest`. Celkem třída obsahuje 40 JUnit testů. Pro testování se použil předem připravený jedinec, který byl načten ze souboru. Za pomoci JUnit testů byla otestována správnost použití hash map a jejich hodnot, na

jejichž základě se testují silné a slabé podmínky. Dále samotná kontrola ukládání počtu splněných silných a slabých podmínek, správný výpočet kvality jedince a správné vyhození výjimky při obdržení nevalidních dat.

Poslední testovanou věcí třídy bylo správné uložení školních rozvrhů do formátu, který vrací oba dva algoritmy.

5.2.4 Testování algoritmu založeného na prohledávání a ukládání jen nekolizních položek

Testování algoritmu založeného na prohledávání a ukládání jen nekolizních položek je prováděno ve třídě `NovyAlgoritmusTest`. Třída, jak je znázorněno v tabulce na obrázku 5.1, obsahuje 23 JUnit testů.

Nejprve bylo otestováno nainicializování pole objektů, které nese rozvrh tříd a seznam učitelů, kteří předměty vyučují. Dále pak nainicializování matice učitelů, která představuje umístění učitele v rozvrhu. Bylo tedy otestováno, zda se aplikace při inicializaci chová správně. V případě chybných dat dokáže vyhodit příslušnou výjimku. Dále byly testovány samotné metody spojené s generování.

Důležitou třídou pro generování a samotné nainicializování objektů tříd je třída `NovyAlgoritmusTrida`. Veškeré metody této třídy byly otestovány ve třídě `NovyAlgoritmusTridaTest`, která obsahuje 13 JUnit testů. Testuje správné přiřazení učitelů dle specializace jednotlivým předmětům při inicializaci pole objektů s rozvrhem a seznamem učitelů v hlavní třídě algoritmu. Dále testuje postup při vkládání nekolizních položek, jestli je předmět maximálně umístěný v rozvrhu, zda již není vyučován v konkrétní den, samotné uložení položky do rozvrhu a snížení hodnoty vloženého předmětu.

Při vkládání je ještě otestováno, zda se učitel v daný den a čas nachází v rozvrhu, to je zajištěno ve třídě `NovyAlgoritmusUcitel` a otestováno ve třídě `NovyAlgoritmusUcitel`.

Po úspěšném vygenerování školních rozvrhů třída ještě testuje, zda se hodnota každého předmětu třídy rovná 0, tedy, že jsou všechny předměty v rozvrhu umístěny. Aby bylo možné tento stav otestovat a zjistit, že se aplikace chová správně, byly připravena data s validním řešení, tak nevalidním. Na základě vytvořených dat byla tato metoda otestována.

Poslední věcí při testování třídy byl převod vygenerovaných dat do stejného formátu pro oba dva algoritmy, které následně třída `Main` za pomoci příslušných funkcí používá k zápisu do souborů.

5.3 Zhodnocení výsledků

Po naprogramování a řádném otestování celé aplikace jsem provedl zhodnocení výsledků generovaných oběma algoritmy. Tedy jaká je časová náročnost pro vygenerování školních rozvrhů, jak moc jsou rozvrhy kvalitní a zda by bylo možné je použít v praxi.

5.3.1 Testovací data pro zhodnocení výsledků

Aby bylo možné porovnat více výsledků, vytvořil jsem 3 kombinace vstupních dat, které se liší v počtu tříd a počtu učitelů. Každou kombinaci jsem pustil 5 krát každým algoritmem.

První kombinací je rozvrh tvořený 5 třídami (1A až 5A) představující první stupeň základní školy a 13 učitelů. Každý učitel má alespoň 2 specializace.

Druhá kombinace obsahuje taktéž 5 tříd (1A až 5A) jako první kombinace s tím rozdílem, že učitelů je jednou tolik (26). Bude tedy možné porovnat, zda aplikace vrací lepší výsledky při vyšším počtu učitelů nebo ne.

Třetí kombinací je školní rozvrh všech tříd základní školy (1A až 9A). Použijeme k vytváření 26 učitelů, kdy prvních 13 učitelů je stejných jako v případě 1. kombinace. Druhá polovina je obohacená o specializace předmětů druhého stupně. Např. Dějepis, Zeměpis a další.

V tabulce 5.2 jsou dvě tabulky s daty, na kterých bylo provedeno zhodnocení výsledků. Konkrétně tabulky znázorňují, kolik učitelů jaký předmět dle specializace může vyučovat. V případě 1. kombinace byla použita data z levé tabulky, která byla získána ze specializací 13 učitelů. V případě 2 kombinace byly použity zdvojnásobené hodnoty levé tabulky. Pravá tabulka obsahuje data získaná z 26 učitelů, která byla použita pro generování rozvrhů celé školy (pro třídy 1A až 9A).

