UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A
INFORMATIKY

Evolučné algoritmy pre tvorbu rozvrhov

Diplomová práca

2013 Bc. Danica Zajacová

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A
INFORMATIKY

Evolučné algoritmy pre tvorbu rozvrhov

Diplomová práca

Študijný program:	Informatika
Študijný odbor:	2508 Informatika
Školiace pracovisko:	Katedra Informatiky FMFI
Školiteľ:	Mgr. Pavel Petrovič, PhD.

2013 Bc. Danica Zajacová

Abstrakt

V rámci výskumného projektu bol na Katedre aplikovanej informatiky vyvinutý kľúčový komponent systému RoGeR - automatického generátora rozvrhov, ktorý je postavený na evolučných algoritmoch. Cieľom práce je preskúmať možné vylepšenia prehľadávacieho algoritmu kombináciou s heuristikami, deterministickými prístupmi, inými metaheuristikami, paralelizáciou, alebo interaktívnym prehľadávaním.

Kľúčové slová: tvorba rozvrhov, evolučné algoritmy

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Danica Zajacová
Študijný program: informatika (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: 9.2.1. informatika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský

Názov: Evolučné algoritmy pre tvorbu rozvrhov

Cieľ: V rámci výskumného projektu bol na Katedre aplikovanej informatiky vyvinutý kľúčový komponent systému RoGeR - automatického generátora rozvrhov, ktorý je postavený na evolučných algoritmoch. Cieľom práce je preskúmať možné vylepšenia prehľadávacieho algoritmu kombináciou s heuristikami, deterministickými prístupmi, inými metaheuristikami, paralelizáciou, alebo interaktívnym prehľadávaním. Vyžaduje sa aktívny záujem študenta zapojiť sa do výskumného projektu.

Literatúra: E.C. Eiben, J. E. Smith: Introduction to Evolutionary Computing, Springer 2007.

Vedúci: Mgr. Pavel Petrovič, PhD.

Dátum zadania: 17.10.2011

Dátum schválenia: 02.11.2011

prof. RNDr. Branislav Rovan, PhD.
garant študijného programu

.....
študent

.....
vedúci

Čestne prehlasujem, že som túto diplomovú prácu vypracovala samostatne s použitím citovaných zdrojov.

.....

Poďakovanie

Touto cestou by som sa chcela poďakovať môjmu školiteľovi Mgr. Pavlovi Petrovičovi, PhD za pomoc pri tvorbe tejto práce.

Obsah

Úvod

Táto práca vznikla na základe požiadavky jednej firmy, ktorá sa zaoberá tvorbou a správou informačného systému pre univerzity najmä v Čechách a na Slovensku. Táto firma by chcela univerzitám využívajúcim jej systém poskytnúť nástroj, ktorý by im pomohol s tvorbou rozvrhov - najlepšie vygeneroval rozvrh pre univerzitu. Vygenerovať použiteľný rozvrh je NP-ťažký kombinatorický optimalizačný problém, a preto sa na generovanie zvyknú používať heuristiky a metaheuristiky, ktoré sú schopné vygenerovať rozumne kvalitný rozvrh v rozumnom čase. Podľa požiadaviek firmy bol už vyvinutý prototyp generátora rozvrhov, ktorý generuje rozvrhy využitím evolučných algoritmov.

Cieľom tejto práce je preskúmať možné vylepšenia generátora. Vylepšenia budeme hľadať najmä v oblasti evolučných algoritmov.

Prvá kapitola popisuje zadananie práce - definuje problém, ktorý riešime. Druhá kapitola popisuje použité metódy - najmä evolučné algoritmy. Tretia sa zaoberá všeobecným prehľadom - ako generujú univerzitné rozvrhy inde vo svete a zameriava sa na práce využívajúce evolučné algoritmy. Štvrtá popisuje pôvodný generátor, ktorý chceme vylepšiť a piata vlastný výskum.

Kapitola 1

Definícia problému

Ako bolo spomenuté v úvode, táto práca vznikla na základe požiadavky jednej firmy zaoberajúcej sa tvorbou a správou univerzitného informačného systému nasadeného okrem iného na univerzitách v Čechách a na Slovensku. Firma by chcela pre univerzity vytvoriť program, ktorý by im pomohol s tvorbou rozvrhov. Snaha je vytvoriť generátor, ktorý by vedel generovať rozvrhy pre *rôzne* univerzity. Cieľom nie je vytvoriť generátor špecializovaný pre jednu univerzitu s jej špecifickými potrebami, ale generátor použiteľný pre generovanie rozvrhov na viacerých univerzitách s rôznymi požiadavkami a potrebami. *Vstupné údaje pre generátor - údaje, na základe ktorých má generátor vytvárať rozvrhy sú dané dátovým modelom existujúceho akademického systému, na ktorý sa má tvorba rozvrhov naviazať. Podobne sú dané aj požiadavky na rozvrhy, ktoré akademický informačný systém dovoľuje zadať a očakávaný formát výstupu.*

Z hľadiska charakterizácie problému sa *táto práca zaoberá tvorbou kurzových univerzitných rozvrhov*[1]. Rozvrh sa tvorí na obdobie jedného týždňa, ktorý sa opakuje počas celého semestra. Týždeň pozostáva zo 7 dní tvorených určitým počtom časových intervalov stanovených dĺžok označovaných ako vyučovacie hodiny. Výučba sa nasadzuje do týchto intervalov - vyučovacích hodín. Univerzita môže učiť na viacerých miestach, pričom nezávisle od miest výuky sa delí na fakulty. Fakulty ponúkajú pre študentov kurzy, ktoré sa skladajú z jednej alebo viac rozvrhových udalostí, zjednodušene nazývaných lekcie. Lekcie môžu byť rôznych typov(napr. prednáška a cvičenie) a môžu pozostávať z viacerých častí. Výučba časti lekcie môže trvať dlhšie ako jednu vyučovaciu hodinu. Časť lekcie môže byť súčasťou viacerých lekcií rôznych kurzov. Časti lekcie rovnakej lekcie sa nesmú vyučovať naraz. S časťami lekcií sa narába ako so samostatnými lekciami. Ak ďalej povieme,

[1] Pre viac informácií ohľadom druhov univerzitných rozvrhov pozrite časť 3.1

že lekcia trvá 2 vyučovacie hodiny, znamená to, že sa skladá z jednej časti trvajúcej 2 vyučovacie hodiny. Lekcie(všetky jej časti) sa nemusia vyučovať každý, ale aj iba každý 2., 3. alebo i-ty týždeň. Jednu lekciu môže učiť viac ako jeden učiteľ a lekcia pre svoju výučbu môže požadovať naraz viacero miestností. Lekcia sa nasadzuje do skupiny miestností, kde skupina miestností môže pozostávať z jednej a viac miestností. Jedna miestnosť môže byť súčasťou viacerých skupín miestností. Každá miestnosť sa nachádza na nejakom mieste výučby a spravuje ju nejaká fakulta. Miestnosť má definovanú svoju kapacitu miest na sedenie a môže obsahovať rôzne zariadenia potrebné pre výuku. Lekcie od miestností, kde majú byť nasadené, požadujú aby mali dostatočnú kapacitu voľných miest na sedenie. Lekcie môžu tiež požadovať, aby miestností obsahovali určité zariadenie alebo si rovno určiť miestnosti do ktorých môže/má byť nasadená. Miestnosti môžu byť v určitých vyučovacích hodinách vyhradené pre konkrétnu fakultu alebo aj niekoho iného. Lekcia nemusí mať presne dané, kto ju má učiť. Pre jeden konkrétny typ lekcií konkrétneho kurzu je definovaná jedna alebo viac skupín učiteľov, ktoré môžu učiť iba lekcie daného typu pre daný kurz. Skupina učiteľov pozostáva z jedného alebo viac učiteľov a jeden učiteľ môže byť súčasťou viacerých skupín učiteľov. Generátor má na starosti lekcii priradiť skupinu učiteľov, ktorí ju majú vyučovať. Ak skupina učiteľov obsahuje viac učiteľov, vníma sa to tak, že učitelia učia lekciu naraz. Pre skupinu učiteľov možno definovať obmedzenia určujúce presný počet lekcií, ktoré majú učiť a maximálny počet lekcií, ktoré môžu vyučovať. Niektorí učitelia nemôžu alebo nechcú učiť v niektoré vyučovacie hodiny. Pre jednotlivých učiteľov preto možno nastaviť, v ktoré vyučovacie hodiny je pre nich viac/menej vhodné alebo nemožné učiť.

Cieľom generátora je vytvoriť rozvrh - priradenie, ktorý každej lekcii priradí vyučovacie hodiny, kedy sa má učiť, miestnosti, kde sa má v dané vyučovacie hodiny učiť a učiteľov, ktorí ju majú vyučovať. Rozvrhom je každé jedno možné priradenie. Na rozvrh sú samozrejme kladené obmedzenia ako: žiaden učiteľ nemá učiť na dvoch miestach naraz, v žiadnej miestnosti sa nemajú naraz učiť dve rôzne lekcie alebo, že skupina učiteľov neučí viac lekcií než je povolené maximum. Nedodržanie jednotlivých obmedzení má rozličnú závažnosť. Viac o obmedzeniach kladených na rozvrhy nájdete v časti 1.2.

Vo vstupných údajoch pre tvorbu rozvrhov sú tiež dvojice lekcií, pre ktoré nie je žiadúce aby boli vyučované v rovnakú vyučovaciu hodinu. Bývajú to spravidla lekcie, ktoré zdieľajú rovnakú miestnosť, učia ich rovnakí učiteľa alebo si obe v predzápise zapísali niektorí študenti. Podľa prístupu k študentom existujú dva typy rozvrhov: po-zápisové a kurikulárne (cieľovoprogramové). Po-zápisový rozvrh vniká tak, že študenti sa najprv zapíšu na kurzy a potom podľa zápisov študentov sa vytvára rozvrh, ktorý sa snaží aby žiaden študent nemusel naraz navštevovať lekcie dvoch rôznych kurzov. Kurikulárny

vzniká pred zápisom študentov na kurzy a vytvára sa na základe študijných programov a študijných skupín navštevujúcich lekcie kurzov. *Naša práca sa zaoberá tvorbou kurikulárnych rozvrhov, ale berie do úvahy aj dobrovoľný predzápis študentov na jednotlivé predmety.* Kvôli vysokej flexibilnosti kreditového štúdia vo všeobecnosti nie je možné vyhovieť všetkým študentom a vždy sa nájdu jednotlivci, ktorí zvolia predmety tak, že niektoré budú kolidovať. Preto sa systém nezaoberá jednotlivými študentami a ich výberom predmetov, ale iba celkovými počtami študentov a kolízií medzi jednotlivými predmetmi. Teoreticky medzi po-zápisovou a kurikulárnou tvorbou rozvrhov nie je žiaden rozdiel, lebo zo študentov zapisujúcich si kurzy sa dajú vytvoriť študijné skupiny navštevujúce kurzy. Prakticky sa to aj tak zvykne robiť. V našom probléme sú dvojice lekcií, ktoré nemajú byť nasadené naraz do rozvrhu určované aj na základe študijných skupín, odborov a ročníkov alebo zadefinovaných dvojíc kurzov s neprázdnym prienikom študentov. Sú dvojice lekcií, ktoré nemajú byť nasadené naraz, u ktorých je viac alebo menej závažné ak sú nasadené naraz v rozvrhu. Napr. ak sa študentom prekrývajú dva povinné predmety je to závažnejšie ako keď sa im kryje povinný s výberovým. Jednotlivé kurzy sa môžu vyučovať na viacerých miestach výuky, pričom vstupné údaje môžu obsahovať pre každé miesto výuky povinnosť - či sa kurz na danom mieste učí ako povinný, povinne voliteľný alebo výberový predmet. Vstupné údaje pre kurz môžu obsahovať aj počty študentov z jednotlivých miest, ktoré navštevujú kurz s danou povinnosťou. Na lekcie kurzov môžu byť zapísané študijné odbory s príslušnými ročníkmi alebo(aj) študijné skupiny. Študijná skupina predstavuje študentov nejakého jedného študijného odboru v konkrétnom ročníku, ktorá je zapísaná na jeden a viac kurzov v nejakom mieste výuky. Nie je žiadúce aby v rozvrhu boli nasadené naraz rôzne lekcie s rovnakou študijnou skupinou alebo odborom. Univerzita môže zadať aj dvojice kurzov, ktoré majú neprázdny prienik študentov a ktorých lekcie nechce aby boli nasadzované naraz do rozvrhu. Hovoríme, že dve lekcie sú v kolízií, ak majú neprázdny prienik študentov, miestností alebo učiteľov. *Univerzity majú voľnosť v zadávaní vstupných údajov. Vstupné údaje môžu a nemusia obsahovať študijné skupiny, odbory, ročníky a kolízie kurzov.*

Ďalšie časti detailnejšie popisujú vstupné údaje, obmedzenia kladené na rozvrhy, pre ktoré nie je žiadúce aby boli porušované a údaje tvoriace výstup generátora.

1.1 Vstupné údaje

Táto časť popisuje štruktúru vstupných údajov, z ktorých sa generuje rozvrh. Vstupné údaje získava generátor vo forme údajových objektov v pamäti alebo

XML súboru. Ako príklady vstupných dát môžu poslúžiť vstupné testovacie súbory pre generátor - anonimizované údaje z 5 univerzít, ktoré nájdete na priloženom CD. V ďalšom si popíšeme a vysvetlíme časti vstupných údajov.

Miesta výuky

Vyučovanie môže prebiehať na niekoľkých miestach, kde vrámci jedného miesta môže mať výučbu vo viacerých areáloch. Miesta predstavujú väčšinou mestá ako Bratislava, Trnava vrámci ktorých má univerzita areály v rôznych mestských častiach. S miestami výuky sa dodáva aj matica vzdialeností určujúca vzdialenosť medzi dvojicami miest. Podobne sa dodáva aj matica vzdialeností areálov vrámci jedného miesta. Vzdialenosť je meraná počtom minút, ktoré treba na presun medzi dvojicami miest alebo areálov. Ak vzdialenosť medzi miestami nie je určená počíta sa, že miesta sú vzdialené 12 hodín a podobne areály 2 hodiny.

Vyučovacie hodiny

Rozvrh sa tvorí na obdobie jedného týždňa, ktorý sa opakuje počas celého semestra. Týždeň pozostáva zo 7 dní tvorených rovnakým počtom rovnako dlho trvajúcich vyučovacích hodín. Výučba sa nasadzuje do týchto vyučovacích hodín. Univerzita si definuje počet vyučovacích hodín v týždni, trvanie a ich začiatky vyjadrené v minútach dňa. Hodiny začínajú každý deň v rovnakom čase. Okrem toho si univerzita pre každú vyučovaciu hodinu definuje vhodnosť - ako je vhodné pre danú vyučovaciu hodinu nasadzovať lekcie do rozvrhu. Napríklad je možné stanoviť, že vyučovanie v pondelok ráno a v piatok poobede je menej vhodné, keďže študenti odchádzajúci na víkend domov môžu čas vyhradiť na cestovanie. Rovnako je možné zadať, že v pondelok sa na 8. a 9. hodinu nesmie zaradiť žiadna výuka, lebo je to čas vyhradený pre schôdze. Vhodnosť sa vyjadruje prirodzeným číslom v intervale od -1 po 100, kde -1 znamená zákaz výučby v danú hodinu a 100 predstavuje najvhodnejšiu dobu pre výučbu. Údaje o vyučovacích hodinách sa vo vstupných údajoch definujú na mieste zvanom všeobecná dostupnosť.

Fakulty, miestnosti

Univerzita pozostáva z fakúlt. Fakulta vyučuje kurzy a spravuje miestnosti. Každá miestnosť sa vyskytuje v práve jednom areáli a spravuje ju práve jedna fakulta. Okrem areálov a fakulty má každá miestnosť definovanú kapacitu - koľko študentov sa do nej zmestí, a zoznam zariadení s ich počtami, ktoré

obsahuje. Miestnosť môže byť počas niektorých vyučovacích hodín rezervovaná nejakou fakultou alebo niekym iným externe. Každá miestnosť preto ešte obsahuje zoznam rezervácií, ktorý ku každej vyučovacej hodine priradí 0 ak miestnosť nie je rezervovaná, identifikátor fakulty, pre ktorú je miestnosť rezervovaná alebo -1 pre externú rezerváciu.

Študijné odbory a skupiny

Vstupné údaje obsahujú aj zoznamy študijných odborov, skupín študentov a kurzov. Univerzita poskytuje študentom na štúdium študijné odbory. Študijný odbor môže byť aj pod-odborom iného študijného odboru. Napríklad univerzita môže ponúkať študijný odbor matematika a vrámci študijného odboru matematika ponúkať študijný odbor matematická analýza a študijný odbor štatistika. Študenti odboru štatistika sú aj študenti odboru matematika.

Skupina študentov má definovaný študijný odbor, pod ktorý patrí, ročník, v akom sa jej študenti nachádzajú a zoznam kurzov s miestami výuky, na ktorých chcú dané kurzy navštevovať. Ako príklad môže byť skupina študentov študijného odboru matematika, ktorá má zapísané kurzy algebra v Bratislave, analýza v Nitre, programovanie v Bratislave.

Typy lekcií

Lekcie v rozvrhu môžu byť rôznych typov. Napríklad prednáška alebo cvičenie. Pre každý typ lekcie sú definované tieto údaje: id, meno, koeficient kapacity a priorita. Id predstavuje jednoznačný identifikátor typu a meno predstavuje názov typu lekcie. K lekciám môže byť stanovená požiadavka na kapacitu voľných miest na sedenie. Koeficient kapacity určuje pre daný typ lekcií akú časť požadovanej kapacity musí miestnosť spĺňať. Hodnota 0,7 určuje, že ak lekcia daného typu požaduje 10 voľných miest, miestnosť ich musí mať minimálne 7. S kapacitou súvisia obmedzenia kapacita v limite a kapacita mimo limitu - pozrite časť s obmedzeniami.

Kurzy, ich požiadavky a skupiny miestností

Kurz pozostáva z jednoznačného identifikátoru, názvu, fakulty, ktorá ho vedie garanta a lekcií. Garant predstavuje učiteľa, ktorý má učiť tie lekcie kurzu, ku ktorým generátor nenájde skupinu učiteľov, čo by ju učila. Kurz ďalej pozostáva z miest výuky, kde sa má učiť a úrovní povinností s akými sa na jednotlivých miestach má učiť. Úrovne povinnosti kurzu sú: 1-povinný,

2-povinne voliteľný, 3-výberový. Kurz môže obsahovať aj pre každú dvojicu miesto výuky a povinnosť počty študentov, ktorí majú kurz zapísaný v mieste s povinnosťou. Pre typy lekcií môže mať kurz definované, koľko lekcií toho typu sa môže maximálne učiť v jeden deň a či lekcie daného typu môžu prebiehať paralelne alebo nie. Ak má typ lekcie pre maximálny počet lekcií na deň zadané -1 alebo 0, nie je určený limit pre počet lekcií, čo sa môžu učiť v jeden deň. Ak nie je určené inak, lekcie rovnakého typu nemôžu byť nasadzované paralelne. Pre jednotlivé typy lekcií môže kurz obsahovať aj zoznam s počtami zariadení, ktoré lekcie daného typu vyžadujú od miestností, kde majú byť učené. Každá lekcia kurzu má definovaný typ lekcie, kapacitu voľných miest, ktorú požaduje od miestností, kde má byť nasadená a periodicitu (spacing) s akou má byť vyučovaná - má byť učená len každý i-ty týždeň. Ak údaj o periodicite obsahuje 0, 1 alebo -1, lekcia sa má učiť každý týždeň. Na lekciu sa môžu zapísať študijné skupiny a odbory, ktoré budú lekciu navštevovať. Jednotlivé odbory zapíšu ročník v akom študenti odboru budú lekciu navštevovať alebo ako ročník uvedú -1 - vtedy lekciu môže navštevovať celý odbor. Lekcia pozostáva z jednej alebo viac častí. Každá časť má jednoznačný identifikátor a dĺžku predstavujúcu, koľko za sebou idúcich vyučovacích hodín sa má vyučovať. Časť lekcie môže byť súčasťou viacerých lekcií viacerých kurzov. Časti lekcie nemôžu byť vyučované paralelne. Časti lekcií sa pre jednoduchosť vnímajú ako lekcie lebo väčšinou lekcia pozostáva len z jednej časti.

