VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

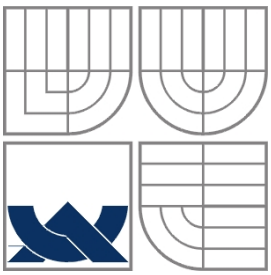
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

BARVENÍ GRAFŮ

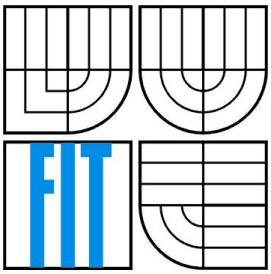
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE LUKÁŠ PROCHÁZKA
AUTHOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

BARVENÍ GRAFŮ
GRAPH COLORING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR
LUKÁŠ PROCHÁZKA

VEDOUCÍ PRÁCE
SUPERVISOR
RNDR. TOMÁŠ MASOPUST, PH.D.

BRNO 2009

Abstrakt

Tato práce se zabývá barvením grafů, což je přiřazování barev jednotlivým vrcholům grafu tak, aby žádné dva vrcholy spojené hranou neměly stejnou barvu. Tento problém je velmi výpočetně náročný, protože je NP-úplný. Zároveň je velmi důležitý, protože má řadu praktických aplikací. Zde jsou popsány některé heuristické algoritmy, které se tento problém snaží řešit pomocí postupného zlepšování počátečního řešení při zadaném počtu barev. Tři algoritmy byly implementovány a poté otestovány na různých grafech a porovnány vzhledem k různým kritériím.

Abstract

This thesis is about graph coloring, which is assigning colors to vertices of a graph such that no two vertices, which are linked with an edge, have the same color. This problem is very computational hard, because it's NP-complete. It's also very important, because it has many practical applications. Here are described some of the heuristic algorithms, which try to solve this problem by iteratively improving the initial solution with given number of colors. Three of them have been implemented, tested on different graphs and compared considering several criteria.

Klíčová slova

graf, barvení grafu, tabucol, gh, hca, antcol

Keywords

graph, graph coloring, tabucol, tabu search, gh, genetic hybrid, hca, hybrid coloring algorithm, antcol, new ant algorithm

Citace

Lukáš Procházka: Barvení grafů, bakalářská práce, Brno, FIT VUT v Brně, 2009

Barvení grafů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením RNDr. Tomáše Masopusta, Ph.D..
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Lukáš Procházka
13.5.2009

Poděkování

Chtěl bych poděkovat svému vedoucímu práce RNDr. Tomášovi Masopustovi, Ph.D., za vedení při zpracovávání bakalářské práce.

Obsah

1 Úvod

Graf je množina vrcholů a hran, které spojují některé dvojice vrcholů. Barvení grafu pak je přiřazování barev vrcholům grafu tak, aby žádné dva vrcholy spojené hranou neměly stejnou barvu. Barvení grafů nachází uplatnění v mnoha různých oborech lidské činnosti, například plánování rozvrhů, přidělování registrů v počítači, řešení hry Sudoku a mnohé další [1]. Nalezení co nejlepšího barvicího algoritmu je tedy určitě zajímavým a důležitým problémem. Zároveň je však barvení grafů velmi výpočetně náročné a proto musí být pro jakýkoliv složitější graf řešeno heuristickými algoritmy, které se snaží postupně zlepšovat určité počáteční obarvení.

Tato práce se zabývá různými heuristickými algoritmy, které se rozličnými způsoby snaží co nejlépe obarvit zadaný graf. Budou zde popsány barvicí algoritmy TABUCOL, GH (HCA), VNS, AMACOL, ANTCOL a VSS.

Cílem této práce je seznámit čtenáře s teorií grafů a jejich barvení, popsat existující metody pro barvení grafů, některé z nich otestovat a poté je porovnat vzhledem k určitým kritériím. Následovat bude také srovnání získaných výsledků s oficiálními výsledky od autorů barvicích metod.

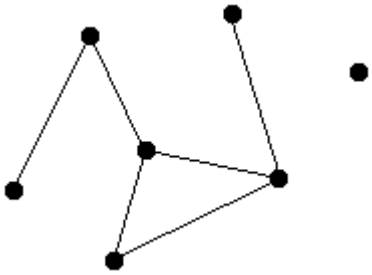
Kapitola 2 uvádí základy teorie grafů a jejich barvení. Kapitola 3 popisuje šest nejzajímavějších sofistikovaných barvicích algoritmů. Kapitola 4 popisuje postup implementace algoritmů a dalších částí programu. V kapitole 5 je stručný návod na překlad a spuštění programu. Kapitola 6 obsahuje výsledky testů barvicích algoritmů a informace z nich odvozené.

2 Teorie grafů

Slovo graf má v matematice více významů. Jedním z nich je grafické znázornění průběhu nějaké funkce. V této práci se však jedná o graf ve smyslu *množiny vrcholů a hran*, které tyto vrcholy spojují.

Označme $[V]^k$ množinu všech k-prvkových podmnožin množiny V. Formálně řečeno je pak graf dvojice množin $G = (V,E)$, kde G je graf (anglicky "graph"), V je množina vrcholů (nebo též uzlů, anglicky "vertices") a E je množina hran (anglicky "edges") taková, že $E \subseteq [V]^2$, čili každý prvek z množiny E je dvouprvková množina prvků z množiny V. To znamená, že každá hrana spojuje právě dva vrcholy. Tyto vrcholy se nazývají *sousední*. Stejně tak se dvě hrany nazývají *sousední*, pokud obě končí jedním svým koncem ve stejném vrcholu. Hrany mohou být orientované nebo neorientované, zde budeme uvažovat hrany neorientované. Graficky se graf znázorňuje tak, že každý vrchol je znázorněn jako bod a každá hrana jako čára spojující příslušné dva body (viz obrázek 2.1). Rozmístění bodů a tvary čar nejsou podstatné, důležité je vyjádřit, které vrcholy jsou spojené hranou a které ne. (Informace v tomto odstavci byly převzaty z [2].)

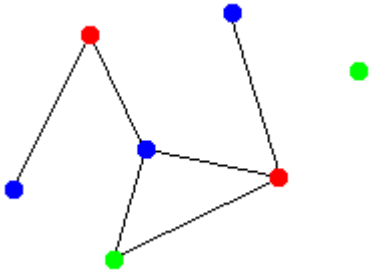
Vrcholy bývají obvykle označeny čísly z množiny přirozených čísel, hrana je pak identifikována čísly dvou vrcholů, které spojuje.