Učitelé:

1 kombinace 13 učitelů
2 kombinace 26 učitelů
3 kombinace 26 učitelů

Třídy:

1A
Český jazyk psaní 5x, Český jazyk čtení 4x, Matematika 4x, Prvouka 2x, Tělesná výchova 2x, Hudební výchova 1x, Výtvarná výchova 1x, Pracovní

činnosti 1x

2A
Český jazyk psaní 5x, Český jazyk čtení 5x, Matematika 5x, Prvouka 2x, Tělesná výchova 2x, Hudební výchova 1x, Výtvarná výchova 1x, Pracovní činnosti 1x

3A
Český jazyk psaní 4x, Český jazyk čtení 4x, Matematika 5x, Prvouka 3x, Tělesná výchova 2x, Hudební výchova 1x, Výtvarná výchova 2x, Pracovní činnosti 1x, Anglický jazyk 3x

4A
Český jazyk psaní 4x, Český jazyk čtení 4x, Matematika 5x, Přírodověda 2x, Tělesná výchova 2x, Hudební výchova 1x, Výtvarná výchova x, Pracovní činnosti 1x, Anglický jazyk 3x, Vlastivěda 2x

5A
Český jazyk psaní 4x, Český jazyk čtení 4x, Matematika 5x, Přírodověda 2x, Tělesná výchova 2x, Hudební výchova 1x, Výtvarná výchova x, Pracovní činnosti 1x, Anglický jazyk 3x, Vlastivěda 2x

6A
Český jazyk 5x, Matematika 4, Anglický jazyk 3x, Fyzika 2x, Zeměpis 2x, Dějepis 2x, Přírodopis 2x, Výtvarná výchova 2x, Informatika 1x, Pracovní činnosti 1x, Tělesná výchova 2x, Hudební výchova 1x, Občanská výchova 1x, Výchova ke zdraví 1x

7A
Český jazyk 5x, Matematika 4, Anglický jazyk 3x, Fyzika 2x, Zeměpis 2x, Dějepis 2x, Přírodopis 2x, Výtvarná výchova 2x, Informatika 1x, Pracovní činnosti 1x, Tělesná výchova 2x, Hudební výchova 1x, Občanská výchova 1x, Výchova ke zdraví 1x

8A
Český jazyk 5x, Matematika 4, Anglický jazyk 3x, Fyzika 2x, Zeměpis 2x, Dějepis 2x, Přírodopis 2x, Výtvarná výchova 2x, Informatika 1x, Pracovní činnosti 1x, Tělesná výchova 2x, Hudební výchova 1x, Občanská výchova 1x, Výchova ke zdraví 1x, Chemie 2x

9A
Český jazyk 5x, Matematika 4, Anglický jazyk 3x, Fyzika 2x, Zeměpis 2x,

Dějepis 2x, Přírodopis 2x, Výtvarná výchova 2x, Informatika 1x, Pracovní činnosti 1x, Tělesná výchova 2x, Hudební výchova 1x, Občanská výchova 1x, Výchova ke zdraví 1x, Chemie 2x

1. stupeň	
předměty	počet možných učitelů
Matematika	5
Český jazyk psaní	4
Český jazyk čtení	3
Prvouka	3
Tělesná výchova	2
Vlastivěda	2
Hudební výchova	3
Přírodověda	1
Anglický jazyk	3
Pracovní činnosti	2
Výtvarná výchova	2

celá škola (9 tříd)	
předměty	počet možných učitelů
Matematika	9
Český jazyk psaní	4
Český jazyk čtení	3
Prvouka	3
Tělesná výchova	4
Vlastivěda	4
Hudební výchova	5
Přírodověda	1
Anglický jazyk	6
Pracovní činnosti	3
Výtvarná výchova	4
Český jazyk	5
Dějepis	1
Zeměpis	2
Přírodopis	2
Občanská výchova	3
Výchova ke zdraví	2
Informatika	2
Fyzika	2
Chemie	1

Tabulka 5.2: Tabulky možných učitelů, kteří mohou vyučovat dle specializace daný předmět.

5.3.2 Zhodnocení školních rozvrhů prvního stupně základní školy

Jak již bylo napsáno v sekci 5.3.1, pro generování byla použita dvojí data. Na jejich základě jsem vygeneroval školní rozvrhy oběma algoritmy.

Při generování genetickým algoritmem za použití 13 učitelů při maximálním počtu 500 generací, bylo ve všech případech nalezeno splnitelné řešení. Bohužel však rozvrhy nebyly příliš kvalitní. Obsahovaly více volných hodin během dne, což není úplně ideální. Celé generování trvalo kolem 5 minut. Přesné časy můžeme vidět v tabulce 5.3.

Při použití 26 učitelů ke generování byly výsledky mnohem přívětivější. Samotné vygenerované školní rozvrhy vypadaly lépe než za použití menšího počtu dat. Sice také obsahovaly několik volných míst v rozvrhu během dne, které by se však dokázali bez většího problému manuálně poupravit. Časová náročnost při generování byla podobná, jako v případě použití 13 učitelů pro generování.