Zvyčajne sa pri tvorbe rozvrhu pridelí jednej lekcii jedna miestnosť s požadovanou kapacitou a vybavením. Existujú však lekcie, ktoré na výučbu potrebujú viac ako jednu miestnosť. Rovnako existujú lekcie, ktoré na svoju výučbu potrebujú špeciálnu miestnosť z definovanej množiny miestností, alebo množinu miestností z definovanej množiny množín miestností. Napríklad lekcia laboratórne cvičenia z chémie môže potrebovať 2 z 5 laboratórií. Kvôli podobným prípadom sa zaviedol pojem skupina miestností. K jednotlivým lekciám sa neprideľujú miestnosti, ale vždy nejaká skupina miestností. Skupina miestností má v sebe definované miestnosti, ktoré obsahuje, kurz a typ lekcie pre aký je zamýšľaná. Skupina miestností môže pozostávať aj len z jednej miestnosti a nemusí obsahovať kurz a typ lekcie - je využiteľná pre všetky lekcie. Skupina miestností s definovaným kurzom a typom lekcie je určená len pre lekcie daného typu a kurzu. Ak má kurz v sebe definované, že mu môžu byť pridelené len miestnosti zo zoznamu miestností pre daný kurz a typ lekcie, znamená to, že lekciám z kurzu toho typu môže byť pridelená jedna zo skupín miestností z definovaným rovnakým kurzom a typom lekcie. Skupina miestností v sebe navyše obsahuje atribút sumovaná kapacita, ktorý určuje, či ako kapacita skupiny miestností sa má vziať suma kapacít všetkých jej miestností alebo minimum z ich kapacít.

Kolízie kurzov

Sú kurzy, ktoré majú neprázdny prienik študentov. O dvojici kurzov s neprázdnym prienikom študentov hovoríme, že sú v kolízií a pri tvorbe rozvrhu je snaha, aby lekcie daných kurzov neboli vyučované v rovnakom čase. Vstupné údaje môžu obsahovať zoznam dvojíc kurzov v kolízií. Pre každú kolíziu si pamätá kurzy, ktoré sú v kolízií, miesta výuky študentov kurzov, povinnosť s akou majú študenti zapísané jednotlivé kurzy a počet študentov, ktorý majú zapísané oba kurzy.

učitelia, požiadavky učiteľov a skupiny učiteľov

Každý učiteľ má okrem svojho identifikátora a mena aj požiadavky kedy môže/nemôže/chce/nechce učiť.

Podobne ako generátor pozná vhodnosť pridelenia lekcií do jednotlivých vyučovacích hodín, aj pre každého učiteľa je pre každú vyučovaciu hodinu na vstupe definovaná vhodnosť, ako je vhodné, aby učiteľ učil v danom čase. Rovnako ako pri všeobecnej dostupnosti vhodnosť je vyjadrená prirodzeným číslom v intervale od -1 po 100, kde -1 zakazuje nasadiť učiteľa na danú hodinu. Ak pre nejakú hodinu nemá učiteľ určenú preferenciu s akou chce učiť, jeho vhodnosť na danú hodinu sa nastaví na 0.

Podobne ako ku lekciám sa priraďujú skupiny miestností, priraďujú sa aj skupiny učiteľov. Skupina učiteľov pozostáva zo zoznamu učiteľov, kurzu a typu lekcie aký smie učiť. Skupina učiteľov môže pozostávať aj len z jedného učiteľa. Lekciu môže vyučovať iba skupina učiteľov s rovnakým kurzom a typom lekcie aký má skupina učiteľov. Skupina učiteľov v sebe môže zahŕňať aj 2 parametre: počet lekcií a maximálny počet lekcií. Ak je definovaný prvý parameter, daná skupina učiteľov by mala byť pridelená k presnému počtu lekcií uvedeného typu a kurzu. Maximálny počet lekcií zakazuje prideliť učiteľom viac ako maximálny počet lekcií uvedeného kurzu a typu.

Fixovanané hodiny, učitelia, miestnosti, periodicita

Niektoré univerzity chcú aby niektoré lekcie kurzov mali napevno pridelenú skupinu miestností alebo vyučovaciu hodinu, kedy majú začať alebo skupinu učiteľov, čo má lekciu učiť alebo skupinu miestností alebo offset určujúci, koľkatý týždeň sa má lekcia začať učiť ak má periodicitu väčšiu ako 1. Ak má lekcia periodicitu 2 a offset 0, znamená že sa má učiť od prvého týždňa a potom každý nepárny týždeň. Ak má offset 1, tak sa má začať učiť druhý týždeň a potom každý párny. Vstupné údaje obsahujú tieto pevné pridelenia.

1.2 Obmedzenia kladené na rozvrh

Rozvrh je jedno konkrétne priradenie, ktoré pridelí ku každej definovanej lekcií skupinu učiteľov, skupinu miestností a množinu za sebou v rovnaký deň nasledujúcich vyučovacích hodín, ktorých počet je rovný dĺžke trvania lekcie. Rozvrhom je akékoľvek priradenie. Môže to byť rozvrh použiteľný v praxi, ale aj rozvrh, ktorý všetky lekcie nasadí na prvú pondelňajšiu hodinu. Na generované rozvrhy sú kladené obmedzenia, ktorých dodržanie/nedodržanie určujú kvalitu rozvrhu. Naša práca podľa závažnosti delí obmedzenia na obmedzenia s najvyššou, vysokou, strednou a nízkou prioritou. Porušenie obmedzenia s vyššou prioritou je závažnejšie ako porušenie obmedzenia s vysokou prioritou a podobne pre nižšie priority. Rozvrh A je kvalitnejší od rozvrhu B ak A obsahuje menej porušení obmedzení najvyššej priority ako B. Ak ich majú rovnako veľa, tak A je kvalitnejší ak porušuje menej obmedzení s vysokou prioritou ako B a tak ďalej. Úlohou generátora je nájsť a zostrojiť rozvrh čo najvyššej kvality v čo najkratšom čase.

V ďalšom si predstavíme jednotlivé obmedzenia rozdelené podľa priorít.

Obmedzenia s najvyššou prioritou

1. Učiteľ nesmie učiť 2 lekcie naraz.
2. V žiadnej miestnosti nesmú prebiehať naraz 2 lekcie.
3. Ak skupina učiteľov má určený počet lekcií, ktoré má učiť, nesmie učiť iný počet lekcií.
4. Ak skupina učiteľov má určený maximálny počet lekcií, ktoré má učiť, nesmie učiť viac lekcií.
5. Učiteľ nesmie učiť vyučovaciu hodinu, kedy má zakázané učiť. 6. Lekcií nesmie byť priradená vyučovacia hodina, kedy je podľa všeobecnej dostupnosti zakázané učiť[2] 7. Ak má lekcia fixovanú nejakú hodnotu(učiteľ, miestnosť, vyučovacia hodina, offset), nesmie jej byť pridelená iná
8. Ak sa učiteľ musí vrámci jedného dňa presúvať medzi miestami výuky alebo areálmi, musí mať na to dostatok času - medzi skončením lekcie a začatím lekcie, na ktorú sa má presunúť musí byť časový rozdiel väčší ako časová vzdialenosť miest, kde sa lekcia učí

Obmedzenia s vysokou prioritou

9. Ak má kurz pre typ lekcie zadefinovaný maximálny počet lekcií, ktoré sa smú učiť v jeden deň, tak tento počet nesmie byť prekročený.

[2] pozrite časť 1.1.Vyučovacie hodiny

10. Ak lekcia pozostáva z viacerých častí, tak prienik im pridelených vyučovacích hodín musí byť prázdny.
11. Ak má kurz pre typ lekcie zadané, že lekcie toho typu sa nesmú učiť paralelne, tak prienik im pridelených vyučovacích hodín musí byť prázdny.
12. Ak kurz pre daný typ lekcie vyžaduje iba skupinu miestností z požadovaných skupín miestností, nesmie lekcii toho typu a kurzu byť pridelená iná skupina miestností (miestnosť).
13. Lekcií s požadovanou kapacitou nesmie byť pridelená miestnosť s menšou kapacitou ako lekciou požadovaná kapacita * koeficient kapacity pre daný typ lekcie[3].
14. Lekcií nesmie byť pridelená miestnosť inej fakulty, než je fakulta kurzu, ktorej je lekcia súčasťou. Výnimku tvoria prípady, kedy kurz pre daný typ lekcie požaduje skupinu miestností, ktorej miestnosti patria pod inú fakultu a lekcii je pridelená táto skupina miestností.
15. Lekcií nesmie byť pridelená miestnosť, ktorej areál sa nenachádza v miestach výuky kurzu danej lekcie. Výnimku tvoria prípady, kedy kurz pre daný typ lekcie požaduje skupinu miestností, ktorej miestnosti sa nachádzajú mimo miest výuky kurzu, ku ktorému patrí lekcia.
16. Ak je miestnosť v konkrétnu vyučovaciu hodinu rezervovaná pre konkrétnu fakultu, nesmie byť v danú vyučovaciu hodinu pridelená lekcii kurzu patriaceho pod inú fakultu. Ak je miestnosť v danú vyučovaciu hodinu rezervované pre cudzie inštitúcie, nesmie byť priradená žiadnej lekcii.
17. Ak lekcie dvoch kurzov majú neprázdny prienik študentov a jednu z lekcií navštevujú najmenej ako povinne voliteľnú, lekcie sa nesmú prekrývať(mať spoločný prienik pridelených vyučovacích hodín).
18. Ak lekcie dvoch kurzov majú neprázdny prienik študentov a jednu z lekcií navštevujú najmenej ako povinne voliteľnú a im pridelené miestnosti sa nachádzajú v rôznych areáloch alebo miestach výuky a učia sa v rovnaký deň, študenti musia mať dosť času na presun medzi lekciami.
19. Ak lekcie rovnakého kurzu a typu sa nesmú vyučovať paralelne a im pridelené miestnosti sa nachádzajú v rôznych areáloch alebo miestach výuky a učia sa v rovnaký deň, študenti musia mať dosť času na presun medzi lekciami.

Obmedzenia so strednou prioritou

20. Ak lekcie dvoch kurzov majú neprázdny prienik študentov a jednu z lekcií navštevujú najmenej ako výberovú, lekcie sa nesmú prekrývať(mať spoločný

[3]pozrite 1.1 Typy lekcií

prienik pridelených vyučovacích hodín).
21. Ak lekcie dvoch kurzov majú neprázdny prienik študentov a jednu z lekcií navštevujú najmenej ako výberovú a im pridelené miestnosti sa nachádzajú v rôznych areáloch alebo miestach výuky a učia sa v rovnaký deň, študenti musia mať dosť času na presun medzi lekciami.
22. Lekciám kurzov musia byť pridelené miestnosti obsahujúce zariadenie ktoré požadujú.
23. Kapacita miestnosti pridelená k lekcii nesmie byť menšia ako jej požadovaná kapacita.
24. Učiteľ nemá učiť vyučovaciu hodinu, keď mu to nevyhovuje(vhodnosť hodiny pre učiteľa je menej ako 50 (100 je maximálna vhodnosť))
25. Lekciám by nemali byť prideľované vyučovacie hodiny, ktorých všeobecná vhodnosť je menej ako 50. (Študenti nemajú radi keď majú v danom čase vyučovanie)

Obmedzenia s nízkou prioritou

26. Lekcie, ktoré navštevujú študenti z rovnakého študijného odboru by sa nemali prekrývať.
27. Ak lekciám, ktoré navštevujú študenti rovnakého odboru, sú pridelené miestnosti v rôznych areáloch alebo miestach výuky a vyučovacie hodiny v rovnakom dni, študenti by mali mať dostatočný čas na presun medzi lekciami.
28. Učiteľ by v jeden deň nemal učiť viac ako 4 vyučovacie hodiny.
29. Všeobecná vhodnosť vyučovacích hodín pridelených k lekciám by mala byť čo najvyššia.
30. Vhodnosť vyučovacích hodín pridelených učiteľom by pre nich mala byť čo najvyššia.
31. Lekcie v rozvrhu by mali využívať čo najmenej miestností.

Kapitola 2

Skúmané a použité metódy

Táto kapitola popisuje metódy, ktoré sa skúmajú a využívajú v tejto práci na generovanie rozvrhov. Hľadáme ako generovať rozvrhy využitím evolučných algoritmov, ktoré si občas pomáhajú aj lokálnym prehľadávaním. Preto si v nasledujúcich dvoch častiach uvedieme, čo je a aké má vlastnosti lokálne prehľadávanie a evolučné algoritmy. Lokálne prehľadávanie aj evolučné algoritmy spadajú do kategórie metaheuristických algoritmov. Tie sa zaoberajú hľadaním čo najlepších (nie nutne optimálnych) riešení pre optimalizačné problémy využitím prvku náhody. Riešenia z priestoru všetkých možných riešení problému hľadajú už z nájdených riešení ich úpravou na nové riešenia.

2.1 Lokálne prehľadávanie

Lokálne prehľadávanie slúži na hľadanie čo najlepšieho (nie nutne optimálneho) riešenia spomedzi možných riešení nejakého optimalizačného problému. Pri optimalizačných problémoch máme zadefinovanú hodnotovú funkciu, ktorá vie ohodnotiť kvalitu jednotlivých riešení. Na základe ohodnotenia riešení vieme riešenia medzi sebou porovnávať. Uvažujme ďalej, že hľadáme riešenie s minimálnou hodnotou.

Algoritmus lokálneho prehľadávania na začiatku náhodne vygeneruje alebo dostane na vstupe nejaké riešenie. Následne hľadá v jeho okolí (susedstve) iné riešenia a vymení súčasné riešenie za najlepšie z riešení v jeho okolí. Algoritmus takto postupne vymieňa aktuálne riešenie najlepším riešením z jeho okolia kým sa nezasekne v lokálnom minime, nenájde dostatočne uspokojivé riešenie alebo mu nevyprší časový limit pre hľadanie najlepšieho riešenia. Zaseknutie sa v lokálnom minime nastane, ak v okolí aktuálneho riešenia sa nachádzajú len horšie riešenia ako je ono samo. Lokálne minimum

predstavuje najkvalitnejšie nájdené riešenie, ktoré však nemusí byť najlepšie možné(globálne minimum). Zaseknutie v lokálnom minime sa zvykne riešiť zväčša reštartovaním lokálneho prehľadávania.

Aby mohlo byť aktuálne riešenie nahradené riešením z jeho okolia, musí sa nad priestorom všetkých možných riešení definovať relácia susednosti, ktorá z aktuálneho riešenia bude vedieť vygenerovať susedné riešenia. Susedné riešenia sa zvyknú generovať aplikovaním malej zmeny na pôvodné riešenie. Napríklad pri probléme farbenia grafu k-farbami môže byť malou zmenou riešenia (grafu) zmena farby niektorého jeho vrchola. Susedmi riešenia, ofarbeného grafu, tak budú všetky grafy, ktoré majú iba v nejakom jednom vrchole inú farbu ako dané riešenie.

Lokálne prehľadávanie, ktoré postupne vymieňa riešenia najlepšími ich susednými sa nazýva aj Horolezecký algoritmus. Okrem problému zaseknutia sa v lokálnom minime mu hrozí aj zacyklenie sa v lokálnom minime, ktoré nastane cyklickým obmieňaním niekoľkých riešení s rovnakou hodnotou. Tento problém môže pomôcť vyriešiť dočasné pamätanie si navštívených vrcholov, alebo pravidiel, ktoré by mohli vrátiť algoritmus späť na už navštívené vrcholy v tabu-zozname. Algoritmus by mal zakázané zmeniť aktuálne riešenie za riešenie v tabu zozname alebo by mal zakázané použiť na riešenie operáciu z dočasne zakázaných operácií. Rozšírenie lokálneho prehľadávania využívaním tabu-zoznam sa nazýva Tabu prehľadávanie.

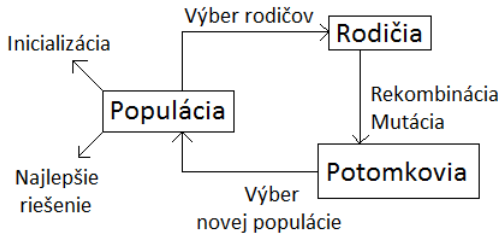
2.2 Evolučné algoritmy

Evolučné algoritmy sa používajú na riešenie ťažkých optimalizačných problémov, kde sa hľadá čo najlepšie (nie nutne optimálne) riešenie. Tieto algoritmy sú inšpirované procesmi prirodzenej evolúcie v prírode tak, ako ich popísal Darwin.

2.2.1 Evolučný algoritmus

Na začiatku evolučného algoritmu sa inicializuje (zväčša náhodne vygeneruje) **populácia** riešení - podmnožina z množiny všetkých možných riešení problému. Populácia pozostáva z jedincov - jednotlivých riešení problému. Jedince v populácií sú zakódované vo forme genotypu. **Genotyp** presne určuje konkrétneho jedinca - riešenie v populácii. Populácia je multimnožina genotypov riešení - môže obsahovať jeden genotyp viac-krát.

Nad všetkými genotypmi sa definuje **účelová funkcia**, ktorá pre každý genotyp vie vrátiť mieru jeho kvality (fitness). Miera kvality genotypu riešenia určuje, nakoľko je dané riešenie dobré a umožňuje navzájom porov-



Obr. 2.1: Evolučný algoritmus [4]

návať kvality riešení. Miera kvality má spravidla formu kladného reálneho čísla, pričom algoritmus sa snaží nájsť genotyp riešenia buď s maximálnou alebo minimálnou mierou kvality[1]. Genotyp riešenia s maximálnou(resp. minimálnou) mierou kvality predstavuje optimálne riešenie problému. Proces prideľovania miery kvality genotypom sa nazýva **evaluácia**.

Populácia sa v za sebou nasledujúcich generáciách obmieňa pomocou **variačných** a **selekčných operátorov**. Cieľom variačných operátorov je vytvárať nové a rôznorodé jedince. Variačné operátory sú tvorené operátormi **kríženia(rekombinácie)** a **mutácie**. Operátor kríženia na vstupe dostáva zväčša 2 genotypy - **rodičov**, z ktorých vzájomnou kombináciou vytvára nové genotypy - **potomkov**, tvoriacich výstup operátora. Operátor mutácie na vstupe dostáva iba jeden genotyp, z ktorého nejakou jeho zmenou vytvorí a vráti ako výstup nový (spravidla iný) genotyp. Tento operátor sa s nejakou pravdepodobnosťou, nazývanou aj pravdepodobnosť mutácie, zvykne aplikovať na novo-vytvorených potomkov vzniknutých operáciou kríženia.

