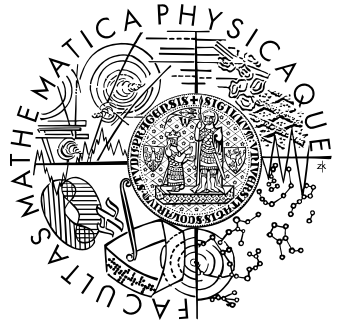
Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Lukáš Ježek

Metody řešení problému přiřazování frekvencí

Katedra aplikované matematiky

Vedoucí bakalářské práce: Mgr. Pavel Nejedlý

Studijní program: Informatika, Programování

2008

Děkuji panu Mgr. Pavlu Nejedlému za odborné vedení mé práce, za čas, rady a doporučení, které mi věnoval.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze dne 5.srpna 2008 Lukáš Ježek

Obsah

Název práce: *Metody řešení problému přiřazování frekvencí*
Autor: *Lukáš Ježek*
Katedra (ústav): *Katedra aplikované matematiky*
Vedoucí diplomové práce: *Mgr. Pavel Nejedlý*
E-mail vedoucího: *bim@kam.mff.cuni.cz*

Abstrakt: *Bezdrátová komunikace je používána ve stále více situacích, například v mobilní telefonii, televizním a rádiovém vysílání, komunikaci satelitů, bezdrátových sítích (WLAN) nebo vojenských operacích. V každé z těchto situací je nutné řešit problém přiřazení frekvencí, aby se spolu komunikující strany domluvily (někdo jim zprostředkoval službu) a při „rozhovoru" nebyly rušeny jinými komunikujícími subjekty. Vzhledem k tomu, že počet způsobů a hlavně účastníků bezdrátové komunikace velmi rychle roste, je tento problém stále více aktuální a důležitý a je nutné jej umět kvalitně řešit.*

V této práci se nachází zadání problému spolu s přehledem používaných technik pro jeho řešení. Vybrané postupy jsou popsány detailněji a je u nich představen konkrétní algoritmus řešící problém přiřazování frekvencí. Tyto algoritmy jsou taktéž implementovány v přiloženém programu, který slouží k řešení zadaných problémů a také k porovnávání jednotlivých použitých algoritmů. V programu je implementován sekvenční algoritmus, algoritmus simulovaného žíhání, algoritmus tabu vyhledávání a dva genetické algoritmy.

Klíčová slova: *problém přiřazování frekvencí, genetické algoritmy, sekvenční algoritmy, simulované žíhání, tabu vyhledávání*

Title: *Solution techniques for the frequency assignment problem*
Author: *Lukáš Ježek*
Department: *Department of Applied Mathematics*
Supervisor: *Mgr. Pavel Nejedlý*
Supervisor's e-mail address: *bim@kam.mff.cuni.cz*

Abstract: *Wireless communication is used in a growing number of situations, for example in mobile telephony, wireless networks (WLAN), television and radio broadcasting, satellite communication or in military operations. It is necessary to solve the frequency assignment problem in each of the above situations, so that the communicating sides get connected and their connection is not interfered by other communicating subjects and connections. It is very important to be able to solve the problem well since the number of the participants in the wireless communication is rapidly growing.*

This work provides introduction to the frequency assignment problem along with the overview of the most frequently used techniques for solving it. Selected methods are described in more details and are also implemented in the enclosed program.

Keywords: *frequency assignment problem, genetic algorithms, sequential algorithms, simulated annealing, tabu search*

1 Úvod

1.1 Organizace práce

Tato práce si klade za cíl seznámit čtenáře s problémem přiřazování frekvencí a nastínit možné způsoby jeho řešení. Na úvod je problém představen včetně formulace jednotlivých řešitelných modelů, testovacích sad a příkladů jeho užití v reálném životě. V kapitole 4 Implementované algoritmy jsou pak detailně rozebrány vybrané algoritmy na řešení jednoho z modelů problému, tyto algoritmy jsou rovněž naimplementovány v přiloženém programu. V další kapitole (5 Porovnání výsledků jednotlivých algoritmů) najdete srovnání jednotlivých postupů z hlediska rychlosti a kvality výsledku. Přiblížení dalších technik pro řešení problému je v kapitole 6 Neimplementované algoritmy. V dalších dvou kapitolách (7 Program a 8 Technický popis programu) následuje popis programu pro uživatele, resp. pro programátory, kteří by se chtěli seznámit s programem blíže a popřípadě jej i rozšířit. V závěru práce je uveden seznam použitých zkratek, referencí a popis obsahu přiloženého CD.

1.2 Motivace a historie

Podle Českého statistického úřadu [CSU] se „v období let 1995 – 2003 zvýšil počet uživatelů mobilních telefonů v Evropské Unii každý rok průměrně o 42%“, v roce 2005 dokonce na jednoho občana České republiky připadal více než jeden mobilní telefon.

Problém přiřazování frekvencí (*FAP* – Frequency Assignment Problem) byl poprvé popsán v šedesátých letech dvacátého století [Metzger70]. Vývoj nových bezdrátových služeb, jako například mobilních telefonních sítí, vedl k nedostatku frekvencí v rádiovém spektru. V té době začaly být frekvence licencovány a prodávány vládou (a nadnárodně Mezinárodní komunikační unií) a operátoři bezdrátových sítí museli začít vytvářet frekvenční plány s ohledem na minimalizaci ceny a zachování kvality služeb. Při tvorbě plánů je nutné brát ohledy na interferenci (vzájemné rušení dvou vysílání na blízkých frekvencích). Sestrojení takovýchto plánů je velmi těžkým problémem, brzy se ukázalo, že řešení problému je podobné problémům známým z teorie grafů, jako například barvení grafů. Do osmdesátých let byly na řešení problému používány zejména heuristiky nalezené právě při řešení barvení grafů. Rozvoj digitálních mobilních sítí *GSM* v devadesátých letech znamenal velký vzrůst zájmu na hledání nových technik pro řešení problému přiřazování frekvencí.

Ucelený pohled na problematiku přiřazování frekvencí poskytuje webová stránka [FAPWEB] a na ní zveřejněný přehledový článek [Aardal03].

1.3 Popis problému

Hlavním úkolem FAPu je sadě bezdrátových antén (nebo spojení) přiřadit frekvence tak, aby byl možný datový přenos mezi dvěma vysílači (konci spojení). Problémem je, že mezi frekvencemi přiřazenými dvěma anténám (nebo spojením) může docházet k interferenci (rušení), čímž nastává ztráta kvality signálu.

Interference dvou signálů nastává, pokud si jsou frekvence blízké v elektromagnetickém spektru a zároveň jsou oba signály geograficky blízko sebe (tak, aby byly dostatečně silné a mohly rušit kvalitu druhého signálu).

Naším cílem je vytvořit frekvenční plány tak, abychom pokryli všechna požadovaná spojení (při rozumně malé úrovni rušení) a přitom zaplatili co nejméně na licenčních požadavcích za používané frekvenční pásmo.

Kvůli konzistenci termínů s anglickou literaturou věnující se problému přiřazování frekvencí je v celé této práci *vysílačem* nazýván přístroj, který vysílá právě na jedné frekvenci. *Anténou* je pak nazývána soustava vysílačů umístěná na jednom geografickém místě.

1.4 Reálné aplikace

Problém přiřazování frekvencí nastává obecně při bezdrátové komunikaci, ta však může mít v různých případech rozdílný charakter (např. pevné versus pohybující se vysílače). V této kapitole je uveden stručný přehled nejdůležitějších výskytů problému a základní rozdíly mezi nimi.

Mobilní telefonní sítě

V mobilní komunikaci je jedním koncem komunikace pevně umístěná anténa a druhým je pohybující se mobilní telefon. Každá anténa pokrývá určitou oblast a dokáže obsluhovat více mobilních zařízení současně (například díky technologii *TDMA* nebo umístění více vysílačů na jednu anténu). Interference se rozděluje na více typů:

- *na stejné anténě* – vysílače umístěné na jedné anténě musí vysílat na dostatečně vzdálených frekvencích (např. 5 jednotek)
- *na sousedních anténách* – sousední antény taktéž nemohou vysílat na blízkých frekvencích (typická vzdálenost frekvencí je 2)
- *blízké antény* – na anténách, které od sebe nejsou dostatečně vzdálené, může být zakázané používat stejné frekvence

V praxi nastává ještě jedna komplikace - mobilní telefony se pohybují, a když dospějí na kraj působnosti jedné antény, je nutné předat řízení anténě jiné. Při tomto předání bývá nutné změnit frekvenci vysílání. Konkrétní informace věnované problému mobilních telefonních sítí jsou uvedeny například v [Eisenblätter01].

Rádio a televize

Televizní a radiové vysílání je podobné tomu mobilnímu. Hlavními rozdíly je jednosměrnost komunikace a používané frekvenční vzdálenosti. Největším problémem u tohoto typu vysílání je poskytnuté frekvenční pásmo, které povoluje výskyt vícenásobných harmonických frekvencí (násobků hlavní nosné frekvence), které spolu velmi výrazně interferují. Problematice rádiového vysílání se věnuje studie [Mannino03].

Případné změny frekvence při pohybu rádia se řeší pomocí služby *AF* (Alternative Frequencies) systému *RDS* (Radio Data System).

Vojenské aplikace

Hlavními případy, které se dají koumat a považovat za odlišné od jiných aplikací, je používání polních a vzdušných telefonů, kdy je nutné řešit dynamický problém přiřazování frekvencí (požadavky na spojení se s časem mění, navíc se jednotlivá komunikující zařízení mohou pohybovat). Každé spojení zde sestává ze dvou pohyblivých telefonů a jsou mu přiřazeny dvě frekvence v pevné vzdálenosti, každá pro komunikaci jedním směrem. Blíže se tomuto problému věnuje například [Tiourine99].

Komunikace přes satelity

Oba dva koncové vysílače jsou umístěny na zemi a komunikují mezi sebou přes satelit (popřípadě více satelitů). Spojení tedy probíhá nejdříve ze země na satelit a poté opět na zem, přičemž největší šance na rušení je právě na satelitu. Frekvence, na kterých satelit najednou komunikuje, od sebe musí být výrazně odděleny (dochází ke střetu dvou silných vysílání na stejném místě). Tento problém řeší *metoda dvou vzdálených pásem*, kdy komunikace v jednom směru probíhá na frekvencích prvního pásma a ve druhém směru na frekvencích z druhého pásma, přičemž jsou tato dvě pásma od sebe výrazně oddělena (mezi žádnými dvěma frekvencemi tak nedochází k interferenci). Proto se má smysl zabývat jenom jedním směrem komunikace (problém přiřazení frekvencí ve druhém směru je stejný). Každému satelitu bývá dopředu přiděleno unikátní pásmo frekvencí (respektive dvě – druhé pro opačný směr), na kterých nemůže vysílat nikdo jiný. Tento problém je řešen v [Thuve81].

Bezdrátové místní sítě

V sítích *WLAN* probíhá komunikace mezi mobilními zařízeními (notebooky, *PDA*, …) a jedním přístupovým bodem (*AP* – Access Point). Jednotlivé *WLAN* standardy (*IEEE 802.11a/b/g/n*) definují počet frekvencí, které může *AP* používat. Standard *802.11g* například přiděluje 13 frekvencí s tím, že se jednotlivé frekvence překrývají – vzdálenost frekvencí je 5 MHz, zatímco frekvence vzdálené minimálně 22 MHz jsou považovány za nepřekrývající se. To znamená, že se dají použít pouze 3 frekvence najednou bez vzniku interference. Proto se z *WLAN* sítí stává problém přiřazení tří frekvencí, což je pro nás nezajímavé. Problém sítí *Wi-Fi* je řešen v [Leung03].

2 Modely FAPu

2.1 Matematická formulace problému

Bezdrátoví operátoři si koupí licenci na používání jedné či více souvislých částí frekvenčního spektra. Zakoupené frekvenční pásmo [f_{min}, f_{max}] pak bývá rozděleno do jednotlivých kanálů o šířkách Δ. Jednotlivé kanály (většinou ve FAPu také nazývány frekvence) jsou poté očíslovány od *1* do *N*, kde $N = \frac{f_{max} - f_{min}}{\Delta}$. Při řešení problému se zabýváme právě přidělováním těchto kanálů (čísel *1* – *N*). *Šířkou frekvenčního pásma* nazýváme rozdíl mezi největším a nejmenším použitým kanálem, tedy *N* – *1*.

Interference dvou signálů závisí na vzdálenosti od místa jejich vzniku, rozdílu jednotlivých frekvencí, síle signálů, směru, ve kterém se střetly, a vlastnostech prostředí. V praxi je téměř nemožné namodelovat takovéto podmínky, a proto se uvažují signály šířící se ve všech směrech stejně a okolní prostředí je zcela zanedbáno. Dále jsou při interferenci uvažovány vždy jen dva signály (v praxi může být rušení vzniklé jedním signálem malé, avšak celkové rušení vzniklé všemi okolními signály neúnosně velké).

Interference v naší formulaci závisí jen na vzdálenosti dvou vysílačů a vzdálenosti frekvencí. Proto můžeme pro každou dvojici zadaných vysílačů určit, v jaké minimální vzdálenosti musí být frekvence jim přiřazené. Tímto nám vzniká *matice rušení A* (pro každou dvojici vysílačů *u* a *v* je na pozici *A[u, v]* uložena hodnota nejmenší frekvenční vzdálenosti tak, aby se signály z těchto vysílačů vůbec nerušily).

Z matice rušení je možné též vyrobit *graf rušení* – vysílače jsou zakresleny do grafu podle jejich geografické pozice. Mezi dvěma vrcholy *u* a *v* existuje hrana, právě tehdy když mezi těmito vysílači

může nastat rušení ($A[u, v] \neq 0$). Váha této hrany je pak minimální frekvenční vzdálenost mezi těmito vysílači, tedy $A[u, v]$.

I přes výše uvedené zjednodušení existuje mnoho hledisek, která můžeme chtít optimalizovat (např. minimalizace počtu přiřazených frekvencí bez interference, minimalizace celkové interference nebo maximalizace počtu poskytovaných spojení při pevně daném počtu frekvencí, které můžeme použít). Rozdělení do těchto kategorií je podrobněji rozebráno v následujících kapitolách.

Pozn.: Téměř veškerá komunikace (vyjma televizního a rádiového vysílaní) je obousměrná a na komunikaci každým směrem bývá potřeba zvláštní kanál. Ve většině modelů se ale řeší jen jednosměrná komunikace s tím, že druhý směr lze vyřešit naprosto stejně v posunutém pásmu.

Vstupem každého z následujících modelů je seznam antén, počet vysílačů na jednotlivých anténách, matice rušení, výstupem je přiřazení frekvencí jednotlivým vysílačům.

2.2 F-FAP (feasible)

Úkolem je přiřadit frekvence (v zadaném pásmu) všem vysílačům tak, aby mezi žádnými dvěma anténami nebylo rušení větší než zadané maximální rušení.

Vstupem algoritmu je navíc ještě počet frekvencí, výstupem je libovolné kompletní přiřazení frekvencí neporušující žádnou podmínku. Pokud žádné takové přiřazení neexistuje, pak není výstupem nic.

2.3 Max-FAP (maximum service)

F-FAP v případě selhání nepřiřadí žádné frekvence (zadané frekvenční maximum je moc malé vzhledem k počtu požadovaných frekvencí), proto je dobré umět přiřadit alespoň co nejvíce frekvencí a obsloužit co nejvíce spojení. Maximalizací počtu poskytovaných spojení se zabývá Max-FAP.

Problémem tohoto modelu však je to, že ve složitých místech na mapě (například uprostřed města), kde je hodně požadavků na spojení, nemusí přiřadit vůbec nic, zatímco anténám na okrajích plánu jsou přiřazovány všechny frekvence. Tento problém lze řešit stanovením minimálního počtu vysílačů s přiřazenými frekvencemi na každé anténě.

Vstupem algoritmu je počet frekvencí (popřípadě ještě minimální počty vysílačů, jimž musí být přiřazena frekvence), výstupem je vždy přiřazení frekvencí (nemusí být úplné – mohou existovat vysílače bez přiřazené frekvence).

2.4 MB-FAP (minimum blocking)

Jiný způsob řešení výše uvedeného problému Max-FAPu. Místo stanovení dolních mezí na počet vysílačů s přidělenými frekvencemi na anténách je minimalizován takzvaný *celkový blokovací faktor*.

Blokovací faktor jedné antény je závislý na poměru přiřazených vysílačů, čím více vysílačům je přiřazena frekvence, tím je blokovací faktor menší. Blokovací faktor může například klesat exponenciálně v závislosti na růstu poměru přiřazených a požadovaných vysílačů na anténě.

Každá anténa pak určuje poměrnou část (podle počtu vysílačů) *celkového blokovacího faktoru*. Vzorcem tedy: $CBF = \sum_{antény} \left(\frac{počet\ vysílačů\ na\ anténě}{celkový\ počet\ vysílačů} * BF\left(\frac{počet\ přiřazených\ vysílačů\ na\ anténě}{počet\ vysílačů\ na\ anténě}\right)\right)$, kde *CBF* je celkový blokovací faktor a *BF* je funkce určující blokovací faktor jedné antény.

Touto metodou jsou dosahována řešení, ve kterých jsou frekvence na všech anténách přiřazeny rozumnému poměru celkového počtu vysílačů.

Vstupem algoritmu je počet frekvencí a výstupem přiřazení frekvencí (opět nemusí být úplné).

2.5 MO-FAP (minimum order)

Tento model vznikl z reálných požadavků v počátcích vzniku frekvenčních plánů v mobilních sítí, kdy se frekvence daly kupovat po jednotkách (a ne po celých souvislých pásmech) a stály velmi mnoho.

Cílem optimalizace je tedy poskytnout všechny požadované služby (jako F-FAP) a přitom minimalizovat počet použitých frekvencí s tím, že není nutné používat frekvence těsně za sebou, ale jen například první, šestou, jedenáctou a platí se jen za ty využité.

Vstupem tohoto modelu je počet frekvencí a výstupem je úplné přiřazení s minimalizovaným počtem použitých frekvencí. Podobně jako u F-FAPu nemusí vždy řešení tohoto modelu existovat.

2.6 MS-FAP (minimum span)

Tento problém vychází ze situace, kdy se platí za celé používané frekvenční pásmo (i když v něm jsou některé frekvence nevyužité). Cílem tedy je minimalizovat rozdíl $f_{max} - f_{min}$ (šířka použitého frekvenčního pásma). Tento problém se od ostatních liší tím, že nemáme zadané dostupné frekvenční pásmo, ale naopak je naším úkolem jej nalézt (a dokonce jej nalézt co nejmenší možné).

Výstupem tohoto modelu je úplné přiřazení frekvencí a šířka použitého pásma (kterou se model snaží minimalizovat).

2.7 MI-FAP (minimum interference)

Ve všech předchozích modelech byla uvažována jen řešení bez interference, v MI-FAPu je naopak interference povolena, ale cílem MI-FAPu je ji minimalizovat. Máme k dispozici frekvenční pásmo o určité šířce a naším úkolem je uspokojit všechny požadavky antén tak, aby celkové rušení bylo co nejmenší.