Na obrázku 5.1 můžeme pro porovnání vidět dva vygenerované rozvrhy při generování rozvrhů pro první stupeň základní školy za pomoci 5 tříd. Horní rozvrh na obrázku byl vytvořen pomocí 13 učitelů a spodní za použití 26 učitelů.

Tabulka 5.3 obsahuje data při generování školních rozvrhů prvního stupně za pomoci genetického algoritmu. V tabulce jsou data získaná při generování za pomoci 13 učitelů a 26 učitelů. Sloupce *splnitelné řešení v generaci* označují generaci, ve které se objevil poprvé jedinec, který neporuše žádnou silnou podmínku (splnitelné řešení).

spuštění	první stupeň (13 učitelů)		první stupeň (26 učitelů)	
	splnitelné řešení v generaci	čas běhu	splnitelné řešení v generaci	čas běhu
1	99	5 min 9 s	72	5 min 17 s
2	111	5 min 3 s	95	5 min 17 s
3	59	5 min 10 s	51	5 min 16 s
4	83	5 min 6 s	104	5 min 15 s
5	186	5 min 7 s	60	5 min 18 s

Tabulka 5.3: Tabulka s daty získaných při testování generování školních rozvrhů pro první stupeň za pomoci genetického algoritmu.

Při generování školních rozvrhů algoritmem založeným na prohledávání a ukládání jen nekolizních položek algoritmus vygeneroval opravdu dobré výsledky. Při použití 13 učitelů jsem prakticky v každém generování obdržel kvalitní rozvrhy, které neobsahovaly žádné volné hodiny s výjimkou té, které náleží oběd. Čas generování trval kolem 1 s. Přesné časové údaje nalezneme v tabulce 5.4.

Při použití 26 učitelů už v žádném případě volné hodiny, s výjimkou pauzy na oběd, nebyly. Čas byl téměř totožný jako při použití menšího počtu učitelů.

Tabulka 5.4 obsahuje časové záznamy z generování školních rozvrhů pro první stupeň za použití 13 a 26 učitelů. Všechny časy jsou uvedeny v sekundách.

spuštění	první stupeň (13 učitelů)	první stupeň (26 učitelů)
1	1,01 s	1,29 s
2	1,01 s	1,29 s
3	1,02 s	1,27 s
4	1,04 s	1,36 s
5	1,03 s	1,26 s

Tabulka 5.4: Tabulka s časovými údaji získanými při testování generování školních rozvrhů pro první stupeň za pomoci algoritmu založeném na prohledávání a ukládání jen nekolizních položek.

Na obrázku 5.2 můžeme pro porovnání vidět dva rozvrhy, které jsme získali při generování rozvrhů prvního stupně (5 tříd) za pomoci algoritmu založeného na postupném prohledávání a ukládání jen nekolizních položek. Horní rozvrh byl získán při použití 13 učitelů. Spodní za požití 26 učitelů.

Třída: 5A

1	2	3	4	5	6	7	8
Čj č	Aj	Pří	Čj p	Tv	Čj p		
M	Hv	Aj	Čj č				Čj č
Tv	Aj	M	Pč			Pří	
Čj p	Vv	Vla	M			M	
Čj č	Čj p	Vla	M				

Třída: 5A

1	2	3	4	5	6	7	8
Čj č	Hv	Aj	Pří		M		
Aj	Čj p	Vla	Pč		Čj č	Vla	
Čj p	Tv	M				Čj p	
Aj	Čj č	M	Pří	M			
Vv	Čj č	Čj p	M	Tv			

Obrázek 5.1: Vygenerované školní rozvrhy za pomoci genetického algoritmu.

Třída: 5A

1	2	3	4	5	6	7	8
M	Hv	Čj p	Aj		Vla		
M	Čj p	Pč	Čj č		Aj		
M	Tv	Pří	Čj č		Čj p		
M	Tv	Pří	Čj č		Aj		
M	Čj p	Vv	Čj č		Vla		

Třída: 5A

1	2	3	4	5	6	7	8
M	Tv	Čj č	Čj p		Vla		
M	Tv	Čj č	Pří		Aj		
M	Hv	Čj p	Pč		Aj		
M	Čj č	Čj p	Pří		Aj		
M	Čj č	Čj p	Vv		Vla		

Obrázek 5.2: Vygenerované školní rozvrhy za pomoci algoritmu založeném na postupném prohledávání a ukládání jen nekolizních položek.

5.3.3 Zhodnocení školních rozvrhů všech tříd základní školy

Při generování rozvrhů celé základní školy bylo použito dle kapitoly 5.3.1 26 učitelů. Počet možných učitelů k jednotlivým předmětům je znázorněn v pravé tabulce 5.2.