Selekčné operátory tvoria operátor výberu rodičov a operátor nahradenia. **Operátor výberu rodičov** pracuje nad celou populáciou a vyberá z nej rodičov pre operátor kríženia. Rodičov z populácie vyberá náhodne, pričom jedince s vyššou mierou kvality majú vyššiu šancu, že budú vybraté za rodičov než tie s nižšou. Tento operátor má svoje korene v teórií evolúcie, kde silnejšie (lepšie) jedince majú väčšiu šancu na reprodukciu než slabšie. Predpokladá sa, že kombináciou kvalitných rodičov vzniknú rovnako kvalitný, ba ešte kvalitnejší potomkovia. Ak má napríklad každý rodič nejakú inú dobrú vlastnosť, je šanca, že ich potomok zdedí obe dobré vlastnosti. **Operátor nahradenia** má na starosti postupné zvyšovanie kvality populácie nahradením nekvalitných riešení, udržiavanie konštantného počtu jedincov v populácii a striedanie generácií. V jednej generácii sa na populáciu aplikujú operátory výberu rodičov, kríženia a mutácie, ktoré majú za následok vznik nových

[1]Miera kvality je prakticky ekvivalentná hodnotovej funkcií definovanej pri optimalizačných problémoch.

jedincov v populácií. Operátor nahradenia potom určí, ktoré zo starých a novovzniknutých jedincov v populácií vytvoria novú populáciu - nasledujúcu generáciu, ktorá nahradí súčasnú populáciu. Existujú rôzne prístupy k výberu jedincov do novej populácie. Jedným z prístupov je, že novovytvorené jedince nahradia najhoršie pôvodné jedince v populácii. Iný prístup spomedzi pôvodných a novovytvorených jedincov vyberie tie najlepšie (s najlepšou mierou kvality) a vytvorí z nich novú populáciu.

Evolučný algoritmus postupne pomocou selekčných a variačných operátorov obmieňa populáciu v za sebou nasledujúcich **generáciách** pokým nenájde riešenie s dostatočne dobrou mierou kvality alebo nevystrieda predom daný počet generácií. Niekedy sa určí aj maximálny časový limit, počas ktorého môže algoritmus pracovať. Výstupom algoritmu je najlepšie riešenie, aké sa našlo počas všetkých generácií populácie alebo najlepšie riešenie z poslednej generácie populácie.

2.2.2 Druhy evolučných algoritmov

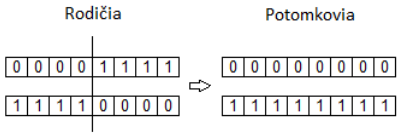
Evolučné algoritmy zdieľajú rovnaké princípy, ale navzájom sa líšia v ich technickom prevedení. Najvýznamnejšie rozdiely sú v spôsobe reprezentácie genotypov a následnom definovaní a používaní variačných operátorov pracujúcich nad danými genotypmi. Podľa spôsobu reprezentácie genotypov a účelu použitia sa v evolučných algoritmoch identifikovali nasledujúce smery:

- **Genetické algoritmy**: genotypy tvoria reťazce definované nad konečnou abecedou. Zvyknú to byť napríklad postupnosti tvoriace permutácie miest pri riešení problému obchodného cestujúceho alebo množstvá jednotlivých druhov tovaru, ktoré si má dať horský nosič do batoha pri riešení problému batoha.

- **Evolučné stratégie**: genotypy majú formu vektora s hodnotami reálnych čísel. Využívajú sa pri problémoch so spojitým priestorom.

- **Evolučné programovanie**: genotypy predstavujú jednotlivé konečnostavové automaty. Tento smer má svoj pôvod v umelej inteligencií, kde evolúciu vnímali ako proces učenia, ktorého cieľom bolo generovať umelú inteligenciu.

- **Genetické programovanie**: genotypy sú tvorené programami. Využitie nachádza pri evolúcií počítačových programov. Napríklad programu pre hranie dámy.



Obr. 2.2: jednobodové kríženie

Na hľadanie rozvrhov pomocou evolučných algoritmov sa používajú najmä genetické algoritmy. Genetické algoritmy sa spomínajú a využívajú v tejto práci. V ďalšom sa preto zameriame na genetické algoritmy.

2.2.3 Genetické algoritmy

Genetické algoritmy kódujú riešenia problémov vo forme reťazcov[2]. Najpopulárnejšie sú reprezentácie riešení vo forme binárnych reťazcov, reálnych vektorov alebo vo forme permutácií. Používajú sa na miestach, kde jedno riešenie predstavuje priradenie hodnôt premenným. Pri probléme batoha môžu premenné predstavovať druhy tovarov a hodnoty k nim priraďované počty kusov tovaru, ktoré má horský nosič vložiť do batoha. Pri probléme obchodného cestujúceho môže jedno riešenie predstavovať permutáciu miest, ktoré má cestujúci obchodník navštíviť. I-ta premenná môže obsahovať i-te navštívené mesto. Pri tvorbe rozvrhov môžu byť premenné lekcie a hodnoty vyučovacie hodiny a miestnosti, určujúce kedy a kde sa má lekcia vyučovať.

Binárne reťazce

Pri kódovaní riešení do binárnych reťazcov je dôležité určiť správnu dĺžky reťazcov a mapovanie z reťazcov na riešenia problému. Je dôležité, aby každý reťazec tvoril genotyp nejakého riešenia. Reťazec (genotyp) zvyčajne tvoria za sebou nasledujúce binárne zakódované hodnoty jednotlivých premenných. Prvých i-bitov kóduje hodnotu prvej premennej, ďalších j-bitov hodnotu druhej a tak ďalej.

Operátor mutácie nad genotypom zvykne väčšinou s nejakou pravdepodobnosťou zmeniť hodnotu jedného alebo viacerých náhodne zvolených bitov. Zvykne sa používať metóda, kde každý bit reťazca zmení svoju hodnotu na opačnú s nejakou vopred určenou pravdepodobnosťou.

Operátory kríženia zvyčajne na vstupe dostanú dvoch rodičov a vrátia 2 potomkov, ktorý vzniknú ich kombináciou. Najčastejšie používanými ope-

[2]zakódované riešenie = genotyp

rátormi kríženia sú jedno-bodové kríženie, n-bodové kríženie a uniformné kríženie.

Jednobodové kríženie náhodne zvolí miesto bodu kríženia k. Prvý potomok získa prvých k-bitov reťazca od prvého rodiča a zvyšné bity doplní bitmi od k+1 bitu po koniec reťazca z druhého rodiča. Druhý potomok opačne získa prvých k-bitov od druhého rodiča a zvyšné od prvého.

N-bodové kríženie namiesto jedného bodu kríženia zvolí náhodne n-bodov kríženia. N-bodov kríženia rozdelí reťazec na n+1 častí. Prvý potomok získa každú druhú časť reťazca od prvého rodiča a zvyšok doplní bitmi od druhého. Druhý potomok naopak.

Uniformné kríženie každému bitu reťazca náhodne pridelí reálne číslo v intervale <0,1>. Na rovnakom intervale sa zvolí parameter p a prvý potomok získa od prvého rodiča všetky bity s prideleným reálnym číslom vyšším ako p a od druhého všetky bity s hodnotou menšou ako p. Druhý potomok získa bity opačne.

Vektory reálnych čísiel

Genotyp pozostáva z vektoru reálnych čísel, kde každá hodnota vektora môže byť ohraničená intervalmi, v ktorých sa má hodnota na pozícií vektora vyskytovať.

Mutácia zvykne ku každej zložke vektoru s určitou pravdepodobnosťou pripočítať hodnotu z intervalu, tak aby neboli porušené podmienky. Zvykom je tiež náhodne zvoliť zložku vektoru a tú nahradiť inou náhodne zvolenou hodnotou z možných hodnôt pre danú zložku.

Na operátor rekombinácie (kríženia) sá zvyknú používať 2 druhy rekombinácií: diskrétna a aritmetická.

Diskrétna rekombinácia používa podobné metódy ako binárna rekombinácia. Jednotlivé zložky vektora však nikdy nerozdelí a potomok získa od rodiča vždy celú zložku vektora. Pri rekombinácií vektorov reálnych čísel na rozdiel od rekombinácie binárnych vektorov nevznikajú nové hodnoty v zložkách vektora.

Aritmetická rekombinácia zvykne potomkov tvoriť kombináciou zložiek rodičov, s tým, že niektoré zložky potomka dostanú hodnotu priemeru hodnôt rodičov alebo hodnotu medzi ich dvomi hodnotami.

Permutácie

Typickým príkladom využitia reprezentácie riešenia vo forme permutácie je príklad obchodného cestujúceho spomenutého v úvode ku genetickým algoritmom. Jedno riešenie tu predstavuje jednu konkrétnu permutáciu miest,

ktoré má obchodník navštíviť. Zmeny nad genotypmi musia byť robené tak, aby sa po zmene žiadne dve zložky genotypu neopakovali a všetky zložky tvorili stále permutáciu.

Mutácie genotypov sa vykonávajú rôznymi spôsobmi: výmenou dvoch prvkov, vložením prvku, permutáciou časti alebo inverziou časti permutácie. **Výmena dvoch prvkov** vyberie náhodne dve pozície v permutácií a vymení prvky na daných pozíciach. **Mutácia vložením prvku** vyberie v permutácií náhodne dve pozície a prvok z druhej pozície vloží za pozíciu prvého. **Permutácia časti** prebehne tak, že sa náhodne vyberie nejaká súvislá časť permutácie, v ktorej sa prvky náhodne premiešajú. **Mutácia inverziou** prebieha tak, že sa zvolí súvislá časť permutácie, ktorej prvky sa preusporiadajú od posledného k prvém. Posledný typ mutácie je výhodný pri problémoch, kde záleží na susednosti dvojíc prvkov v permutácií. Táto mutácia vykoná v tomto probléme nad permutáciou najmenšiu možnú zmenu lebo zmení susednosť prvkov len v 2 prípadoch.

Rekombinácia predstavuje pri permutáciach väčší problém, pretože treba nájsť spôsob ako skombinovať do potomkov vlastnosti rodičov, tak aby aj potomkovia tvorili permutácie. Z viacerých operátorov rekombinácie sme sa rozhodli bližšie opísať PMX a Cyklické kríženie.

PMX (Partially mapped crossover) bol prvý raz predstavený pri riešení problému obchodného cestujúceho. Výhodou tohto operátora je, že v potomkoch zachováva susedné dvojice z rodičov. Algoritmus na vytvorenie 2 potomkov z 2 rodičov vyzerá nasledovne:
1. Náhodne vyber 2 miesta kríženia v permutácií a prvky medzi tými miestami skopíruj z prvého rodiča do prvého potomka
2. Začínajúc od prvého miesta kríženia, nájdi medzi dvoma miestami kríženia prvky v druhom rodičovi, ktoré neboli prekopírované z prvého rodiča do potomka
3. Pre každý z týchto prvkov (povedzme i) pozri v potomkovi, ktorý prvok (j) bol na to miesto skopírovaný z prvého rodiča
4. Umiestni i na pozíciu, ktorú okupovalo j v druhom rodičovi
5. Ak pozícia okupovaná j v druhom rodičovi je už v potomkovi obsadená iným prvkom (k), vlož i na pozíciu okupovanú k v druhom rodičovi
6. Zvyšné prvky, ktoré neboli medzi miestami kríženia vlož z druhého rodiča do prvého potomka.
7. Vytvor druhého potomka tak, že vymeníš úlohy rodičov.

Cyklické kríženie sa snaží v potomkoch zachovať čo najviac informácií o absolútnych pozíciach, na ktorých sa vyskytovali prvky v rodičoch. Algoritmus cyklického kríženia je nasledovný:
1. Vyber si pozíciu na prvom rodičovi a pridaj ju do cyklu.

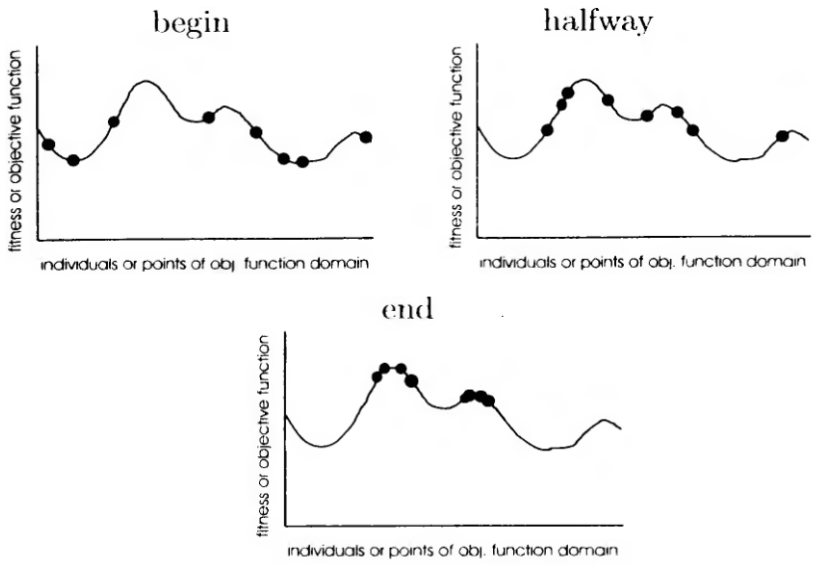
2. Pozri na prvok(i) na rovnakej pozícií v druhom rodičovi
3. Choď na pozíciu v prvom rodičovi, kde je prvok i a tú pozíciu pridaj do cyklu.
4. Opakuj kroky 2 a 3 pokiaľ neprídeš na pozíciu vybranú v kroku 1
5. Prvky v cykle daj na rovnaké pozície z prvého rodiča do prvého potomka. Zvyšné pozície vyplň prvkami z druhého rodiča. Obdobne vytvor druhého potomka.

2.2.4 Vlastnosti evolučných algoritmov

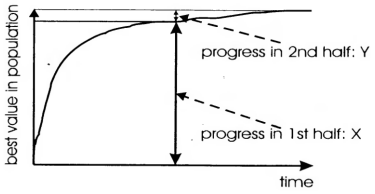
V tejto časti sa zameriame na výpočet a vlastnosti typického evolučného algoritmu a ukážeme problémy, ktoré treba riešiť pri konštrukcií evolučných algoritmov. Pre ilustráciu vlastností evolučných algoritmov budeme uvažovať bežný evolučný algoritmus, ktorý hľadá riešenie s maximálnou mierou kvality (fitness).

Rozmiestnenie populácie

Evolučný algoritmus hľadá čo najlepšie (nie nutne optimálne) riešenie spomedzi všetkých možných riešení problému. Hľadá genotyp s najvyššou mierou kvality v priestore všetkých možných genotypov. Obrázok 2.3 ilustruje rozmiestnenie genotypov populácie v priestore všetkých genotypov v rôznych časoch behu výpočtu. Na začiatku - hneď po inicializácií je populácia jedincov rovnomerne rozmiestnená po celom prehľadávacom priestore. Už po niekoľkých generáciách sa toto rozmiestnenie vďaka použitiu selekčných a variačných operátorov, kde populácia opustí regióny s malou mierou kvality a jedince sa začnú "šplhať"po okolitých kopcoch (regiónoch s vyššou mierou kvality). Na konci prehľadávania je populácia rozmiestnená len okolo pár vrcholov a v lepšom prípade aj okolo najvyššieho, predstavujúceho riešenie s maximálnou mierou kvality. Žiaľ môže nastať aj prípad, kedy populácia skončí na nie najvyššom vrchole - v lokálnom optime. O prehľadávaní priestoru sa dá povedať, že je rozdelené do 2 hlavných fáz: *exploration* - generácia nový jedincov je nasadená do neznámych regiónov prehľadávacieho priestoru a *exploitation* - koncentrácia prehľadávania do okolia dobrých riešení. Dobrý evolučný algoritmus musí nájsť kompromis medzi týmito dvoma fázami. Veľa hľadania v neprebádaných oblastiach priestoru riešení znižuje efektívnosť algoritmu. Príliš skoré zameranie sa na konkrétny cieľ môže viesť k nájdeniu riešenia, ktoré je len lokálne a nie globálne optimálne.



Obr. 2.3: rozmiestnenie populácie v rôznych časoch prehľadávania [4]



Obr. 2.4: miera kvality(fitness) najlepšieho jedinca populácie v čase [4]

Vývoj najlepšej nájdenej miery kvality v čase

Obrázok 2.4 ukazuje hodnoty mier kvality najlepších jedincov v populácií v čase. Na obrázku možno vidieť, že miera kvality najlepšieho jedinca v prvých generáciách prudko rastie a po čase sa rast prudko spomalí až nebadať takmer žiadne zlepšenia. Táto vlastnosť ukazuje, že zvyšovanie počtu generácií na nájdenie želaného riešenia nemá od istého okamžiku zmysel a zároveň, predvypočítavanie si jedincov, aby prehľadávanie začalo s jedincami s lepšou mierou kvality nemusí mať rovnako zmysel, pretože už po pár generáciách sa miera kvality jedincov v populácií prudko zvýši.

Kapitola 3

Univerzitné rozvrhy vo svete

3.1 Univerzitné rozvrhy ako problém

Oblasť generovania rozvrhov pre univerzity sa delí na dva hlavné smery podľa problému, ktorý riešia: Skúškové rozvrhy a kurzové rozvrhy. **Skúškové rozvrhy** sa tvoria pre obdobie skúšok a časovým rozsahom zvyknú pokrývať celé skúškové obdobie. **Kurzové rozvrhy** pokrývajú časový rozsah jedného týždňa, ktorý sa opakuje počas celého semestra. Hlavným rozdielom medzi skúškovými a kurzovými rozvrhmi býva, že pri skúškových rozvrhoch možno umiestniť viacero skúšok do jednej miestnosti v rovnakom čase, zatiaľ čo pri kurzových sa smie v jednej miestnosti a čase vyučovať len jedna lekcia(kurz, udalosť). Pri skúškových rozvrhoch býva snaha nechať študentom medzi jednotlivými skúškami, čo najväčšiu časovú medzeru alebo aspoň nedať študentom viac skúšok na jeden deň. Pri kurzových rozvrhoch naopak býva snaha aby študenti medzi lekciami nemali veľké časové medzery. V tejto práci sa budeme zaoberať kurzovými rozvrhmi, nakoľko problém, ktorý riešime spadá do tohto smeru.

Všeobecná definícia problému tvorby univerzitných rozvrhov má nasledovnú podobu[1]: *Úlohou pri tvorbe univerzitného rozvrhu je umiestniť konečnú množinu udalostí(lekcie kurzov, skúšky, stretnutia) do limitovaného (konečného) počtu vyučovacích hodín a miestností v súlade s množinou obmedzení kladených na rozvrh.*

Obmedzenia kladené na rozvrh sa zvyčajne delia na **silné** a **slabé**. Silnými obmedzeniami zvyčajne bývajú požiadavky aby žiaden učiteľ neučil na dvoch miestach naraz, dve udalosti(lekcie) sa nevyučovali naraz v rovnakej miestnosti, žiaden študent alebo skupina študentov nemala navštevovať naraz 2 kurzy. Rozvrh spĺňajúci všetky silné obmedzenia sa označuje ako **uskutočniteľný**. Slabým obmedzením býva napríklad požiadavka aby študenti ne-

mali medzi dvoma lekciami voľnú vyučovaciu hodinu alebo požiadavka aby učiteľ neučil 2 rôzne lekcie hneď za sebou, ale mal čas na prípravu. Od rozvrhu sa požaduje aby bol uskutočniteľný(spĺňal všetky silné obmedzenia) a dodržiaval čo najviac slabých obmedzení aj keď ich nutne nemusí dodržať všetky. Rozvrh spĺňajúci všetky silné aj slabé obmedzenia (**ideálny**) sa vo väčšine prípadov nedá zostrojiť, preto sa hľadá rozvrh spĺňajúci všetky silné obmedzenia a porušuje čo najmenej slabých.