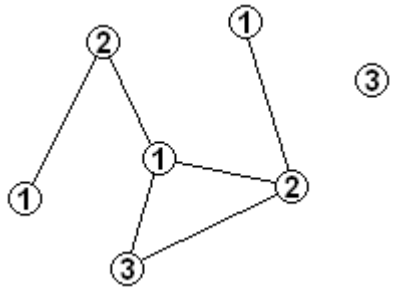
Obrázek 2.1 - Graf

2.1 Barvení grafů

Barvení grafů spočívá v přiřazení "barvy" každému vrcholu grafu. "Barva" je v tomto případě opět obvykle reprezentována číslem z množiny přirozených čísel. Cílem je, aby žádné dva sousední vrcholy neměly stejnou barvu. Pak se takové obarvení nazývá *legální* (viz obrázky 2.2 a 2.3). Pokud se toto nepodaří, pak dva sousední vrcholy obarvené stejnou barvou se nazývají *konfliktní vrcholy*, hrana, která je spojuje, se nazývá *konfliktní hrana* a barva, kterou jsou vrcholy obarveny, se nazývá *konfliktní barva*. Nejmenší počet barev, kterými může být graf legálně obarven, se nazývá *chromatické číslo*. Nalezení chromatického čísla je základním úkolem barvení grafu.



Obrázek 2.2 - Obarvený graf - znázornění



Obrázek 2.3 - Obarvený graf - skutečná reprezentace

2.2 Rovinné grafy

Rovinný graf je takový graf, který lze zakreslit do roviny takovým způsobem, že se žádné dvě jeho hrany nekříží. Bylo matematicky dokázáno, že každý rovinný graf má chromatické číslo nejvýše 5 [2]. Dlouho bylo otevřeným problémem, zda má každý rovinný graf chromatické číslo nejvýše 4. Toto bylo nakonec dokázáno v roce 1976 Appelem a Hakenem prověřením velkého množství grafů pomocí počítače. Byl to první matematický důkaz s takto významným nasazením počítače [3][4].

Praktickým použitím je například barvení států na politické mapě.

2.3 Praktická použití barvení grafů

Barvení grafů není jen matematickou hříčkou, jak by se možná mohlo zdát, ale má široké uplatnění v praxi, například:

- *vytváření rozvrhů, plánování* - máme určitý počet činností (vrcholy) a každé je třeba přiřadit určitý čas (barva), kdy bude probíhat. Některé činnosti nemohou probíhat současně (sousední vrcholy). Chceme použít co nejméně časových oken, tj. obarvit graf co nejméně barvami.
- *přidělování registrů při kompilaci programu v počítači* - v rámci zrychlení běhu určitého programu jsou nejčastěji používané hodnoty programu (vrcholy) uloženy v registrech počítače (barvy). Některé hodnoty jsou potřeba ve stejný časový okamžik (sousední vrcholy), nemohou být tedy uloženy ve stejném registru. Počet barev je stejný jako počet potřebných registrů.
- *nalezení řešení rébusu Sudoku* - Sudoku (viz obrázek 2.4) se skládá z 81 políček (vrcholy), kterým je třeba přiřadit čísla od 1 do 9 (barvy) tak, aby žádná dvě políčka, která jsou ve stejném řádku, sloupci nebo vyznačeném čtverci (sousední vrcholy), neměla stejné číslo. Tento problém lze tedy převést na graf o 81 vrcholech, který je třeba obarvit 9 barvami.
- *přidělování rádiových frekvencí* - na určitém území máme určitý počet rádiových vysílačů (vrcholy) a je třeba jim přidělit co nejméně rádiových frekvencí (barvy) tak, aby se vysílače vzájemně nerušily [5].

(Příklady byly převzaty z [1] a [6].)

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Obrázek 2.4 - Sudoku [7]

2.4 Složitost

Nalezení chromatického čísla bohužel patří do třídy NP-úplných problémů [8], což znamená, že neznáme algoritmus, který by ho vyřešil v polynomiálním čase. (Pokud neplatí, že $P = NP$, což je jeden ze sedmi největších nevyřešených problému matematiky, za jejichž vyřešení je nabízena odměna jeden milion dolarů. Zájemcům o toto téma doporučuji knihu [9].) Proto všechny algoritmy, o kterých dále bude řeč, nefungují tak, že bychom jim předali graf a ony by zjistily jeho chromatické číslo. Naopak jsou to heuristické algoritmy, kterým musíme zadat počet barev, ony vyjdou z nějakého počátečního obarvení a postupně se snaží ho zlepšovat tak, aby při zadaném počtu barev odstranily všechny konflikty. To se samozřejmě může nebo nemusí podařit. Hledání chromatického čísla pak spočívá v opakovaném spouštění algoritmu s postupně klesajícím počtem barev, dokud se nám daří nacházet legální obarvení. Výsledek pak závisí na kvalitě použitého algoritmu a nikdy si nemůžeme být úplně jisti, že jsme skutečně chromatické číslo nalezli.

3 Barvicí algoritmy

Nyní budou popsány principy šesti sofistikovaných algoritmů, z nichž tři byly v rámci této práce implementovány a otestovány. Všechny tyto algoritmy mají na vstupu neobarvený graf a počet barev, kterými se má obarvit. Na výstupu pak mají obarvený graf a informaci o tom, zda se podařilo odstranit všechny konflikty. Všechny také obsahují funkci, která hodnotí kvalitu řešení, obvykle je to funkce vracející počet konfliktních hran v grafu.

Je zajímavé povšimnout si, jak každý algoritmus funguje na úplně jiném principu. Ilustruje to snahu matematiků přicházet se stále novými nápady a tím i důležitost problému barvení grafů jako takového.

Algoritmy jsou seřazeny podle roku, kdy byly publikovány.

3.1 TABUCOL

Název publikace: Using Tabu Search Techniques for Graph Coloring [10]
Autoři: Alain Hertz a Dominique de Werra
Rok publikování: 1987

Algoritmus pracuje na principu hledání mezi řešeními podobnými aktuálnímu řešení. Aby se předešlo zacyklení, ukládají se změny do tabu listu, kde po určitý počet iterací jsou a nesmí být znovu použity. Tento algoritmus je jedním z nejznámějších a na svoji jednoduchost je velmi efektivní. Bývá používán jako součást složitějších metod.