Při použití genetického algoritmu, jak můžeme vidět v tabulce 5.5 v jednom případě z 5 se splnitelný rozvrh nepodařilo vygenerovat. Díky tomu, že se počet tříd zvětšil, zvětšila se i časová náročnost při generování. Celý běh trval více než 14 minut. Přesné časové hodnoty z testování nalezneme opět v tabulce 5.5. Ve všech případech, kdy se podařilo vygenerovat splnitelné rozvrhy, obsahovaly volné hodiny, což by bylo manuálně těžké opravit.

Samotná tabulka 5.5 obsahuje stejně jako v případě tabulky s daty získanými při generování rozvrhů pro první stupeň genetickým algoritmem sloupec s číslem generace, kdy populace obsahovala poprvé splnitelné zadání a čas běhu celého generování.

spuštění	9 tříd (26 učitelů)	
	splnitelné řešení v generaci	čas běhu
1	138	14 min 15 s
2	nepodařilo se	14 min 22 s
3	228	14 min 20 s
4	468	14 min 15 s
5	340	14 min 17 s

Tabulka 5.5: Tabulka s daty při generování školních rozvrhů celé školy (9 tříd) za pomoci genetického algoritmu.

Na obrázku 5.3 můžeme vidět rozvrh vygenerovaný genetickým algoritmem za použití 26 učitelů ke generování. Pro příklad byl vybrán rozvrh třídy, která obsahuje nejvíce předmětů. Jak můžeme vidět, obsahuje volné hodiny během dne, což není ideální.

Za pomoci stejných dat byly rozvrhy vygenerovány i algoritmem založeným na postupném prohledávání a ukládání jen nekolizních položek. Téměř ve všech případech algoritmus vygeneroval kvalitní rozvrhy které by bylo možné použit v praxi. V některých případech se sice stalo, že se v nějakém rozvrhu objevila volná hodina, která by se však poměrně lehce dala manuálně odstranit. Časová náročnost při generování byla podobná jako v případě generování rozvrhů pro 5 tříd (první stupeň). Doba trvání generování tedy byla opět kolem 1 s. Přesné časy nalezneme v tabulce 5.6.

Samotná tabulka 5.6 obsahuje časové záznamy při generování rozvrhů celé školy, tedy všech 9 tříd. Všechny časy jsou uvedeny v sekundách.

spuštění	9 tříd (26 učitelů)
1	1,41 s
2	1,45 s
3	1,44 s
4	1,38 s
5	1,42 s

Tabulka 5.6: Tabulka s daty při generování školních rozvrhů celé školy (9 tříd) za pomoci algoritmu založeného na prohledávání a ukládání jen nekolizních položek.

Na obrázku 5.4 je vidět vygenerovaný školní rozvrh za pomoci algoritmu založeného na postupném prohledávání a ukládání jen nekolizních položek. Jak je vidět, i rozvrh s největším počtem předmětů obsahuje pouze volnou hodinu určenou na oběd. V této formě by byl možný použít v praxi.

Třída: 9A

1	2	3	4	5	6	7	8
Aj	Z	Čj	M		Tv	Ch	
M	Aj	Pří	Čj		D		Vv
Inf	Pč	Vz	Vv	Čj	Fy		M
Aj	D	Ch	Pří		Tv		Fy
Hv	Čj	Z	Ov		M	Čj	

Obrázek 5.3: Rozvrh vygenerovaný genetickým algoritmem.

Třída: 9A

1	2	3	4	5	6	7	8
Čj	M	Tv	Pří		Vv	Ov	
Čj	M	Hv	Pří		Vz	D	Ch
Čj	M	Pč	Aj		Vv	D	Ch
Čj	Tv	Fy	Aj		Z		
M	Čj	Fy	Aj		Inf	Z	

Obrázek 5.4: Rozvrh vygenerovaný algoritmem založeným na postupném prohledávání a ukládání jen nekolizních položek.

5.3.4 Vzájemné porovnání obou algoritmů

Na základě připravených dat byly vygenerovány školní rozvrhy oběma algoritmy a následně vzájemně porovnány a to jak z hlediska kvality rozvrhů, tak časové náročnosti a možnosti zavedení vygenerovaných rozvrhů do praxe.

Ve všech případech dopadl lépe algoritmus založený na postupném prohledávání a ukládání jen nekolizních položek. Téměř ve všech případech a to jak při generování rozvrhů pro první stupeň základní školy, tak celou školu tvořenou z 9 tříd, dokázal na rozdíl od rozvrhů generovaných genetickým algoritmem vygenerovat kvalitní rozvrhy, které by bylo možné použít v praxi. Z hlediska časové náročnosti při generování dopadl mnohonásobně lépe než genetický algoritmu, aby generování u tohoto algoritmu trvalo vždy kolem 1 sekundy a u genetického na velkých datech dokonce více než 14 minut.

Pro užití do praxe bych tedy použil algoritmus, který je založený na postupném prohledávání jen nekolizních položek, který je schopen téměř v každém případě vygenerovat opravdu kvalitní rozvrhy a to za dobu kolem 1 sekundy.