Kvalita rozvrhu sa meria najčastejšie podľa druhu a množstva obmedzení, ktoré porušuje. Ak vieme určiť a porovnávať kvalitu vygenerovaných rozvrhov, úloha vygenerovať najkvalitnejší rozvrh zo vstupných údajov predstavuje **konštrukčný kombinatorický optimalizačný problém**. Problém je konštrukčný a optimalizačný, lebo jeho cieľom je skonštruovať najkvalitnejší rozvrh spomedzi všetkých možných rozvrhov. Problém je kombinatorický, pretože hľadáme rozvrh z konečnej množiny všetkých možných rozvrhov.

Čo sa časovej zložitosti týka, problém zkonštruovania najkvalitnejšieho rozvrhu je **NP-úplný**. Dôkazy tohto tvrdenia možno nájsť vo viacerých publikáciach(napr.[1],[2]). V dôkazoch sa najčastejšie využíva podobnosť problému s problémom farbenia grafu k-farbami, o ktorom sa vie, že je NP-úplný. V probléme farbenia grafu k-farbami máme zadaný neorientovaný graf a k-farieb. Úlohou je prideliť ku každému vrcholu grafu farbu z k-farieb tak, aby susedné vrcholy[1] nemali rovnakú farbu. Problém farbenia grafu k-farbami sa dá prerobiť pomocou deterministického Turingovho stroja v polynomiálnom čase na problém vygenerovania rozvrhu s k-vyučovacími hodinami. Vrcholy sa prevedú na udalosti, pričom ak je medzi vrcholmi hrana, zakážeme, aby sa udalosti učili rovnakú vyučovaciu hodinu.

3.2 Prístupy k hľadaniu rozvrhov

Ako sme uviedli vyššie, problém vytvorenia rozvrhu je NP-úplný kombinatorický optimalizačný problém. Nájsť optimálne(najlepšie) riešenie pre tento druh problému si väčšinou vyžaduje prehľadanie celého priestoru možných riešení, ktorý so zväčšujúcimi sa inštanciami problémov(viac kurzov, miestností, ...) exponenciálne narastá a prakticky sa stáva neprebádateľný. Prehľadávanie celého priestoru riešení je možné len pri malých inštanciách problému.

Z tohto dôvodu sa pre hľadanie takýchto problémov používajú aproximačné algoritmy, ktoré vedia nájsť uspokojivé riešenie, ktoré nemusí byť

[1] vrcholy spojené hranou

optimálne, v rozumnom čase. Väčšina prístupov pre riešenie problému vygenerovania univerzitného rozvrhu spadá do tejto kategórie.

Burke a spol. vo svojej práci [3] identifikujú 4 silné skupiny prístupov k tvorbe rozvrhov: sekvenčné, klastrové, založené na obmedzeniach a metaheuristické.

Sekvenčné prístupy zoraďujú udalosti do radu podľa nejakej heuristiky a potom sa ich snažia sekvenčne priradiť do časov v rozvrhu, tak aby udalosti neboli navzájom v konflikte. Je zvykom problém reprezentovať ako graf, v ktorom udalosti predstavujú vrcholy. Medzi vrcholmi sa nachádza hrana ak predstavujúce udalosti sú v konflikte – t.j. zdieľajú aspoň jeden spoločný prostriedok akým je napríklad študent, učiteľ, a preto nemôžu byť nasadené do rozvrhu v rovnakom čase. Konštrukcia rozvrhov bez konfliktov je prideľovanie časov z konečnej množiny časov k udalostiam, tak aby neboli v konflikte a je modelovaná ako farbenie grafu k-farbami. Pre radenie udalostí do radu na zaradenie do rozvrhu sa používajú prevažne nasledovné heuristiky:

- Vrcholy s najmenším stupňom vrchola ako prvé (Largest degree first). Ako prvé sú do rozvrhu zaraďované udalosti, ktoré majú konflikty s najväčším počtom udalostí. Heuristika vychádza z predpokladu, že takéto udalosti je najťažšie zaradiť do rozvrhu.

- Vrcholy s najväčšou váhou ako prvé(Largest weighted degree). Je to modifikácia predchádzajúcej heuristiky, kde sa neberie už do úvahy počet konfliktov ale suma všetkých váh konfliktov(hrán) udalostí s danou udalosťou. Váha konfliktu zvyčajne predstavuje počet študentov navštevujúcich obe udalosti.

- Stupeň nasýtenia (Saturation degree). V každom kroku sa do rozvrhu zaradí udalosť, ktorá má najmenší počet časov, do ktorých môže byť zadelená.

- Stupeň farby(Colour degree). Heuristika uprednostňuje tie udalosti, ktoré majú najväčší počet konfliktov s udalosťami, ktoré už boli zaradené do rozvrhu.

Po zoradení udalostí pre postupné vkladanie do rozvrhu nasleduje ich postupné vkladanie do rozvrhu. Vyvstáva otázka, do akých časov v rozvrhu ich zadeliť. Používajú sa rôzne prístupy. Jedným z nich je napríklad vybratie prvého vhodného času.

Klastrové metódy rozdeľujú udalosti do klastrov - podmnožín tvoriacich rozklad množiny všetkých udalostí. V klastroch sa udalosti rozdelia do

skupín(spoločných pre všetky klastre), tak aby neporušovali silné obmedzenia. Skupinám sú potom pridelené časy tak, aby bolo porušených čo najmenej slabých obmedzení. Ako príklad nám môže poslúžiť škola s viacerými študijnými odbormi. Každý študijný odbor vyučuje svoje predmety, pričom sú predmety spoločné pre viacero študijných odborov. Algoritmus najprv rozdelí predmety(udalosti) do klastrov podľa študijných odborov. Ak je predmet spoločný pre viacero študijných odborov umiestní sa do špeciálneho klastru. Spraví sa graf predmetov, kde vrcholy budú predstavovať predmety a hrany budú medzi predmetmi, ktoré sú v konflikte a nemôžu byť nasedené v rovnakom čase. Predmety sú v konflikte ak ich učí rovnaký učiteľ alebo navštevuje rovnaký študent. Vrcholy sa potom ofarbia farbami tak aby susedné vrcholy nemali rovnakú farbu.(počet farieb býva menší rovný počtu vyučovacích hodín v týždni). Najprv sa ofarbujú vrcholy v špeciálnom klastri, potom po jednotlivých klastroch, tak aby aby boli splnené silné obmedzenia. Po skončení sa jednotlivým farbám priradeným k predmetom pridelia časy(vyučovacie hodiny), tak aby bolo porušených čo najmenej slabých obmedzení.

Prístupy založené na obmedzeniach[2] modelujú problém tvorby rozvrhu ako problém prideľovania hodnôt k premenným, tak aby bolo splnené množstvo pravidiel (požiadaviek) pre prideľovanie hodnôt k premenným. Premennými sú jednotlivé udalosti a priraďujú sa k ním hodnoty predstavujúce časy, miestnosti príp. ďalšie prostriedky. Ak pre čiastočne vytvorené riešenie sa nedá aplikovať žiadne pravidlo, spustí sa backtrack, ktorý sa vykonáva pokiaľ sa nenájde riešenie(rozvrh) spĺňajúci všetky požiadavky.

Metaheuristické metódy sú charakteristické tým, že riešenie hľadajú v priestore všetkých možných riešení použitím prvku náhody. Na začiatku zvyknú (zväčša náhodne) vygenerovať jedno alebo viacero riešení problému, ktoré aplikovaním rôznych operátorov menia na iné lepšie riešenia, čím sa snažia efektívne prehľadávať priestor riešení a nájsť čo najlepšie. Metaheuristické metódy sa stali v relatívne nedávnej dobe veľmi populárne pri riešení kombinatorických optimalizačných problémov. Svoju popularitu si získali najmä tým, že s malými zmenami sú úspešne aplikovateľné na množstvo rôznych kombinatorických optimalizačných problémov vrátane generovania univerzitných rozvrhov. Medzi metaheuristické metódy patria aj evolučné algoritmy a lokálne prehľadávanie, ktorých využitie na hľadanie univerzitných rozvrhov skúmame v tejto práci.

Jedna z vecí, ktorá oddeľuje problém hľadania univerzitných rozvrhov od iných kombinatorických optimalizačných problémov je prítomnosť dvoch typov obmedzení - silných a slabých. Podľa Lewisovho prieskumu [1] väčšina

[2] constraint-based approaches

metaheuristických[3] algoritmov sa podľa spôsobu riešenia tohto problému delí do troch skupín: jednoprechodové, dvojprechodové a algoritmy umožňujúce relaxáciu.

Jednoprechodové algoritmy sa snažia počas svojho behu naraz riešiť silné aj slabé obmedzenia. Porušenie obmedzenia zvyknú penalizovať pokutou v závislosti od závažnosti porušeného obmedzenia. Silné obmedzenia bývajú pokutované vyššími pokutami ako slabé. Kvalita rozvrhu je vyjadrená ako súčet všetkých porušení obmedzení, kde každé porušenie obmedzenia je zarátané veľkosťou pokuty za jeho porušenie. Algoritmus potom hľadá rozvrh s najnižšou celkovou pokutou. Tento prístup je populárny, lebo do už existujúcich algoritmov sa ľahko pridávajú nové obmedzenia. Pri prehľadávaní priestoru riešení povoľujú jednoprechodové algoritmy riešenia porušujúce silné obmedzenia. Takéto riešenia sú ale silno penalizované a algoritmus preto dáva prednosť riešeniam, ktoré ich neporušujú.

Dvojprechodové algoritmy pracujú v dvoch fázach. V prvej sa snažia nájsť uskutočniteľný rozvrh neporušujúci žiadne silné obmedzenie nehľadiac na slabé. Po nájdení uskutočniteľného rozvrhu prehľadávajú len tie časti priestoru riešení neporušujúcich žiadne silné obmedzenie a zameriavajú sa na minimalizovanie počtu porušenia slabých obmedzení. Tento prístup je aplikovateľný pri problémoch, kde sa ľahko dá nájsť uskutočniteľný rozvrh a následne prehľadávať iné (okolité) riešenia tak aby neboli porušené silné obmedzenia. V praktických aplikáciach je tento prístup málo používaný, pretože nájsť uskutočniteľný rozvrh je ťažké - NP-ťažké.

Algoritmy umožňujúce relaxáciu fungujú podobne ako dvojprechodové s tým rozdielom, že ak pri vytváraní rozvrhu nevie nejakú udalosť doň zaradiť tak, aby nebolo porušené žiadne silné obmedzenie, odloží ju bokom medzi nezaradené udalosti a ďalej sa snaží minimalizovať porušovanie slabých. V neskorších fázach, po vykonaní mnohých zmien na rozvrhu, sa znovu pokúsi zaradiť odložené nezaradené udalosti do rozvrhu. Takýto proces zjednodušovania si problému (napr. odkladaním nepohodlných udalostí bokom) sa nazýva relaxácia. Relaxácia problému sa zvykne uskutočňovať aj pridávaním ďalších vyučovacích hodín do rozvrhu, ak nie je hodina, kam udalosť zaradiť. Pri oprave sa potom algoritmu snaží dať znovu použitý počet vyučovacích hodín do normálneho - daného počtu.

3.3 Rôznosť univerzitných rozvrhov

V predošlej časti sme si predstavili problém tvorby rozvrhov vo všeobecnosti a spomenuli sme, že chceme aby spĺňali nejaké obmedzenia, ktoré sa v hrubom

[3] aj evolučných

zvyknú deliť na silné a slabé. Vo všeobecnosti sme si ukázali aj druhy rôznych prístupov k hľadaniu rozvrhov. V prvej kapitole smi si predstavili problém, ktorý riešime, pričom riešenia hľadáme v oblasti evolučných algoritmov. V tejto kapitole si preto spravíme prehľad napísaných prác venujúcich sa tvorbe univerzitných kurzových rozvrhov pomocou evolučných algoritmov. Dôraz budeme klásť na práce a prístupy, ktoré riešia podobný, podobne zložitý, problém ako my.

Vo svete neexistuje presná definícia toho, čo znamená problém tvorby univerzitného kurzového rozvrhu. Väčšinou si každá univerzita definuje požiadavky od rozvrhu, ktorý chce vytvoriť a potom hľadá taký rozvrh. Požiadavky, aby žiaden učiteľ neučil na dvoch miestach naraz alebo, aby v jednej miestnosti neprebiehali naraz dve lekcie(udalosti) alebo aby študent nemal dve rôzne lekcie v rovnakom čase bývajú spoločné. Rozdiely bývajú v ďalších požiadavkách. Sú napríklad školy, ktoré chcú aby študenti mali výučbu rovnomerne rozmiestnenú v celom týždni. Iné školy zas chcú, aby študenti mali celú výučbu čo najviac v kope a ostali im voľné dni na štúdium, či zamestnanie popri škole. Veľké rozdiely sú aj v definovaní, čo je to "dobrýälebo "dostatočne dobrý"(uskutočniteľný) rozvrh s ktorým by jednotlivá univerzita bola spokojná.

Rozdiely však nie sú iba v obmedzeniach kladených na rozvrh, ale aj v zložitosti problému. Niektorý(väčšina) počítajú iba problém, kde každá lekcia trvá práve jednu vyučovaciu hodinu a vyučuje sa v práve jednej miestnosti. Iní rátajú aj s tým, že lekcia môže trvať viac za sebou nasledujúcich vyučovacích hodín a ďalší k tomu pridajú aj možnosť výučby lekcie vo viacerých miestnostiach naraz. Naša definícia problému ráta s možnosťou, že lekcia trvá viac než jednu vyučovaciu hodinu a môže prebiehať vo viacerých miestnostiach naraz. Aby toho nebolo málo, počítame s tým, že niektoré lekcie sa môžu vyučovať aj len každý druhý, tretí, i-ty týždeň. Narozdiel od väčšiny problémov sa zaoberáme aj otázkou prideľovania lekcií podmnožine učiteľov z množín učiteľov schopných vyučovať danú lekciu. Riešenie posledných dvoch problémov z nás robí pravdepodobne svetový unikát - nepodarilo sa nám totiž nájsť prácu, ktorá by priraďovala učiteľov k lekciám, či nasadzovala lekcie v párny a nepárny týždeň.

Rozdielnosť v definíciách, spôsobuje aj praktickú neporovnateľnosť výsledkov. Autor práce si zadefinuje problém, ktorý rieši a napíše, ako veľmi bol podľa neho úspešný. Na svet sa tak dostalo niekoľko zaujímavých metóda a výsledkov, ktoré sa však ťažko porovnávajú.

V rámci štúdia automatického generovania univerzitných rozvrhov bol pre vedecké účely v 2001 zadefinovaný UCTP(University Course Timetabling problem)[6] problém, ktorý v 2002 poslúžil ako zadanie pre First International Timetabling Competition[5].

UCTP pozostáva z množiny udalostí, miestností, vyučovacích hodín(5 dní po 9 hodín) a študentov, kde každá miestnosť má definovanú svoju kapacitu miest na sedenia a zariadenia, ktoré obsahuje. Každý študent navštevuje nejakú podmnožinu z množiny udalostí a každá lekcia požaduje pre svoju výučbu podmnožinu z množiny zariadení, ktoré by mala obsahovať miestnosť, kde je udalosť nasadená. Cieľom je ku každej miestnosti prideliť práve jednu vyučovaciu hodinu a práve jednu miestnosť tak aby boli splnené všetky silné obmedzenia a čo najviac slabých[4]. Definované sú 3 silné obmedzenia a 3 slabé obmedzenia. Silné obmedzenia sú: 1. Žiaden študent nesmie navštevovať v jednu vyučovaciu hodinu viac udalostí. 2. Všetky udalosti sú pridelené do miestností s postačujúcou kapacitou a vybavením(miestnosť obsahuje všetko potrebné zariadenie). 3. Pre všetky vyučovacie hodiny a miestnosti platí, že maximálne jedna udalosť je pridelená do miestnosti v jednom čase. Slabé obmedzenia sú: 1. Žiaden študent by nemal navštevovať udalosť nasadenú v poslednú hodinu dňa. 2. Žiaden študent by nemal mať viac ako 2 udalosti nasledujúce hneď po sebe. 3. Žiaden študent by nemal mať v dni len jednu udalosť. Porušenie silných obmedzení je závažnejšie ako slabých, pričom silné a slabé obmedzenia sú si ekvivalentné.

Riešením UCTP sa zaoberá množstvo publikácií. Jeho nevýhodou je jeho prílišná jednoduchosť, ktorá nekorešponduje s požiadavkami z praxe. Second International Timetabling Competition UCTP skomplikovala na PE-UCTP(Post Enrollment UCTP) pridaním dvoch silných obmedzení, ktoré riešili nedostupnosť lekcií v niektoré vyučovacie hodiny a precedenciu udalostí, kde udalosť má byť nasadená pred alebo po inej udalosti.

Celkovo existuje málo prác, ktoré by riešili problém kde lekcia môže byť vyučovaná viac ako jednu vyučovaciu hodinu alebo by mohla byť nasadená vo viacerých miestnostiach.

3.4 Evolučné algoritmy v rozvrhoch

V tejto časti si pozrieme ako boli využité evolučné algoritmy pri hľadaní univerzitných rozvrhov. Dôraz budeme klásť na metódy v publikáciach, kde sa zložitosť hľadaného problému podobá našej ([7],[8],[9]) alebo ponúka zaujímavé myšlienky a prístupy[11].

3.4.1 Kódovanie rozvrhu v genotype

Pri zostrojovaní evolučného algoritmu jednou z prvých a najdôležitejších vecí je určenie si spôsobu reprezentácie jedincov v populácií - genotypov. Nad ge-

[4] čo najviac slabých obmedzení bolo porušených

notypmi sa potom definujú operátory rekombinácie a mutácie. Tie majú na starosti hľadanie nových a samozrejme lepších riešení. Operátory rekombinácie a mutácie silne závisia od spôsobu kódovania genotypu.

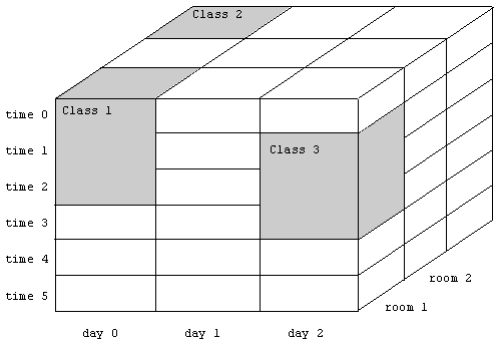
Pri hľadaní rozvrhov pomocou evolučných algoritmov predstavujú genotypy v populácií nejakým spôsobom zakódované rozvrhy. Sú dva hlavné spôsoby kódovania rozvrhov do genotypov: priamy a nepriamy.

Nepriamy spôsob kóduje genotypy vo forme inštrukcií, z ktorých deterministickým algoritmom skonštruuje rozvrh. Príkladom môže byť práca Rossi-Doria a spol.[10] riešiaca UCTP, kde genotypy obsahovali permutácie všetkých udalostí s pridelenými vyučovacími hodinami a deterministický algoritmus na základe poradia v permutácií a pridelenej vyučovacej hodiny pridelil udalosti miestnosť.