Jak funguje: Na začátku je graf náhodně obarven zadaným počtem barev. Poté se v cyklu provádí následující: Nejdříve je vytvořeno okolí aktuálního řešení. Okolím se rozumí určitý počet sousedů aktuálního řešení. Soused vznikne z aktuálního řešení změnou barvy jendoho z konfliktních vrcholů na jinou náhodně zvolenou barvu. Z okolí se pak vybere nejlepší řešení, tj. takové, které má nejmenší počet konfliktních hran a které není v tabu listu. Pokud je ale nalezeno takové obarvení, které je lepší než nejlepší doposud nalezené obarvení, je toto okamžitě použito bez ohledu na to, zda je v tabu listu. Původní barva vrcholu se zaznamená do tabu listu a zůstane tam po určitý předem zadaný počet iterací. Po tuto dobu nesmí daný vrchol opět dostat původní barvu. Cyklus končí, pokud je nalezeno řešení bez konfliktů, nebo pokud je dosaženo předem zadaného maximálního počtu iterací.

3.2 GH (HCA)

Název publikace: Hybrid Evolutionary Algorithms for Graph Coloring [8]
Autoři: Philippe Galinier a Jin-Kao Hao
Rok publikování: 1999

Tento algoritmus je znám pod názvem GH (Genetic hybrid) nebo HCA (Hybrid coloring algorithm). Autoři jej původně nazývají HCA, ale v této práci bude používán název GH, protože tento název je použit v ostatních publikacích a navíc jsou G a H první písmena příjmení autorů. Algoritmus je založen na měnící se populaci složené z různých obarvení, kdy vždy ze dvou rodičů vznikne potomek.

Jak funguje: Na začátku je vytvořena populace předem zadaného počtu různých řešení. Každý člen populace je obarven tímto způsobem: Vždy se vybere vrchol, kterému lze přiřadit nejmenší možný nenulový počet různých barev tak, aby nevznikl konflikt. Je-li takových vrcholů více, je jeden vybrán náhodně. Vrcholu je poté přiřazena bezkonfliktní barva, která je reprezentována co nejnižším číslem. Zbylé vrcholy, které už bezkonfliktně obarvit nejdou, jsou obarveny náhodně. Pak je obarvení ještě vylepšeno TABUCOLem. Po vytvoření populace se v cyklu provádí následující: Jsou vybráni dva nejlepší členové populace (tj. obarvení s nejnižším počtem konfliktních hran). Nazývají se *rodiče* a je z nich vytvořen *potomek*. Potomek se vytváří tak, že v jednom z rodičů se vždy hledá barva, kterou je obarveno nejvíce vrcholů, a tyto vrcholy jsou pak i v potomkovi obarveny všechny stejně.

Toto vytváření potomka proběhne v tolika iteracích, kolik je zadaných barev, přičemž vždy v liché iteraci se hledá v prvním rodičovi a v sudé iteraci se hledá ve druhém rodičovi. Zbylé vrcholy jsou opět obarveny náhodně. Po vytvoření je potomek opět vylepšen TABUCOLem a poté vložen zpět do populace, kde nahradí horšího z rodičů, tj. toho, který má větší počet konfliktích hran (při shodě se vybírá rodič náhodně). Cyklus končí, pokud je nalezeno řešení bez konfliktů, nebo pokud je dosaženo předem zadaného maximálního počtu iterací.

3.3 VNS

Název publikace: A variable neighborhood search for graph coloring [11]
Autoři: Cédric Avanthay, Alain Hertz a Nicolas Zufferey
Rok publikování: 2003

Algoritmus VNS (Variable neighborhood search) je založen na hledání řešení v různě definovaných okolích. Okolí je v tomto případě způsob, kterým se mění barvy vrcholů v aktuálním řešení. Použitím různých okolí se zabrání, aby se algoritmus zasekl v lokálním minimu funkce, která určuje kvalitu řešení. Okolí jsou tří různých typů:

- okolí, která mění barvu některých konfliktních vrcholů
  - například takové, které změní barvu jednoho náhodně vybraného konfliktího vrcholu a pokud touto změnou vzniknou nové konfliktní vrcholy, vybere z nich jeden náhodně, změní jeho barvu a takto pokračuje dál
- okolí, která mění barvu některých nebo všech vrcholů obarvených jistou konfliktní barvou
  - například takové, které vybere barvu, kterou je obarveno nejvíce konfliktních vrcholů, všechny vrcholy obarvené touto barvou přebarví na jinou barvu, poté vybere z grafu stejný počet vrcholů (nejlépe nově vzniklých konfliktních vrcholů) a přebarví je na tuto barvu
- okolí, která mění barvu některých vrcholů tak, aby se nezvýšil počet konfliktních hran

Jak funguje: Na začátku je graf obarven náhodně. Poté se v cyklu provádí následující: Z aktuálního řešení je vytvořeno nové řešení. Způsob vytvoření nového řešení závisí na tom, které okolí je aktuálně používáno. Nové řešení je vylepšeno TABUCOLem. Je-li nové řešení lepší než původní, pak nahradí původní. Naopak pokud proběhl určitý předem zadaný počet iterací bez zlepšení řešení, přejde se k následujícímu okolí. Cyklus končí, pokud je nalezeno řešení bez konfliktů, nebo pokud je dosaženo předem zadaného maximálního počtu iterací, které uběhly od posledního zlepšení řešení.

3.4 AMACOL

Název publikace: An Adaptive Memory Algorithm for the k-Colouring Problem [12]
Autoři: Philippe Galinier, Alain Hertz a Nicolas Zufferey
Rok publikování: 2003

Algoritmus AMACOL (Adaptive memory algorithm) používá centrální paměť, která obsahuje různé části řešení. Z těchto částí se pak řešení vytváří a částmi vytvořeného řešení se opět aktualizuje centrální paměť.