Genetický algoritmus by pravděpodobně v praxi použitelný nebyl. V případě, že se použilo ke generování větší množství učitelů pro méně tříd, dokázal získat sice splnitelné rozvrhy, ale ne tak kvalitní. Musely by se rozvrhy manuálně poupravit. Při použití většího počtu tříd v některých případech nedokázal vygenerovat splnitelné řešení. Řešením tohoto problému by bylo rozdělit generování zvlášť pro první a druhý stupeň (rozdělit učitele a třídy), nebo zvýšit maximální počet generací, čímž se ale zvýší časová náročnost generování. I v případě nalezení splnitelného řešení pro 9 tříd rozvrhy nebyly příliš kvalitní, neboť obsahovaly velký počet volných hodin během dne a bylo by náročné je manuálně opravit. Stejně tak časová náročnost při generování jak na malých datech, tak velkých, byla vysoká. Při generování rozvrhů pro 9 tříd dokonce více než 14 minut.

6 Závěr

Nejprve jsem se v této práci věnoval analýze v sekci 2, kde jsem zjistil, jakými způsoby je možné školní rozvrhy generovat. Na základě získaných poznatků jsem se rozhodl, že se pokusím rozvrhy generovat pomocí genetického algoritmu a algoritmu, který pracuje na způsobu postupného prohledávání a ukládání jen nekolizních položek. Oba dva navržené algoritmy jsou důkladně popsány v sekci 3.

Dle návrhu se mi podařilo vytvořit aplikaci, která je schopna generovat školní rozvrhy malé školy. Celý program tvoří 17 tříd a 1 interface. Pro otestování bylo vytvořeno 17 testovacích tříd, které celkem obsahují 277 JUnit testů, kterými jsem aplikaci důkladně otestoval.

Po naprogramování funkční aplikace jsem oba dva algoritmy vzájemně porovnal a to jak na základě vygenerovaných rozvrhů, časové náročnosti při generování, tak možnosti použití v praxi.

Na základě ověření na několika typech dat jsem dospěl k názoru, že algoritmus založený na postupném prohledávání a ukládání jen nekolizních položek funguje efektivně a dokáže vygenerovat kvalitní školní rozvrhy pro základní školu, která má 9 tříd. Bylo by tedy možné použít algoritmus ke generování v praxi.

Naopak genetický algoritmus by příliš vhodný na generování školních rozvrhů pro základní školu, která má 9 tříd nebyl. Při vygenerování rozvrhů celé základní školy rozvrhy obsahovaly poměrně dost volných hodin. Musela by se tedy provádět dodatečná manuální úprava a ta by v případě vyššího počtu tříd byla pravděpodobně náročná. V některých případech se dokonce řešení nalézt nepodařilo. Řešením by bylo rozdělení generování zvlášť pro první stupeň a druhý stupeň, nebo zvýšit maximální počet generací. Stejně tak časová náročnost při generování je poměrně vysoká. V případě menších dat, jako je například generování rozvrhů pouze pro první stupeň, vrací relativně dobré výsledky, které bychom však museli také lehce manuálně poupravit. Pro uvedení do praxe tedy není příliš vhodný, proto bych se při opětovném řešení této problematiky rozhodl raději pro jiný algoritmus, například nějaký heuristický.

Podařilo se tedy vytvořit aplikaci, která je schopna generovat školní rozvrhy malé školy. Práce tedy splňuje zadání.

Literatura

[1] *Asc rozvrhy* [online]. c2019 aSc Applied Software Consultants. [cit. 2019-12-22]. Dostupné z: `https://www.asctimetables.com/`.

[2] *Bakaláři – mezi školou a rodinou | Bakaláři* [online]. BAKALÁŘI software, c2020. [cit. 2020-04-08]. Dostupné z: `https://www.bakalari.cz/Timetable`.

[3] *Genetický algoritmus* [online]. Vojtěch Hordějčuk, c2008-2019 Vojtěch Hordějčuk. [cit. 2019-12-22]. Dostupné z: `http://voho.eu/wiki/geneticky-algoritmus/`.

[4] *Statistika a pravděpodobnost | Přírodovědecká fakulta Masarykovy univerzity* [online]. Masarykova univerzita, 2016. [cit. 2020-04-03]. Dostupné z: `https://is.muni.cz/do/rect/el/estud/prif/ps15/statistika/web/pages/rovnomerne-diskretni-rozd.html`.

[5] *Škola OnLine - nejrozšířenější webový školní informační systém* [online]. ŠKOLA ONLINE, c2020. [cit. 2020-04-08]. Dostupné z: `https://www.skolaonline.cz/Skolni_informacni_system.aspx`.

[6] Cigler, L. Analýza a implementace algoritmů prosestavování středoškolských rozvrhů. Bakalářská práce, Univerzita Karlova v Praze, Matematicko-fyzikální fakulta, Katedra aplikované matematiky, Praha, 2006.