Priamy spôsob zvyčajne reprezentuje genotyp ako vektor udalostí, ku ktorým sú pridelené vyučovacie hodiny a miestnosti určujúce kde a kedy majú udalosti prebiehať. Okrem tohto spôsobu sa ešte zvykne používať reprezentácia pomocou matice vyučovacích hodín a miestností, do ktorej sú vkladané udalosti. Priamu reprezentáciu využívajú napríklad práce [7],[8],[9],[11] a pôvodný generátor rozvrhov, ktorý sa snažíme vylepšiť v tejto práci.

Výhodou priamej reprezentácie je, že narozdiel od nepriamej netreba z genotypu už vytvárať rozvrh a rozvrh je priamo prístupný pre operátory evolučného algoritmu. Nevýhodou priamej reprezentácie je, že operátory rekombinácie a mutácie takmer vždy vytvoria neuskutočniteľný rozvrh, ktorý treba nejaký spôsobom opravovať. Napriek tomu sa ďalej budeme venovať najmä riešeniam využívajúcim priamu reprezentáciu.

Práca od Sigl a spol.[8] pracuje s lekciami trvajúcimi rôzne veľa vyučujúcich hodín, pričom môžu byť umiestnené do viacerých miestností. Lekcie tam môže navštevovať jedna a viac študijných skupín a vyučovať ju môže jeden alebo viac učiteľov. Učitelia a skupiny sú pridelené k lekciám. Genotyp je tu reprezentovaný troma 3D štruktúrami pre miestnosti, študijné skupiny a učiteľov, kde x-ová súradnica predstavuje deň, y-nová vyučovaciu hodinu a z-ová v jednej štruktúre predstavuje miestnosti, v druhej študijné skupiny a v tretej učiteľov. Udalosti po pridelení vyučovacích hodín a miestností vkladané do týchto štruktúr tak aby bola na mieste určenom jej dňom, hodinou a miestnosťou, študijnou skupinou alebo učiteľom.



Obr. 3.1: 3D štruktúra Sigl a spol.[8]

Deb a spol.[7] podobne ako Sigl a spol. v [8] pracuje s lekciami s rôznym počtom vyučovacích hodín s možnosťou umiestnenia do viac miestností. Deb a spol. pracujú dokonca ešte so zložitejším problémom. Genotypy sú tvorené vektormi vyučovacích hodín a miestností, kde i-ta pozícia vo vektore predstavuje i-tu lekciu. Tento spôsob je všeobecne využívaný a využíva ho aj Peachter a spol. v [9] hoci oni sa nezaoberajú prideľovaním miestností k lekciám, ale lekcia má dané v akej miestnosti sa má vyučovať.

Lewis a Peachter v [11] pri riešení UCTP reprezentujú genotyp vo forme matice vyučovacia hodina x miestnosť, kam vkladá lekcie.

3.4.2 Inicializácia populácie

Po zadefinovaní si spôsobu kódovania genotypov a pred spustením evolučného algoritmu si treba zadefinovať ako veľká má byť populácia a aké genotypy ju budú na začiatku tvoriť. Existujú dva prístupy: prvý prístup vygeneruje genotypy v populácií náhodne, druhý vytvára genotypy v populácií použitím heuristík. Druhý prístup väčšinou lekcie vzostupne utriedi podľa počtu možností koľkými ich možno zaradiť do rozvrhu. Snaží sa tak na začiatku inicializovať populáciu uskutočniteľnými rozvrhmi a ak to nejde, tak aspoň rozvrhmi, ktoré majú blízko k uskutočniteľnému. Zástancovia prvého prístupu argumentujú, že druhý prístup spôsobí zameranie sa iba na jednu časť prehľadávaného priestoru a neumožní hľadanie v iných, kde môže byť najlepšie riešenie. Pri mnohých iných optimalizačných problémoch sa totiž ukázalo, že je lepšie na začiatku inicializovať populáciu náhodne. Zástancovia druhého prístupu tvrdia, že prvý prístup pri zložitých problémov ako je tvorba rozvrhov dlho hľadá v neužitočných regiónoch prehľadávacieho priestoru a vďaka istej konvergencií algoritmu sa uskutočniteľné a dobré rozvrhy vôbec nemusí podariť nájsť. Deb a spol v [7] uvádza na obhajobu druhého prístupu ilustračný príklad: Uvažujme, že máme školu, v ktorej sa učí 5dní v týždni po 7 vyučovacích hodinách. Školá má 10 miestností, takže ak chceme lekciu zaradiť do rozvrhu máme 350 možností. Uvažujme, že lekcia sa môže učiť len vo fyzikálnom laboratóriu. To nám počet možností zredukuje na 35. Nech lekcia trvá dve hodiny, t.j. nemožno ju nasadiť poslednú hodinu - možností je 30. Uvažujme, že lekciu učí externista, ktorý nemôže učiť skôr ako 6-tu hodinu - počet možností klesol na 6. Ak by sa táto lekcia do rozvrhu umiestňovala ako posledná, takmer určite by spôsobila porušenie nejakého silného obmedzenia. V ďalšom si ukážeme 2 príklady inicializácie populácie využitím heuristického prístupu zoraďovania lekcií.

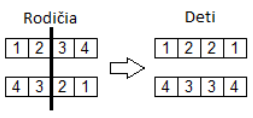
Deb a spol. v [7] zoraďuje lekcie nasledujúcim spôsobom: 1.Lekcie si najprv utriedi vzostupne podľa počtu možností, koľkými ich možno zaradiť do rozvrhu. 2.Ak lekcie majú rovnaký počet možností zoradí ich zostupne

podľa ich trvania(počtu za sebou idúcich vyučovacích hodín, ktoré sa majú učiť). 3.Ak sú stále rovnaké, uprednostní lekcie s menším počtom miestností, kde ich možno umiestniť. 4.Ak sú stále rovnaké, prednosť dostane lekcia, ktorú navštevuje viac krúžkov (odborov). 5. Pri rovnakosti vo všetkých 4 bodoch dáva prednosť lekciám patriacich do "split-classes"špecifických pre jeho problém. 6. Prednosť dostávajú lekcie, ktoré potrebujú viac miest na sedenie.

Lewis a Peachter v [11] pri riešení UCTP inicializujú populáciu iba uskutočniteľnými rozvrhmi. Ku každej lekcii na začiatku priradia množinu voľných okienok v rozvrhu, kam ich možno zaradiť. Okienko je konkrétna vyučovacia hodina v týždni v konkrétnej miestnosti. Okienko je voľné, ak v danú hodinu v miestnosti neprebieha žiadna lekcia. 1.Algoritmus zoradí lekcie vzostupne podľa počtu voľných okienok. 2.Zoberie lekciu s najnižším počtom voľných okienok - ak ich je viac náhodne jednu vyberie. 3.Pre každé okienko spočíta počet lekcií, ktoré ho obsahujú ako svoje voľné okienko a počet lekcií, ktoré sú zaradené do rozvrhu v túto vyučovaciu hodinu. 4.Vyberie okienko s minimálnym počtom lekcií z 3 - ak ich je viac, náhodne jedno vyberie a zaradí lekciu do toho okienka. 5.Vybrané okienko vymaže z okienok lekcií v zozname a lekciu vymaže zo zoznamu lekcií, ktoré treba zaradiť do rozvrhu. Pri umiestňovaní lekcií do rozvrhy sa opakujú kroky 1-5, pričom ak algoritmus zistí, že nemá lekciu kam zaradiť - spustí tvorbu rozvrhu odznova. Pri riešení zadaných inštancií UCTP sa autorom nestalo, že by program uviazol v cykle.

3.4.3 Rekombinácia a mutácia

Operátor rekombinácie z genotypov populácie (rodičov) vytvára kombináciou ich vlastností nové genotypy (potomkov). Zvyčajne z dvoch rodičov vytvára dvoch potomkov. Operátor mutácie vytvára z jedného genotypu zmenením nejakej jednej alebo viacerých vlastností nový genotyp. Operátor selekcie a rekombinácie tlačí riešenie do lokálneho optima, zatiaľ čo mutácia objavuje nové miesta v prehľadávanom priestore riešení a zabezpečuje jeho spojitosť - každé(aspoň uskutočniteľné) riešenie bolo možné algoritmom nájsť. Pri priamom spôsobe reprezentácie genotypov je klasické kríženie využívané v genetických algoritmoch[5] veľmi nevhodné, pretože takmer zaručene produkuje nevhodné - neuskutočniteľné rozvrhy. Obrázok 3.2 ilustruje ako by dopadlo tradičné kríženie dvoch ge-



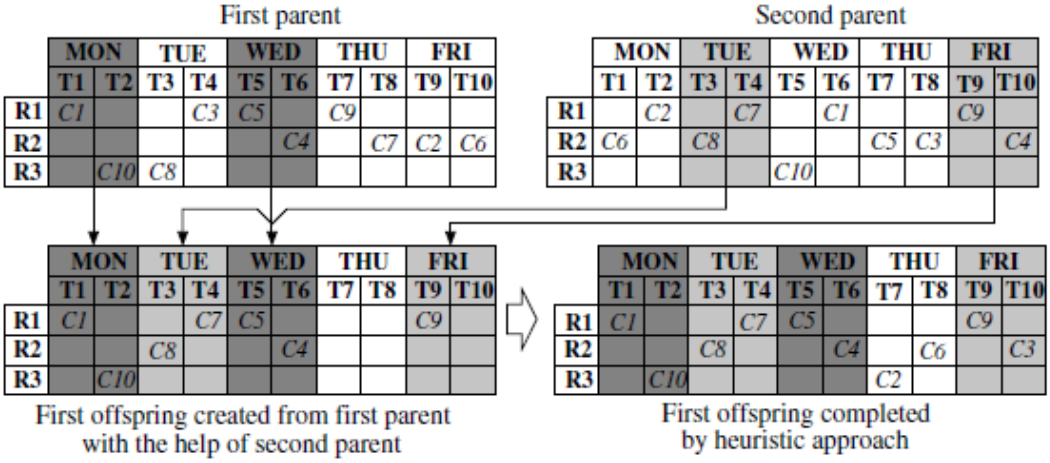
Obr. 3.2: Tradičné kríženie

[5]Podtrieda evolučných algoritmov. Všetky evolučné algoritmy v tejto práci sú genetické ak nie je uvedené ináč.

notypov - čísla predstavujú vyučovacie ho-
diny alebo miestnosti pridelené na pozície predstavujúce udalosti. Existujú 2 spôsoby ako tieto nepríjemnosti odstrániť. Prvý na novo-vzniknuté jedince aplikuje akýsi opravujúci operátor, druhý sa snaží vymýšľať rekombinácie schopné viac menej alebo striktne zachovávať silné obmedzenia a zabezpečovať tak uskutočniteľnosť. Nakoľko prvý spôsob je časovo oveľa náročnejší, vývoj sa uberá druhým spôsobom.

Sigl a spol. v [8] ako operátor rekombinácie používajú uniformné kríženie. Z dvoch rodičov vytvára potomka tak, že operátor postupne pre každú lekciu porovná či sú jej pridelené rovnaké hodnoty(hodina, skupina, učiteľ). Ak áno, prekopíruje lekciu s hodnotami do potomka. Ak nie, náhodne si vyberie rodiča a z neho prekopíruje hodnoty pridelené lekcii do potomka. Takto operátor vytvorí aj druhého potomka. Mutácia postupne pre každú lekciu vygeneruje náhodne číslo od 0 po 1 a ak je číslo menšie ako pravdepodobnosť mutácie(definovaný parameter), zmení k lekcii pridelené hodnoty (vyučovaciu hodinu, skupinu, učiteľa). Algoritmus s využitím ruletovej selekcie pre výber rodičov vykazoval veľmi slabé výsledky a autori článku ho zmenili výberom inej selekcie - pomocou turnaja a pozmenením rekombinácie. Ak rodičia mali k lekcii pridelené rôzne hodnoty, operátor vypočítal, ktoré pridelené hodnoty z rodičov by spôsobili porušenie viacerých obmedzení a vybral hodnoty toho rodiča, ktoý spôsobil menšiu škodu. Inovovaný algoritmus vykazoval oveľa lepšie výsledky.

Lewis a Peachter sa v [11] zaoberali hľadaním a testovaním rôznych operátorov rekombinácie pre UCTP, ktoré by po aplikácií zachovávali uskutočniteľnosť rozvrhov. Vo svojej práci používali dvoj-fázový algoritmus, ktorý najprv naplnil populáciu uskutočniteľnými rozvrhmi a následne nad populáciou spustil evolučný algoritmus, ktorý strážil aby na žiadnom rozvrhu neboli porušené silné obmedzenia a teda ostali uskutočniteľné. Spolu otestovali 5 operátorov rekombinácie: sektorový, denný, študentský, konfliktný a náhodný. Všetky rekombinácie z dvoch rodičov produkujú dvoch potomkov, pričom prvý potomok je najprv naklonovaný z prvého rodiča a druhý z druhého. Každá rekombinácia potom do prvého potomka vloží (zmení) hodnoty vybraných lekcií z druhého rodiča. Obdobne vloží hodnoty vybraných lekcií z druhého rodiča do prvého potomka. **Sektorová rekombinácia** hľadá v rodičovských rozvrhoch oblasti, ktoré by mohli mať málo porušení slabých obmedzení. Po nájdení nejakej z týchto oblastí v prvom rodičovi, vloží túto oblasť do druhého potomka. **Dňová rekombinácia** vychádza z predpokladu, že porušenia slabých ani silných obmedzení sa neprenášajú medzi dňami. Preto do prvého potomka vloží celé dni z druhého rodiča. **Študentská rekombinácia** vyberie náhodného študenta a v prvom rodičovi vyberie všetky lekcie, ktoré má vybraný študent zapísané. Hodnoty lekcií z prvého



Obr. 3.3: XVRA z [7]

rodiča potom prekopíruje do druhého potomka. **Konfliktná rekombinácia** v prvom rodičovi vyberie náhodne lekciu a všetky lekcie, ktoré sú s ňou v konflikte. Lekcie sú v konflikte ak majú spoločného študenta. Operátor hodnoty vybraných lekcií skopíruje do druhého potomka. **Náhodná rekombinácia** vyberie v prvom rodičovi náhodne hodnoty lekcií a tie prekopíruje do druhého rodiča. Všetky rekombinácie môžu vyprodukovať porušenie nejakého silného obmedzenia, preto pri kopírovaní lekcie z vybranej udalosti do potomka skontroluje algoritmus, či vloženie lekcie do potomka nespôsobí porušenie silného obmedzenia. Ak áno, hľadá pre danú lekciu nejakú voľnú hodinu v miestnosti, v ktorej sa lekcia mala učiť a po nájdení ju tam vloží. Po aplikovaní rekombinácie algoritmus použije s nejakou pravdepodobnosťou na novovzniknutého potomka operátor mutácie. Autori článku otestovali rekombinácie na skúškových inštanciách UCTP problému a prišli k nasledovným pozorovaniam. Najlepšie si počínala konfliktná rekombinácia nasledovaná sektorovou a študentskou. Sklamaním bola hlavne dňová rekombinácia, ktorá priniesla podobné výsledky ako náhodná, či algoritmus používajúci len mutáciu bez rekombinácie. Autori pozorujú, že konfliktná rekombinácia bola prekonaná v prípadoch, kedy riešenie mohlo vrámci časového limitu dlhšie evolvovať sektorovou a náhodnou rekombináciou. Tento jav u týchto dvoch rekombinácií vysvetľujú ich väčšou flexibilitou pri výbere lekcií v rodičoch a opierajú o fakt, že dňová rekombinácia ponúka malú flexibilitu pri výbere lekcií, a preto si počínala tak zle.

Deb a spol v [7] predstavili ako operátor rekombinácie **XVRA**(Crossover for valid resource allocation). Podobne ako všetky uvedené rekombinácie aj XVRA z dvoch rodičov vytvára dvoch potomkov. Fungovanie XVRA ilustruje

obrázok 3.4.3. Pri vytváraní potomka sa v prvom aj druhom rodičovi vyberie zopár vyučovacích hodín. Na obrázku 3.4.3 sú to v prvom rodičovi hodiny T1, T2, T5, T6 a v druhom hodiny T3, T4, T9, T10. Lekcie prebiehajúce v hodinách T1, T2, T5 a T6 sú priamo prekopírované z prvého rodiča do prvého potomka. Aj všetky lekcie v T3, T4, T9, T10, ktoré sa ešte nenachádzajú v prvom potomkovi sú prekopírované z druhého rodiča do prvého potomka. Zvyšné lekcie, ktoré neboli prokopírované zo žiadneho rodiča sú zoradené podľa heuristiky a postupne umiestňované do miestností v časoch, kedy v nich neprebieha výuka.

Deb a spol v [7] priniesli a otestovali aj 4 druhy mutácie: MFRA, MIRA, MRRA a MSIS. MFRA(Mutation for Fresh Resource Allocation) pre vybrané lekcie hľadá nejakú miestnosť a vyučovaciu hodinu, počas ktorej by bola miestnosť voľná. Nevypláca sa pri problémoch s nedostatkom voľných hodín v miestnostiach. MIRA(Mutation for Interchanging Resource Allocation) vymieňa lekcie medzi dvoma vyučovacími hodinami. MRRA(Mutation for Reshuffling Resource Allocation) najprv náhodne vyberie dve vyučovacie hodiny a následne lekcie v rovnakých miestnostiach vymieňa medzi sebou. Ak lekcia trvá viac ako hodinu, vymieňajú sa aj ďalšie hodiny v ktorých lekcia trvá. Ak v jednu vybranú hodinu je miestnosť prázdna a v druhej vybranej prebieha lekcia, lekcia sa presunie na čas kedy bola miestnosť prázdna a v jej pôvodnom čase ostane miestnosť prázdna. Ak by výmena lekcií v nejakej miestnosti spôsobila porušenie silného obmedzenia, pre danú miestnosť sa náhodne vyberú iné dve vyučovacie hodiny. MSIS(Mutation for Steering Infeasible Solution) je mutácia, ktorá sa používa len v prípadoch keď riešenie porušuje nejaké obmedzenia. Vyberie lekciu, ktorá porušuje obmedzenie a vymení ju s inou náhodne vybranou. Autori v článku testovali na svojom probléme mutácie MFRA, MIRA a MRRA. Najlepšie výsledky dosiahla MRRA.

3.4.4 Selekcia a veľkosť populácie

K evolučným algoritmom patria aj operátory selekcie. Sú to výber rodičov pre rekombináciu(kríženie) a výber novej populácie - genotypov, ktoré budú tvoriť populáciu v ďalšej generácií.

Sú používané dva spôsoby výberu rodičov(a) pre rokombináciu: ruleta a turnaj. Pri rulete sa z populácie náhodne vyberie genotyp, pričom gentotypy s lepšou mierou kvality majú väčšiu šancu, že budú vybrané. Turnaj náhodne(uniformná pravdepodobnosť) vyberie nejaký počet jedincov (zväčša 2) a vyberie najlepšieho. Evolučný algoritmus zvykne porovnateľne dobre fungovať pri použití rulety aj turnaja. Výber turnajom je oveľa menej časovo náročný ako ruleta, preto sa pri tvorbe rozvrhov používa vo väčšine

prípadov. [7], [11], [9] používajú turnaj a [8] sa vo svojej práci dopracovali k záveru, že turnaj je lepší ako ruleta.