Jak funguje: Na začátku je inicializována centrální paměť. Ta se skládá z *nezávislých množin* (stable sets), což jsou množiny vrcholů, z nichž žádné dva nejsou spojeny hranou. Inicializace probíhá tak, že je vytvořen určitý předem zadaný počet náhodných obarvení grafu, každé je vylepšeno TABUCOLem a poté jsou v něm hledány největší nezávislé množiny a tyto jsou ukládány do centrální paměti. Po inicializaci centrální paměti se v cyklu provádí následující: Z částí paměti je vytvořeno řešení. To se provede tak, že pro každou barvu je vybrán náhodný vzorek nezávislých množin z centrální paměti, z těchto množin je vybrána taková, která má největší počet dosud neobarvených vrcholů, a tyto vrcholy jsou obarveny danou barvou. Pokud nakonec zůstanou nějaké neobarvené vrcholy, jsou obarveny náhodně. Vytvořené řešení je poté vylepšeno TABUCOLem. Pak je centrální paměť aktualizována kusy získaného řešení. To je opět provedeno tak, že v získaném

řešení jsou hledány největší nezávislé množiny a tyto jsou vloženy do centrální paměti, kde nahradí jiné náhodně vybrané nezávislé množiny. Cyklus končí, pokud je nalezeno řešení bez konfliktů, nebo pokud je dosaženo předem zadaného maximálního počtu iterací, nebo pokud rozmanitost (diversity) centrální paměti klesne pod určitou mez.

3.5 ANTCOL

Název publikace: A New Ant Algorithm for Graph Coloring [6]
Autoři: Alain Hertz a Nicolas Zufferey
Rok publikování: 2006

Algoritmus ANTCOL pracuje na principu pohybu *mravenců*. Mravenci se pohybují po vrcholech grafu. Každý mravenec má svou barvu a každý vrchol je pak obarven podle barev mravenců na něm. Mravenci se pohybují tak, že vždy dva mravenci na dvou sousedních vrcholech si vymění místa. Pohyb mravenců je řízen jednak tím, že mravenci stejné barvy se snaží dostat se k sobě (*greedy force*), jednak tím, že pohyb mravenců zanechává na vrcholech stopy (*trails*). Zajímavé je, že role každého mravence je malá, má vliv pouze na vrchol, na kterém se nachází, nikoliv na celý graf. Přesto lze tímto způsobem dosáhnout zajímavých výsledků.

Jak funguje: Na začátku je na každý vrchol umístěn jeden mravenec od každé barvy. Poté je graf obarven procedurou Color. Ta funguje následovně: Vybere vždy vrchol připojený k vrcholům nejvíce různých barev. Je-li takových více, vybere vrchol připojený k nejvíce vrcholům. (Je-li více i těch, pak vybere jeden z nich náhodně.) Vybraný vrchol dostane barvu, kterou má alespoň jeden mravenec na něm, a která minimalizuje konflikty. Je-li takových více, pak vybere barvu, kterou má nejvíce mravenců na daném vrcholu. (Je-li více i těch, pak vybere jednu z nich náhodně.) Po obarvení se spustí vlastní cyklus ANTCOLu: V každé iteraci je úkolem změnit barvu jednoho z konfliktních vrcholů. Toho se dosáhne tak, že z vrcholu odstraníme všechny mravence jeho barvy. Nejdříve zvolíme nejlepší pohyb mravence (spočítá se z greedy force a trails) a pak odstraníme z daného vrcholu zvoleného mravence a také všechny ostatní mravence stejné barvy. Daná barva je pro daný vrchol označena tabu na určitý počet iterací. Po tuto dobu nesmí žádný mravenec této barvy opět vstoupit na daný vrchol. Dále se aktualizují hodnoty trails (stopy se s časem vypařují, tj. hodnoty trails klesají). Nakonec jsou pomocí procedury Color přebarveny vrcholy, které byly minule konfliktní, a vrcholy, kde nastal pohyb mravenců. Cyklus končí, pokud je nalezeno řešení bez konfliktů, nebo pokud je dosaženo předem zadaného maximálního počtu iterací, které uběhly od posledního zlepšení řešení.

3.6 VSS

Název publikace: Varibale Space Search for Graph Coloring [13]
Autoři: Alain Hertz, Matthieu Plumettaz a Nicolas Zufferey
Rok publikování: 2006

Algoritmus VSS (Variable space search) hledá nejen v různých okolích (jako VNS, viz kapitola 3.3), ale i v různých hledacích prostorech. Barvení grafu je pak popsáno více různými způsoby, z nichž každý má svůj vlastní hledací prostor, vlastní okolí a vlastní funkci, která určuje kvalitu nalezeného řešení. Tyto různé způsoby se při barvení střídají a tím je vyřešen únik z lokálního minima hodnotící funkce. VSS používá 3 různé hledací prostory:

- První může obsahovat obarvení s konfliktními hranami a všechny vrcholy musí být obarveny. Kvalita řešení je určena počtem konfliktních hran. Je použit TABUCOL pro zlepšení řešení.
- Druhý nesmí obsahovat konfliktní hrany, ale některé vrcholy mohou zůstat neobarvené. Kvalita řešení je určena počtem neobarvených vrcholů.
- Třetí nesmí obsahovat konfliktní hrany, všechny vrcholy musí být obarveny, ale není omezen počtem barev.

Jak funguje: Na začátku algoritmus vytvoří počáteční řešení. Poté se v cyklu provádí následující: Hledá se řešení v daném hledacím prostoru pomocí příslušného hledacího algoritmu

(například TABUCOL pro první hledací prostor). Pokud se řešení nezlepší po určitý předem zadaný počet iterací, pak je transformováno do dalšího prostoru a dále se bude hledat tam. Cyklus končí, pokud je nalezeno řešení bez konfliktů, nebo pokud uběhl předem zadaný maximální čas.

3.7 Algoritmy zvolené pro implementaci

Z výše zmíněných algoritmů byly pro implementaci a testování zvoleny TABUCOL, GH a ANTCOL.

TABUCOL byl vybrán, protože je to jeden z nejznámějších algoritmů. Prakticky ve všech ostatních publikacích se o něm mluví a bývá často použit jako součást složitějších algoritmů. Přitom je velice jednoduše definovaný.

ANTCOL byl vybrán, protože je zajímavý svým nápadem přesouvajících se mravenců a také tím, že dává zajímavé výsledky, přestože každý mravenec má sám o sobě malý vliv na celkové obarvení grafu.