[7] Cormen, T. H. et al. *Introduction to Algorithms*. Mass.: MIT Press, 3rd edition edition, c2009. ISBN 9780262033848.

[8] Dechter, R. *Constraint processing*. Morgan Kaufmann Publishers, c2003. ISBN 978-1-55860-890-0.

[9] Fiala, J. GENEROVANÁNÍ ŠKOLNÍCH ROZVRHŮ. Diplomová práce, Vysoké učení technické v Brně, Fakulta informačních technologií, Brno, 2012.

[10] Rozenberg, D. Aplikace pro tvorbu rozvrhu. Diplomová práce, České vysoké učení technické v Praze, Fakulta elektrotechnická, Katedra počítačové grafiky a interakce, Praha, 2014.

[11] Tesařová, P. Aplikace pro vytváření školních rozvrhů. Diplomová práce, VŠB - Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky, Katedra informatiky, Ostrava, 2017. Dostupné z:

https://dspace.vsb.cz/bitstream/handle/10084/119161/TES0038_FEI_N2647_2612T025_2017.pdf?sequence=1&isAllowed=n.

[12] Vlček, L. Interaktivní generátor rozvrhu. Diplomová práce, Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Katedra informatiky a výpočetní techniky, Plzeň, 2013.

[13] Švadlenka, J. Informační systém pro školy s automatickou tvorbou rozvrhů. Diplomová práce, Vysoké učení technické v Brně, Fakulta informačních technologií, Brno, 2008. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=116324.

A Příloha

A.1 Uživatelská příručka

V této části bude popsána uživatelská příručka pro správné ovládání celého programu. Tedy možnost použití předpřipravených vstupních souborů, ukázka spuštění aplikace v příkazové řádce a dokumentace programu.

A.1.1 Vstupní soubory

Aby uživatel nemusel vytvářet všechny vstupní soubory, bylo připraveno pár souborů, které si uživatel dle vlastního uvážení může poupravit k vlastnímu použití. Připravené soubory jsou uložené ve složkách `dataCSV` a `dataXML`.

A.1.2 Spuštění programu

Program se spouští z příkazové řádky. Pro spuštění byl vytvořen spustitelný soubor s názvem `generatorSkolnichRozvrhu.jar`.

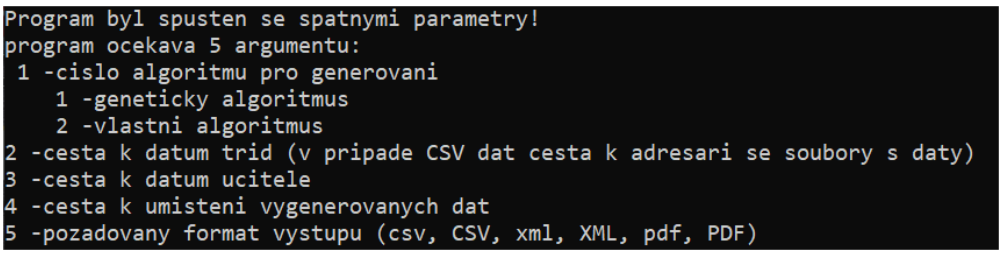
Otevřeme si příkazovou řádku s umístěním spustitelného souboru. Následně můžeme zadat příkaz pro spuštění, který vypadá následovně.

```
java -jar generatorSkolnichRozvrhu.jar parametry programu
```

Aby program fungoval správně, je třeba zadat 5 parametrů v pořadí číslo algoritmu, cesta k datům tříd, cesta k datům učitelů, cesta k adresáři, do kterého se uloží vygenerované rozvrhy a označení formátu souborů vygenerovaných rozvrhů. Pokud jsou parametry zadány špatně, program vyhodí výjimku a vypíše informace o správném spuštění programu.