Za každou porušenou podmínku v zadané matici rušení je určena penalizace. Hodnotou této penalizace může být například váha odpovídající hrany v grafu rušení (tedy minimální požadovaná frekvenční vzdálenost).

Vstupem je počet frekvencí a výstupem přiřazení, ve kterém je povoleno rušení (toto je však minimalizované).

2.8 Úlohy k řešení

V předcházejících kapitolách je popsán problém a jeho jednotlivé varianty. Pro řešení každého modelu nás ale mohou zajímat dvě různé věci: buď chceme znát nejlepší přiřazení frekvencí optimalizující dané kritérium, nebo nás může zajímat dolní odhad optimalizovaného kritéria (například počtu frekvencí, rušení atd.).

Hledání přiřazení

Problém optimálního přiřazení frekvencí je sám o sobě NP-těžký (není tedy možné v rozumném čase nalézt optimální řešení), proto jsou na hledání přiřazení frekvencí uspokojujícího zadané kritérium používány různé heuristické algoritmy. Tyto algoritmy se snaží najít alespoň co nejlepší řešení a navíc ještě v rozumném čase. Přehled používaných metod je v kapitolách 4 Implementované algoritmy a 6 Neimplementované algoritmy.

Dolní odhady

Na druhou stranu dolní odhady nevytvářejí žádná přiřazení frekvencí, pouze se snaží vytvářet odhady omezující zespoda hodnotu optimálního řešení (optimalizované kritérium). S pomocí dolních odhadů pak bývá prokazováno, že řešení nalezené heuristickým algoritmem je dostatečně dobré. Za všech okolností platí: $dolní\ odhad \leq optimum \leq$ řešení $heuristiky$. Například když se dolní odhad od nalezeného řešení liší o 5%, pak jistě také platí, že nalezené řešení je maximálně o 5% horší než optimum. V případě, že se hodnota nalezená dolním odhadem rovná hodnotě získané algoritmem pro hledání přiřazení, pak tato hodnota je optimálním řešením daného problému.

FAP lze formulovat jako úlohu celočíselného programování, ta je ovšem NP-těžká, takže se řeší jen přibližně pomocí *lineární relaxace* [Kolen94], metody *cutting planes* nebo *braching rules* [Aardal96].

2.9 Restrikce práce

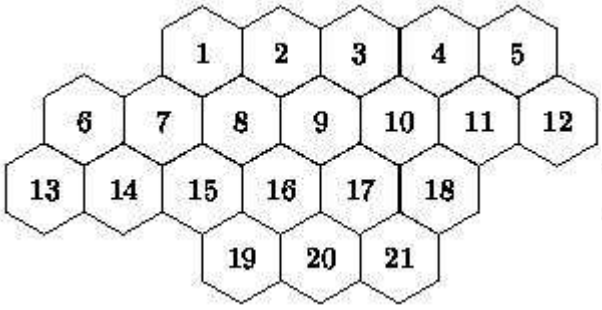
Nadále se v této práci budeme zabývat pouze hledáním co nejlepšího přiřazení frekvencí pro model MS-FAP. V následující podkapitole jsou představeny hlavní testovací sady používané pro zkoumání kvality řešení, v dalších kapitolách pak hlavní algoritmy používané pro získání co nejlepšího přiřazení frekvencí při minimalizaci šířky pásma. Některé z těchto metod jsou podrobněji rozebrány až do podoby konkrétního algoritmu a tyto algoritmy jsou taktéž naimplementovány v přiloženém programu.

3 Testovací sady

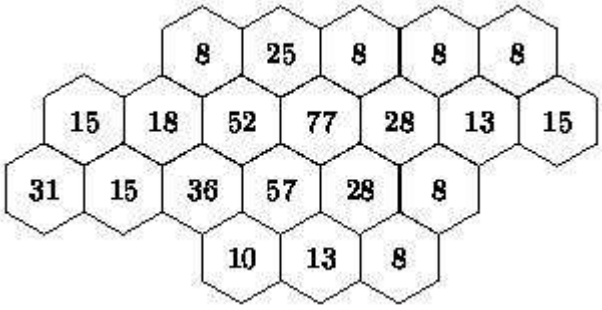
Při hledání algoritmů na řešení problému je nutné mít podklady, na kterých by se dala získaná řešení testovat. Proto v praxi vzniklo několik sad instancí, na kterých se provádí různé testy a srovnání algoritmů.

3.1 Philadelphia

Nejznámější testovací sada pro FAP je tzv. Philadelphia [Anderson73]. Vychází z reálných plánů mobilních sítí okolo Philadelphie. Jedná se o 21 šestiúhelníkových buněk dotýkajících se hranami (viz. Obrázek 1), kde každá buňka představuje plochu pokrývanou jednou anténou (vysílání z ní však sahá i mimo šestiúhelník). Antény jsou umístěny vždy uprostřed jednotlivých buněk, vzdálenost sousedních antén (středů sousedních buněk) je 1 (viz. Obrázek 3). Na každé anténě může být různý počet vysílačů, přičemž každému vysílači je potřeba přiřadit právě jednu frekvenci (viz. Obrázek 2).



Obrázek 1: Schéma Philadelphie



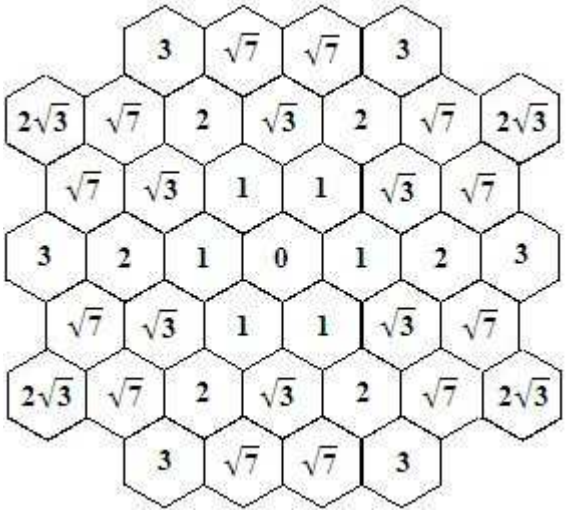
Obrázek 2: Požadavky instance P1

Matice rušení (tj. minimální vzdálenosti frekvencí přidělených anténám *v* a *w* vzdálených od sebe *d(v, w)*) je zadána pomocí vektoru *znovupoužitelných vzdáleností* (reuse distances) – posloupnost nerostoucích čísel d_0, …, d_k. Platí:

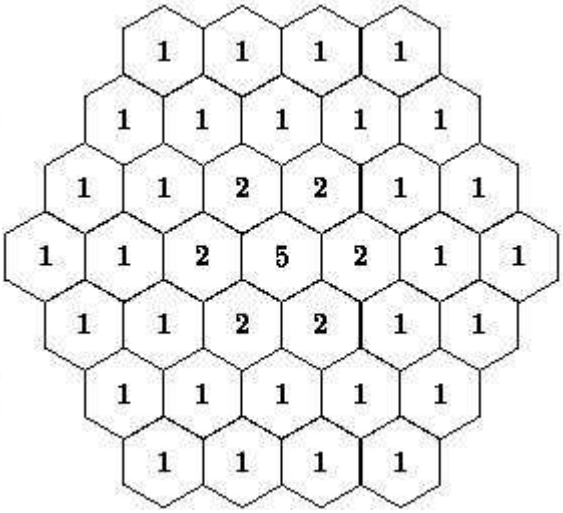
$$\text{pokud} \quad d_j \leq \boldsymbol{d}(\boldsymbol{v}, \boldsymbol{w}) < d_{j-1}, j \in (1, \dots, k), \quad \text{pak} \quad |f_v - f_w| \geq j$$

Jinak řečeno: z intervalů $[d_0, d_1)$, $[d_1, d_2)$, …, $[d_{k-1}, d_k)$ se vybere ten, ve kterém leží vzdálenost dvou vybraných antén. Minimální frekvenční vzdálenost těchto antén je poté index tohoto intervalu (číslováno od 1).

Jedním z používaných vektorů znovupoužitelných vzdáleností je (d0, …, d5) = $(2\sqrt{3}, \sqrt{3}, 1, 1, 1, 0)$, odpovídající Obrázku 4. Hodnoty tohoto vektoru určují, že minimální vzdálenost frekvencí na dvou vysílačích na stejné anténě musí být minimálně 5, protože $d_5 = 0 \leq \mathbf{0} < 1 = d_4$. Vzdálenost frekvencí na sousedních anténách je minimálně 2 ($d_2 = 1 \leq \mathbf{1} < \sqrt{3} = d_1$) a frekvence na všech anténách do vzdálenosti $2\sqrt{3}$ se musí lišit.



Obrázek 3: Vzdálenosti buněk od středové buňky ve Philadelphii.



Obrázek 4: Minimální frekvenční vzdálenosti instancí Philadelphie P1, P3, P5, P7 a P9. Minimální frekvenční vzdálenost ke všem vzdálenějším buňkám je 0.

Kombinací různých vektorů požadavků a znovupoužitelných vzdáleností získáme 9 základních instancí Philadelphie (pojmenovaných P1 – P9), viz. Tabulka 1.

Inst.	Vektor požadavků	Znovupoužitelné vzdálenosti
P1	(8,25,8,8,8,15,18,52,77,28,13,15,31,15,36,57,28,8,10,13,8)	$(2\sqrt{3}, \sqrt{3}, 1, 1, 1, 0)$
P2	(8,25,8,8,8,15,18,52,77,28,13,15,31,15,36,57,28,8,10,13,8)	$(\sqrt{7}, \sqrt{3}, 1, 1, 1, 0)$
P3	(5,5,5,8,12,25,30,25,30,40,40,45,20,30,25,15,15,30,20,20,25)	$(2\sqrt{3}, \sqrt{3}, 1, 1, 1, 0)$
P4	(5,5,5,8,12,25,30,25,30,40,40,45,20,30,25,15,15,30,20,20,25)	$(\sqrt{7}, \sqrt{3}, 1, 1, 1, 0)$
P5	(20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20)	$(2\sqrt{3}, \sqrt{3}, 1, 1, 1, 0)$
P6	(20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20,20)	$(\sqrt{7}, \sqrt{3}, 1, 1, 1, 0)$
P7	(16,50,16,16,16,30,36,104,154,56,26,30,62,30,72,114,56,16,20,26,16)	$(2\sqrt{3}, \sqrt{3}, 1, 1, 1, 0)$
P8	(8,25,8,8,8,15,18,52,77,28,13,15,31,15,36,57,28,8,10,13,8)	$(2\sqrt{3}, 2, 1, 1, 1, 0)$
P9	(32,100,32,32,32,60,72,208,308,112,52,60,124,60,144,228,112,32,40,52,32)	$(2\sqrt{3}, \sqrt{3}, 1, 1, 1, 0)$

Tabulka 1: Jednotlivé instance Philadelphie

Testování MS-FAPu na Philadelphii má tu výhodu, že pro každou z výše uvedených instancí je znám optimální počet frekvencí a navíc je na Philadelphii testován velký počet dalších algoritmů, takže je vytvořené algoritmy s čím porovnávat.

3.2 COST 259

Další z testovacích sad [COST259] FAPu pro mobilní technologie vznikla za podpory Evropské Unie za účelem porovnání různých algoritmů a povzbuzení do vývoje nových metod řešení problému. Celkem vzniklo 32 realistických instancí GSM sítí. Požadavky na počet frekvencí jsou větší než u Philadelphie – od 900 až do 1400.

3.3 CALMA

Na rozdíl od výše zmíněných instancí je CALMA (Combinatorial ALgorithms for Military Applications, [CALMA05]) navržena pro účely vojenské komunikace – ta probíhá mezi dvěma pohybujícími se polními telefony. Každému spojení musí být přiřazeny dvě frekvence – každá na komunikaci jedním směrem. Tyto dvě frekvence jsou do sebe vzdáleny vždy v pevně daných vzdálenostech. Pod tímto projektem byly navrženy dvě sady instancí: CELAR – jedenáct reálných instancí a GRAPH – 14 počítačem vygenerovaných instancí.

3.4 Testovací sady v programu

V přiloženém programu je možné vytvořit libovolné zadání problému pomocí uživatelsky zadané mapy, kde je nutné zadat pozici každé antény, počet požadovaných vysílačů na anténě a vektor znovupoužitelných vzdáleností. Ukázky vytvoření testovacích sad takovýmto způsobem jsou v kapitole 7.2 Vstupní XML a na přiloženém CD.

Program navíc zná rozložení vrcholů testovací sady Philadelphie a umožňuje vytvořit libovolnou její instanci zadáním jejího čísla, popřípadě vytvoření vlastní instance definováním počtu požadovaných vysílačů na jednotlivých anténách a zadáním vektoru znovupoužitelných vzdáleností.

Navíc je možné program rozšířit o další předdefinované mapy – více informací o rozšiřování programu je v kapitole 8.2 Rozšiřitelnost.

4 Implementované algoritmy

V této kapitole se seznámíme s několika vybranými obecnými postupy pro řešení MS-FAPu, každá z těchto technik je pak detailněji rozebrána až do popisu konkrétního algoritmu. Obdržené výsledky a diskuze o nastaveních jednotlivých parametrů algoritmů pak následují za každým popsaným algoritmem. V následující kapitole (5 Porovnání výsledků jednotlivých algoritmů) je uvedeno srovnání výsledků všech implementovaných algoritmů. K řešení problému MS-FAP lze taktéž použít i další postupy, které nebyly do programu implementovány, stručný přehled těchto tříd algoritmů je uveden v kapitole 6 Neimplementované algoritmy.

4.1 Sekvenční (hladové) algoritmy

Sekvenční (neboli hladové) algoritmy vytváří přiřazení frekvencí tak, že nejdříve vysílače seřadí podle nějakého kritéria a poté postupně berou vysílače a přiřazují jim frekvence.

Hlavní charakteristikou sekvenčních algoritmů je nenávratnost tahu (jednoho přiřazení). Přehled používaných technik pro jednotlivé kroky algoritmu je uveden níže.

Implementovaný sekvenční algoritmus

Základní prvky vybraného sekvenčního algoritmu jsou popsány v [Hurley97]. Základní schéma algoritmu je následující:

```
(1)   seřaď vysílače
      vyber první vysílač a přiřaď mu frekvenci 1
      dokud nejsou přiřazeny všechny vysílače
(2)         vyber následující vysílač
(3)         přiřaď mu vhodnou frekvenci
      konec
```

Pozn. v kroku (2) není nutné, aby vybraný vysílač byl následující v prvotním setřídění (z kroku 1), ale tento krok může na prvotní setřídění brát ohled.

Každá z výše označených operací má více variant provedení a všechny se dají vzájemně kombinovat. Tímto se dá dosáhnout 56 různých sekvenčních algoritmů.

(1) **prvotní setřídění vysílačů** – většinou vychází z grafu rušení, který jsme popsali v kapitole Modely FAPu, a snahou je přiřadit frekvence nejdříve těm vysílačům, které vytváří nejvíce omezujících podmínek (z nich vede v grafu nejvíce hran). Použité strategie jsou:
   a. *podle stupně* (*LargestFirst*) – sestupné setřídění podle počtu porušení, které vysílače vytvářejí.
   b. *podle stupně s vypouštěním* (*LargestFirstExclude*) – opět sestupné setřídění podle počtu porušení, ale tentokrát jsou v potaz brána jen porušení s těmi vysílači, které ještě nejsou zařazeny (jinými slovy: od chvíle, kdy vybereme vysílač do výsledné posloupnosti, už nebereme v potaz porušení, ve kterých figuroval).
   c. *nejmenší stupeň nakonec* (*SmallestLast*) – postupně jsou ze všech vysílačů odebírány ty, které vytvářejí nejmenší počet porušení, po odebrání posledního vysílače je výsledný seznam otočen.
   d. *vážené (generalizované) verze setřídění* (*Generalized**) – ke všem výše uvedeným strategiím ještě existují jejich vážené varianty, které místo s počtem porušení počítají se součtem vah těchto porušení (takto vzniknou další tři algoritmy prvotního setřídění).

e. *náhodné setřídění* (*Random*) – vysílače jsou seřazeny náhodně.

(2) **výběr následujícího vysílače** – prvotním setříděním nám sice vzniklo setřídění vysílačů, které bychom mohli používat po celou dobu, ale některé skutečnosti mohly vyplynout na povrch až během výpočtu, a proto je možné brát v potaz i aktuální situaci. Jsou k dispozici následující strategie:
   a. *sekvenční* (*Sequential*) – nijak nemění prvotní setřídění.
   b. *podle stupně nasycení* (neimplementováno, slouží k lepšímu pochopení následující strategie) – přeuspořádávání vysílačů podle aktuální situace, snaží se upřednostňovat ty, které jsou v kritickém území (kde okolní vysílače zakazují nejvíce frekvencí). Pro každý vysílač je určen počet zakázaných frekvencí (*stupeň nasycení*). Frekvence f je pro vysílač zakázaná, pokud je přiřazena některému z jeho sousedů v grafu rušení. Z vysílačů je vždy vybrán ten, který má největší stupeň nasycení, v případě rovnosti je vybrán první podle prvotního setřídění.
   c. *vážený stupeň nasycení* (*GSD*, *Generalized saturation degree*) – tato strategie vychází z předchozí, avšak je volen jiný (vážený) výpočet stupně nasycení. Vliv zakázané frekvence f pro vysílač v je největší z vah hran v grafu rušení spojující vysílač v s vysílačem, jemuž je přiřazena frekvence f. Vážený stupeň nasycení je pak pro každý vysílač roven součtu vlivů jemu zakázaných frekvencí.

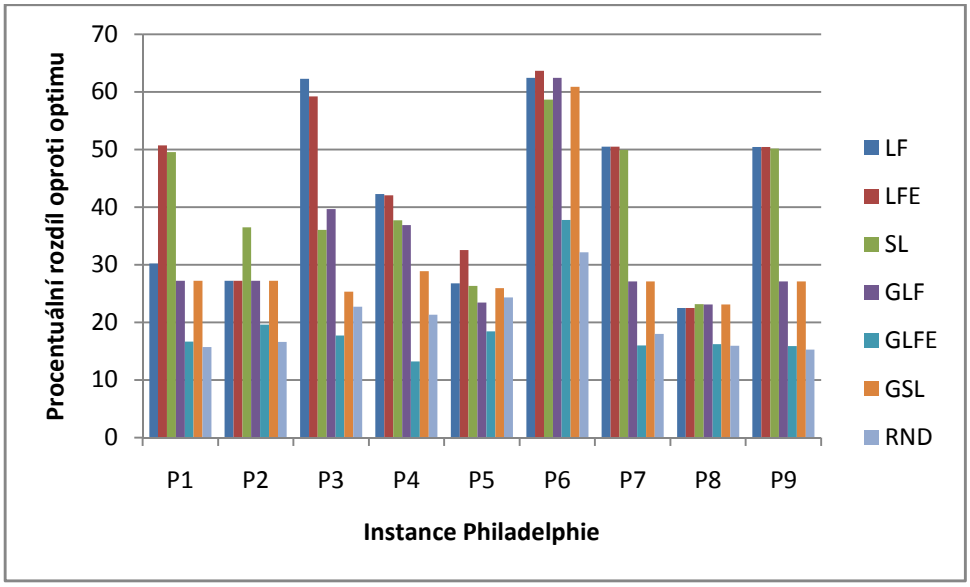
(3) **výběr frekvence** – vybranému vysílači je nutné ještě přiřadit frekvenci. Protože se snažíme minimalizovat celkový počet frekvencí, jsou použity strategie upřednostňující nižší frekvence před většími:
   a. *nejmenší vhodná* (*Smallest*) – výběr nejmenší možné frekvence.
   b. *obsazená frekvence* (*Occupied*) – vybrána je libovolná z již přiřazených frekvencí. Pokud z nich žádná nelze přiřadit, tak se použije první strategie.
   c. *nejmenší obsazená frekvence* (*SmallestOccupied*) – výběr nejmenší vhodné z již přiřazených frekvencí. Pokud není žádná vhodná, použije se první strategie.
   d. *nejmenší co nejvíce obsazená frekvence* (*HeaviestOccupied*) – vybrána je nejmenší z nejvíce použitých frekvencí. Pokud žádná použitá není vhodná, použije se první strategie.