GH byl vybrán, protože je zajímavý tím, že funguje na principu evoluce, kde silnější jedinci nahrazují slabší. Stejně jako v případě ANTCOLu jde opět o použití principů vypozorovaných z jevů v přírodě.

4 Implementace

Vlastní implementace programu byla provedena v jazyce C. Program byl vyvíjen v editoru PSPad a překládán z příkazové řádky systému Windows pomocí gcc verze 3.4.2. Program byl testován také v CVT.

Vývoj programu probíhal v těchto fázích:

- programování funkce main - kontrola vstupních parametrů, načtení grafu ze souboru a jeho uložení do struktury v paměti
- programování metody ANTCOL
- programování metody TABUCOL
- programování metody GH
- závěrečné úpravy, nastavení parametrů barvicích metod
- napsání programu pro generování náhodných grafů
- napsání programu pro generování rovinných grafů

4.1 Načtení grafu ze souboru

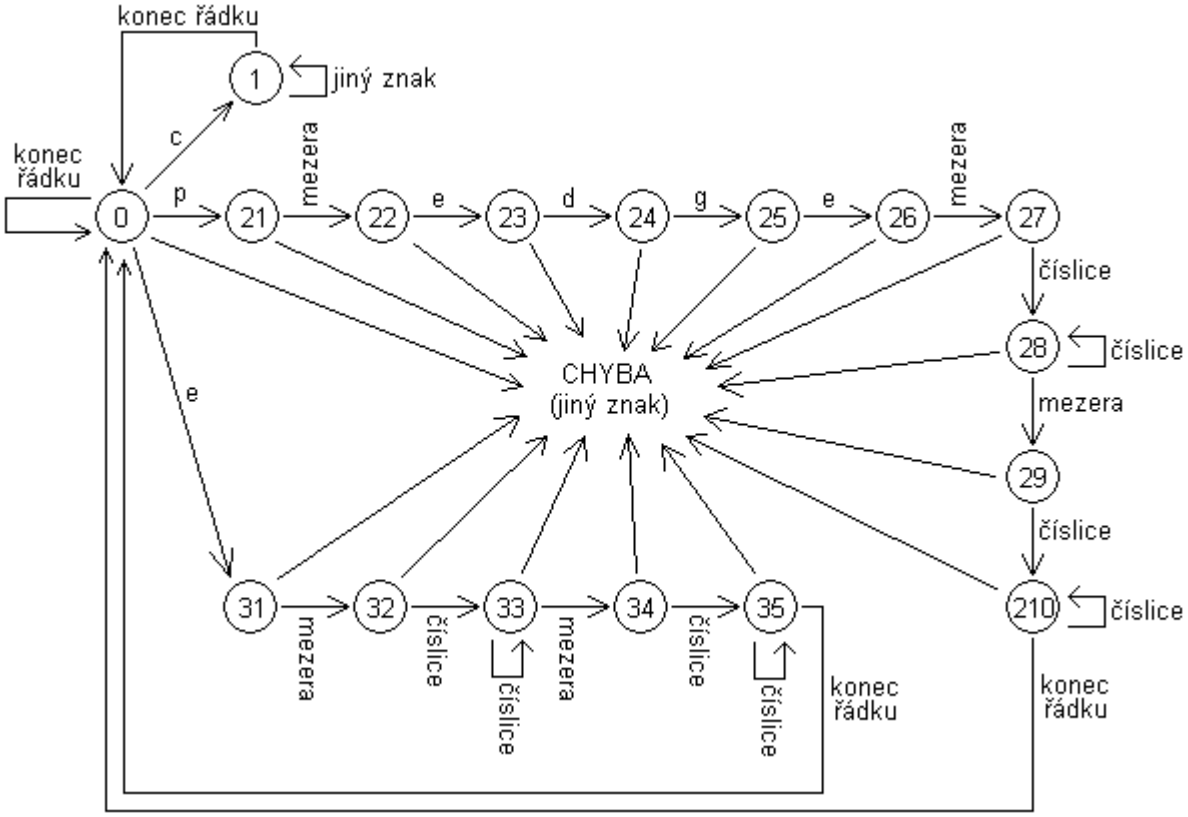
Každý graf je zadán v textovém souboru, kde je specifikován počet vrcholů, počet hran a všechny hrany jsou popsány dvojicí čísel vrcholů, které spojují. Formát souboru vychází ze specifikace v [14], protože grafy zmiňované v publikacích o šesti výše zmíněných barvicích algoritmech, jsou zadané právě v tomto formátu. Tyto grafy se pak dají stáhnout z internetu, například na adrese *ftp://dimacs.rutgers.edu/pub/challenge/graph/benchmarks/color/*. Zmíněná specifikace se netýká pouze grafů určených pro barvení, proto byla zjednodušena pouze pro tento účel.

Každý řádek začíná jedním z vybraných písmen, identifikujících typ řádku. Mezera mezi jednotlivými částmi je vždy právě jedna a za poslední částí bezprostředně následuje konec řádku. Jiné než níže specifikované řádky jsou považovány za chybu.

- **c***libovolné_znaky* - komentářový řádek, může se vyskytovat kdekoliv v souboru a nemá žádný význam pro definici grafu.
- **p** edge *počet_vrcholů počet_hran* - zadání počtu vrcholů a počtu hran grafu. Tento řádek musí předcházet první výskyt řádku **e**. Slovo edge je zde kvůli kompatibilitě s výše zmíněnou specifikací v [14].
- **e** *číslo_vrcholu číslo_vrcholu* - definice hrany pomocí čísel vrcholů, které spojuje. Každá hrana je zadána pouze jednou, čili pokud už soubor obsahuje řádek **e** 4 8, pak již nesmí obsahovat řádek **e** 8 4. Program to kontroluje a ohlásí chybu. Stejně tak ohlásí chybu, pokud zadaný počet hran nesouhlasí se skutečným počtem řádků **e**.
- prázdný řádek - prázdný řádek je ignorován.

Soubor má obvykle příponu .col.

Načítání grafu je řešeno pomocí konečného automatu. Ten je zobrazen na obrázku 4.1. Nejdříve se v souboru pouze hledá řádek **p**, abychom zjistili počet vrcholů a hran. Poté se soubor prochází pomocí tohoto konečného automatu znovu od začátku, kontroluje se a graf se postupně ukládá do paměti. Konečný automat začíná ve stavu 0 a graf je považován za správně načtený pouze tehdy, pokud po dosažení konce souboru automat skončí opět ve stavu 0.