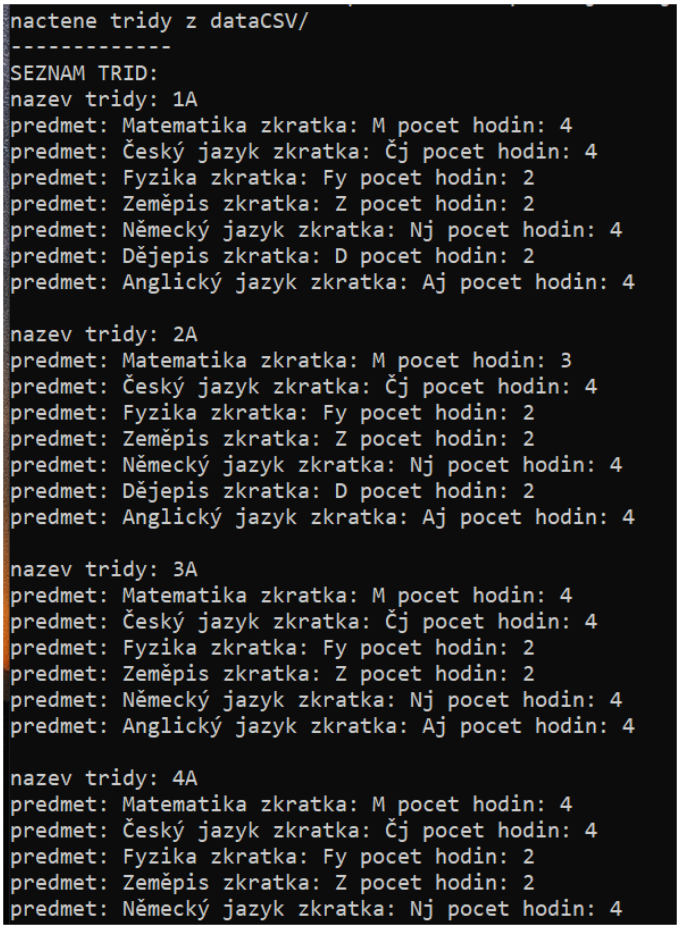
```
java -jar generatorSkolnichRozvrhu.jar 2 dataCSV/ dataCSV/Ucitele.csv rozvrhyPDF pdf
```

Obrázek A.1: Správné spuštění programu v příkazové řádce.

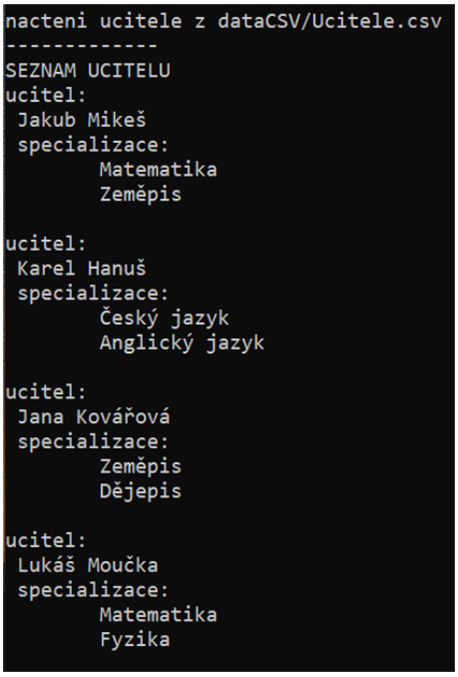


Obrázek A.2: Vypsaná nápověda v případě špatného spuštění.

Pokud se program podaří spustit, nejdříve aplikace vypíše seznam načtených tříd (obrázek A.3) a učitelů (obrázek A.4). Následně vypíše vybraný algoritmus a spustí se samotné generování.

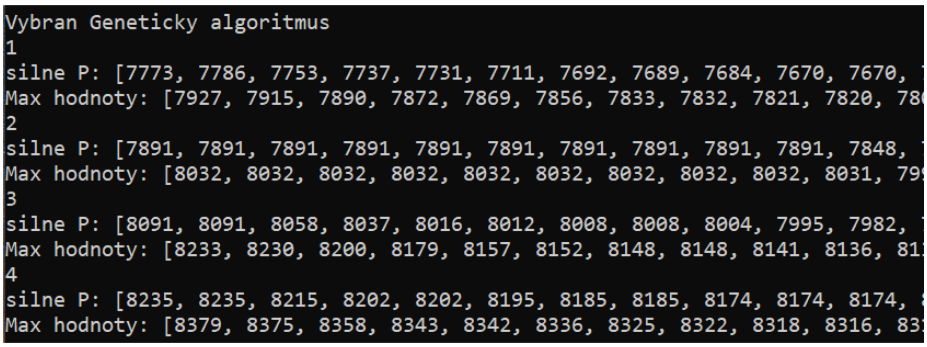


Obrázek A.3: Výpis načtených tříd.



Obrázek A.4: Část výpisu načtených učitelů.

Při zvolení generování pomocí genetického algoritmu aplikace postupně vypisuje nejsilnější jedince v generaci (obrázek A.5). První seznam ukazuje, kolik je splněných silných podmínek. Pokud se v poli objeví hodnota 10000, znamená to, že jedinec obsahuje splnitelné řešení. Druhé pole znázorňuje fitness hodnotu jedinců. V obou polích tedy můžeme sledovat, jak je jedinci postupně zlepšují. Tento výpis byl zvolen záměrně, neboť samotné generování trvá poměrně dlouho. Uživatel tedy může kontrolovat průběh.



Obrázek A.5: Výpis nejsilnějších jedinců jednotlivých generací při průběhu generování genetickým algoritmem.