Porovnání jednotlivých strategií

V této kapitole je provedeno porovnání jednotlivých použitých strategií. Výsledků v testech bylo dosaženo spuštěním výpočtu všech instancí Philadelphie pomocí všech kombinací strategií (celkem 56). Při porovnávání každé ze strategií je jako její hodnota brán průměr, kterého tato strategie dosáhla v kombinaci se všemi dalšími volbami sekvenčního algoritmu (např. pro třídění je jako výsledek jednoho třídění brán průměr z výsledků dosažených tímto tříděním se všemi možnými kombinacemi výběru následujícího vysílače a výběru frekvence – tedy průměr z osmi hodnot).

Grafy u porovnávání jednotlivých strategií (prvotního setřídění, výběru vysílače a výběru frekvence) ukazují o kolik procent je průměrný výsledek s použitím té které strategie horší než optimum.

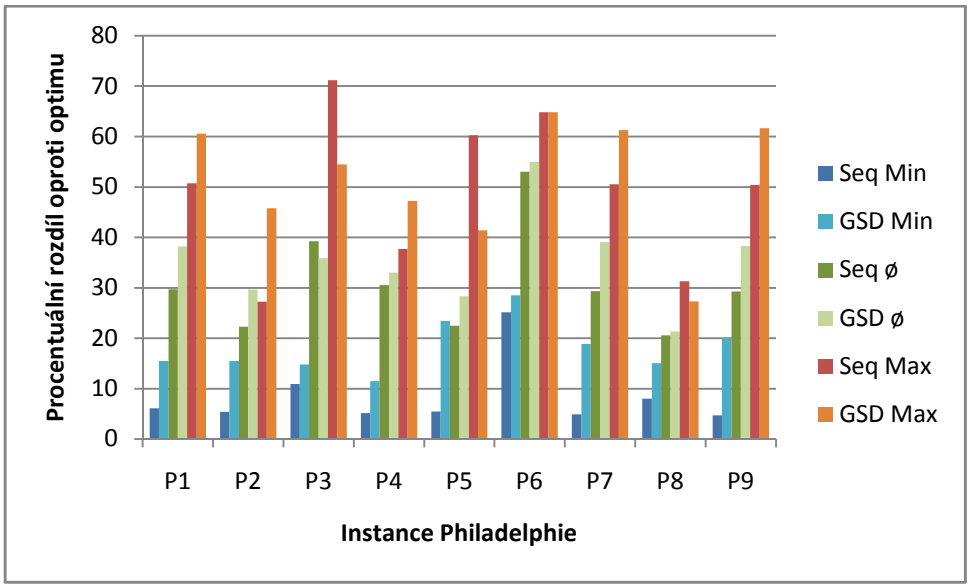
Srovnání prvotních setřídění



Z výsledků porovnání prvotních setřídění je patrné, že vážené varianty algoritmů (beroucí v potaz váhy jednotlivých omezení) dosahují lepších výsledků než jejich jednodušší varianty. Celkově nejlepší možností se ukázalo být setřídění podle váženého stupně s vypouštěním (*GeneralizedLargestFirstExclude*). Překvapivé jsou velmi dobré výsledky náhodného setřídění.

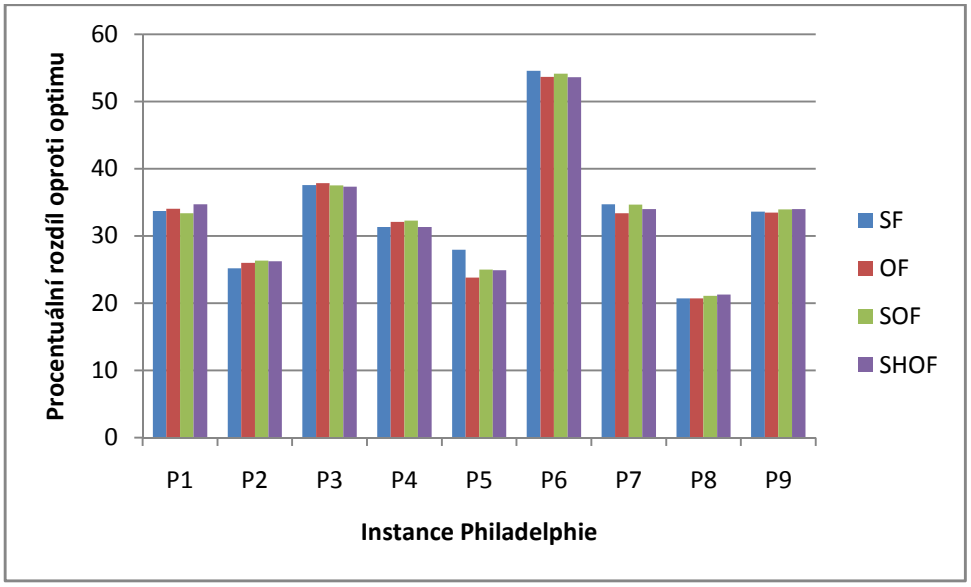
Srovnání výběru následujícího vysílače

Pro výběr následujícího vysílače jsou implementovány pouze 2 strategie: sekvenční – tmavší barvy a vážený stupeň nasycení – světlejší barvy. Pro každou strategii je zobrazena minimální (modrá), průměrná (zelená) a maximální (červená) hodnota z provedených testů.



Na instance problému Philadelphia se více hodí sekvenční výběr následujícího vysílače. I když na výsledcích instance *P3* a *P5* je vidět, že nejhorší výsledek získaný právě sekvenčním výběrem je o hodně horší, než ten získaný váženým stupněm nasycení.

Srovnání výběru frekvence



Ze srovnání výběru frekvence je patrné, že všechny čtyři algoritmy jsou průměrně stejně dobré. Minima bylo u většiny výsledků dosaženo strategií výběru nejmenší vhodné frekvence (*Smallest*).

4.2 Genetické algoritmy

Genetické algoritmy se inspirují pozorováním přírody, zejména evoluce a přírodního výběru. V přírodě se každý živý organismus snaží co nejvíce přizpůsobit okolnímu prostředí, zkušenosti, které živočišný druh během vývoje získal, má uloženy v chromozomech. Při reprodukci dochází ke kombinaci chromozomů mezi jednotlivými živočichy a vyvíjejí se druhy, které mají větší šanci na přežití než jiné. Takovéto druhy mají pro budoucnost větší šanci, že předají dobré části svých chromozomů svým potomkům. Samozřejmě, ne každá změna musí vést k lepšímu organismu a ty, které jsou horší, mají tendenci vymřít.

Genetické algoritmy tento přirozený proces napodobují a používají biologické termíny jako *chromozomy* (jedno řešení), *gen* (hodnota jedné veličiny – proměnné – v řešení) a *populace* (sada řešení). Při množení se chromozomy dvou druhů kombinují a vznikají z nich nové chromozomy (*crossover, křížení*) a jednou za čas dochází k *mutaci* (nahodilé změně v chromozomu). Z každé populace jsou do další generace vybrány jen ty slibnější chromozomy (*selekce*).

Hlavními prvky každého genetického algoritmu jsou:

- reprezentace chromozomu – jak spolu souvisí chromozomy a řešení problému
- tvorba prvotní populace
- hodnotící funkce – funkce, která určuje jak moc je který chromozom dobrý
- selekce – mechanismus výběru lepších chromozomů do dalších populací (většinou na základě výsledků hodnotící funkce)
- mutace – drobné změny jednotlivých chromozomů
- crossover – vznik nového řešení (popřípadě i více nových) ze dvou rodičovských řešení

Ve FAPu může být chromozomem například vektor frekvencí přiřazených k jednotlivým vysílačům (c_j = f ↔ frekvence f je přiřazena vysílači j) nebo pořadí vysílačů, ve kterém jim má být přiřazena

frekvence. První z možností je vhodná k použití při minimalizaci celkového rušení (MI-FAP), zatímco druhá reprezentace je používána k minimalizaci šířky pásma (náš případ).

Mutací většinou bývá jednoduchá změna chromozomu jako například změna jednoho genu (jedné hodnoty) nebo prohození hodnot dvou genů v chromozomu.

V literatuře je možné najít mnoho typů crossoverů, například *jednobodový* crossover – je vybrána pozice, kde jsou chromozomy překříženy, do této pozice se berou hodnoty genů z prvního otce, zatímco od této pozice dále geny od druhého otce (z otců *(a1, a2, …, an)* a *(b1, b2, …, bn)* a pozice *k* vznikne *(a1, a2, …, ak, bk+1, …, bn)*), nebo *dvoubodový* crossover, kde je každý chromozom rozdělen na 3 části. Tyto typy crossoverů však nejsou použitelné v případě, že používáme druhou zmíněnou reprezentaci chromozomů (pořadí vysílačů), protože by nám vznikly chromozomy s opakujícími se čísly. V takovémto případě je třeba využít sofistikovanějších metod, jako například *cyklický* crossover (viz. Permutační algoritmus) nebo *permutační* crossover (viz. Třífázový genetický algoritmus).

Třífázový genetický algoritmus

Náčrtek algoritmu je uveden v [Fu06], kde je k řešení MS-FAPu navržen třífázový algoritmus, z čehož jsou první dvě fáze sekvenční a až třetí fáze genetická:

1. *pravidelné přiřazení* – přiřazení co nejmenšího počtu frekvencí té anténě, na které je nejvíce vysílačů. Vysílačům je přiřazeno co nejméně frekvencí, tedy v pravidelných intervalech po nejmenší možné frekvenční vzdálenosti na jedné anténě (tedy od *1* do *(N – 1) * INT + 1*).
2. *hladové přiřazení* – nalezení tzv. hladového regionu – část mapy, kde je potřeba přiřadit nejvíce frekvencí a snaha přiřadit frekvence všem vysílačům v tomto regionu (pouze ale do frekvence určené první fází). Pokud se nepovede všem prvkům v regionu přiřadit, tak je region dále rozšiřován o sousedy všech již obsažených antén do té doby, než zahrnuje všechny antény nebo než se povede přiřadit frekvence všem anténám.
3. *genetický algoritmus* – genetický algoritmus generuje pořadí zbylých vysílačů, ve kterém jim mají být přiřazeny frekvence. Na finální přiřazení frekvencí se poté použije sekvenční algoritmus, který vysílačům v daném pořadí přiřazuje vždy nejmenší vhodné frekvence (sekvenční algoritmus s vysílači seřazenými podle genetického algoritmu s volbami *Sequential* a *Smallest*).

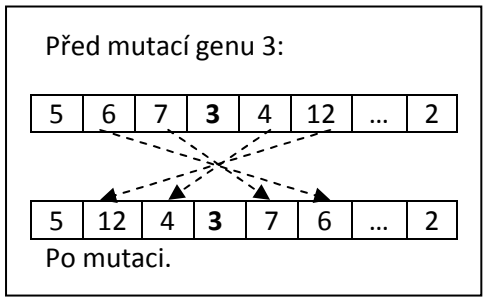
```
(1)   přiřaď frekvence v pravidelném intervalu největší anténě
(2)   najdi hladový region
      dokud nejsou v hladovém regionu všechny antény
            přiřaď frekvence do hodnoty spočtené první fází všem anténám v
            hladovém regionu
            pokud se povedlo, tak vyskoč
            rozšiř hladový region o sousedy všech antén v regionu
      konec
(3)   náhodně vygeneruj první populaci
      od 1 do počtu generací
(3a)        vyhodnoť všechny chromozomy
(3b)        proveď selekci
(3c)        zmutuj všechny vybrané chromozomy
            odstraň všechny nevybrané chromozomy
(3d)        doplň populaci pomocí crossoveru (náhodní rodiče z vybraných)
      konec
(4)   přiřaď frekvence podle historicky nejlepšího chromozomu
```

První fáze (1 – pravidelné přiřazení) přiřadí frekvence anténě s největším počtem požadavků a vytvoří tak dolní odhad na počet frekvencí. Dále v algoritmu přiřazujeme frekvence pouze do tohoto dolního odhadu – to znamená, že se nám nemusí povést přiřadit frekvence všem vysílačům.

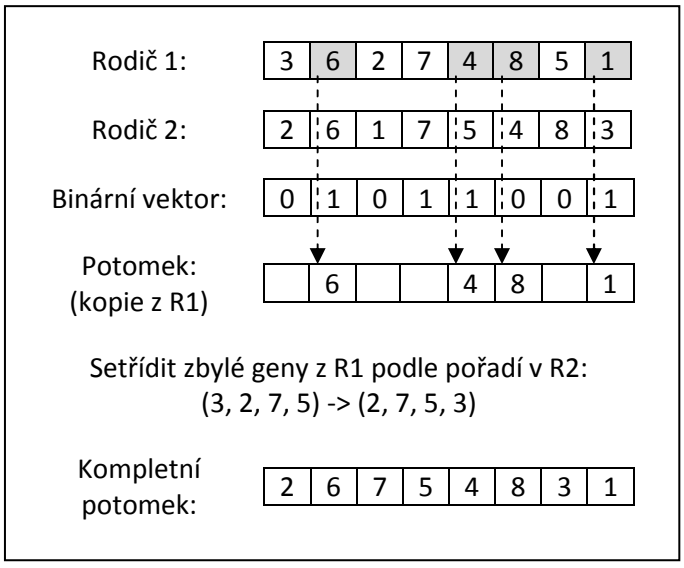
Ve fázi genetického algoritmu (3) je na začátku vygenerována počáteční populace. Každý chromozom reprezentuje pořadí vysílačů, ve kterém jim mají být přiřazeny frekvence (pomocí sekvenčního algoritmu). V chromozomech se vyskytují jen ty vysílače, kterým se nepovedlo v předchozích dvou fázích přiřadit frekvenci.

Hodnotou vyhodnocovací funkce (3a) je pro každý chromozom počet nepřiřazených vysílačů (viz. bod 1). Čím větší její hodnota je, tím je chromozom (řešení) horší. Ve fázi selekce (3b) je pro každý chromozom určena pravděpodobnost výběru podle jeho hodnoty (čím lepší – menší – hodnota, tím větší pravděpodobnost) a poté je pro každý chromozom rozhodnuto, zda je vybrán (pokud je náhodné číslo – vygenerované pro každý chromozom zvlášť – menší než pravděpodobnost jeho selekce, pak je vybrán, v opačném případě odmítnut).

Každý vybraný chromozom je zmutován (3c). Pro tuto fázi mutace musí být zvolena mutace, která zachovává reprezentaci chromozomu – musí zachovávat hodnoty, může je jen prohazovat (jinak bychom již neměli v chromozomu uložené pořadí vysílačů). Proto je zvolena jednoduchá transpoziční mutace – pro každý gen je rozhodnuto (náhodně s pravděpodobností rovnou mutačnímu faktoru – *mutation*), zda zmutuje. Pokud ano, pak si jeho sousedé vymění pozice, stejně provedou i jeho „ob-sousedé“ (viz. ukázka mutace).



Chromozomy, které nebyly vybrány v selekci, jsou zapomenuty a je třeba místo nich doplnit nové, aby byla populace úplná. Proto přichází fáze crossoveru (3d), kde se ponechané (a zmutované) chromozomy kříží a vytváří tak nová řešení (ve kterých by měly být zachovány dobré vlastnosti rodičů). Jelikož nám křížení musí zachovat reprezentaci genu, tak nemůžeme použít obyčejný jedno- nebo dvou-bodový crossover, ale nějaký jiný, který by zachovával permutační charakter chromozomu. Například *permutační* crossover: k vytvoření nového individua jsou použiti vždy dva náhodně vybraní jedinci, rodiče R1 a R2. Dále je vygenerován náhodný binární vektor stejné velikosti, jako mají chromozomy. Z rodiče R1 jsou do nového řešení vybrány geny z těch pozic, kde jsou v binárním vektoru jedničky. Nevybrané geny z rodiče R1 jsou setříděny podle pořadí jejich výskytu v chromozomu R2 a v tomto pořadí vyplněny do volných míst nového chromozomu. Tímto se do nového genu projeví získané uspořádání vysílačů z obou rodičů, což je žádoucí, a ten má šanci na dobré výsledky.



Poté, co je vygenerován dostatečný počet generací, algoritmus vybere nejlepší chromozom z celé historie (po celou dobu výpočtu jej zná a aktualizuje při změnách k lepšímu) a podle něj provede přiřazení frekvencí (4) – tentokrát však již bez horního omezení, takže vznikne korektní přiřazení. Frekvence jsou přiřazovány sekvenčním algoritmem s výběrem nejmenší dostupné frekvence (stejný algoritmus, jako na určování kvality genu).

Na běh algoritmu má vliv nastavení základních konstant, jako jsou velikost populace, počet generací a mutační faktor. Nastavení jejich hodnot je podrobně rozebráno v podkapitole Nastavení parametrů.

Úprava algoritmu

Ve výše zmíněném návrhu genetického algoritmu nastává problém ve chvíli, kdy se anténa s největším počtem vysílačů nachází v oblasti s velkou hustotou vysílačů. Ne vždy musí být výhodné, aby této anténě byly přiřazeny frekvence s nejmenšími možnými rozestupy. Mějme například 3 sousedy s požadavky 5, 4, 4 a minimální frekvenční vzdálenost na jedné anténě 5 a mezi dvěma sousedy *2*, pak v pravidelném přiřazení přiřadíme první anténě frekvence ve vzdálenosti 5. Mezi frekvence každých dvou vysílačů první antény se „vejde“ jen jeden vysílač souseda, takže celkový rozsah frekvencí bude 1 + 5 * 4 + 4 * 2 = 29. Pokud bychom první fázi nespustili a zvolili rozestupy 6, tak se nám povede vměstnat vysílače na druhé a třetí anténě mezi vysílače z první antény a mohli bychom použít maximální frekvenci 1 + 6 * 4 = 25.

Poté, co jsem si tuto skutečnost uvědomil, jsem do algoritmu přidal druhou variantu, ve které jsem vypustil první dvě fáze a rovnou pustil genetický algoritmus bez horního omezení počtu frekvencí. Touto změnou se nám ale ztratilo určení dolního odhadu počtu frekvencí, takže je potřeba upravit vyhodnocovací funkci – tou je zde šířka využitého frekvenčního pásma.