Obrázek 4.1 - Konečný automat pro načtení grafu z textového souboru

4.2 Struktura pro uložení grafu

Pro reprezentaci každého vrcholu byla vytvořena struktura, která obsahuje:

- číslo barvy
- počet sousedů
- ukazatel na první prvek seznamu sousedních vrcholů

Graf je pak reprezentován polem těchto struktur, kde číslo vrcholu je indexem do tohoto pole (dekrementovaným o jedničku).

Každý prvek seznamu sousedních vrcholů obsahuje:

- číslo jednoho sousedního vrcholu
- ukazatel na další prvek seznamu

Všechny tyto struktury jsou vytvořeny a naplněny údaji při načítání grafu ze souboru. Později při barvení už se mění pouze čísla barev jednotlivých vrcholů.

4.3 Uložení výsledku do souboru

Po dokončení barvení je výsledek uložen do výstupního souboru specifikovaného uživatelem. Soubor má tento formát:

- 1. řádek: `Zadany pocet barev:` *`počet_barev`* - počet barev, kterými měl být graf obarven.
- 2. řádek: `Pocet konfliktnich hran:` *`konfliktní_hrany`* - počet konfliktních hran, které zbyly. Pokud se podaří obarvit graf bezkonfliktně, je zde hodnota 0.
- (3. řádek je prázdný.)
- 4. řádek: `uzel` *`číslo_uzlu`* `- barva` *`číslo_barvy`* - pro všechny uzly (vrcholy) pořadě je vypsán údaj o jejich barvě. Na každém dalším řádku je právě jeden takový údaj.

4.4 Problém s ANTCOLem

Při implementaci metody ANTCOL se objevil problém s normalizací hodnot trails. Trails jsou stopy, které po sobě mravenci při pohybu zanechávají a podle kterých se především rozhodují o svém dalším pohybu. Trails se postupně "vypařují", čili jejich hodnota se snižuje. V zadání ANTCOLu bylo, že v každé iteraci se mají hodnoty trails normalizovat do intervalu (0,1), aby s nimi bylo možno dále jednoduše počítat. Normalizace však vedla k příliš rychlému snižování hodnot trails na nulu. Tím pádem rychle mizela nejdůležitější informace pro rozhodování mravenců a to vedlo k degradaci algoritmu. Rozhodl jsem se tedy normalizaci obejít. Hodnoty trails nikdy nemohou být menší než 0 ani větší než 1,7. Pozorováním však bylo zjištěno, že jen málokdy jsou větší než 1 a když ano, tak jen o nějakou desetinu. Normalizaci tedy byla nahrazena ořízntutím hodnoty trails na 1, pokud byla větší než 1.

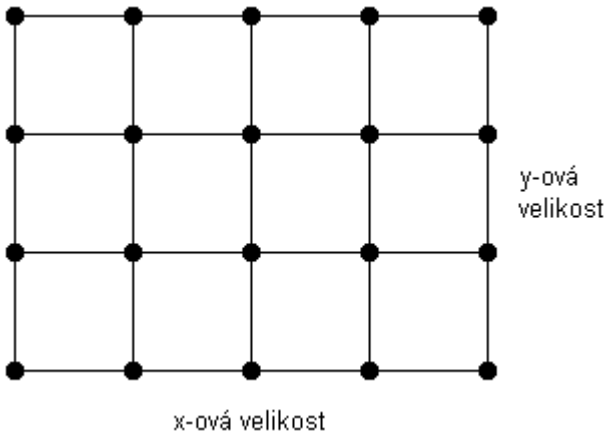
4.5 Program pro generování náhodných grafů

Pro účely testování byl dále vytvořen program, který generuje náhodný graf se zadaným počtem vrcholů a určitou hustotou. Hustota je průměrný počet hran mezi dvěma vrcholy, jinými slovy je to pravděpodobnost, že dva vrcholy budou spojeny hranou [6]. Pokud je tedy hustota například 0,5, pak se každá možná dvojice vrcholů spojí hranou s pravděpodobností 0,5.

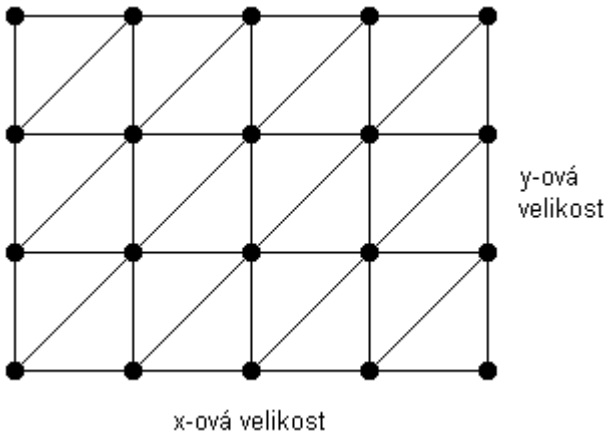
Programu je tedy třeba zadat počet vrcholů, hustotu a název výstupního souboru. Program generuje textový soubor s grafem podle specifikace uvedené v kapitole 4.1.

4.6 Program pro generování rovinných grafů

Dále byl vytvořen ještě jeden program, který generuje rovinné grafy. Byly navrženy dva jednoduché typy rovinného grafu a tyto byly nazvány typ 1 a typ 2 (viz obrázky 4.2 a 4.3.).



Obrázek 4.2 - Rovinný graf typu 1



Obrázek 4.3 - Rovinný graf typu 2

Programu je třeba zadat počet vrcholů na ose x, počet vrcholů na ose y, typ grafu a název výstupního souboru. Program generuje textový soubor s grafem podle specifikace uvedené v kapitole 4.1.

5 Návod na použití

Následující kapitola podává stručný návod na použití barvicího programu a programů pro generování grafů. Programy byly spouštěny přes příkazový řádek systému Windows.