Výber novej populácie zabezpečuje konštantnú veľkosť populácie a tlačí na zvyšovanie kvality riešení v populácií mazaním najhorších riešení. Zaužívané sú tieto spôsoby: 1.Nové genotypy, ktoré vznikli rekombináciou a mutáciou sa pridajú do populácie a z populácie sa následne vymažú najhoršie riešenia. 2.Nové genotypy po vzniknutí nahrádzajú najhoršie 3.Nová populácia nahrádza celú starú. Najmä pri treťom spôsobe existuje šanca zmazania si najlepšieho riešenia, preto sa používa elitizmus, ktorý zabezpečuje prežitie jedného alebo viacerých najlepších riešení do ďalšej generácie.

Veľkosť populácie tvorí dôležitý parameter evolučného algoritmu. Ak obsahuje príliš málo jedincov, rýchlo konverguje a prehľadáva malý priestor riešení. Ak obsahuje priveľa jedincov, okrem toho, že je časovo neefektívny, prakticky prestáva konvergovať k lepším riešeniam. Dobrý jedinec má malú šancu, že bude vybratý selekčným operátorom, pretože operátor má na výber z veľkého množstva iných priemerných riešení. Deb a spol. v [7] používajú populáciu veľkosti 50, Sigl a spol. v [8] 5120 a Lewis s Peachterom v [11] pre UCTP populáciu veľkosti 100.

Kapitola 4

Pôvodný generátor rozvrhov

Cieľom tejto práce je preskúmať možné vylepšenia prehľadávacieho algoritmu, ktorý používa pôvodný generátor rozvrhov. V tejto kapitole si preto predstavíme fungovanie pôvodného generátora a jeho prehľadávací algoritmus, ktorý používa pri hľadaní čo najlepšieho rozvrhu.

Pôvodný generátor na vstupe príma *špecifikáciu behu* algoritmu obsahujúcu vstupné údaje a parametre výpočtu, ktoré spracuje a spustí výpočet - prehľadávací algoritmus založený na evolučných algoritmoch. Ako výsledok vráti vygenerovaný rozvrh spolu so zoznamom všetkých obmedzení, ktoré rozvrh porušuje a metrikou vyjadrujúcou jeho kvalitu.

Parametre **špecifikácie behu** určujú napr. počet výpočtových vlákien, ktoré má generátor pri výpočte použiť, vstupné semeno pre generátor náhodných čísel, ktorý bude používať, aby výpočet mohol byť opakovateľný s rovnakými výsledkami, miesta kam má generátor zapisovať logy a iné výstupy a miesta odkiaľ má brať vstupné údaje. Špecifikácia behu v sebe obsahuje aj vstupné údaje, parametre evolučného algoritmu a parametre pre evaluátor rozvrhov. **Vstupné údaje** získava generátor vo forme údajových objektov alebo XML súborov, ktoré spracúva na údajové objekty. Predstavujú údaje z univerzity na základe ktorých má generátor vygenerovať rozvrh. Bližšie sú popísané v prvej kapitole. **Parametre evolučného algoritmu** nastavujú evolučný algoritmus, ktorý používa generátor rozvrhov pri hľadaní rozvrhu pre univerzitu. Nastavujú veľkosť populácie, maximálny počet generácií, ktoré má algoritmus vystriedať, počet elít, ktoré má zachovávať, pravdepodobnosť rekombinácie dvoch rodičov, pravdepodobnosť mutácie novovzniknutého potomka a iné parametre evolučného algoritmu. **Parametre pre evaluátor** rozvrhov obsahujú váhy pokút, akými majú byť pokutované porušenia jednotlivých obmedzení ale aj doporučený maximálny počet vyučovacích hodín, ktoré má učiteľ učiť a iné nastavenia pre evaluátor, ktoré sa majú použiť pri rátaní miery kvality nájdených riešení.

Generátor na základe špecifikácie behu spúšťa prehľadávací algoritmus. Prehľadávací algoritmus možno spustiť na nezávislom výpočtovom vlákne a možno ho v ľubovolnej chvíli pozastaviť a znovu spustiť alebo prerušiť. Možno ho dokonca nastaviť, aby každých 10 generácií ukladal čiastočné riešenia a v prípade prerušenia výpočtu i celého programu úmyselne alebo neúmyselne[1] mohol znovu pokračovať vo výpočte, kde pred prerušením približne skončil. Počas výpočtu sa tiež možno priebežne dotazovať na predbežný výsledok prehľadávacieho algoritmu obsahujúci zatiaľ najlepšie nájdené riešenie.

Výsledok prehľadávacieho algoritmu obsahuje najlepší nájdený (vygenerovaný) rozvrh, metriky, popisujúce kvalitu nájdeného rozvrhu a zoznam nevyriešených konfliktov - zoznam porušení jednotlivých obmedzení kladených na rozvrh. Metriky obsahujú počty porušení jednotlivých obmedzení, celkovú mieru kvality riešenia (fitness) ale aj iné kvality, ako počet využitých miestností, rozptyl vyučovania pre učiteľov - učia príliš veľa dní v týždni[2], a celkový stres rozvrhu pre učiteľov tvorený tým, že v jeden deň musia oducčiť príliš veľa lekcií. Zoznam nevyriešených konfliktov opisuje detaily jednotlivých porušení obmedzení umožňujúce identifikovať miesto porušenia v rozvrhu. Napríklad pri porušení obmedzenia, že učiteľ nemá učiť naraz dve rôzne lekcie vypíše lekcie, ktoré sú učené jedným alebo viacerými rovnakými učiteľmi a vyučovaciu hodinu, kedy má (majú) naraz učiť uvedené lekcie.

Niektoré univerzity chcú aby niktoré ich lekcie mali niektoré hodnoty pevne dané - fixované a mali tak napevno pridelenú vyučovaciu hodinu, kedy majú začínať, skupinu miestností kde sa majú učiť, skupinu učiteľov, ktorá ich má vyučovať alebo offset určujúci ktorý týždeň majú začínať ak sú vyučované každý i-ty týždeň. Fixovať hodnoty lekciám možno pred, ale aj po spustení prehľadávacieho algoritmu.

4.1 Prehľadávací algoritmus

Prehľadávací algoritmus je prakticky evolučný - genetický algoritmus. Na základe údajov zo špecifikácie behu inicializuje populáciu genotypov, ktoré ohodnotí pomocou evaluátora a priradí im ich mieru kvality. Na základe miery kvality usporiada genotypy v populácií od najlepšieho po najhorší a 4(ak nie je určené inak) najlepšie genotypy (elity) skopíruje do novej populácie. Algoritmus sa snaží ďalej zaplniť novú populáciu genotypmi, tak aby obsahovala toľko genotypov koľko má populácia. Parametrami evolučného algoritmu sa

[1] napr. najpopulárnejší operačný systém sa reštartuje, lebo práve nainštaloval nové aktualizácie

[2] Učitelia majú radi, ak vrámci týždňa majú voľné dni, kedy sa môžu napríklad naplno venovať výskumu.

dá určiť aj koľko genotypov populácie(okrem elít) má byť v novej generácií nahradených. Počet genotypov, ktoré majú byť nahradené určuje počet genotypov, čo sa má zachovať (okrem elít) do novej generácie. Najlepšie genotypy v počte genotypov, ktoré sa majú zachovať sa skopírujú do novej populácie. Ak nie je určené inak, prehľadávací algoritmus nahrádza celú populáciu okrem elít novou - žiaden ďalší genotyp sa neskopíruje do novej populácie. Evolučný algoritmus môže vytvoriť niektoré jedince v novej populácií tak, že skopíruje niekoľko genotypov do novej populácie a aplikuje na ne lokálne prehľadávanie (horolezecký algoritmus), ktorým sa ich snaží vylepšiť. Počet skopírovaných a následne vylepšovaných genotypov lokálnym prehľadávaním je určený v parametroch evolučného algoritmu. Zvyšok novej populácie algoritmus zapĺňa nasledovne: Použitím operátora selekcie algoritmus vyberie z celej populácie 2 genotypy - rodičov. Náhodným generátorom vygeneruje reálne číslo v intervale <0;1>. Ak je číslo menšie ako parametrami daná pravdepodobnosť rekombinácie, klonovaním vytvorí z rodičov dvoch potomkov. Ak je vygenerované číslo väčšie, vytvorí z rodičov dvoch potomkov použitím operátora rekombinácie. Na každého potomka v parametroch definovanou pravdepodobnosťou mutácie aplikuje operátor mutácie. Vzniknutí potomkovia potom algoritmus vloží do novej populácie. Na novovytvorené genotypy vzniknuté rekombináciou, mutáciou alebo lokálnym prehľadávaním algoritmus aplikuje evaluáciu a pridelím im ich mieru kvality. Po zaplnení novej populácie nahradí nová populácia nahradí pôvodnú populáciu - populácia prešla do novej generácie. Populácia sa znovu utriedi od najlepšieho riešenia a zopakuje sa celý ďalej uvedený postup. Populácia sa takto obmieňa, kým algoritmus nie je zvonku prerušený alebo sa nevystrieda stanovený maximálny počet generácií. Ďalšie podčasti tejto časti obsahujú popis genotypu, inicializáciu genotypu, operátory nad genotypmi ako rekombinácia, mutácia a lokálne prehľadávanie a popis evaluácie genotypov.

4.1.1 Genotyp a jeho inicializácia

Genotyp pozostáva zo zakódovaného riešenia a jemu prislúchajúcej miery kvality určujúcej jeho kvalitu.

Zakódované riešenie obsahuje zoznamy, ktoré pre každú lekciu obsahujú začiatočnú vyučovaciu hodinu určujúcu, na kedy je lekcia zaradená do rozvrhu, skupinu miestností, kde sa má učiť, skupinu učiteľov, ktorá ju má vyučovať a perióda určujúca, ktorý týždeň sa má začať vyučovať ak má periodicitu väčšiu ako 1 - t.j. má sa učiť každý i-ty týždeň. Tieto zoznamy sú zakódované vo forme polí, kde indexy poľa predstavujú identifikátory lekcií.

Inicializácia genotypu znamená vytvorenie nového (zakódovaného) riešenia, ktoré treba ohodnotiť evaluátorom a priradiť mu mieru kvality. Pri

vytvorení nového riešenia sa na začiatku lekciám priradia ich zafixované hodnoty(ak nejaké mali zafixované). Následne sa zoberú skupiny učiteľov, ktoré majú presne stanovený počet, lekcií, ktoré majú vyučovať. Pre každú skupinu učiteľov spomedzi vybraných skupín bude náhodne vybraný počet lekcií typu a kurzu skupiny učiteľa, ktoré ešte nemajú pridelenú skupinu učiteľov, ktorá by ich učila. Skupina učiteľov je následne pridelená k týmto lekciám. Môže sa stať, že nie je dosť voľných lekcií pre učiteľa, vtedy sa priradia všetky čo ostávajú aj keď ich bude málo. Generátor samozrejme ráta s už zafixovanými skupinami učiteľov pre lekcie. Algoritmus ďalej ku každej lekcii snaží v cykle priradiť skupinu učiteľov, skupinu miestností a následne vyučovaciu hodinu, kedy má začať. Priradenie skupiny učiteľov vyzerá nasledovne: ak už má lekcia pridelenú skupinu učiteľov algoritmus ide lekcii priraďovať miestnosť. Inak spraví zo skupín učiteľov náhodnú permutáciu a postupne skúša skupiny učiteľov z permutácie. Ak skupina učiteľov už nemôže učiť túto lekciu(prekročný maximálny počet lekcií, ktoré môže učiť alebo už učí predpísaný počet lekcií) ide skúšať lekcii priradiť ďalšiu skupinu učiteľov z permutácie. Ak nie je vhodné priradiť lekcii žiadnu skupinu učiteľov, bude jej priradená posledná skupina učiteľov v permutácií. Po priradení učiteľa sa algoritmus snaží lekcii priradiť skupinu miestností. Najprv sa určí spomedzi ktorých skupín miestností má algoritmus vyberať. Ak lekcia má nastavené, že jej môžu byť pridelená iba miestnosť z preferovaných miestností, algoritmus vyberá iba z permutácie preferovaných miestností. V opačnom prípade vyberá najprv z permutácie preferovaných miestností a neskôr z permutácie všetkých miestností. Ak miestnosti vybranej skupiny miestností obsahujú miestnosti, ktoré nevedie fakulta ponúkajúca kurz lekcie, s pravdepodobnosťou ovplyvnenou pomerom miestností z iných fakúlt a všetkých miestností sa vyberie iná skupina miestností v permutácií. Po výbere miestnosti sa algoritmus snaží lekcii prideliť vyučovaciu hodinu, kedy má začať. Vytvorí si permutáciu všetkých vyučovacích hodín, z ktorých následne sa snaží vybrať hodinu, ktorá by bola vhodná. Algoritmus vyberá inú vyučovaciu hodinu ak nastane aspoň jeden z nasledovných prípadov: *a)* v danú vyučovaciu hodinu je zakázané učiť *b)* nejaký učiteľ z vybranej skupiny učiteľov nemôže v danom čase učiť *c)* lekcia je vyučovaná paralelne s inou lekciou rovnakého kurzu a typu hoc je to pre daný kurz a typ zakázané *d)* vybraná miestnosť je rezervované pre inú fakultu alebo externe *e)* je prekročený maximálny počet lekcií za deň pre daný kurz a lekcie rovnakého typu *f)* lekcia nie je zarovnaná. Ak sa nepodarí nájsť pre lekciu vyučovacia hodina, algoritmus skúša postupne vybrať lekcii inú skupinu miestností a pre ňu vyučovaciu hodinu. Ak sa nepodarí pre žiadnu skupinu miestností nájsť vyučovacia hodina, algoritmu skúsi prideliť lekcii inú skupinu učiteľov, po ktorej znovu vyberá skupinu miestností a vyučovaciu hodinu. Ak algoritmus nenájde pre lekciu hodnoty, pridelí

jej posledné pridelené.

4.1.2 Operátor rekombinácie

Operátor rekombinácie vytvorí z dvoch rodičovských genotypov dva-krát jedného potomka - dohromady 2 potomkov. Na vytvorenie potomka používa uniformné kríženie. Vytvorenie potomka vyzerá tak, že sa doň skopírujú údaje z prvého rodiča. Operátor si potom zvolí náhodne *pravdepodobnosť prenosu* s akou sa budú prenášať údaje z kurzov druhého rodiča do potomka. Pre každý kurz operátor vygeneruje reálne číslo v intervale <0;1> a ak je číslo väčšie ako pravdepodobnosť prenosu, prenesie z druhého rodiča do potomka všetky lekcie, ktorých skupiny učiteľov pridelené k lekciám v prvom aj druhom rodičovi nemajú určený presný stanovený počet lekcií, ktoré majú učiť.

4.1.3 Operátor mutácie

Operátor mutácie pri mutácií genotypu používa niekoľko typov mutácií: vytvorenie nového genotypu, zmena vyučovacej hodiny, zmena skupiny miestností, zmena skupiny učiteľov, výmena vyučovacích hodín a skupín miestností medzi lekciami, výmena skupín miestností medzi lekciami, výmena vyučovacích hodín medzi lekciami, výmena skupín učiteľov vrámci kurzu a posun vyučovacej hodiny.

Pre každý z typov mutácie je v parametroch evolučného algoritmu definovaná pravdepodobnosť s akou sa má daný typ mutácie použiť.

Na začiatku mutácie sa s danou pravdepodobnosťou namiesto pôvodného genotypu inicializuje nový genotyp. Ak sa tak stane, operátor mutácie skončí. V opačnom prípade vypočíta počet opakovaní s akými bude skúšať zvyšné typy mutácií. Počet opakovaní sa určí náhodne na základe parametru určujúceho s akou pravdepodobnosťou sa má opakovať mutácia, aktuálnej generácie populácie(čím neskoršia generácia, tým vyššia šanca opakovania mutácií) a parametrov určujúcich maximálny a minimálny počet opakovaní. Operátor potom v cykle zopakuje nasledovnú vec: Pre každý typ mutácie okrem vytvorenia nového genotypu aplikuje na genotyp s pravdepodobnosťou prislúchajúcou typu mutácie daný typ mutácie. V nasledujúcom si popíšeme jednotlivé typy mutácií:

Zmena vyučovacej hodiny: Operátor náhodne vyberie lekciu spomedzi všetkých lekcií. Ak má lekcia fixovaný začiatok výučby, náhodne vyberie ďalšiu lekciu. Toto zopakuje maximálne 20-krát. Ak nenájde lekciu s nefixovaným začiatkom výučby, mutácia nevykoná nič. V opačnom prípade sa pre lekciu pokúsi nájsť začiatočnú vyučovaciu hodinu rovnako ako pri vytváraní zakódovaného riešenia. Ak sa nepodarí lekcii priradiť vyučovacia hodina,

kedy má začať, ani po 20 pokusoch, mutácia nevykoná nič. V opačnom prípade zmení lekcii pridelený začiatok výučby na ten nájdený mutáciou.
zmena skupiny miestností: Operátor hľadá náhodne lekciu spomedzi lekcií, ktorá nemá zafixovanú skupinu miestností. Ak sa mu nepodarí takú nájsť ani po 20 pokusoch, nevykoná nič. V opačnom prípade sa snaží vybranej lekcii náhodne prideliť skupinu miestností spomedzi jej preferovaných miestností a všetkých miestností. Ak lekcia môže byť nasadená len do jednej z preferovaných skupín miestností, operátor vyberá len medzi ňou preferovanými. V opačnom prípade hľadá aj v ostatných skupinách miestností, pričom s polovičnou pravdepodobnosťou vyberie skupinu miestností z preferovaných. Ak vybraná skupina miestností obsahuje miestnosť spravovanú inou fakultou ako je fakulta kurzu lekcie alebo je v danom čase jedna z jej miestností rezervovaná externe alebo inou fakultou, operátor vyberie náhodne inú skupinu miestností. Operátor sa takto pokúša nájsť vhodnú skupinu miestností maximálne 20-krát. V prípade neúspechu nevykoná zmenu.
zmena skupiny učiteľov: Podobne ako pre zmenu skupiny miestností vyberie operátor náhodne lekciu bez zafixovanej skupiny učiteľov. Spomedzi všetkých skupín učiteľov, ktorí môžu vyučovať lekciu vyberie náhodne skupinu učiteľov. Ak má skupina určený presný počet lekcií, ktoré má učiť, operátor vyberie náhodne novú skupinu. Takto sa pokúsi nájsť vyhovujúcu skupinu učiteľov maximálne 20-krát. V prípade neúspechu nevykoná žiadnu zmenu, inak pridelí lekcii nájdenú skupinu učiteľov.
výmena vyučovacích hodín a skupín miestností medzi lekciami: Vyberie N náhodných lekcií a na základe jej permutácie vytvorí vo vybraných lekciách dvojice vrámci ktorých sa snaží vymeniť vyučovacie hodiny a skupiny miestností medzi lekciami. Ak by však pri výmene vyučovacích hodín a miestností mala byť pridelená jednej z lekcii skupina miestností, ktorá sa nenachádza medzi jej preferovanými skupinami miestností hoc vyžaduje len miestnosť z preferovanej skupiny, mutácia nevykoná žiadnu zmenu pre žiadnu dvojicu lekcií. Podobne aj keby jeden z učiteľov po výmene vyučovacích hodín nemohol učiť. Ak nenastane žiadna z komplikácií, vymenia sa vyučovacie hodiny a skupinu miestností medzi dvojicami lekcií.
výmena skupín miestností medzi lekciami: Prebieha podobne ako výmena vyučovacích hodín a skupín miestností medzi lekciami, ale vymieňajú sa len skupiny miestností.
výmena vyučovacích hodín medzi lekciami: Prebieha podobne ako výmena vyučovacích hodín a skupín miestností medzi lekciami, ale vymieňajú sa len vyučovacie hodiny.
výmena skupín učiteľov vrámci kurzu: Náhodne vyberie kurz spomedzi kurzov. Ak kurz obsahuje maximálne jednu lekciu, skončí tento typ mutácie a nevykoná žiadnu zmenu. V opačnom prípade vyberie z kurzu náhodne lekcie

spomedzi jeho lekcií. Ak má lekcia zafixovanú skupinu učiteľov, algoritmus opakuje maximálne niekoľko krát výber nového kurzu a novej lekcie. Ak taký kurz s lekciou nenájde, nevykoná žiadnu zmenu. V opačnom prípade vyberá z daného kurzu postupne lekcie rovnakého typu, ktoré nemajú zafixovanú skupinu učiteľov a skúša, či by výmenou skupín učiteľov medzi lekciami neučil nejaký učiteľ spomedzi skupín učiteľov v čase, kedy nemôže učiť. Ak žiaden učiteľ nemá problém, vymení lekciám učiteľov a ukončí tento typ mutácie.
posun vyučovacej hodiny: Podobne náhodne ako zmena skupiny miestností vyberie náhodne lekciu s nezafixovanou vyučovacou hodinou a pokúsi sa ju posunúť o vyučovaciu hodinu skôr alebo neskôr v rozvrhu. Pri posune kontroluje, či po posune lekcia nezačína skôr ako začiatok vyučovania alebo sa neučí po poslednej vyučovacej hodine dňa alebo nejaký učiteľ, čo ju učí nemôže v danú hodinu učiť alebo jedna z pridelených miestností je rezervovaná inou fakultou alebo externe.