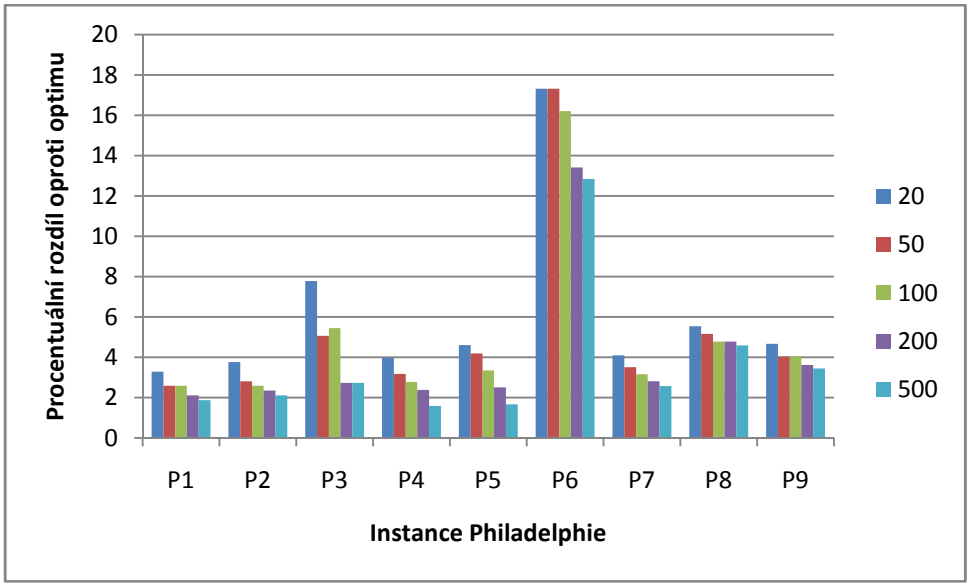
Po tomto vylepšení začal algoritmus generovat cca o 10% lepší výsledky a na několika instancích Philadelphie dokonce dosáhl optima.

Nastavení parametrů

Hodnoty vynesené v níže uvedených grafech byly získány jedním spuštěním algoritmu (pro každou hodnotu) na všech instancích Philadelphie. Volba dalších parametrů je vždy uvedena u každého testu. Na grafech je zobrazeno, o kolik procent byl získaný výsledek horší, než je optimální řešení pro danou instanci Philadelphie. Všechny testy byly spuštěny se změněným algoritmem (viz. Úprava algoritmu).

Počet kroků

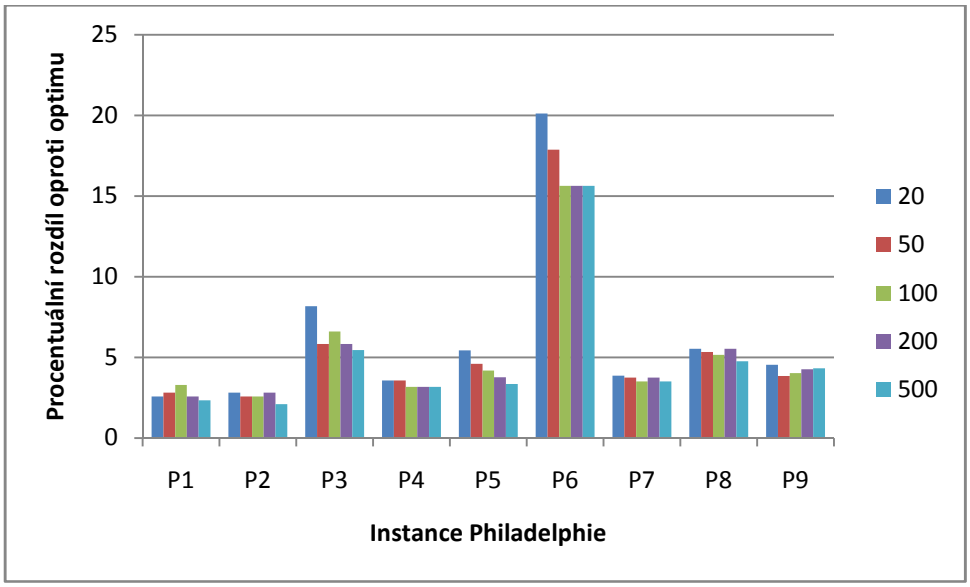
Tento parametr udává, kolik generací algoritmus vyprodukuje, než skončí. Velikost generace (*size*) byla *100* a mutační faktor (*mutation*) *0,1*.



Z grafu je vidět, že genetickému algoritmu prospívá, když může vytvářet více generací. Výsledky dosažené s větším počtem generací jsou daleko lepší. Na druhou stranu však jejich vytvoření trvá déle (doba výpočtu závisí lineárně na počtu generací).

Velikost populace

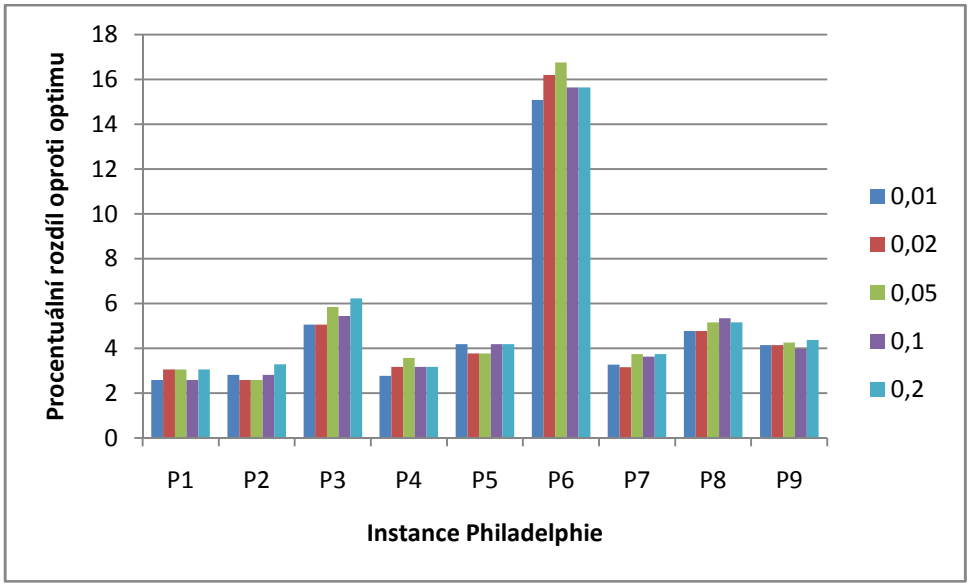
Velikost populace udává, kolik chromozomů (řešení) koexistuje v jedné populaci. V testovacích datech byl zvolen *počet kroků* (loops) *100* a *mutační faktor* (mutation) *0,1*.



Přestože výsledky nastavení tohoto parametru nejsou tak jednoznačné jako u počtu kroků algoritmu, je z nich patrné, že čím větší je populace chromozomů, tím větší je zpravidla šance na dosažení dobrého výsledku. Doba výpočtu algoritmu je však opět lineárně závislá na velikosti generace, proto je lepší volit větší počet kroků algoritmu se zachováním rozumné velikosti generace.

Mutační faktor

Mutační faktor udává pravděpodobnost mutace jednoho genu v chromozomu. *Velikost populace* (size) byla při testování *100* a *počet smyček* (loops) *50*.



Z výsledků obdržených porovnáním jednotlivých nastavení mutačního faktoru lze usoudit, že menší hodnoty mají tendenci dosahovat lepších výsledků (nezanáší se do chromozomů tolik chyb). Doba výpočtu na hodnotě mutačního faktoru nijak nezávisí.

Permutační algoritmus

Permutační genetický algoritmus, který byl navržen a popsán v [Hurley98], využívá stejné principy jako předchozí popsaný genetický algoritmus. Tedy každý chromozom je permutace vysílačů a udává, v jakém pořadí jim mají být přiřazeny frekvence. Přiřazení probíhá jednoduchým sekvenčním algoritmem bez přeuspořádávání vstupního pořadí a jako frekvence je vybrána vždy ta nejmenší dostupná. Cílem tohoto genetického algoritmu je tedy, jako v předchozím případě, vyprodukovat co nejlepší pořadí vysílačů.

```
(1)   vygeneruj N náhodných permutací
(2)   uprav každý chromozom pomocí algoritmu GSD a spočti šířky pásem pro
      jednotlivé chromozomy
      ulož nejlepší chromozom
(3)   dokud není splněna ukončovací podmínka
            pro každý chromozom
                  aktuální chromozom = rodič1
(4)               náhodně vyber rodiče2
(5)               zkřiž rodiče a vytvoř 1 potomka
(6)               zmutuj potomka
                  přiřaď frekvence pomocí potomka
(7)               pokud je potomek lepší než slabší z rodičů
                        nahraď potomkem slabšího rodiče
                        pokud je potomek lepší než nejlepší chromozom
                              ulož potomka jako nejlepší chromozom
            konec
      konec
      výsledek je přiřazení frekvencí s použitím nejlepšího chromozomu
```

Na začátku algoritmu je potřeba vytvořit (1) počáteční generaci chromozomů, na kterých pak bude simulován vývoj „živočišného druhu“, v našem případě řešení FAPu. Všechny tyto chromozomy jsou poté upraveny tak, aby se v nich nacházely správné sekvence a byli tak perspektivnější pro další vývoj a celý algoritmus rychlejší. Tato úprava se provádí pomocí sekvenčního algoritmu s přeuspořádáním frekvencí podle váženého stupně nasycení (2 – viz. kapitola 4.1 Sekvenční (hladové) algoritmy). Tímto

krokem se jednak přeuspořádají permutace vysílačů a rovnou k tomu i vypočtou šířky pásem využitých jednotlivými chromozomy.

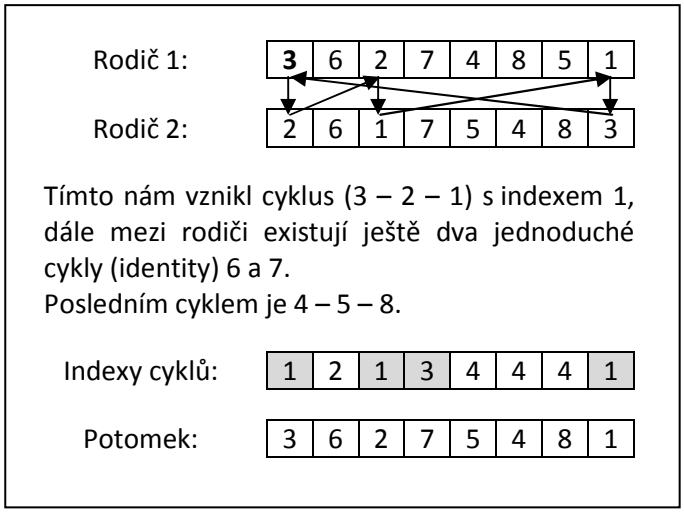
Algoritmus skončí (3) ve chvíli, kdy po určitý počet generací nedošlo k žádné změně v kroku (7) – to znamená ve chvíli, kdy se populaci po delší dobu nedaří vylepšovat aktuální chromozomy.

Výběr prvního rodiče v každé generaci je sekvenční (v každé generaci je každý chromozom jednou prvním rodičem pro crossover), zatímco druhý rodič se vybírá náhodně (4). Výběr druhého rodiče je náhodný, ale ne úplně – každý chromozom má pravděpodobnost výběru úměrnou tomu, jak moc je dobrý:

$$pravděpodobnost\ výběru\ chromozomu = \frac{velikost\ populace + 1 - pořadí\ chromozomu}{\sum pořadí\ chromozomů}$$

Kde je každý chromozom ohodnocen šířkou využitého pásma v přiřazení frekvencí a pořadí je pořadí chromozomů podle jejich hodnoty vzestupně. Tímto je zajištěno, že se budou lepší jedinci reprodukovat častěji než ti horší.

Na vybrané rodiče je poté aplikován *cyklický* crossover (5) za vzniku nového potomka. Cyklický crossover (Cycle crossover [Oxypedia]) hledá permutační cykly přes oba rodiče. Cykly hledáme tak, že si rodiče umístíme nad sebe, vezmeme první takový gen (*počátek*) z prvního rodiče, který ještě není v žádném cyklu, a podíváme se na hodnotu tohoto genu v druhém rodiči (*cílový* gen). Poté se v prvním rodiči přesuneme do genu se stejnou hodnotou, jako má *cílový* gen. Podíváme se do druhého rodiče na aktuální pozici – v něm je uložen nový *cílový* gen. Poslední dva kroky opakujeme do té doby, než je *cílový* gen rovný *počátečnímu* (dokončení cyklu). Každému takto nalezenému cyklu přiřadíme index (od jedné až do počtu cyklů). Nakonec geny z cyklů s lichými indexy zkopírujeme do potomka z prvního rodiče a geny ze sudých cyklů z druhého rodiče.



Každý potomek vzniklý výše popsaným křížením je zmutován (6 – zanesení náhody do genotypu). V tomto algoritmu používáme jednoduchý dvoubodový transpoziční mutační operátor – náhodně se vyberou dva geny a prohodí se (prohození pořadí dvou vysílačů).

Pokud je potomek lepší než slabší z rodičů (šířka využitého pásma je menší), pak je slabší rodič v populaci nahrazen tímto potomkem.

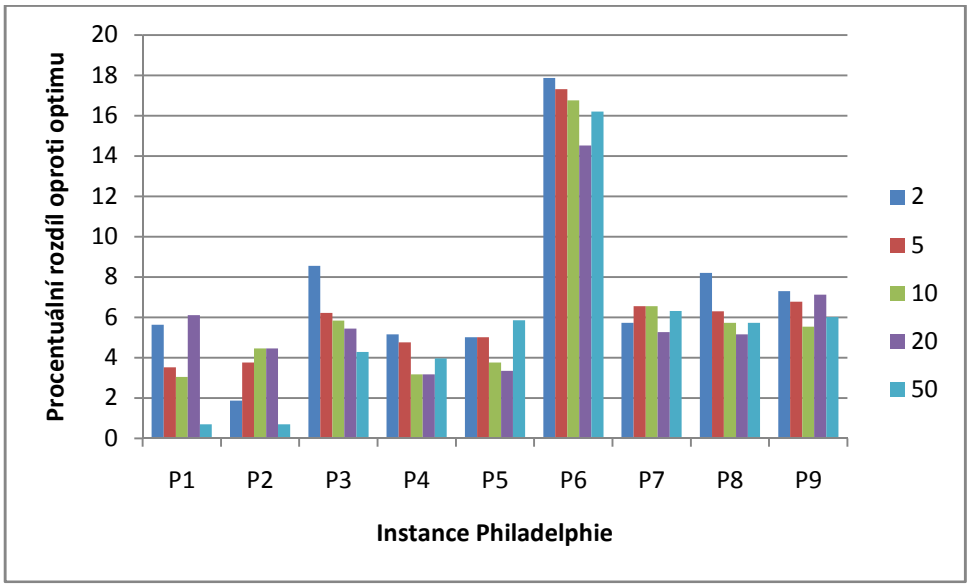
Na konec algoritmu nepřichází žádné další optimalizace – na pořadí vysílačů získané genetickým algoritmem je aplikován sekvenční algoritmus, který provede finální přiřazení frekvencí stejnou strategií jako při vyhodnocování genů (výběr nejmenší vhodné frekvence).

Nastavení parametrů

Hodnoty vynesené v níže uvedených grafech byly získány jedním spuštěním algoritmu (pro každou hodnotu) na všech instancích Philadelphie. Volba druhého parametru je uvedena u obou testů. Na grafech je zobrazeno, o kolik procent byl získaný výsledek horší, než je optimální řešení pro danou instanci Philadelphie.

Počet kroků

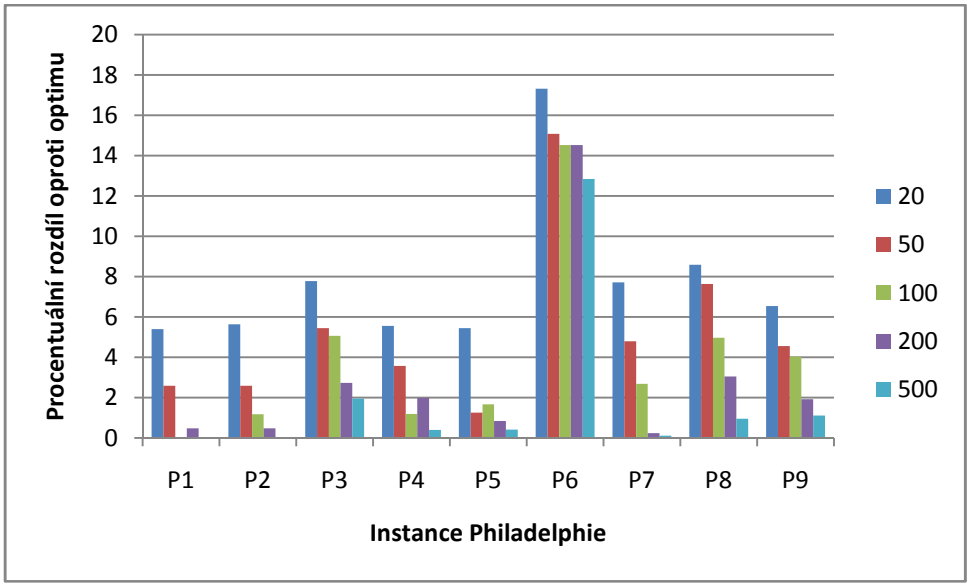
Podle počtu kroků je určován konec výpočtu permutačního algoritmu. Pokud se v posledních *loops* generacích nepovedlo zlepšit některé řešení v populaci, tak je algoritmus ukončen. *Velikost generace* (size) byla pro všechny testy *30*.



Na počtu kroků bez změny nutných k ukončení algoritmu záleží, ale výsledky nejsou o moc lepší. Čím více je těchto kroků, tím větší je šance, že se povede některé z řešení vylepšit. Avšak na druhou stranu toto vylepšení se většinou netýká nejlepšího řešení, takže nemáme žádnou záruku, že k němu někdy dojde. Doba výpočtu závisí na počtu kroků algoritmu lineárně.

Velikost populace

Velikost populace stanovuje počet chromozomů v jedné populaci (resp. počet řešení, kterými se najednou zabýváme). *Počet kroků* (loops) nutných k ukončení algoritmu byl v testech *5*.



Z dosažených výsledků je zřejmé, že na velikosti populace v permutačním algoritmu velmi záleží. Velké populace mají tendenci dosahovat velmi dobrých řešení, i přesto, že byl počet kroků nutných k ukončení algoritmu relativně malý. Doba výpočtu závisí lineárně na velikosti populace. V permutačním algoritmu je dobré upřednostňovat zvětšení tohoto parametru před zvětšováním počtu kroků.

4.3 Tabu search

Tabu search (metoda prohledávání se zákazy) je složitější variantou *local search*. Local search je programovací technika, při níž si neseme po celou dobu výpočtu jedno její řešení. Toto řešení se snažíme zlepšovat – hledáme „sousedy“ tohoto řešení, a pokud je soused lepší, pak aktuální řešení nahradíme tím sousedním. V případě, že v aktuálním sousedství neexistuje žádné lepší řešení, prohlásíme aktuální řešení za nejlepší (více v kapitole 6.2 Local search).