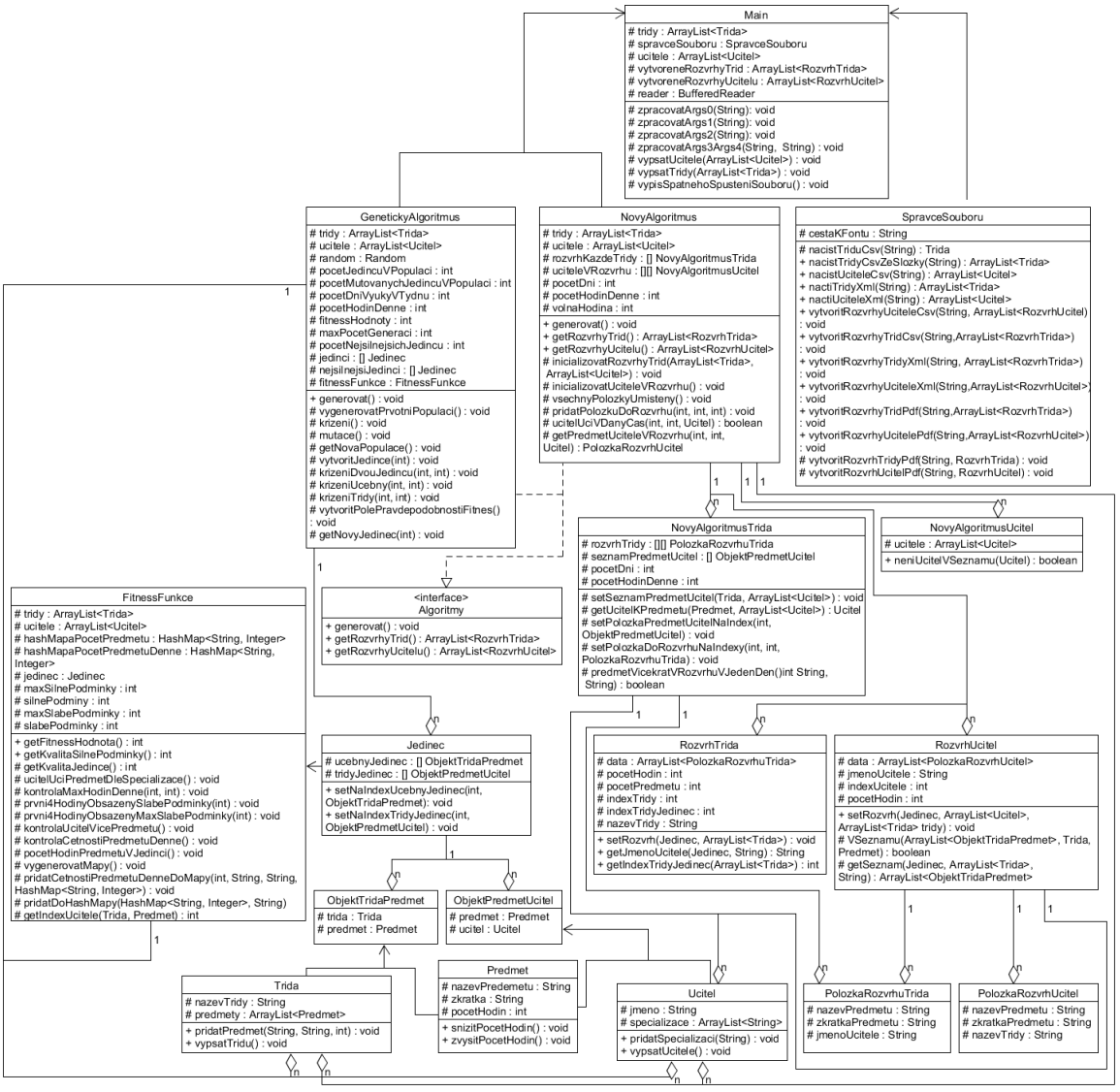
A.1.3 Vygenerované soubory

Vygenerované soubory jsou uloženy v nově vytvořeném adresáři, který byl vytvořen na základě čtvrtého parametru zadaného při spuštění programu.

```
soubor 1A.pdf uspesne vytvoren
soubor 2A.pdf uspesne vytvoren
soubor 3A.pdf uspesne vytvoren
soubor 4A.pdf uspesne vytvoren
soubor Jakub Mikeš.pdf uspesne vytvoren
soubor Karel Hanuš.pdf uspesne vytvoren
soubor Jana Kovářová.pdf uspesne vytvoren
soubor Lukáš Moučka.pdf uspesne vytvoren
soubor Jan Lukeš.pdf uspesne vytvoren
soubor Roman Novák.pdf uspesne vytvoren
soubor Zdeněk Kovář.pdf uspesne vytvoren
soubor Ivan Křešnička.pdf uspesne vytvoren
soubor František Křešnička.pdf uspesne vytvoren
soubor Roman Hájek.pdf uspesne vytvoren
soubor Jan Novák.pdf uspesne vytvoren
soubor Kateřina Mladá.pdf uspesne vytvoren
```

Obrázek A.6: Výpis vygenerovaných souborů s rozvrhy.

B Příloha



Obrázek B.1: Diagram tříd obsahující hlavní metody a atributy.