Kapitola 5

Vlastná práca a výskum

Táto kapitola popisuje vlastnú prácu. Začiatok predstavuje údaje, na ktoré budú slúžiť ako vstupné údaje pre generátor a na základe ktorých má generátor generovať rozvrhy. Generátor a jeho zmeny budeme testovať na týchto údajoch.

5.1 Testovacie údaje

Táto práca svoje výsledky testuje na dátach získaných z univerzít používajúcich informačný systém firmy. Konkrétne budeme používať 5 sád údajov získaných z rôznych univerzít. Nasledujúca tabuľka sa pokúša charakterizovať vstupné údaje jednotlivých sád. Všetky nižšie uvedené testy na dátach boli namerané z piatich rôznych behov generátora používajúceho vždy iný inicializačný prvok pre ním používaný generátor náhodnosti.

	Sada 1	Sada 2	Sada 3	Sada 4	Sada 5
Počet lekcií	180	113	349	544	779
Poček kurzov	20	20	99	544	195
Počty lekcií v kurzoch	(av=9, min=5 max=18)	(av=6, min=2 max=12)	(av=4, min=1 max=20)	(av=1, min=1 max=1)	(av=4, min=3 max=4)
študentské skupiny	nie	nie	nie	58	nie
Štud.odbory(počet)	nie	nie	áno (16)	áno (58)	áno(353)
Používa odbory ich v lekciách	nie	nie	nie	áno	nie
kolízie kurzov	nie	nie	áno	nie	nie
kurzy požadujú miestnosti	180	108	nie	0	168
počet lekcí s requiredRoomOnly	0	0	0	0	0
požiadavky učiteľov (časové)	áno	nie	áno	áno	áno
má maximum lekcií na deň	áno	áno	nie	nie	áno
Periodcita – počty lekcií s per.	1(178), 2(2)	1(90), 2(23)	1(268), 2(81)	1(529), 2(1), 3(14)	nie
zarovnané lekcie	nie	nie	nie	nie	nie
počet kolíznych dvojíc	32	17	30353	29308	563

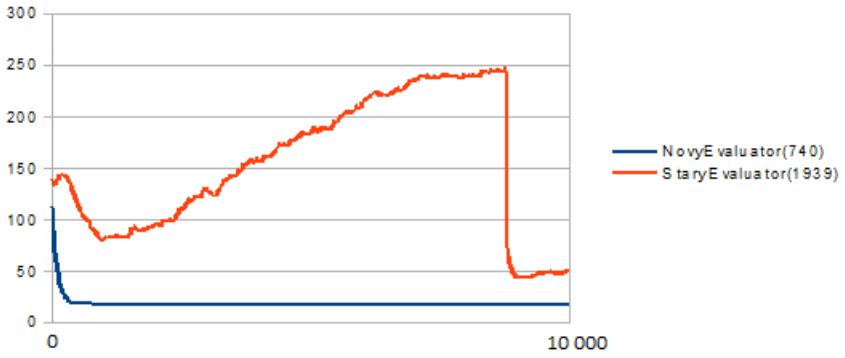
Obr. 5.1: popis a porovnanie dát z jednotlivých fakúlt

5.2 Nový evaluátor

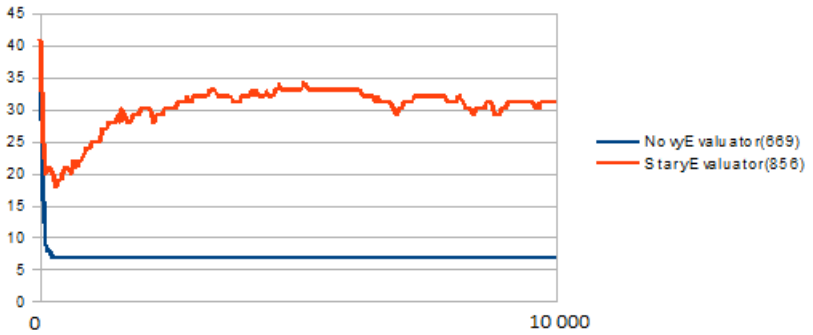
Pôvodný generátor obsahuje evaluátor, ktorý prideľuje genotypom mieru kvality v závislosti od generácie, v ktorej sa nachádza populácia a teda miera kvality toho istého genotypu môže počas rôznych generácií nadobúdať rozdielne hodnoty. Pôvodný evaluátor tiež nereflektuje zmenu priorít niektorých obmedzení kladených na rozvrh a pridanie nových obmedzení. Pôvodný evaluátor napríklad zaraďoval porušenie 3. a 4. obmedzenia len medzi obmedzenia s vysokou prioritou. Pre potreby tejto práce však vznikla potreba vytvoriť nový evaluátor, ktorý by vedel nezávisle od generácie presne určiť kvalitu riešení. V prípade testovania iných evaluátorov a metód môže tento evaluátor nezávisle určiť kvalitu rozvrhu. Vytvorený nový evaluátor v sebe zahŕňa nové požiadavky zadané koncovými užívateľmi generátora a tiež implementuje inovatívne nápady. Nový evaluátor hodnotí rozvrhy na základe definície problému a počtu porušení jednotlivých obmedzení definovaných v 2. kapitole. Mieru kvality rozvrhov určuje druhý evaluátor podľa počtu porušení jednotlivých druhov obmedzení - s najvyššou, s vysokou, so strednou a nízkou prioritou. Na základe tejto miery rozvrh A je lepší od B ak porušuje menej-krát obmedzenia s najvyššou prioritou. Ak A a B porušujú rovnako veľa krát obmedzenia s najvyššou prioritou, potom A je lepší od B ak porušuje menej-krát obmedzenia s vysokou prioritou. Ak sa zhodujú v počte porušení obmedzení s vysokou prioritou, potom sa rozhoduje podľa obmedzení so strednou prioritou a následne nízkou.

Po vytvorení nového evaluátora vznikla otázka, ako sa bude správať pôvodný generátor, ktorý namiesto pôvodného evaluátora bude používať na hodnotenie rozvrhov nový evaluátor. Otestovali sme na spomínaných piatich sadách vstupných údajov výpočet pôvodného generátora, ktorý prestrieda 10 000 generácií a pri ohodnocovaní vrámci prehľadávacieho algoritmu používa raz nový a raz pôvodný evaluátor. Aby sa dala porovnať kvalita jednotlivých medzivýsledkov výpočtov, ktoré tvoria v každej fáze prehľadávacieho algoritmu doteraz najlepšie nájdené rozvrhy(podľa jednotlivých použitých evaluátorov), výsledné rozvrhy sú nezávisle ohodnotené novým generátorom, ktorý rovnaké riešenie vždy oohodnotí rovnako a berie do úvahy najposlednejšie požiadavky budúcich koncových užívateľov. Nasledujúce obrázky predstavujú grafy znázorňujúce počty porušení obmedzení najvyššej priority najlepších nájdených riešení počas výpočtu prehľadávajúceho algoritmu, ktorý vymenil 10000 generácií.

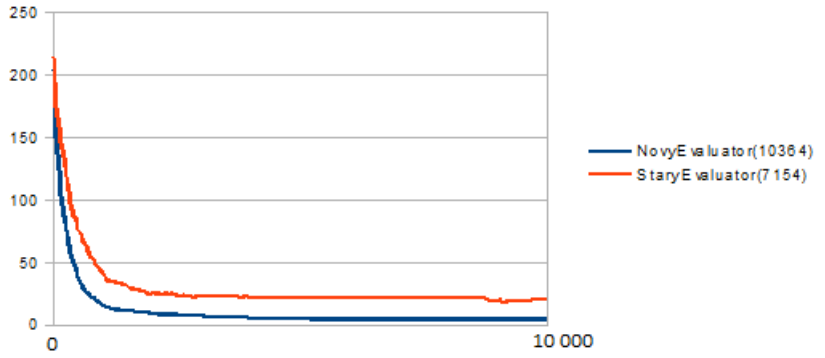
Výsledky ukázali, že použitie nového evaluátora dáva pre všetky sady lepšie výsledky a dokonca pre sady 1,2 a 4 je rýchlejší. Pre sady 3 a 5 je pomalší, ale nie až tak výrazne. Na základe porovnania budeme ďalej v práci pri skúmaní heuristík a operátorov rekombinácie a mutácie používať na evaluáciu



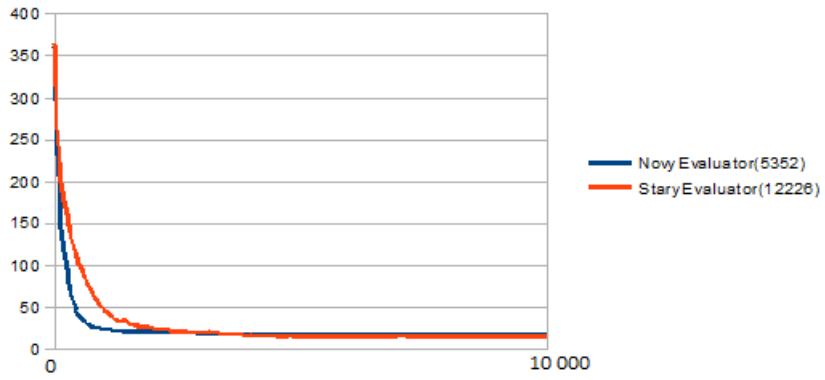
Obr. 5.2: Porovnanie nového a starého evaluátora na sade1. V zátvorke je uvedený čas výpočtu generátora v sekundách.



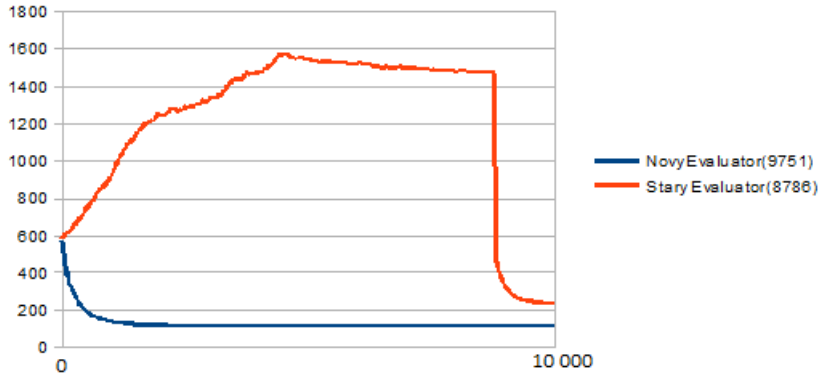
Obr. 5.3: Porovnanie nového a starého evaluátora na sade2. V zátvorke je uvedený čas výpočtu generátora v sekundách.



Obr. 5.4: Porovnanie nového a starého evaluátora na sade3. V zátvorke je uvedený čas výpočtu generátora v sekundách.



Obr. 5.5: Porovnanie nového a starého evaluátora na sade4. V zátvorke je uvedený čas výpočtu generátora v sekundách.



Obr. 5.6: Porovnanie nového a starého evaluátora na sade5. V zátvorke je uvedený čas výpočtu generátora v sekundách.

nový evaluátor.

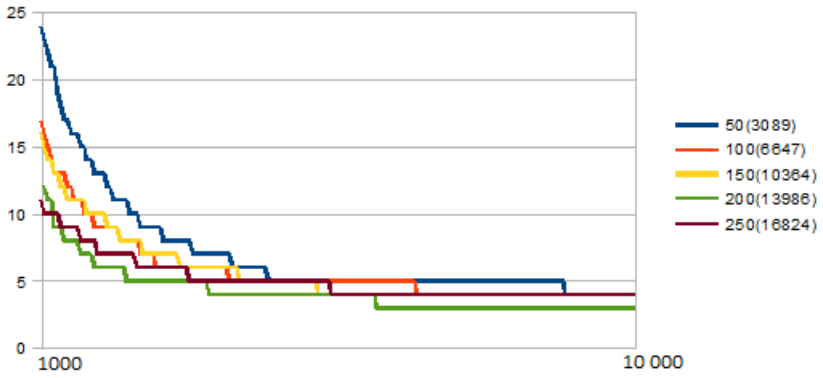
5.3 Vplyv veľkosti populácie

Tretia kapitola tejto práce spomína, aký vplyv na prehľadávací algoritmus a teda aj nájdené výsledné riešenie má veľkosť populácie. Ak populácia obsahuje príliš málo jedincov, rýchlo konverguje a algoritmus prehľadáva malý priestor riešení. Ak obsahuje priveľa jedincov, okrem toho, že je algoritmus časovo neefektívny, prakticky prestáva konvergovať k lepším riešeniam. Dobrý jedinec má malú šancu, že bude vybratý selekčným operátorom, pretože operátor má na výber z veľkého množstva iných priemerných riešení. V tejto práci sme sa rozhodli otestovať rôzne veľkosti populácií pre prehľadávací algoritmus pôvodného generátora používajúceho na evaluáciu riešení nový evaluátor. Algoritmus používajúci rôzne veľké populácie sme otestovali na všetkých piatich testovacích sadách. Algoritmus počas svojho behu vystrieda 10 000 generácií. Nasledujúce grafy ukazujú počty porušení obmedzení najvyššej priority zatiaľ najlepšieho nájdené riešenia v konkrétnej generácií populácie počas behu výpočtu. Niektoré grafy znázorňujú výpočet od 500 alebo 1000-cej generácie - ak by obsahovali výpočet od 0-tej generácie, boli by nečitateľné.

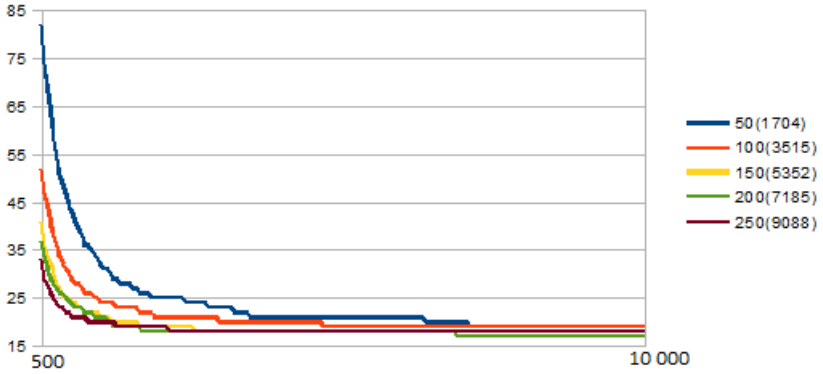
V prvých dvoch sadách veľkosť populácie prakticky neovplyvnila výsledok prehľadávacieho algoritmu. Rozdiel bol však v čase výpočtu generátora, kde prehľadávací algoritmus trval pri populácii tvorenej 250 jedincami viac

Obr. 5.7: Generátor používajúci pôvodný evolučný algoritmus s rôzne veľkými populáciami testovaný na sade1. V zátvorke je uvedený čas výpočtu generátora v sekundách.

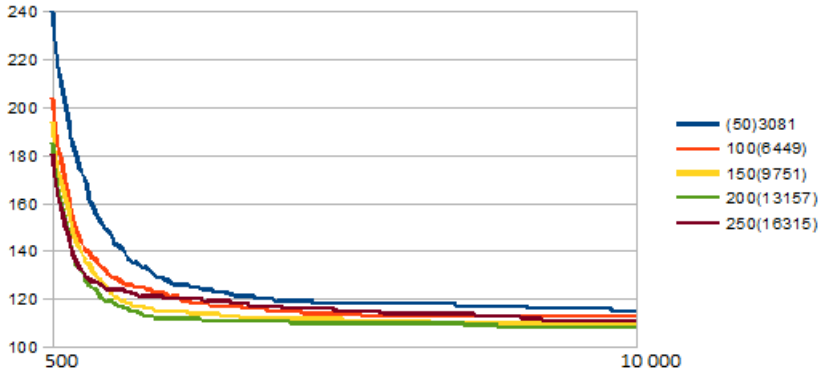
Obr. 5.8: Generátor používajúci pôvodný evolučný algoritmus s rôzne veľkými populáciami testovaný na sade2. V zátvorke je uvedený čas výpočtu generátora v sekundách.



Obr. 5.9: Generátor používajúci pôvodný evolučný algoritmus s rôzne veľkými populáciami testovaný na sade3. V zátvorke je uvedený čas výpočtu generátora v sekundách.



Obr. 5.10: Generátor používajúci pôvodný evolučný algoritmus s rôzne veľkými populáciami testovaný na sade4. V zátvorke je uvedený čas výpočtu generátora v sekundách.



Obr. 5.11: Generátor používajúci pôvodný evolučný algoritmus s rôzne veľkými populáciami testovaný na sade5. V zátvorke je uvedený čas výpočtu generátora v sekundách.

ako 5-násobok času potrebného pri populácii tvorenej len 50 jedincami. V zvyšných 3 sadách sa ako najlepšia voľba pre beh algoritmu na 10 000 generáciách ukázala populácia veľkosti 200. Algoritmus na menších populáciách trvá výrazne kratšie ako na väčších. Nevyriešenou otázkou ostáva, či by pri prestriedaní viacerých generácií algoritmus s menšou populáciou sa vyrovnal alebo bol lepší než algoritmus s väčšou ak by ich výpočet trval rovnako dlho. Ukázalo sa tiež, že populácia s 250 jedincami je už príliš veľká a dáva horšie výsledky v horšom čase.

5.4 Nový prehľadávací algoritmus - Multiple

Táto časť reflektuje výskum nového prehľadávacieho algoritmu, ktorý používa iný spôsob inicializácie genotypov ako pôvodný generátor. Nový prehľadávací evolučný algoritmus dostal pracovný názov "multiple", pretože je nastaviteľný vstupnými parametrami. Multiple umožňuje použitie dvoch typov rekombinácie(dňovú a jednoduchú - obe vyvinuté vrámci tejto práce) a novú mutáciu - inú ako používa pôvodný generátor. Ďalšie časti tejto podkapitoly obsahujú popis novej inicializácie, nových operátorov rekombinácie a mutácie.