Tabu search na rozdíl od local search povoluje i zhoršující pohyby (*pohybem* nazýváme přechod od jednoho řešení k druhému – sousednímu – například změna frekvence přiřazené nějakému vysílači). V každé iteraci algoritmu je stávající řešení (jako řešení považujeme přímo vektor přiřazení frekvencí jednotlivým vysílačům) nahrazeno jeho nejlepším „sousedním“ řešením. Toto nové řešení však může být i horší. Aby se předešlo zacyklení, je nutné zakázat tahy zpět do již navštívených řešení. K tomu zavádíme historii tabu pohybů – je zakázáno obrátit tah, který jsme provedli v posledních *k* iteracích, parametr *k* nazýváme délka tabu historie. Algoritmus je ukončen po předem daném počtu kroků a jeho výsledkem je nejlepší řešení, které bylo v historii nalezeno. Pro tabu search jsou důležité zejména následující rozhodnutí:

- tvorba prvotního přiřazení
- hodnotící funkce – funkce, kterou se snažíme minimalizovat. Hraje roli při výběru nejlepšího jedince ze sousedství
- výběr sousedních řešení – algoritmus hledání sousedních řešení, v nichž budeme hledat nové řešení. Podle [Aardal03] je nejčastějším algoritmem tzv. 1-výměnové sousedství – neboli všechna řešení, která můžeme z aktuálního získat změnou jedné hodnoty (frekvence přiřazené jednomu vysílači)

- restrikce sousedních řešení – vygenerované sousedství bývá většinou moc velké na to, aby se dalo v rozumném čase prozkoumat a tak je třeba jej ještě nějak omezit
- délka tabu historie a celkový počet iterací

Implementovaný tabu search algoritmus

Tento algoritmus vychází z popisu [Hurley97]. Tabu search algoritmy popsané výše jsou vhodné pro minimalizaci interference při pevně daném frekvenčním spektru, ale už ne pro minimalizaci počtu frekvencí. Proto budeme postupovat trochu jinak:

```
(1)   vygeneruj počáteční řešení
      dokud jsou všechny podmínky neporušeny
            přenastav všechny vysílače s největší frekvencí na náhodné
                  nižší frekvence
            odeber největší frekvenci
(2)         spusť Tabu search algoritmus pro minimalizaci rušení
      konec
      vrať se k poslednímu platnému přiřazení // v aktuálním je porušení
```

Pozn.: Pro vytvoření počátečního řešení (1) je možné použít libovolný jiný algoritmus (viz. 7.2 Vstupní XML*).*

Vlastní tabu search algoritmus je v každém kole spuštěn na přiřazení frekvencí, ve kterém je porušena nějaká podmínka (vysílače, kterým byla přiřazena nejvyšší frekvence, jsou přeladěny na náhodně zvolené nižší frekvence a tím mohlo vzniknout porušení) a má za úkol toto porušení eliminovat. Pokud se mu to povede, může se pokračovat se snižováním počtu frekvencí, pokud však během daného počtu kroků není TS algoritmus schopen najít neporušující přiřazení, pak už se dále nepokračuje a předchozí přiřazení je považováno za výsledek algoritmu.

Hodnotící funkcí řešení je součet vah všech porušených podmínek.

Pro generaci a restrikci sousedství je používán algoritmus o*mezeného náhodného sousedství*, kdy je předem dána velikost sousedství (N) a každé řešení v sousedství je náhodně vytvořeno tak, že je z aktuálního řešení vybrán vysílač, který porušuje alespoň jednu podmínku, a je mu přiřazena náhodná jiná frekvence.

Kromě *krátkodobé historie* (kdy je zakázáno přiřadit jednomu vysílači stejnou frekvenci, kterou již měl v posledních *k* iteracích) je ještě zavedena *dlouhodobá historie*. Ta se stará o to, aby jeden vysílač nebyl měněn příliš často na úkor ostatních vysílačů. K realizaci dlouhodobé historie je dáno reálné číslo α, $0 < \alpha < 1$, které říká, že pokud počet změn vysílače ku počtu iterací je větší než α, pak je další změna tohoto vysílače považována za tabu pohyb.

Ze sousedství je vždy vybrán nejlepší pohyb, který není tabu (zakázaný). Avšak v případě, že je řešení získané některým tabu pohybem nejlepší v sousedství a zároveň nejlepší v celé historii všech řešení, tak je tento tabu pohyb vybrán.

Optimalizační algoritmus tabu search může vypadat přibližně takto:

```
dokud existuje nějaké porušení a nepočítám moc dlouho
      vygeneruj omezené náhodné sousedství
      vyhodnoť všechna řešení v sousedství
      vyber nejlepší ne-tabu pohyb (N)
      vyber nejlepší tabu pohyb (T)
      pokud je f(T) lepší než f(N) a zároveň je f(T) nejlepší v historii
            přejdi do T
```

```
      jinak
            přejdi do N
konec
```

Pozn.: f(X) je hodnota hodnotící funkce.

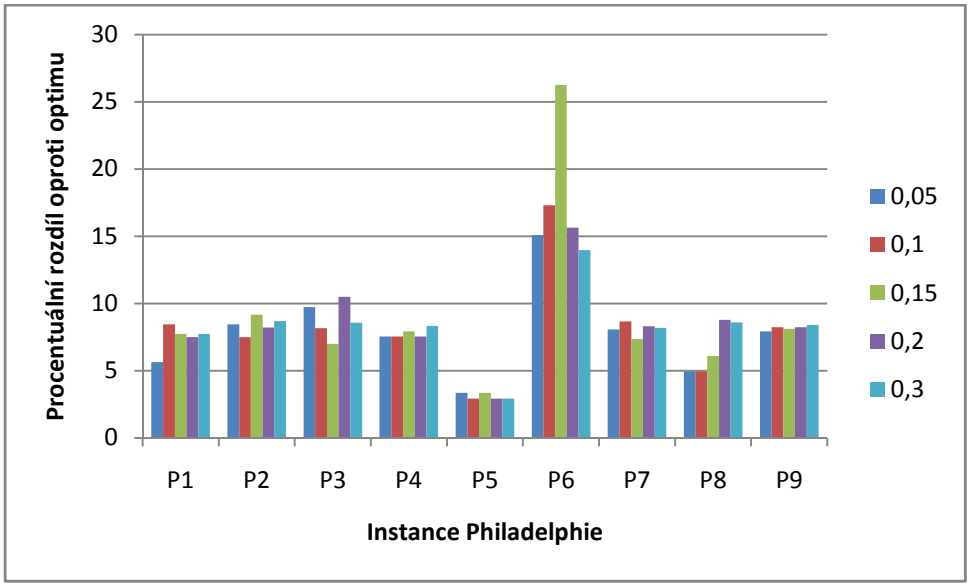
Nastavení parametrů

Hodnoty vynesené v níže uvedených grafech byly získány jedním spuštěním algoritmu (pro každou hodnotu) na všech instancích Philadelphie. Na grafech je zobrazeno, o kolik procent byl získaný výsledek horší, než je optimální řešení pro danou instanci Philadelphie.

Pro srovnání vlivu nastavení jednotlivých parametrů u tabu search byl jako počáteční algoritmus zvolen *sekvenční algoritmus* s počátečním setříděním podle váženého stupně s vypouštěním (*GeneralizedLargestFirstExclude*), vysílače byly přeuspořádávány metodou váženého stupně nasycení (*GSD)* a frekvence byla volena nejmenší možná (*Smallest*). Sekvenční algoritmus byl vybrán proto, že pokaždé dosáhne stejného výsledku. Volby netestovaných hodnot nastavení tabu search jsou uvedeny vždy u každého testu.

Dlouhodobá historie

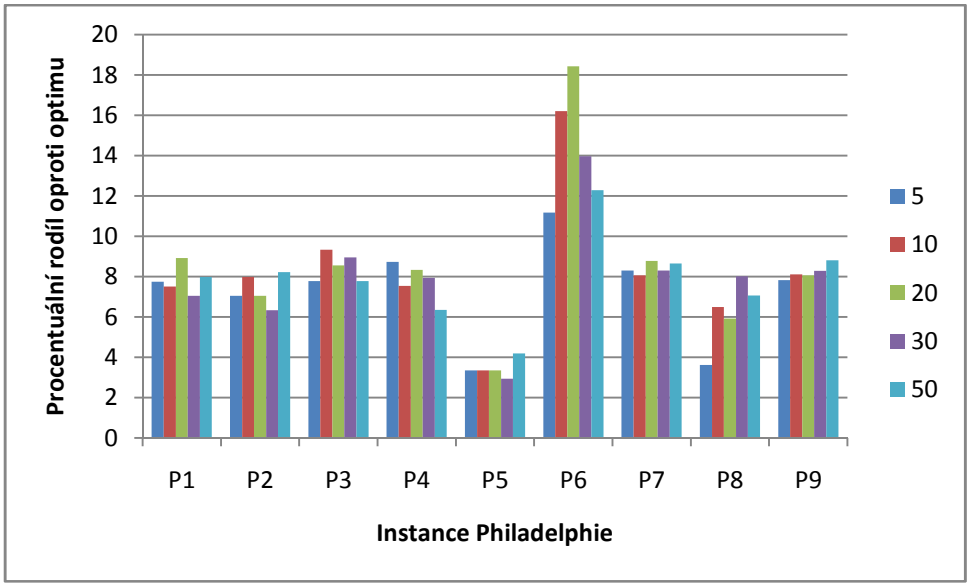
Parametr dlouhodobé historie (alpha) určuje největší povolený poměr počtu změn jednoho vysílače vůči celkovému počtu změn v historii. *Délka krátkodobé historie* (history) byla u pro všechny testy *10* a *počet smyček* (loops) *200*.



Z grafu není vidět žádná zřetelná závislost kvality výsledku na poměru dlouhodobé historie, ale je pravděpodobné, že menší hodnoty parametru alpha vedou k lepším výsledkům – méně často jsou prováděny zpětné změny. Doba výpočtu na dlouhodobé historii nijak nezávisí.

Krátkodobá historie

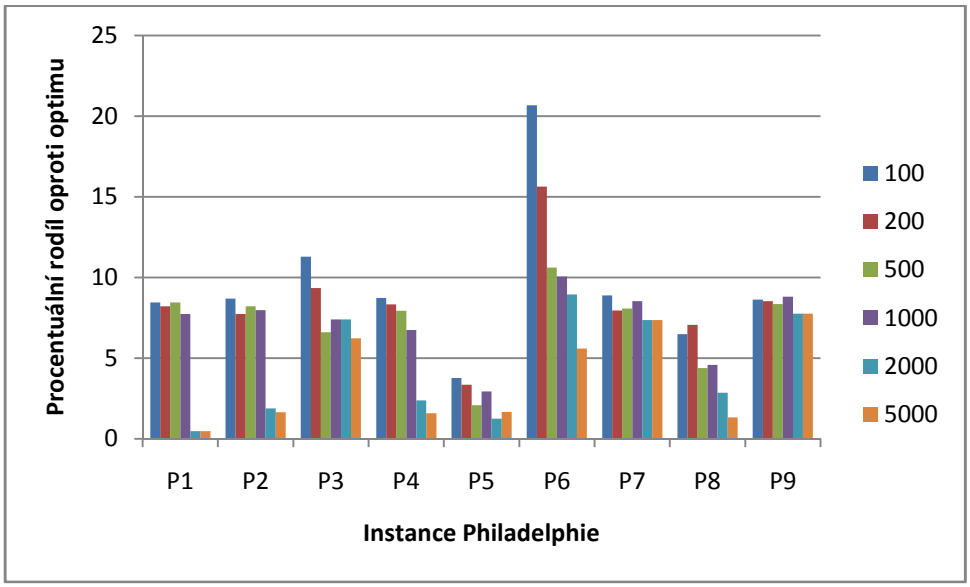
Délka krátkodobé historie (history) udává, počet tahů, který si tabu search pamatuje a zakazuje je vracet zpět. *Počet smyček* (loops) byl *200* a *poměr dlouhodobé historie* (alpha) *0,1*.



Na délce krátkodobé historie nezávisí ani kvalita výsledku ani doba výpočtu.

Počet smyček

Tento parametr určuje kolikrát maximálně je provedena vnitřní smyčka tabu search algoritmu. *Délka krátkodobé historie* (history) byla *10* a *poměr dlouhodobé historie* (alpha) *0,1*.



Kvalita výsledku velmi záleží na maximálním počtu kroků vnitřní smyčky, zvětšením tohoto parametru je dodávána algoritmu šance počítat i přesto, že s menší hodnotou by už skončil. Lepší kvalita výsledků je vykoupena lineární závislostí doby výpočtu na počtu smyček.

4.4 Simulované žíhání

Motivace pro simulované žíhání (simulated annealing) vznikla z pozorování mrznutí a krystalizování kapalin, popřípadě žíhání a tuhnutí kovů. Když se kapalina ochlazuje pomalu, tak se její atomy uspořádají velmi symetricky do stavu minimální energie – krystalu. Toho se dá využít i při tvorbě optimalizačních algoritmů.

Simulované žíhání je optimalizační algoritmus opět vycházející z local search a podobně jako tabu search povolující i zhoršující pohyby.

Simulované žíhání v každém kroku najde nějaký blízký stav, do kterého by se dalo přejít. Pokud je tento nový stav lepší (má menší energii), tak do něj přejde. Aby se algoritmus nezastavil v lokálním minimu, tak jsou s určitou pravděpodobností povoleny i zhoršující pohyby. Pravděpodobnost přechodu do horšího stavu je přímo úměrná aktuální *teplotě* látky a nepřímo úměrná energii, kterou nás tento přechod bude stát. Teplota je kontrolována mechanizmem chlazení - na začátku tuhnutí je látka horká a tak umožňujeme dělat i velké změny, s postupem času teplota klesá a tak je stále složitější provádět zhoršující tahy, až nakonec látka úplně ztuhne a jsou povoleny jen přechody do lepších stavů.

Základní vlastnosti ovlivňující chování simulovaného žíhání:

- tvorba prvotního přiřazení
- mechanismus hledání pohybů
- hodnotící funkce
- chladící strategie
- počáteční a koncová teplota

Implementované simulované žíhání

Pro implementaci simulovaného žíhání byl vybrán popis [Hurley97]. Podobně jako tabu search je i původní verze simulovaného žíhání navržená spíše na minimalizaci interference než minimalizaci počtu frekvencí a proto je základní schéma algoritmu podobné, pouze v kroku (2) se spustí algoritmus simulovaného žíhání (více v kapitole Implementovaný tabu search algoritmus). Schéma vlastního algoritmu simulovaného žíhání se však od tabu search podstatně liší (pro vyváznutí z lokálního minima se používá teplota místo historie tabu pohybů):

```
(1)   máme řešení R s energií E
(2a)  zinicializuj T
(2b)  dokud T > Tmin
(3)         pro i od 1 do počtu smyček
(4)               vytvoř nové řešení N z aktuálního řešení R
(5)               spočti energii EN
                  pokud EN < E nebo náhoda < pravděpodobnost(EN – E, T)
                        přejdi do stavu N s energií EN // R = N, E = EN
            konec
(6)         ochlaď látku // zmenši T
      konec
```

Pozn.: Náhoda je náhodné číslo (v každém kroku jiné) a pravděpodobnost je pravděpodobnostní funkce pro přechod do stavu horšího o $E_N - E$ při teplotě T. Například 0 < náhoda < 1 a pravděpodobnost = $e^{-\frac{E_N - E}{T}}$.

Jako energii (1 a 5) řešení bereme pro každou porušenou omezující podmínku takovou hodnotu, o kterou byla podmínka porušena. Cílem simulovaného žíhání je vygenerování řešení, které má nulovou energii (žádná podmínka není porušena).

Pro výše popsané pravděpodobnostní schéma je nutné, aby byla teplota po celou dobu běhu větší než 0. Na startovní a koncové teplotě závisí doba výpočtu algoritmu a také jeho správný výsledek. Startovní teplota (2a) se dá ovlivnit parametrem *acceptratio*. Před startem algoritmu je teplota

nastavena tak, aby byl poměr počtu přijatých tahů a celkového počtu tahů minimálně takový, jako je tento parametr. Čím větší je tato hodnota, tím větší je startovní teplota a tím déle se bude chladit. Cílová teplota je přímo ovlivnitelná v nastavení parametru *end*. Je očekávána nízká hodnota (bod ztuhnutí) – např. defaultní 0,001. Na počtu smyček provedených při jedné teplotě velmi závisí rychlost a výsledky algoritmu. Uživatel má možnost zadat koeficient startovního počtu smyček (koeficient udává, kolikrát větší má být počet smyček než celkový počet vysílačů).

Pro generování sousedního řešení ze stávajícího (4) je použita strategie omezeného jednoduchého pohybu, kde jeden krok je změna frekvence na náhodnou jinou frekvenci u vysílače, který porušuje alespoň jednu podmínku. Takovéto změny vedou k dobrým výsledkům, protože přenastavují ty vysílače, které přiřazení kazí.

Na rychlosti chlazení (6) a dalších jeho vlastnostech, které chlazení má, velmi závisí rychlost a kvalita algoritmu (většinou jdou bohužel proti sobě). V programu FAP jsou k dispozici tři různé chladící strategie (parametrizované chladícím faktorem α, $0 < \alpha < 1$):

- *geometrické chlazení* – v každém kroku algoritmu je teplota snížena na α-násobek původní teploty. To má za následek rozumně rychlé chlazení zezačátku a pomalé tuhnutí na konci, což je žádaný efekt.
- *geometrické chlazení se zvyšováním počtu smyček* [Costa93] – teplota se snižuje stejně jako v geometrickém chlazení, navíc je však v každém kroku zvýšen počet provedených smyček – *PočetSmyček* = *PočetSmyček* / α – a látce trvá déle, než se ochladí.
- *chlazení závislé na výsledcích poslední iterace* [Hurley95] – tato chladící strategie bere v úvahu výsledky sousedů vzniklých v poslední iteraci algoritmu. Čím rychleji se algoritmus zlepšuje, tím rychleji teplota klesá.

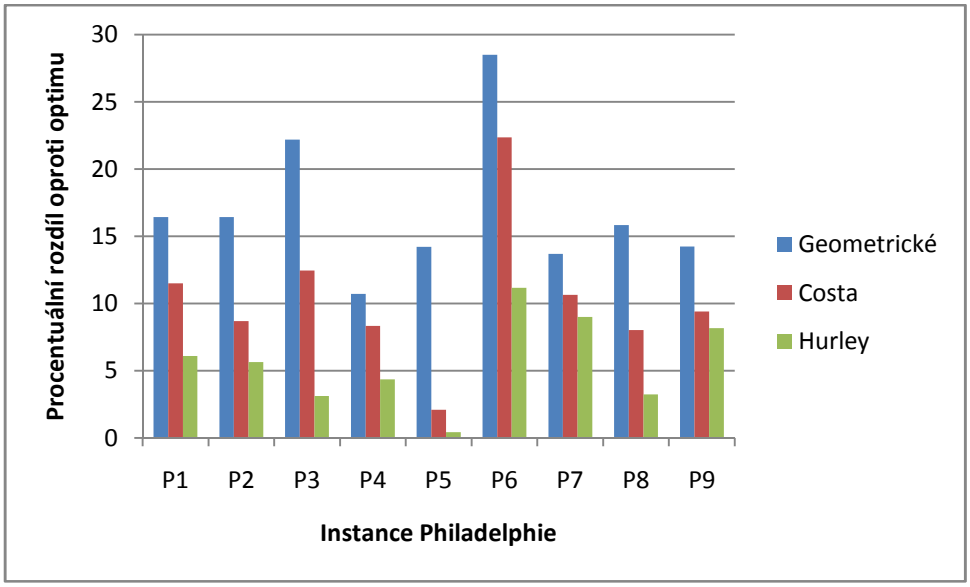
Nastavení parametrů

Hodnoty vynesené v níže uvedených grafech byly získány jedním spuštěním algoritmu (pro každou hodnotu) na všech instancích Philadelphie. Na grafech je zobrazeno, o kolik procent byl získaný výsledek horší, než je optimální řešení pro danou instanci Philadelphie.