5.1 Barvicí program Color

Tento program slouží k samotnému obarvení grafu. Programu je třeba zadat, kterou barvicí metodou má graf obarvit, jakým počtem barev, název vstupního souboru, kde je graf uložen, a název výstupního souboru, kam se uloží výsledek. Počet barev je nejméně 2 a nejvíce 120. Soubor s grafem musí být uložen ve stejné složce jako vlastní program. Po spuštění programu se vypíšou údaje, které jsme zadali, údaje o grafu načteném ze souboru a poté začne samotné barvení. Aby uživatel viděl, že se něco děje (což má velký psychologický význam), vypíše se do terminálu tečka za každých 100 iterací. Po dokončení barvení se vypíšou údaje o výsledku, například počet iterací, počet konfliktních hran a uplynulý čas. Výsledné obarvení grafu se uloží do výstupního souboru. Jestliže soubor zadaného jména již existuje, pak se přepíše.

Překlad:
`make`

Spuštění:
`Color.exe barvicí_metoda počet_barev vstupní_soubor výstupní_soubor`
- spustí barvení, parametr barvicí metoda je `-ant`, `-tabu` nebo `-gh`.

`Color.exe -h`
- vypíše nápovědu.

Příklad použití:
`Color.exe -ant 8 Graf.col Vysledek.txt`

5.2 Program Maker

Tento program slouží ke generování náhodného grafu. Programu je třeba zadat počet vrcholů grafu, pravděpodobnost výskytu hrany mezi dvěma vrcholy a název výstupního souboru, kam se uloží výsledek. Pravděpodobnost se zadává v procentech, tedy 0-100. Po spuštění se vypíšou námi zadané údaje a vytvořený graf se uloží do výstupního souboru. Jestliže soubor zadaného jména již existuje, pak se přepíše.

Překlad:
`make`

Spuštění:
`Maker.exe počet_vrcholů pravděpodobnost_hrany výstupní_soubor`
- vytvoří graf a uloží ho do souboru.

`Maker.exe -h`
- vypíše nápovědu.

Příklad použití:
`Maker.exe 30 50 Graf.col`

5.3 Program Planar

Tento program slouží ke generování rovinného grafu. Programu je třeba zadat počet vrcholů v x-ové ose, počet vrcholů v y-ové ose, typ grafu (viz kapitola 4.8) a název výstupního souboru, kam se uloží výsledek. Po spuštění se vypíšou námi zadané údaje a vytvořený graf se uloží do výstupního souboru. Jestliže soubor zadaného jména již existuje, pak se přepíše.

Překlad:
`make`

Spuštění:
`Planar.exe x-ový_rozměr y-ový_rozměr typ_grafu výstupní_soubor`
- vytvoří graf a uloží ho do souboru, typ je `1` nebo `2`.

`Planar.exe -h`
- vypíše nápovědu.

Příklad použití:
`Planar.exe 10 5 1 Graf.col`

6 Testování barvicích algoritmů

Po dokončení implementace bylo přistoupeno k samotnému testování vlastností barvicích algoritmů. Testy byly prováděny na mém osobním notebooku Lenovo, procesor Intel Core Duo 2,00 GHz, 504 MB RAM, operační systém Microsoft Windows XP.

6.1 Nastavení parametrů

Nejdříve bylo třeba nastavit parametry. Následuje jejich seznam:

Metoda ANTCOL:

- maximální počet iterací

Metoda TABUCOL:

- maximální počet iterací
- velikost okolí, tj. počet sousedů, mezi kterými se hledá nejlepší
- velikost tabu listu

Matoda GH:

- maximální počet iterací
- maximální počet iterací pro TABUCOL volaný z GH
- velikost populace, tj. počet řešení v populaci

Maximální počet iterací ANTCOLu byl podle jeho zadání ([6]) nastaven na 5000. U TABUCOLu a GH nebylo maximum iterací specifikováno, takže bylo nastaveno rovněž na 5000, aby se daly jednotlivé metody porovnávat.

Velikost okolí TABUCOLu také nebyla přesně specifikována v jeho zadání ([10]), byla zde pouze tabulka výsledků barvení, podle které bylo pro graf o velikosti 100 vrcholů použito okolí o velikosti 50, pro 300 vrcholů 170, pro 500 vrcholů 250 a pro 1000 vrcholů 600. Z těchto hodnot jsem určil, že okolí se bude nastavovat na polovinu počtu vrcholů v grafu.

Velikost tabu listu pro TABUCOL byla podle jeho zadání ([10]) nastavena na 7. (To znamená, že každá změna řešení je zaznamenána v tabu listu po 7 iterací.) Zde však nastal problém, protože při malém počtu zadaných barev docházelo k zacyklení. To bylo způsobeno tím, že při malém počtu barev a malém počtu konfliktních vrcholů je málo možných změn a může se stát, že všechny jsou v tabu listu. Toto bylo vyřešeno tak, že je-li zadáno méně než 5 barev, pak se velikost tabu listu vypočítá podle vzorce *tabu_velikost = 2 * počet_barev - 3*. Je-li zadáno 5 nebo více barev, pak zůstává původní hodnota 7.

Maximální počet iterací pro TABUCOL volaný z GH opět nebyl v zadání GH ([8]) konkrétně specifikován. Bylo zde však napsáno, že zkoušeli 5 různých hodnot - 250, 500, 1000, 2000 a 4000 - a zjistili, že nižší hodnoty dávají horší výsledky, ale algortimus je rychlejší, zatímco vyšší hodnoty dávají lepší výsledky, ale algoritmus je pomalejší. Zvolil jsem tedy prostřední hodnotu 1000 jako kompromis.

Velikost populace pro GH byla podle jeho zadání ([8]) nastavena na 10.

Většina těchto nastavení se nachází v souboru *Struktury.h*.

6.2 Změny kvůli časové náročnosti

Původně bylo v plánu provést testy po vzoru testů v publikaci o ANTCOLu ([6]), tedy na grafech s hustotou 0,5 a počty vrcholů 100, 300, 500 a 1000. Ukázalo se však, že by to bylo neúnosně časově náročné. Například algoritmus GH by podle mých výpočtů pro graf s 1000 vrcholy mohl trvat až 202 hodin (pokud by nenašel řešení a vykonal všech 5000 iterací). Bylo tedy rozhodnuto provést barvení na grafech s menším počtem vrcholů a místo hodnot 100, 300, 500 a 1000 byly zvoleny hodnoty 10, 30, 50, a 100. (Hustota 0,5 zůstala zachována.) Za tímto účelem byl také vytvořen výše zmíněný program, který požadované grafy vytvořil.