5.4.1 Nová inicializácia

Prvý rozdiel v inicializácií genotypu - riešenia od pôvodnej pre pôvodný generátor predstavuje počiatočné pridelenie skupín učiteľov k lekciám. Nová inicializácia pridelí lekcie najprv učiteľom, ktorý majú určené, že majú učiť presný počet lekcií. Následne pridelí zvyšné lekcie. Pôvodný generátor na nepridelené lekcie postupne skúšal rôznych učiteľov, kým sa nepodarilo nájsť vhodnú kombináciu skupina učiteľov, miestnosť a začiatočná vyučovacia hodina pre miestnosť. Multiple si zjednodušuje prácu pridelením skupín učiteľov k lekciám hneď na začiatku. Multiple zabezpečuje tiež náhodný výber skupiny učiteľov pre lekciu a následnú variabilitu podobne ako pôvodný generátor. Multiple dodržiava obmedzenia, aby lekcie s fixovanými hodnotami, mali správne pridelené hodnoty. Po pridelení skupín učiteľov k lekciám sa pre každú lekciu postupne vyberá začiatočná vyučovacia hodina a miestnosť. Multiple sa snaží dodržiavať čo najviac obmedzenie týkajúce sa výberu vyučovacej hodiny pre lekciu, kedy aspoň jeden zo skupiny učiteľov nemôže v danú hodinu učiť. Využíva na to predspracovanie vstupných údajov, ktoré robí nový prehľadávací algoritmus ešte pred spustením prehľadávania. Pri predspracovaní pridelí každej skupine učiteľov množinu vyučovacích hodín, kedy môžu všetci učitelia zo skupiny začať učiť typy lekcií kurzu, ktoré majú učiť. Ak táto množina pre skupinu učiteľov, čo má učiť lekciu nie je prázdna, lekcií sa pridelí náhodne vybraná začiatočná vyučovacia hodina z definovej množiny. Ak je prázdna, lekcií sa pridelí náhodne vyučovacia hodina spomedzi vyučovacích hodín, kedy je možné nasadzovať výuku do rozvrhu. Po pridelení začiatočnej hodiny, pridelí inicializácia lekcii skupinu miestností. Tu tiež využije počiatočné predspracovanie údajov, ktoré každej lekcií pridelí zoznamy dobrých, horších a zlých skupín miestností. Dobré skupiny miestností obsahujú miestnosti, ktoré spĺňajú: sú spravované fakultou, ktorá ponúka kurz, kam lekcia patrí; nachádzajú sa na jednom z miest výuky kurzy, kam lekcia patrí; obsahujú potrebné zariadenie a majú pre lekciu dostatočnú kapacitu. Miestnosti z horšej skupiny miestností narozdiel od prvej miestnosti nemajú dostatočnú kapacitu alebo vybavenie. Lekcia však prekračuje kapacitu miestnosti v povolenom limite. Skupina zlých miestností pozostáva z miestností, ktoré spĺňajú to čo skupina horších s tým rozdielom, že lekcia prekračuje kapacitu miestnosti nad povolený limit. Ak má lekcia aspoň jednu skupinu miestností v skupine preferované, dobré, horšie alebo zlé, inicializácia jej pridelí náhodnu skupinu miestností z jednej zo skupín. Ak nie, pridelí jej náhodnú miestnosť.

Nová inicializácia sa snaží prideľovaniu hodnôt lekciám ponechať čo najväčšiu voľnosť, aby populácia obsahovala čo najrôznorodejšie riešenia. Snaží sa však zabrániť prideľovaniu vyučovacích hodín, kedy nemôže nejaký učiteľ

učiť alebo je zakázané všeobecne učiť. Rovnako sa snaží vcelku zmysluplne prideľovať lekciám miestnosti. Vďaka predspracovaniu vstupných údajov vyberá lekcia na rozdiel od pôvodného generátora z viacmenej použiteľných miestností, pričom dôraz sa kladie na neporušenie obmedzení 14, 15 definovaných v druhej kapitole.

5.4.2 Jednoduchá rekombinácia

Jednoduchá rekombinácia používa podobné uniformné kríženie ako pôvodný generátor. Z dvoch rodičov sa vytvárajú dvaja nový potomkovia. Na začiatku prvý rodič sa skopíruje do prvého potomka a druhý do druhého. Potom pre každú lekciu v rozvrhu si rekombinácia zvolí náhodne s rovnakou pravdepodobnosťou číslo 0 alebo 1. Ak si zvolí 1, prvý potomok si pre danú lekciu pridelí miestnosť a začiatočnú vyučovaciu hodinu z druhého rodiča. Skupinu učiteľov si však zachová. Toto zabezpečuje, aby rekombináciou neboli porušené počty lekcií, čo učia jednotlivé skupiny učiteľov. Ak však skupina učiteľov v prvom potomkovi by po pridelení mala učiť, keď nemôže, rekombinácia pridelí lekcií náhodne iný slot, kedy skupina učiteľov môže učiť. Rovnako ako prvý potomok si nastavuje pre lekciu hodnoty z druhého rodiča, si druhý nastavuje hodnoty pre lekciu z prvého. Ak si rkombinácia zvolí 0, nevykoná sa žiadna zmena.

5.4.3 Dňová rekombinácia

Dňová rekombinácia podobne ako jednodňová vytvára z dvoch rodičov dvoch potomkov. Na začiatku skopíruje údaje z prvého rodiča do prvého potomka a z druhého rodiča do druhého potomka. Následne si zvolí náhodne dva dni a pre každú lekciu z vybraných dní urobí nasledovné: Prvý potomok si k lekcií priradí začiatočnú vyučovaciu hodinu a miestnosť akú má druhý rodič. Ak má však skupina učiteľov z prvého potomka pridelená k lekcií(rovnaká akú má prvý rodič) zakázané učiť v nový pridelený čas, zaradí sa lekcia medzi vylúčené miestnosti prvého potomka a priebežne si pre lekciu zachová hodnoty získané z prvého rodiča. Podobne pre druhého potomka.

Rekombinácia na konci pridelí lekciám náhodné vyučovacie hodiny, počas ktorých učitelia môžu lekcie učiť.

Dňová rekombinácia sa stará podobne ako jednoduchá o to, aby počty lekcií pridelených učiteľom ostali zachované.

5.4.4 Mutácia

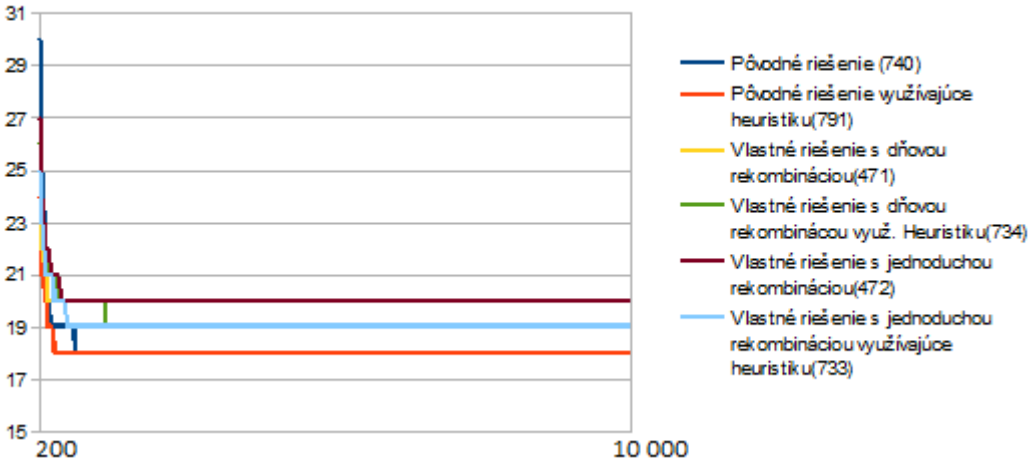
Mutácia funguje rovnako ako mutácia pôvodného generátora s tým, že využíva iba tieto typy mutácií: zmeň vyučovaciu hodinu, zmeň miestnosť, zmeň učiteľa a vymeň lekciám učiteľov.
zmeň vyučovaciu hodiny: pridelí lekcii náhodne vyučovaciu hodinu, kedy nie je zakázané učiť a pridelená skupina učiteľov môže v danom časom učiť. Ak taká vyučovacia hodina nie je, pridelí náhodnú, kedy nie je všeobecne zakázané učiť.
zmeň miestnosť: pridelí lekcií miestnosť rovnako ako prideľuje multiple inicializácia
zmeň učiteľa: zmení lekcií skupinu učiteľov za inú iba ak sa tým nenarušia obmedzenia týkajúce sa počtu lekcií pre učiteľa - presný alebo maximálny počet lekcií, ktoré smie skupina učiteľov učiť.
vymeň lekciám učiteľov: vyberie náhodne kurz, typ lekcií a dve lekcie vybraného typu, ktoré patria pod vybraný kurz. Vymení skupiny učiteľov medzi lekciami. Ak by nejaká skupina učiteľov mala po výmene učiť keď nemôže, pridelí jej náhodne začiatočnú vyučovaciu hodinu, kedy môže učiť.

5.5 Použitie heuristiky

Deb vo svojej práci [7] uviedol, že rozvrhy pri inicializácii mali používať heuristiku, ktorá pred vygenerovaním riešenia zoradí lekcie do postupnosti podľa počtu možností, koľkými je ju schopné zaradiť do rozvrhu. Pre túto prácu sme preto vyvinuli aj heuristiku, ktorej efekt budeme ďalej sledovať. Táto heuristika zoradí lekcie do postupnosti a jednotlivé druhy inicializácií, ktoré sa používajú v tejto práci budú zaraďovať lekcie do rozvrhu v poradí daným heuristikou. Heuristika zoradí lekcie podľa počtu možností ich zaradenia do rozvrhu, pričom bude brať do úvahy pre každú lekciu len dobré miestnosti a vyučovacie hodiny, kedy miestnosť nie je rezervovaná inou fakultou alebo externe a hodiny, kedy môžu učitelia, čo majú lekciu učiť, učiť. Problém pri našom probléme je, že prideľujeme lekciám učiteľov a nie je možné si pre lekciu predvypočítať počty možností jej zaradenia. Ak generátor pri experimentoch použije heuristiku, ako prvé pri inicializácií pridelí lekciám skupiny učiteľov.

5.6 Porovnanie metód na testovacích sadách

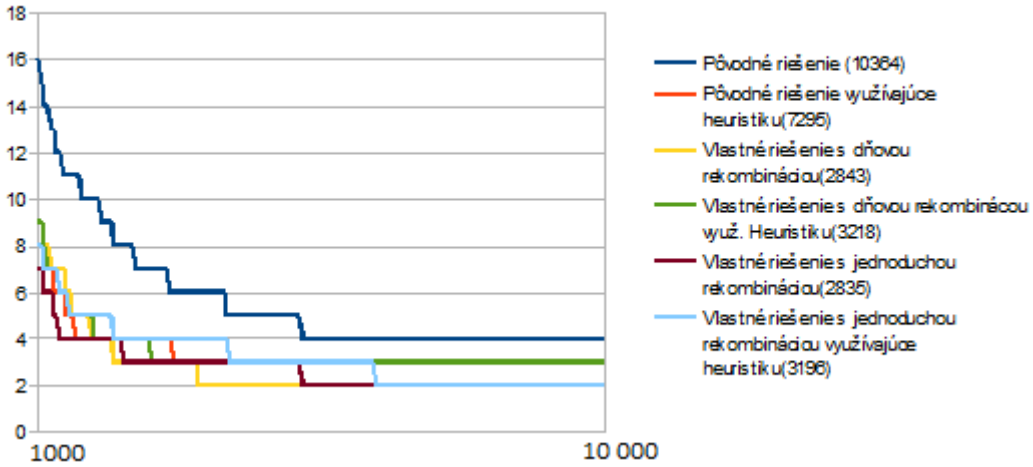
Táto podkapitola porovnáva použitie rôznych prehľadávacích evolučných algoritmov na testovacích sadách. Každý prehľadávací algoritmus hľadá rieše-



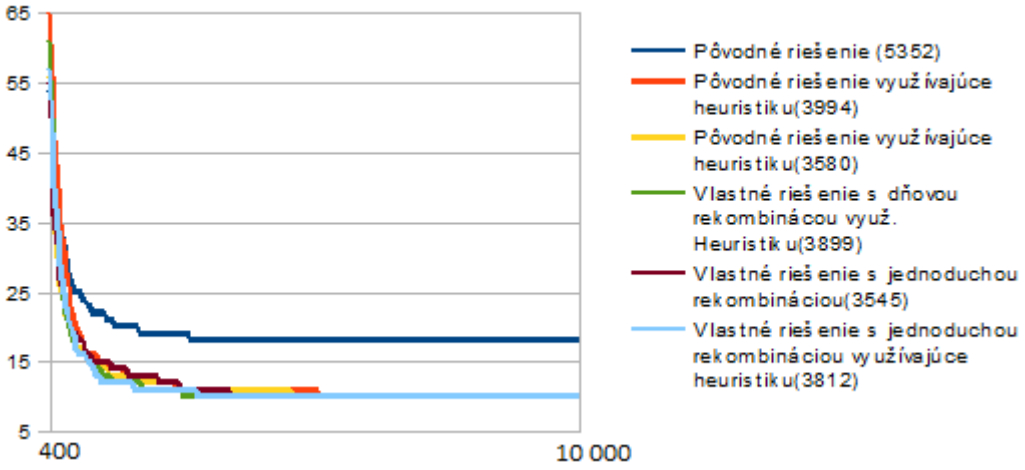
Obr. 5.12: Použitie rôznych metód prehľadávacieho evolučného algoritmu na sade1. V zátvorke je uvedený čas výpočtu generátora v sekundách.

nie počas 10 000 generácií a používa populáciu veľkosti 150. Grafy zobrazujú pre jednotlivé prehľadávacie algoritmy(metódy) počty porušení obmedzení najvyššej priority najlepšieho nájdeného riešenia pre danú generáciu. Niektoré grafy vykresľujú výpočet až od istej generácie, lebo graf kreslený od 0-tej by bol veľmi neprehľadný

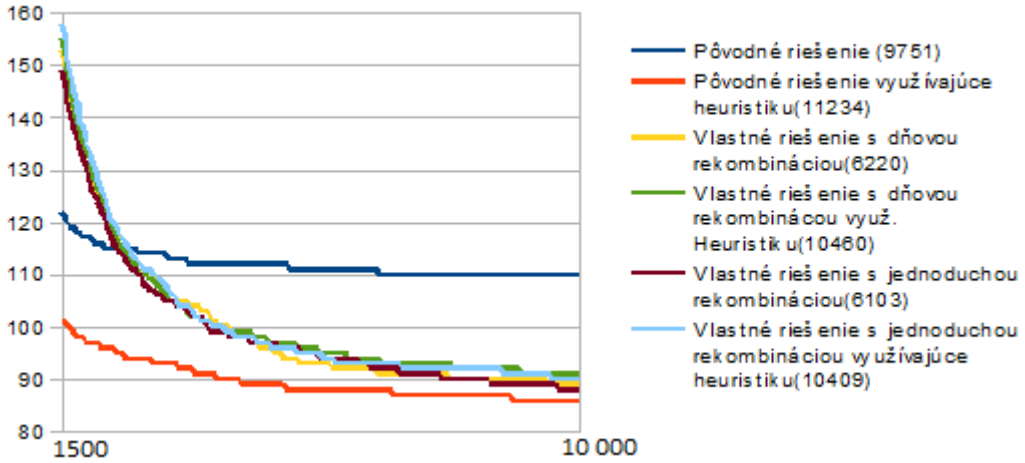
Obr. 5.13: Použitie rôznych metód prehľadávacieho evolučného algoritmu na sade2. V zátvorke je uvedený čas výpočtu generátora v sekundách.



Obr. 5.14: Použitie rôznych metód prehľadávacieho evolučného algoritmu na sade3. V zátvorke je uvedený čas výpočtu generátora v sekundách.



Obr. 5.15: Použitie rôznych metód prehľadávacieho evolučného algoritmu na sade4. V zátvorke je uvedený čas výpočtu generátora v sekundách.



Obr. 5.16: Použitie rôznych metód prehľadávacieho evolučného algoritmu na sade5. V zátvorke je uvedený čas výpočtu generátora v sekundách.

Kapitola 6

Záver

V tejto práci sme popísali problém tvorenia univerzitných rozvrhov a vytvorili prehľad riešení tohto problému so zameraním sa na riešenia využívajúce evolučné algoritmy. Popísali sme fungovanie pôvodného generátora a vymysleli sme, implementovali a otestovali nové metódy, ktoré vylepšujú pôvodný generátor.

Literatúra

[1] R.M.R. Lewis, Metaheuristics for University Course Timetabling, 2006

[2] Tim B. Cooper, Jeffrey H. Kingston *The Complexity of Timetable Construction Problems*, Selected papers from the First International Conference on Practice and Theory of Automated Timetabling, pp.283-295, 1996

[3] E.K. Burke, S. Petrovic *Recent research directions in automated timetabling*, European Journal of Operational Research 140, pp. 266–280, 2002

[4] E.C. Eiben, J.E. Smith *Introduction to Evolutionary Computing*, Springer 2007

[5] http://www.idsia.ch/Files/ttcomp2002/oldindex.html

[6] http://www.idsia.ch/Files/ttcomp2002/IC_Problem/node1.html

[7] D.Datta, K.Deb, C.Fonseca *Multi-Objective Evolutionary Algorithm for University Class Timetabling Problem*, Studies in Computational Intelligence Volume 49, 2007, pp 197-236

[8] B.Sigl, M.Golub, V.Mornar *Solving Timetable scheduling problem by Using Genetic Algorithms*, Faculty of Electrical Engineering and Computing, University of Zagreb

[9] B.Peachter, R.C.Rankin, A.Cumming, T.C.Fogarty *Timetabling the Classes of an Entire University with an Evolutionary Algorithm*, V, Springer, LNCS, 1998

[10] O.Rossi-Doria, M.Sampels, M.Birattari, M.Chiarandini, M.Dorigo, L.M.Gambardella, J.Knowles, M.Manfrin, M.Mastrolilli, B.Paechter, L.Paquete, T.Stutzle *A Comparison of the Performance of Different Metaheuristics on the Timetabling Problem*, PATAT 2002, LNCS 2740, pp. 329–351, 2003

[11] R.Lewis, B.Peachter *New Crossover Operators for Timetabling with Evolutionary Algorithms*, In Proceedings of the 5th International Conference on Recent Advances in Soft Computing (RASC 2004)

[12] Tom M. Mitchell *Machine Learning*, McGraw-Hill Science/Engineering/Math March 1, 1997

[13] E.K. Burke, J.P. Newall *A multi-stage evolutionary algorithm for timetable problem.* IEEE Transactions on Evolutionary Computation, 3:1 pp.63-74, 1999

[14] E.K. Burke, J.P. Newall, R.F. Weare *Initialization strategies and diversity in evolutionary timetabling* Evolutionary Computation, 6:1 pp.81-103, 1998

[15] B. Paechter, R.C. Rankin, A. Cumming, T.C. Fogarty *Timetabling the classes of an entire university with an evolutionary algorithm* In: Eiben et al. [116], pp. 865-874