Pro srovnání vlivu nastavení jednotlivých parametrů u simulovaného žíhání byl, stejně jako u tabu search, za počáteční algoritmus zvolen *sekvenční algoritmus* s počátečním setříděním podle váženého stupně s vypouštěním (*GeneralizedLargestFirstExclude*), vysílače byly přeuspořádávány metodou váženého stupně nasycení (*GSD)* a frekvence byla volena nejmenší možná (*Smallest*).

Chladící strategie

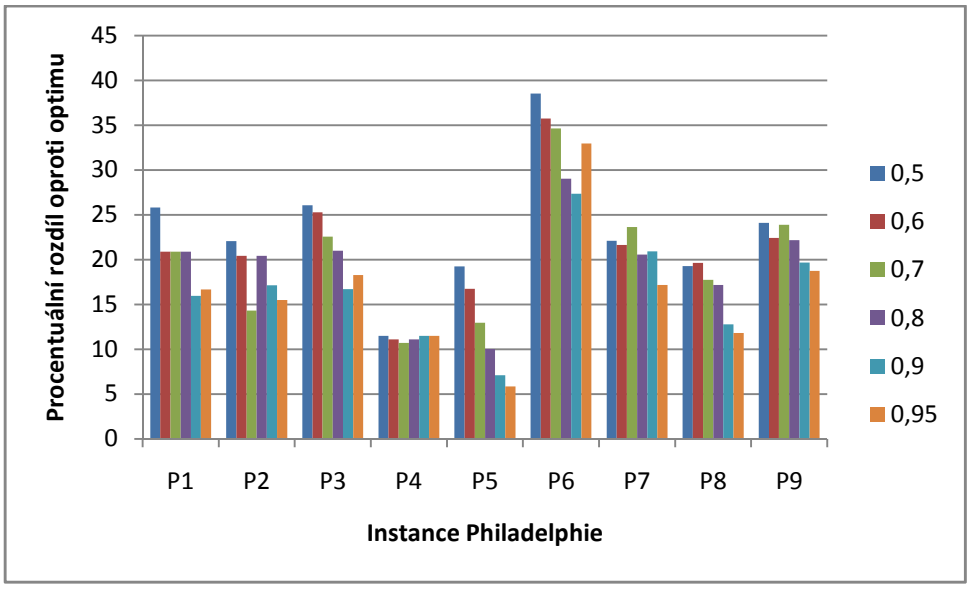
Na výběru chladící strategie (cooling) závisí rychlost tuhnutí látky. *Chladící faktor* (alpha) byl ve všech testech 0,7, *faktor počtu smyček* (loops) byl 0,5, počáteční *poměr přijímaných tahů* (acceptratio) byl 0,05 a *koncová teplota* (end) 0,001.



Nejlepší chladící strategií se ukázalo být nejpomalejší chlazení – závislé na výsledcích poslední iterace (Hurley), kterým byly dosahovány velmi kvalitní výsledky, chlazení se zvyšováním počtu smyček (Costa) se ukázalo být vcelku rychlé a spolehlivé. Výsledky simulovaného žíhání s geometrickým chlazením byly získány velmi rychle, zato však nebyly nijak dobré, často se touto chladící strategií nepovedlo ani zlepšit počáteční přiřazení získané sekvenčním algoritmem. Costa chlazení bylo řádově (cca 5-10 krát) pomalejší než geometrické, Hurley chlazení bylo ještě o 2 řády pomalejší.

Chladící faktor

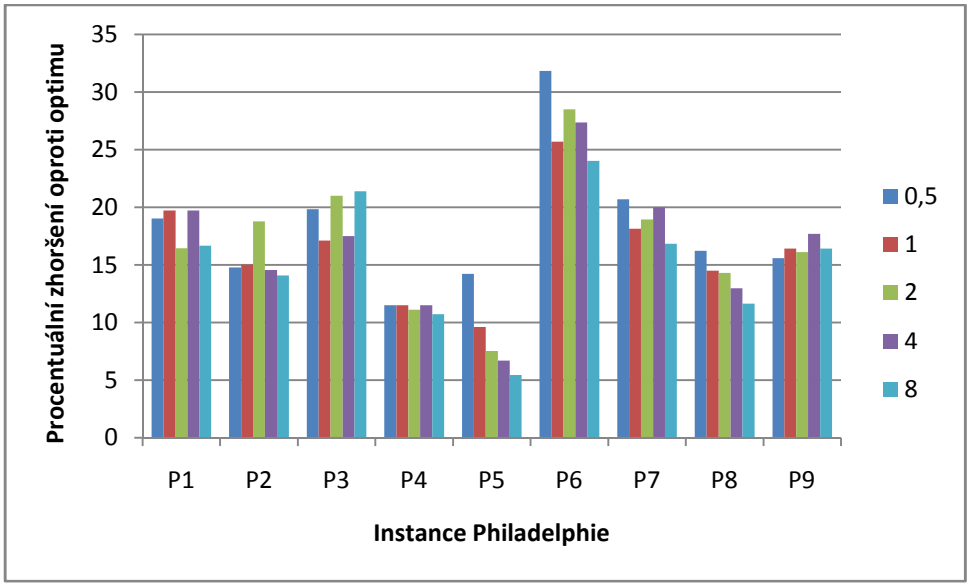
Tento parametr (alpha) určuje rychlost tuhnutí látky, čím je větší, tím látka tuhne pomaleji. Jako *chladící strategie* (cooling) byla použita geometrická (*Geometric*), *faktor počtu smyček* (loops) byl 1, počáteční *poměr přijímaných tahů* (acceptratio) byl 0,3 a *koncová teplota* (end) 0,001.



Z grafu je patrné, že kvalita výsledku získaného simulovaným žíháním velmi závisí na hodnotě chladícího faktoru, čím pomaleji je látka chlazena (faktor je větší), tím lepší je výsledek. Cena za lepší výsledky je bohužel veliká – doba výpočtu roste s hodnotou chladícího faktoru exponenciálně!

Faktor počtu smyček

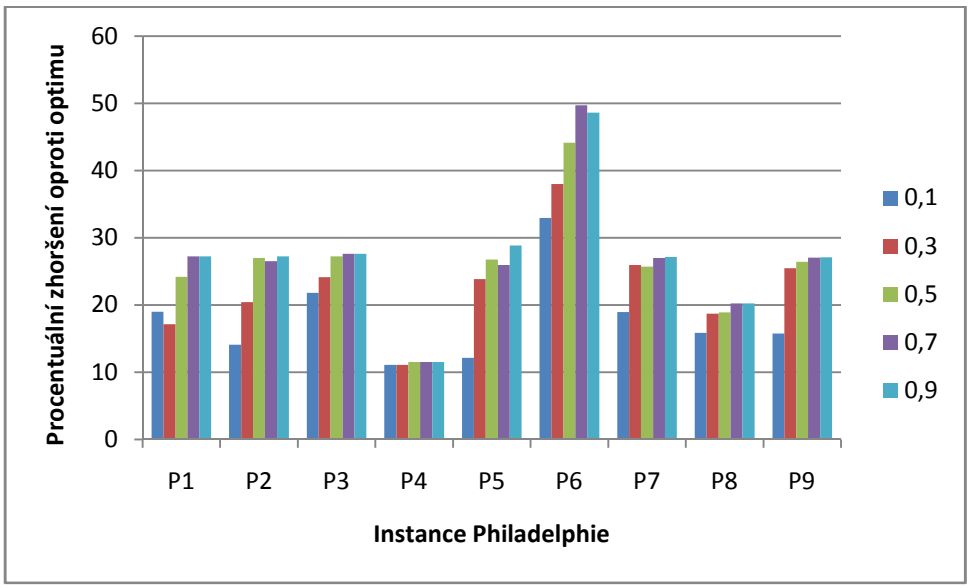
Faktor počtu smyček (loops) určuje, kolikrát větší bude počáteční počet smyček, než je celkový počet vysílačů. Jako *chladící strategie* (cooling) byla použita geometrická (*Geometric*), *chladící faktor* (alpha) byl 0,7, počáteční *poměr přijímaných tahů* (acceptratio) byl 0,05 a *koncová teplota* (end) 0,001.



Čím větší je počet smyček algoritmu, tím lepší jsou získané výsledky, avšak také doba výpočtu (ta roste lineárně s počtem smyček). U geometrického chlazení se spíše vyplatí zvyšovat tento parametr než chladícího faktor.

Poměr přijímaných tahů

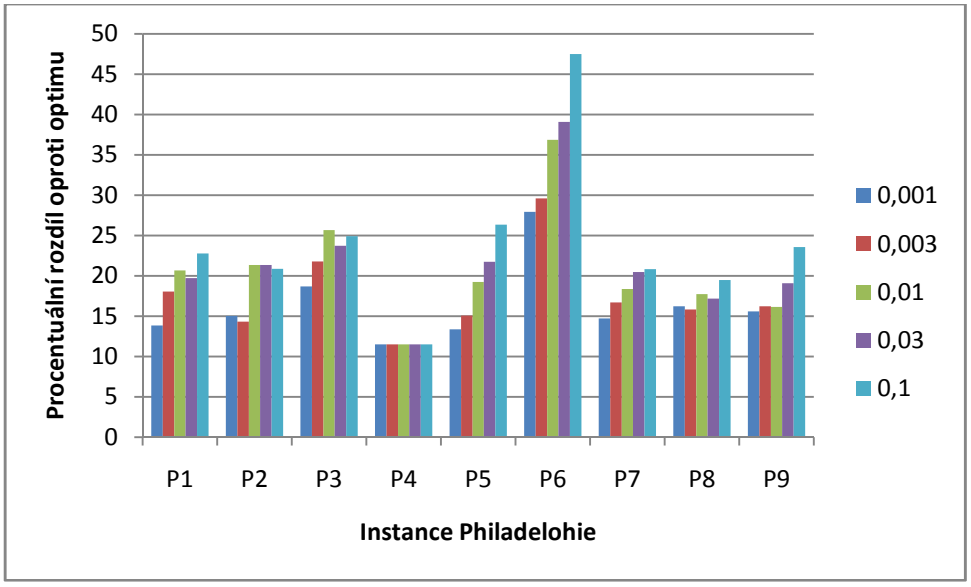
Počáteční poměr přijímaných tahů (acceptratio) určuje minimální poměr přijatých „sousedních“ řešení v prvním kole žíhacího algoritmu. Čím větší je tento parametr, tím je větší počáteční teplota, tím více zhoršujících změn je na začátku přijato a tím déle trvá chlazení látky. Jako *chladící strategie* (cooling) byla použita geometrická (*Geometric*), *chladící faktor* (alpha) byl 0,7, *faktor počtu smyček* (loops) byl 0,5 a *koncová teplota* (end) 0,001.



Z grafu je patrné, že lepších výsledků je dosahováno s nižším poměrem přijímaných tahů (tím pádem i nižší počáteční teplotou a menší šancí na zkažení aktuálního řešení hned zpočátku). Doba výpočtu nezávisí na hodnotě tohoto parametru.

Koncová teplota

Tento parametr (end) udává teplotu, při níž látka ztuhne a nadále už nebude přijímat zhoršující tahy. Jako *chladící strategie* (cooling) byla použita geometrická (*Geometric*), *chladící faktor* (alpha) byl 0,7, *faktor počtu smyček* (loops) byl 0,5 a počáteční *poměr přijímaných tahů* (acceptratio) 0,05.



Čím menší je koncová teplota, tím větší má algoritmus šanci na získání lepšího výsledku (déle trvá chlazení při nízké teplotě a delší dobu jsou přijímány jen nepatrně zhoršující změny). Čím menší je koncová teplota, tím algoritmus počítá déle (doba výpočtu je logaritmicky nepřímo úměrná koncové teplotě).

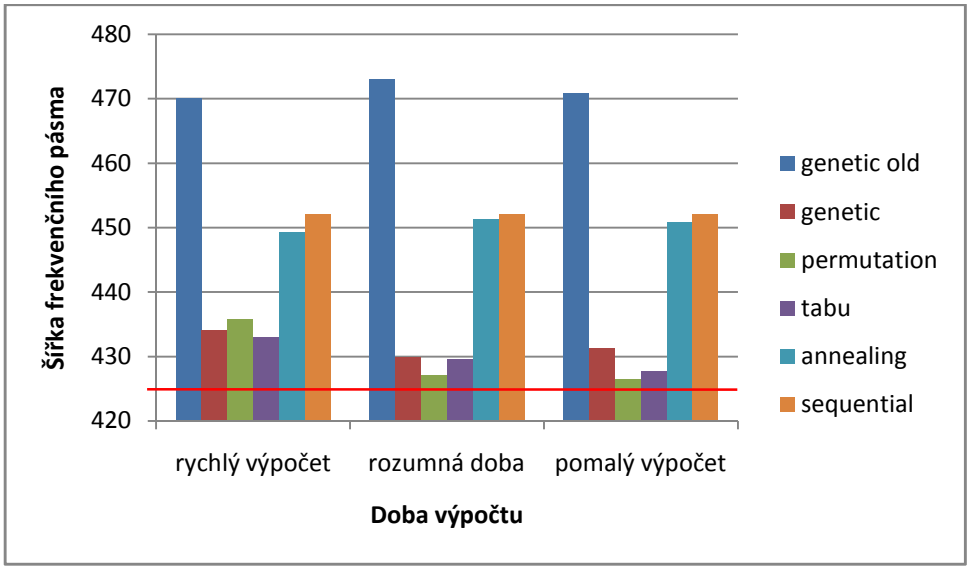
5 Porovnání výsledků jednotlivých algoritmů

V této kapitole se nachází srovnání výsledků získaných jednotlivými implementovanými algoritmy na třech vybraných instancích Philadelphie – P1, P3 a P6. Právě tyto instance byly vybrány proto, že každá z nich má odlišný charakter (požadavky na počty vysílačů se v nich liší) a jsou zároveň dostatečně malé, aby na nich i pomalé algoritmy stihly dopočítat.

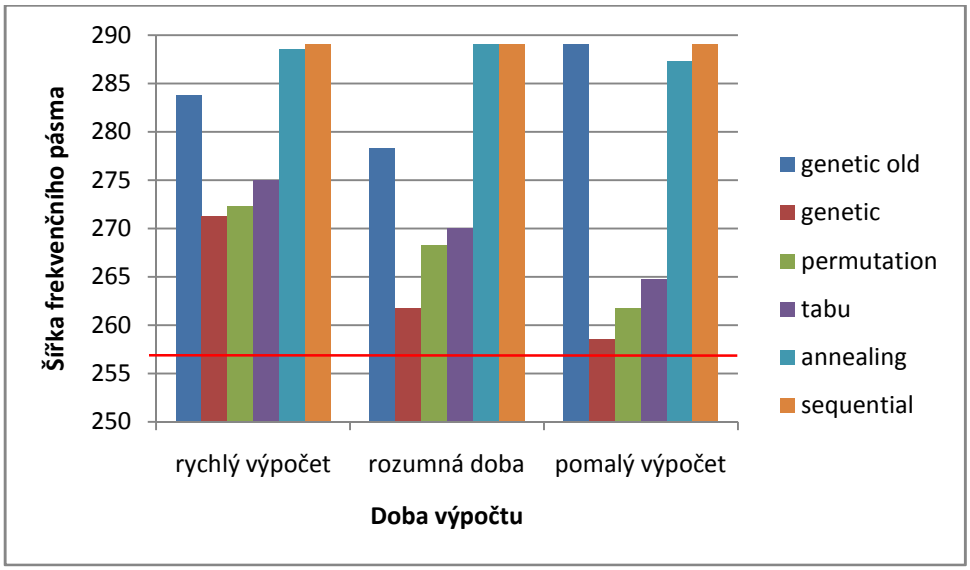
Implementované algoritmy jsou srovnávány ve třech kategoriích – *rychlý výpočet* (cca 30 sekund), *rozumná doba* výpočtu (cca 5 minut) a *pomalý výpočet* (cca 45 minut výpočetního času). *Pozn.: Uvedené výpočetní časy byly měřeny při samostatném běhu programu FAP na PC s Intel Core 2 Duo 2,2 GHz, 2 GB RAM a Windows XP Professional.*

Volby parametrů jednotlivých algoritmů byly vybrány tak, aby algoritmus dosáhl ve stanoveném časovém limitu co nejlepších výsledků. Každý z algoritmů byl pro daný test spuštěn čtyřikrát a hodnoty vynesené v grafech jsou vždy průměry z těchto čtyř výsledků. Horizontální červené čáry v grafech naznačují hodnoty optimálních řešení jednotlivých instancí Philadelphie.

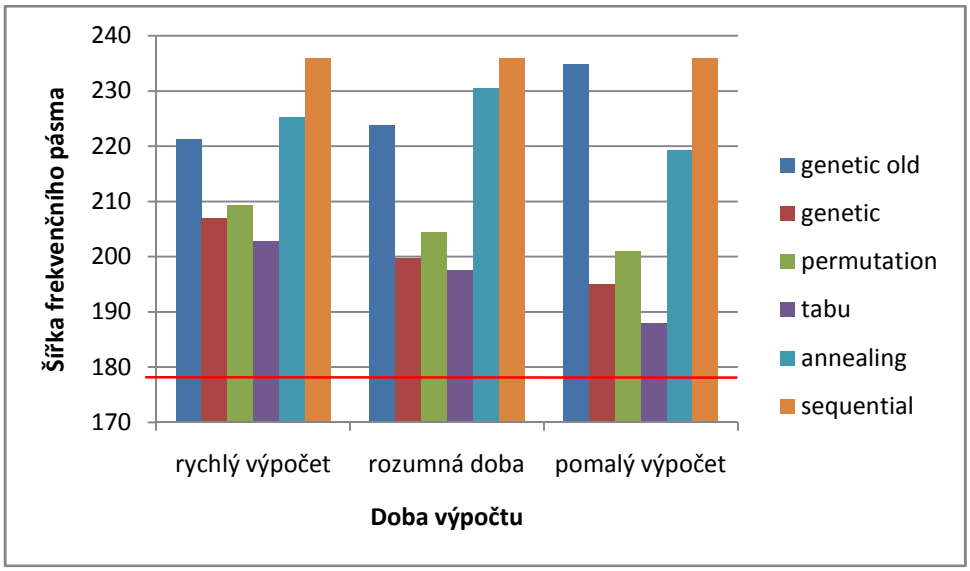
Philadelphia P1



Philadelphia P3



Philadelphia P6



Nejúspěšnějšími algoritmy mezi všemi implementovanými byly oba genetické algoritmy (upravený třífázový a permutační genetický algoritmus) a tabu search. Výsledky získané sekvenčním algoritmem jsou uvedeny pro porovnání s optimalizačními algoritmy (simulované žíhání a tabu search), které z nich vycházely. Simulované žíhání potřebuje na výpočet odlišné nastavení (především pomalejší chladící strategii) vyžadující daleko více času a je méně spolehlivé (často skončí se špatným výsledkem). Výsledky získané neupraveným třífázovým genetickým algoritmem jsou velmi špatné a nezávisí na době, která je algoritmu přidělena.

6 Neimplementované algoritmy

6.1 Exhaustive search

Řešení problému hrubou silou zcela bez pochyby vede k optimálním výsledkům. Bohužel, zjištění minimálního počtu frekvencí (a nalezení odpovídajícího přiřazení) je NP-těžký problém. Navíc je tu prohledávaný stavový prostor obrovský – k tomu, abychom zjistili minimální počet frekvencí, musíme vyzkoušet, že nižší počet frekvencí přiřadit nelze. I kdyby nám někdo zadal hodnotu optima, tak by nalezení správného řešení trvalo obecně čas n^m, kde n je počet frekvencí a m počet vysílačů, což je neúnosné.

Při řešení problému backtrackingem můžeme využít znalosti podmínek a vytvářet jenom taková řešení, kde není porušena žádná podmínka.

6.2 Local search

Local search je nejjednodušší optimalizační technikou pro řešení FAPu. Jako vstup bere přípustné řešení FAPu a nahrazuje jej lepším řešením z nějaké omezené podmnožiny všech řešení (té množině říkáme sousedství). Pokud LS nenajde žádné lepší řešení v sousedství, tak skončí. Problémem LS je, že může uváznout v *lokálním minimu* – řešení, ze kterého se nedá jednoduše dostat do lepšího, ale přesto nějaké lepší řešení existuje. Při tvorbě LS je potřeba zamyslet se nad volbou sousedství, protože není možné prozkoumat všechna řešení, ale je nutné jich prozkoumat dostatečně mnoho, aby byla šance v nich najít ta lepší. Mezi nejpoužívanější techniky volby sousedství patří:

- *plné sousedství* – získané všemi možnými jednobodovými tahy (změna frekvence na jednom vysílači) – celkem je jeho velikost *m * (n - 1)*, kde *m* je počet vysílačů a *n* počet frekvencí
- *omezené sousedství* – jednobodové tahy jsou prováděny pouze s vysílači, které porušují nějakou podmínku
- *omezené náhodné sousedství* – předem je dán počet řešení v sousedství, každé z nich je vytvořeno náhodnou změnou frekvence na vysílači porušující libovolnou z podmínek

Local search bývá v této podobě užíván jen k výpočtům určitých podúloh (například v [Park96]), jelikož je velká šance, že uvízne v lokálním minimu a k tomu globálnímu se nedopočítá. Proto z této techniky vycházejí další optimalizační techniky jako například t*abu search* nebo s*imulované žíhání*, které povolují i zhoršující tahy a tím jsou schopny dostat se z lokálního minima.

6.3 Ant colony optimization

Algoritmy inspirované chováním mravenců v přírodě. K výpočtu máme k dispozici kolonii „mravenců“ (jednoduchý výpočetní modul). Každý z těchto mravenců řeší zadaný problém a je ovlivňován řešením ostatních mravenců.

Mravenec může být interpretován jako sekvenční hladový algoritmus, který iterativně vytváří řešení problému tvorbou jednotlivých tahů. Výběr pohybu je ovlivňován dvěma parametry, *atraktivitou* – přímo závisí na kvalitě pohybu (kolik nás bude stát – porušené podmínky, …) a *silou feromonové stopy* – výsledky tahů prováděných ostatními mravenci. Feromonové stopy jsou upraveny vždy, když mravenci dokončí své tahy, pro jejich inicializaci je potřebné umět spočítat dolní odhad řešení problému. Řešení získaná jednotlivými mravenci mohou být dále vylepšována, například pomocí local search.

Jeden z algoritmů řešících MI-FAP simulací na kolonii mravenců je popsán v [Maniezzo00].

6.4 Artificial neural networks

Neuronové sítě v programování vychází z principů neuronových soustav živočichů. Každý neuron nese nějakou drobnou informaci (je v nějakém *stavu* – drží hodnotu nějaké proměnné daného problému) a dohromady dávají dohromady řešení celého problému. Výpočet probíhá tak, že každý neuron dynamicky mění svůj stav v závislosti na stavech okolních neuronů. Tímto procesem je optimalizována hodnotící funkce. Mezi hlavní rozhodnutí, která musí při tvorbě neuronové sítě padnout, jsou vztah neuronu a řešení, hodnotící funkce, váhy jednotlivých propojení mezi neurony (synapse) a pravidla pro upravení stavu (funkce okolních stavů a synapsí).

Běžně bývá neuronová síť pro řešení FAPu reprezentována následovně: každá možná dvojice (vysílač, frekvence) je reprezentován jedním neuronem. Neurony jsou propojeny, když spolu odpovídající vysílače sousedí v grafu rušení. Neuronové sítě bývají většinou používány k minimalizaci rušení při pevně daném frekvenčním pásmu a tomu odpovídají i vyhodnocovací funkce – většinou vážený součet hodnot porušených podmínek.

Jedním z prvních algoritmů řešících FAP pomocí neuronových sítí je [Kunz91].

7 Program

Program FAP umístěný na přiloženém CD slouží k počítání řešení problému přiřazování frekvencí, konkrétně modelu MS-FAP. V programu jsou naimplementovány všechny algoritmy zmíněné v kapitole 4 Implementované algoritmy, tedy sekvenční algoritmus, simulované žíhání, tabu search a dva genetické algoritmy.

Program si načte zadání problémů ze vstupního souboru a poté, co spočte výsledky jednotlivých problémů, je zapíše do odpovídajících výstupních souborů. Mimo to během výpočtů ještě vytváří logy, ve kterých může zadavatel najít přesnější informace o průběhu výpočtu.

V této kapitole následuje stručné seznámení s programem, jeho ovládání, popis souborů se zadáním problémů a výstupních souborů. Technické informace o programu jsou uvedeny v kapitole 8 Technický popis programu a programátorská dokumentace je umístěna na přiloženém CD.

7.1 Ovládání programu

Aplikace je dostupná ve formě programu spustitelného z příkazové řádky (FAP.exe). Jako jediný parametr očekává jméno XML souboru se zadáním problémů, které má řešit, a definicí algoritmů, jimiž je má řešit.

Příklad: `> FAP.exe example.xml`

Program spustí všechny testy v souboru a do odpovídajících výstupních souborů (podle zvoleného nastavení) vypíše spočtené výsledky. Během výpočtu vypisuje na obrazovku informace o aktuálním průběhu algoritmu.

7.2 Vstupní XML

Zadání každého FAP problému je uloženo v XML souboru. Celý XML dokument musí být validní. Soubor se zadáním je uspořádán do *testů*, každý test odpovídá jednomu problému FAP. Každý test má jedno zadání (mapu) a libovolný počet (větší než nula) algoritmů, kterými má být dané zadání vyřešeno. Všechna řešení jednoho testu budou ve výsledcích za sebou a zadavatel si poté může vybrat nejlepší z nich.

```
<?xml version=”1.0” ?>
<settings output=”…” log=”…” />
<test>
      <map>…</map>
      <algorithm>
            …
            …
            …
      </algorithm>
</test>
…
…
…
<test>
      <map>…</map>
      <algorithm>
            …
            …
            …
      </algorithm>
</test>
```

Ukázkové schéma vstupního souboru.

Každý soubor se zadáním by měl mít nastavené cesty k výstupním souborům, do těchto souborů bude prováděn výpis dosažených výsledků. Výběr výstupních souborů se provádí pomocí kořenového elementu *settings* a jeho atributů *output* a *log*, které určují cestu k výstupnímu (resp. logovacímu) souboru. Cesty souborů jsou relativní vzhledem k adresáři se spuštěným programem FAP.

Dále je nutné specifikovat zadání problémů a vybrat algoritmy, kterými mají být problémy řešeny. K tomuto účelu slouží elementy *test*. V každém testu je (pomocí vnořených elementů *map* a *algorithm*) zadán jeden problém a jsou v něm vybrány algoritmy na jeho řešení. Nepovinné atributy elementu *test*, *output* a *log* určují, stejně jako v případě elementu *settings*, cestu k výstupnímu a logovacímu souboru tohoto testu. Pokud jsou vynechány, je použito globální nastavení. Dále je možné každý test pojmenovat textovým atributem *name*, toto jméno pak bude vystupovat i ve výsledcích.

Pozn.: Pro více testů není možné uvést stejné jméno výstupních souborů. V tomto případě by došlo pouze k uložení výsledků posledního testu (předchozí by jimi byly přepsány).

Zadání problému (umístění vysílačů)

K zadání problému testu slouží element *map*, jehož atributem *name* lze pojmenovat vytvářenou mapu. Element map musí obsahovat právě jeden ze dvou následujících podelementů, které přesně určují výběr mapy (plná kolečka znamenají element, prázdná atribut):

- *philadelphia* – základní testovací instance (viz. kapitola 3.1 Philadelphia), zadání parametrů Philadelphie je možné provádět dvěma rozdílnými způsoby:
  - výběr známé instance Philadelphie
    - *instance* (celé číslo, 1 – 9) – identifikátor problému, v případě, že je uvedeno číslo instance, tak se neberou v potaz žádná další nastavení
  - přímé zadání počtu vysílačů a vektoru znovupoužitelných vzdáleností
    - *demands* – seznam 21 celočíselných počtů vysílačů na jednotlivých anténách (resp. požadovaných frekvencí)
    - *reuse* (reálná čísla oddělená mezerami) – vektor znovupoužitelných vzdáleností
- *coordinates* – mapa zadaná pomocí souřadnic
  - *antena* (opakovatelný) – zadání polohy a počtu frekvencí na anténě
    - *x*, *y* (povinné, reálné, nezáporné) – souřadnice umístění antény
    - *demand* (přirozené číslo, default = 1) – počet vysílačů na této anténě (počet požadavků na frekvence)
  - *reuse* – stejný význam jako element *reuse* u zadání Philadelphie

```
<map>
  <philadelphia instance="4" />
</map>
…
<map>
  <philadelphia>
    <demands>8 25 8 9 2 5 8 52 77 28 3 15 31 5 36 57 28 8 9 13 8</demands>
    <reuse>3.2 2.5 1 1 1 0</reuse>
  </philadelphia>
</map>
…
<map>
  <coordinates>
    <antena x="0.5" y="0" demand="10" />
```

```
    <antena x=”3” y=”2.5” demand=”4” />
    …
    <reuse>5 3 2 2 0</reuse>
  </coordinates>
</map>
```

Příklad zadání map.

Výběr algoritmů pro řešení problému

Kromě zadání problému je ještě třeba programu říci, jakými algoritmy se má pokusit tento problém vyřešit. Každý problém může být řešen více algoritmy (tyto výsledky pak budou ve výstupních souborech za sebou). Jednotlivé algoritmy musí být dohromady uzavřeny do jednoho elementu *algorithm* a jejich výběr je následující:

- *sequential* – sekvenční algoritmus, viz. kapitola Implementovaný sekvenční algoritmus
  - *ordering* (defaultně *GeneralizedLargestFirstExclude*) – prvotní setřídění, jedno z následujících: *LargestFirst*, *LargestFirstExclude*, *SmallestLast*, *GeneralizedLargestFirst*, *GeneralizedLargestFirstExclude*, *GeneralizedSallestLast*, *Random*
  - *next* (defaultně *Sequential*) – strategie výběru následujícího vysílače, jedno z: *Sequential*, *GSD*
  - *frequency* (defaultně *Smallest*) – strategie výběru frekvence pro vybraný vysílač, jedno z: *Smallest*, *Occupied*, *SmallestOccupied*, *HeaviestOccupied*
  - *all* (0 nebo 1, defaultně 0) – v případě, že je rovno 1, tak postupně pustí všech 56 možností algoritmu (všechny kombinace strategií)
- *genetic* – genetický algoritmus, viz. kapitola Třífázový genetický algoritmus
  - *size* (přirozené, defaultně 100) – velikost populace
  - *loops* (přirozené, defaultně 50) – počet vytvářených generací
  - *mutation* (reálné 0 – 1, defaultně 0,01) – procentuální pravděpodobnost mutace jednoho genu v chromozomu
  - *original* (0 nebo 1, defaultně 0) – použít původní algoritmus podle [Fu06] bez úprav zmíněných v kapitole Úprava algoritmu
- *permutation* – permutační genetický algoritmus, viz. kapitola Permutační algoritmus
  - *size* (přirozené, defaultně 100) – velikost populace
  - *loops* (přirozené, defaultně 50) – počet generací, které musí uplynout beze změny hodnoty některého z genů k lepšímu, aby algoritmus skončil
- *annealing* (optimalizační algoritmus, potřebuje výstup jiného algoritmu, na kterém dále pracuje) – simulované žíhání, viz. kapitola Implementované simulované žíhání, prvotní algoritmus je zadán jako podelement, může to být libovolný algoritmus (dokonce i optimalizační, ale nakonec musí být někdy zadán standardní algoritmus)
  - *alpha* (reálné číslo 0 – 1, defaultně 0,9) – faktor klesání teploty
  - *loops* (reálné číslo > 0, defaultně 1) – koeficient počtu běhů každého kola algoritmu v závislosti na počtu vysílačů (počtu požadovaných frekvencí) – počet běhů = loops * počet vysílačů
  - *acceptratio* (reálné číslo 0 – 1, defaultně 0,3) – minimální počáteční poměr přijatých tahů. Na základě něj je stanovena startovní teplota
  - *end* (reálné číslo 0 – 1, defaultně 0,001) – nejmenší možná hodnota teploty (když pod ní teplota klesne, algoritmus skončí)
  - *cooling* (defaultně *Geometric*) – chladící strategie, jedna z: *Geometric*, *Costa*, *Hurley*

    - *all* (0 nebo 1, defaultně 0) – v případě, že je rovno 1, tak potupně spustí algoritmus se všemi třemi chladícími strategiemi
    - prvotní algoritmus
- *tabu* (optimalizační algoritmus, platí to samé, co pro annealing, viz. výše) – tabu search, viz. kapitola Implementovaný tabu search algoritmus
    - *alpha* (reálné číslo 0 – 1, defaultně 0,1) – maximální poměr změn frekvence vysílače, při kterém tento pohyb nebude považován za tabu (pokud je poměr počtu změn frekvence vysílače ku počtu všech změn větší než alpha, bude další jeho změna považována za tabu)
    - *history* (přirozené číslo, defaultně 10) – délka krátkodobé historie (pokud byla vysílači přiřazena ta samá frekvence, kterou teď zkoušíme přiřadit, v krátkodobé historii, tak tento pohyb považujeme za tabu)
    - *loops* (přirozené číslo, defaultně 1000) – maximální počet opakování jednoho kola
    - prvotní algoritmus

```
<?xml version="1.0" ?>
<settings output="test.out.xml" log="test.log" />
<test name="Pokus 1">
  <map>
    <philadelphia instance="1" />
  </map>
  <algorithm>
    <sequential frequency="SmallestOccupied" />
    <permutation size="100" loops="200" />
  </algorithm>
</test>
<test output="test2.out.xml">
  <map>
    <philadelphia instance="2" />
  </map>
  <algorithm>
    <annealing cooling="Costa" alpha="0.8">
      <sequential ordering="SmallestLast" next="GSD" frequency="Smallest"/>
    </annealing>
    <tabu alpha="0.05" history="5">
      <annealing>
        <permutation />
      </annealing>
    </tabu>
  </algorithm>
</test>
```

Ukázkový soubor (test/example.xml na CD).

Další okomentované ukázkové vstupní soubory je možné nalézt na přiloženém CD ve složce *test*.

7.3 Výstupní soubory

Hlavní výstup programu je prováděn do jednoho (či více – podle nastavení) XML souboru. Tento soubor obsahuje pro každý test vstupního souboru jeden element *test*, ve kterém jsou informace o výsledcích testu. Výsledky jednotlivých řešení testu jsou uloženy v elementech *solution*. Obsah jeho podelementu *result* určuje celkový počet přiřazených frekvencí, v elementu *assigment* je uloženo nejlepší vygenerované řešení problému a v elementu *time* doba výpočtu řešení. Pokud je řešení testu uloženo v jiném souboru, pak je v hlavním výstupním souboru místo elementu *solution* element *file*, jehož obsahem je jméno souboru s výsledky testu.

```
<?xml version="1.0" ?>
<test id="1" name="Population size test">
  <map>Philadelphia P1</map>
```

```
  <solution algorithm="genetic: NS20C50M0.10">
    <result>437</result>
    <assignment>
      <antena id="1">2 12 36 74 141 184 220 272</antena>
      …
      <antena id="21"> 17 22 69 80 104 114 129 158</antena>
    </assignment>
    <time>7.25</time>
  </solution>
  …
  <solution>
    <result>438</result>
    <assignment>
      <antena id="1"> 40 69 79 84 100 119 124 251</antena>
      …
      <antena id="21"> 11 18 30 35 45 55 69 119</antena>
    </assignment>
    <time>15.969</time>
  </solution>
</test>
<test id="…">
  <map>…</map>
  <solution algorithm="…">
    <result>…</result>
    <assignment>
      <antena id="…">…</antena>
      …
    </assignment>
    </time>
  </solution>
  <solution>
    …
  </solution>
</test>
```

Ukázkové schéma výstupního XML souboru.

V logovacím souboru jsou uvedeny detaily výpočtu řešení. Asi nejzajímavější částí log souborů je prostřední část, kde se vyskytují data o průběhu výpočtu oddělená středníky – dají se vyjmout a použít jako CSV soubor pro analýzu průběhu algoritmu.

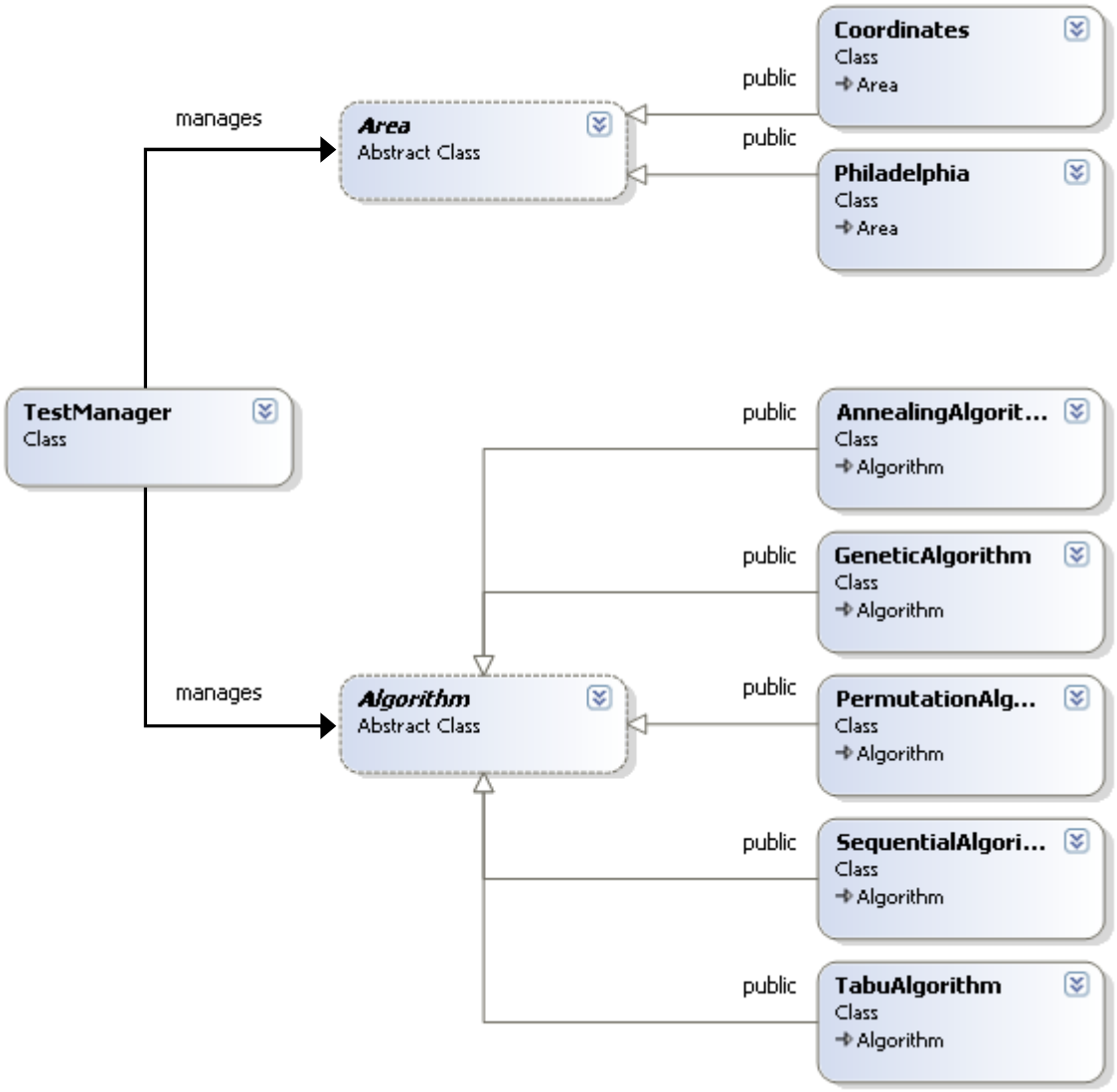
8 Technický popis programu

Přiložený program na řešení problému přiřazování frekvencí je naprogramován v objektově orientovaném programovacím jazyce C++ s využitím standardní knihovny jazyka (STL).

O rozdělování všech úkolů se stará třída *TestManager*, která je navržena pro co nejjednodušší použití zvenčí (aby se co nejvíce usnadnilo použití programu FAP – například kdyby jej někdo chtěl využít jako výpočetní knihovnu). Tato třída se stará o načítání XML souboru se zadáním a vytváří jednotlivé zadání problému (třídy zděděné z *Area*) a zvolené algoritmy (třídy zděděné z *Algorithm*) a poté spouští algoritmy nad vytvořenými problémy. Po skončení výpočtu do historie a příslušných souborů uloží výsledek a další informace o výpočtu.

Zadání problémů (map) je načítáno do tříd zděděných ze základní třídy *Area*, která poskytuje základní operace pro práci s mapou – udržuje informace o pozicích jednotlivých antén, počtech vysílačů na jednotlivých anténách, matici rušení a v neposlední řadě i hodnoty frekvencí přidělených jednotlivým vysílačům. Dále poskytuje funkce na uložení aktuálního stavu přiřazení a načtení stavu z posledně uloženého, což se hodí ve všech algoritmech, které v průběhu tvorby finálního řešení tvoří další pomocná přiřazení (například pro vyhodnocení kvality).

Výpočetní algoritmy jsou třídy zděděné ze základní třídy *Algorithm* a k výpočtu využívají informace a funkce, které jim poskytují třídy pro práci s mapami. *TestManager* se stará o to, aby každému algoritmu správně vyrobil mapu, poté ji algoritmu předal a následně mu dal příkaz na vyřešení zadané mapy (zavolá virtuální metodu algoritmu *Solve()*).



Obrázek 4: Class diagram

Kvůli oddělení „ošklivé“ práce s načítáním vstupu od „hezkých“ algoritmů existují pro každý algoritmus a mapu ještě třídy vlastností (zděděné z *Options*). Jejich hlavní metodou je *Read()*, která jako parametr dostane XML element s nastavením příslušné komponenty a má za úkol jej načíst a takto získané hodnoty v „hezké“ podobě poskytovat algoritmům (a mapám). Další vlastností tříd nastavení je to, že poskytují algoritmům (a mapám) přístup k logovacím souborům.

Detailnější popis vytvořených tříd je možné nalézt v programátorské dokumentaci na přiloženém CD.

8.1 Použité knihovny

Pro načítání vstupního XML souboru byla použita knihovna *ticpp* volně distribuovaná pod licencí *zlib* (umožňující volné použití a provádění libovolných změn), která je dostupná ke stažení na adrese http://code.google.com/p/ticpp/. S knihovnou se mi pracovalo velmi dobře, díky ní jsem se nemusel zabývat téměř žádnými specifiky práce s XML soubory.

8.2 Rozšiřitelnost

Přidání nových prvků (algoritmů a map) do programu je celkem jednoduché. V podstatě stačí podědit od odpovídajících základních tříd (*Algorithm* resp. *Area*) nový algoritmus a od základní třídy *Options* třídu, která bude načítat nastavení daného algoritmu (popřípadě mapy) a předefinovat několik virtuálních metod.

Přidání nového algoritmu

Při přidávání nového algoritmu je nutné podědit vytvářený algoritmus od třídy *Algorithm*, vytvořit v nové třídě konstruktor beroucí ukazatel na třídu *Options* a předefinovat minimálně čistě virtuální metody:

- `*Algorithm(Options * o)` – konstruktor, který je defaultně volaný při tvorbě algoritmu manažerem testů
- `void Set(Area * a)` – přiřazení testovacího problému k algoritmu, je voláno vždy jednou před začátkem výpočtu algoritmu, vhodné k provedení inicializací
- `bool Solve()` – hlavní funkce, která počítá řešení zadaného problému a vrací, zda se povedlo spočítat či ne. U standardních algoritmů je volána pouze jednou, zatímco u algoritmů, které mají definovánu volbu all, je volána pro každou možnou konfiguraci algoritmu (viz. kapitola 7.2 Vstupní XML)
- `void PrintInfo()` – tisk informací o algoritmu do logu
- `std::string GetDetail()` – krátký popis algoritmu včetně konkrétního nastavení (do 28 znaků)

K tvorbě vlastního algoritmu je vhodné ještě vytvořit novou třídu zděděnou z třídy *Options*. Třídy Options* jsou určeny k tomu, aby načetly nastavení v zadání a poskytovaly je algoritmu (popřípadě mapě) v požadované podobě. Dále poskytují i přístup k logovacímu streamu. Při přidávání nového algoritmu je tedy vhodné vytvořit novou třídu zděděnou z třídy Options a předefinovat v ní tyto metody:

- `void Read(ticpp::Element * settings)` – načtení nastavení algoritmu z odpovídajícího vstupního XML elementu
- `void Print()` – tisk podrobných informací o nastavení algoritmu (mapy) do logu
- `std::string GetTitle()` – krátký popis nastavení algoritmu
- `bool Next()` – v případě, že umožníte volbu *all* v nastavení, je třeba předefinovat tuto metodu. Ta je poté volána vždy, když je nutné přepnout na další možné nastavení.

Přidání nové mapy

Při tvorbě nové mapy je nutné vytvořit potomka třídy *Area*, který se musí při konstrukci postarat o následující (v udaném pořadí):

- nastavit položku `double * reuse_distances` na pole obsahující minimální vzdálenosti mezi vysílači, po kterých se dají znovupoužít blízké frekvence (viz. kapitola 3.1 Philadelphia) a položku `int reuse_size`, která udává celkový počet vzdálenostních omezení
- nastavit počet antén pomocí `void setCellCount(int cell_cnt)` – vytvoří potřebná pole pro ukládání informací
- vyplnit pole `Cell * cells` – seznam jednotlivých antén, důležité položky jsou:
  - `double x, y` – souřadnice antény
  - `int demands` – počet vysílačů na anténě
  - `int index` – index prvního prvku v poli vysílačů – tuto položku společně s prvkem demands je možné vyplňovat pomocí metody `void set(int demands, int index)`
- nechat spočítat vzdálenosti mezi anténami a tabulku vzniklých omezení pomocí metody void `countDistances()`
- zinicializovat potřebné pomocné struktury pomocí metody `void initialize(int size)`, kde size je celkový počet vysílačů

K načtení nastavení mapy je doporučeno použít potomka třídy *Options*, který načte nastavení mapy a té je pak předá v rozumné podobě (viz. kapitola Přidání nového algoritmu).

Obecné změny nutné k zavedení nových prvků

Přidání nových tříd zděděných z odpovídajících základních tříd bohužel nestačí, ještě je nutné říci hlavnímu rozdělovači práce (třídě *TestManager*), že přibyl nový algoritmus (popřípadě mapa) a že jej má umět rozpoznávat ve vstupním souboru a spouštět. K tomu je ještě nutné provést níže uvedené kroky v následujících souborech:

- **base.h** – přidat identifikátor nového algoritmu do *enumu Algorithms* a jeho název (ten, co se bude vyskytovat ve vstupním souboru) do pole *AlgorithmNames* (v případě nové mapy odpovídající *enum Area* a pole *AreaNames*)
- **test_manager.h** – přidat načtení (#include) nově vytvořených hlavičkových souborů
- **test_manager.cpp** – do příkazu switch ve funkci *GetAlgorithm* (popřípadě *GetArea*) přidat identifikátor algoritmu (popřípadě mapy) a v tomto „casu“ vytvořit požadovaný algoritmus (resp. mapu) a nechat načíst jeho nastavení (podobně jako v ostatních „casech“)

9 Závěr

Výsledkem této práce je především program umožňující řešit jeden z modelů problému přiřazování frekvencí (MS-FAP) a testovat kvalitu jednotlivých naprogramovaných algoritmů, mezi něž patří sekvenční algoritmus, dva genetické algoritmy a dvě optimalizační techniky – simulované žíhání a hledání tabu. Těmto algoritmům je možné jednoduše upravovat nastavení. Pro účely testování je do programu zabudována podpora testovací sady Philadelphia, dále je možné definovat libovolnou „mapu“ antén a na této mapě nechat vyřešit problém přiřazování frekvencí.

S použitím naprogramovaných technik bylo na většině z testovacích instancí Philadelphie dosaženo optimální hodnoty šířky frekvenčního pásma, nejúspěšnějšími z algoritmů byl upravený třífázový genetický algoritmus a tabu search, které oba počítaly rozumně rychle, a při dobrém nastavení spolehlivě dosahovaly výborných výsledků.

Dále byla během prací jedna z nastudovaných technik vylepšena – jednalo se o třífázový genetický algoritmus, tímto vylepšením se získaná řešení zlepšila přibližně o 10 % při zachování krátkých výpočetních časů.

Dalším přínosem této práce je ucelený pohled na problém přiřazování frekvencí a přehled nejčastěji používaných technik pro jeho řešení v českém jazyce.

10 Obsah CD

Na přiloženém CD se v adresáři *src* nachází kompletní *zdrojové kódy* programu v projektu pro MS Visual Studio 2008 a Makefile pro linux.

V adresáři *bin* jsou umístěny *binární soubory* programu pro MS Windows a linux.

V adresáři *doc* je pak uložena *programátorská dokumentace* k programu (ve formě jednoho PDF a více HTML souborů).

Složka *test* obsahuje *ukázkové vstupní soubory* programu spolu s dosaženými výsledky. Vybrané z nich obsahují i slovní popis a komentář.

Tento *text* je uložen *v elektronické podobě* (MS Word 2007 a PDF) v kořenovém adresáři CD.

11 Použité zkratky

AP (Access Point)

Přístupový bod k bezdrátové síti *Wi-Fi*. K tomuto zařízení jsou připojeni všichni uživatelé jedné bezdrátové sítě a všechna komunikace, jak do vnější sítě (pokud existuje), tak i mezi jednotlivými uživateli sítě, probíhá právě přes její přístupový bod.

FAP (Frequency Assignment Problem)

Problém přiřazování frekvencí, někdy také nazýván Channel Assignment Problem. Téma celé této práce.

GSM (Groupe Spécial Mobile – Global System for Mobile Communications)

Nejpoužívanější standard pro mobilní telefony, ve více než 200 zemích GSM mobilní telefony používá přes miliardu lidí.

IEEE (Institute of Electrical and Electronics Engineers)

Mezinárodní organizace usilující o rozvoj elektrotechniky. Za tímto účelem vyvíjí nové technologie a vydává standardy, mezi které patří například i standard lokálních sítí, který definuje mimo jiné standard pro *Wi-Fi* (802.11).

IEEE 802.11

Standard *Wi-Fi* pro lokální bezdrátové sítě, definuje mimo jiné různé druhy modulace (kódování) rádiového signálu (ty jsou popsány v dodatcích ke standardu označovaných písmeny, mezi nejznámější patří *a*, *b*, *g* a *n*).

PDA (Personal Digital Assistant – palmtop)

Malý kapesní počítač obvykle vybavený dotykovou obrazovkou a perem. PDA se používají primárně k organizaci času a kontaktů, avšak většina z nich poskytuje plnohodnotné počítačové programy a také umožňuje přístup na internet.

RDS (Radio Data System)

Systém určený k přenosu doplňkových informací v rádiových sítích. Mezi hlavní služby patří například přenos názvu rozhlasové stanice, informace o tom, že jsou právě vysílány informace o dopravě, přenos dopravních informací pro navigace, přenos času a data a v neposlední řadě také automatické přelaďování.

TDMA (Time Division Multiple Access)

Metoda pro rozdělení přístupu k sítím se sdíleným médiem – nosičem (většinou se jedná o rádiové sítě – například *GSM*). Tato metoda povoluje více zařízením sdílet stejný frekvenční kanál díky rozdělení času na jednotlivé intervaly (časové sloty). Každý uživatel sítě pak má přidělen vlastní časový slot, ve kterém může vysílat.

Wi-Fi (WiFi, Wifi, …)

Standard pro lokální bezdrátové sítě, používaný pro připojování (přenosných) zařízení do bezdrátových sítí a dále jejich připojením na sítě lokální. Dnes bývá často využíváno i k bezdrátovému připojení do sítě internet.

WLAN (Wireless Local Area Network)

Bezdrátová lokální síť sloužící k propojení dvou a více zařízení (například počítačů) bez pomoci drátů. Hlavní z využívaných technologií pro tvorbu bezdrátové lokální sítě je *Wi-Fi*.

XML (eXtensible Markup Language)

Obecný značkovací jazyk umožňující vytváření konkrétních značkovacích jazyků pro různé účely. Mezi nejznámější konkrétní příklady patří *XHTML* (zdrojový kód webových stránek), *RSS* (novinky na webových stránkách), *Office Open XML* (jazyk pro ukládání dokumentů Microsoft Office 2007) nebo *OpenDocument* (dokumenty aplikace OpenOffice).

12 Reference

[Aardal03] Aardal, K. I., van Hoesel, C. P. M., Koster, A. M. C. A., Mannino, C., Sassano, A.:
„Models and solution techniques for the frequency assignment problem“.
4OR, 1 (2003/4), 261–317.

[Aardal96] Aardal, K. I., Hipolito, A., van Hoesel, C. P. M., Jansen, B.:
„A branch-and-cut algorithm for the frequency assignment problem“.
Research Memorandum 96/011 (1996), Maastricht University.

[Anderson73] Anderson, L. G.:
„A simulation study of some dynamic channel assignment algorithms in a high capacity mobile telecommunications system“.
IEEE Transactions on Communications, 21 (1973), 1294–1301.

[CALMA95] CALMA website (1995). EUCLID CALMA project:
http://www.win.tue.nl/~wscor/calma.html

[COST259] COST 259 website (2001):
http://www.lx.it.pt/cost259/

[Costa93] Costa, D.:
„On the use of some known methods for T-colourings of graphs“.
Annals of Operation Research, 41 (1993), 343–358.

[CSU] Český statistický úřad
http://www.czso.cz/csu/tz.nsf/i/mobilni_telefony_v_evropske_unii

[Eisenblätter01] Eisenblätter, A.:
„Frequency assignment in GSM networks: Models, heuristics, and lower bounds“.
PhD Thesis (2001), Technische Universität Berlin, Berlin, Germany.

[FAPWEB] FAP web (2000 – 2007):
http://fap.zib.de/

[Fu06] Fu, X., Bourgeois, A.G., Fa, P., Pan, Y.:
„Using a genetic algorithm approach to solve the dynamic channel-assignment problem.”
Int. J. Mobile Communications, Vol. 4, No. 3 (2006), 333–353.

[Hurley95] Hurley, S., Smith, D. H.:
„A comparison of local search algoithms for radio link frequency assignment problems“.
ACM Symposium on Applied Computing (1995), 373–378.

[Hurley97] Hurley, S., Smith, D. H., Thiel, S. U.:
„FASoft: A system for discrete channel frequency assignment“.
Radio Science, 32 (1997), 1921–1939.

[Hurley98] Hurley, S., Smith, D.H., Valenzuela, Ch.:
„A Permutation Based Genetic Algorithm for Minimum Span Frequency Assignment“.
A.E. Eiben: PPSN V, LNCS 1498 (1998), 907–916.

[Kolen94] Kolen, A. W. J., van Hoesel, C. P. M., van der Wal, R.:
„A constraint satisfaction approach to the radio link frequency assignment problem“.
Technical Report 2.2.2 (1994), EUCLID CALMA project.

[Kunz91] Kunz, D.:
„Channel assignment for cellular radio using neural networks“.
IEEE Transactions on Vehicular Technology, 40 (1991), 188–193.

[Leung03] Leung, K. K., Kim, B.-J.:
„Frequency assignment for IEEE 802.11 wireless networks“.
Proceedings of VTC 2003-Fall (2003), Orlando, FL.

[Maniezzo00] Maniezzo, V., Carbonaro, A.:

„An ants heuristic for the frequency assignment problem“.
Future Generation Computer Systems, 16 (200), 927–935.

[Mannino03] Mannino, C., Sassano, A.:
„An enumerative algorithm for the frequency assignment problem“.
Discrete Applied Mathematics, 129 (2003/1), 155–169.

[Metzger70] Metzger, B. H.:
„Spectrum management technice“.
Presentation at 38th National ORSA meeting (1970), Detroit, MI.

[Oxypedia] Oxypedia:
http://www.oxypedia.org/Cycle-Crossover-01.htm

[Park96] Park, T., Lee, C. Y.:
„Application of the graph coloring algorithm to the frequency assignment problem“.
Journal of the Operations Research Society of Japan, 39 (1996), 258–265.

[Thuve81] Thuve, H.:
„Frequency planning as a set partitioning problem“.
European Journal of Operational Research, 6 (1981), 29–37.

[Tiourine99] Tiourine, S.R.:
„Decision Support by Combinatorial Optimization: Case Studies“.
PhDThesis (1999), Eindhoven University of Technology