Později bylo rozhodnuto otestovat, jak se změní výsledky, když místo hustoty 0,5 budou mít grafy hustotu 0,1. Opět byly vytvořeny čtyři grafy s počty vrcholů 10, 30, 50 a 100.

Nakonec bylo provedeno barvení rovinných grafů. (Ty mají chromatické číslo maximálně 4.) Toto však žádné moc zajímavé výsledky nepřineslo, takže to bylo vyzkoušeno pouze na dvou grafech typu 1 a typu 2 (viz kapitola 4.8) o velikosti 7x7 vrcholů.

6.3 Výsledky testů

Testování probíhalo následovně: Pro každý graf byla spouštěna každá barvicí metoda s postupně klesajícím počtem barev, dokud byla tato metoda schopná nalézt řešení bez konfliktů. Pro každý počet barev byly provedeny tři pokusy. Výstupem je počet iterací, počet konfliktních hran a celkový čas barvení (v sekundách).

6.3.1 Výsledky pro grafy s hustotou 0,5

V tabulce 5.1 jsou výsledky testů na náhodném grafu s 10 vrcholy a hustotou hran 0,5. Na takto malém počtu vrcholů se zatím žádné rozdíly v nejmenším počtu barev neprojevily. Je zde ale jasně vidět velká časová náročnost algoritmu GH, který oproti ostatním běžel daleko déle, pokud řešení nenašel.

	ANTCOL			TABUCOL			GH		
barvy	iterace	konf. h.	čas [s]	iterace	konf. h.	čas [s]	iterace	konf. h.	čas [s]
4	0	0	0,000	8	0	0,000	0	0	0,000
4	0	0	0,000	4	0	0,000	0	0	0,000
4	0	0	0,000	8	0	0,000	0	0	0,000
3	5001	1	0,422	5000	1	0,093	5000	1	34,015
3	5001	1	0,438	5000	1	0,094	5000	1	34,250
3	5000	1	0,469	5000	1	0,093	5000	1	35,562
2	5000	6	0,235	5000	6	0,093	5000	6	34,203
2	5064	6	0,218	5000	6	0,078	5000	6	34,546
2	5014	6	0,250	5000	6	0,078	5000	6	34,359

Tabulka 5.1 - 10 vrcholů, hustota 0,5

V tabulce 5.2 jsou výsledky testů na náhodném grafu s 30 vrcholy a hustotou hran 0,5. Zde už byly algoritmy TABUCOL a GH schopny nalézt řešení se 7 barvami, zatímco ANTCOL nejlépe s 8 barvami. Pro 6 barev legální obarvení nalezeno nebylo, v počtu zbylých konfliktních hran byl ANTCOL nejhorší, TABUCOL lepší a GH nejlepší (ale trvalo mu to vždy skoro 7 minut).

	ANTCOL			TABUCOL			GH		
barvy	iterace	konf. h.	čas [s]	iterace	konf. h.	čas [s]	iterace	konf. h.	čas [s]
8	2	0	0,015	22	0	0,015	0	0	0,015
8	0	0	0,000	40	0	0,015	0	0	0,000
8	45	0	0,234	40	0	0,000	0	0	0,015
7	5000	1	36,094	1644	0	0,218	0	0	0,297
7	5000	1	37,000	1335	0	0,187	0	0	0,110
7	5743	1	41,578	437	0	0,093	0	0	0,046
6	5906	4	50,156	5000	3	0,500	5000	3	405,922
6	5008	4	42,015	5000	4	0,500	5000	3	413,063
6	6393	4	53,015	5000	3	0,515	5000	3	416,656

Tabulka 5.2 - 30 vrcholů, hustota 0,5

V tabulce 5.3 jsou výsledky testů na náhodném grafu s 50 vrcholy a hustotou hran 0,5. Algoritmy TABUCOL a GH byly v počtu barev opět lepší než ANTCOL. ANTCOL nebyl schopný nalézt legální řešení s 9 barvami, z důvodů časové náročnosti proto u něj již nebyly provedeny pokusy o obarvení s 8 barvami. Algoritmus GH se v případě nenalezení řešení už dostal přes 25 minut, a i algoritmus ANTCOL už skoro k 10 minutám.

	ANTCOL			TABUCOL			GH		
barvy	iterace	konf. h.	čas [s]	iterace	konf. h.	čas [s]	iterace	konf. h.	čas [s]
11	0	0	0,000	52	0	0,031	0	0	0,031
11	0	0	0,000	65	0	0,031	0	0	0,016
11	0	0	0,015	75	0	0,031	0	0	0,031
10	5000	1	219,594	101	0	0,062	0	0	0,062
10	0	0	0,000	89	0	0,062	0	0	0,031
10	216	0	5,235	92	0	0,046	0	0	0,031
9	9053	1	507,234	354	0	0,187	0	0	0,297
9	5039	3	277,671	330	0	0,171	0	0	0,219
9	10005	1	590,609	792	0	0,328	0	0	0,391
8				5000	4	1,594	5000	3	1525,640
8				5000	4	1,610	5000	3	1552,000
8				5000	3	1,609	5000	3	1491,281

Tabulka 5.3 - 50 vrcholů, hustota 0,5

V tabulce 5.4 jsou výsledky testů na náhodném grafu se 100 vrcholy a hustotou hran 0,5. Toto byl nejsložitější graf, na kterém byly prováděny testy. Algoritmus ANTCOL nebyl schopen nalézt legální řešení s 16 barvami, bylo u něj proto opět upuštěno od testů s nižším počtem barev. Algoritmy TABUCOL a GH nalezly řešení s počtem barev dokonce o 2 nižším než ANTCOL. Při neúspěchu se časová náročnost u ANTCOLu vyšplhala k 75 minutám a u GH se dostala přes 3,5 hodiny. TABUCOL netrval déle než 13 vteřin!

	ANTCOL			TABUCOL			GH		
barvy	iterace	konf. h.	čas [s]	iterace	konf. h.	čas [s]	iterace	konf. h.	čas [s]
18	0	0	0,016	178	0	0,360	0	0	0,281
18	0	0	0,016	121	0	0,281	0	0	0,328
18	0	0	0,016	169	0	0,391	0	0	0,235
17	5131	1	3033,656	223	0	0,516	0	0	0,313
17	61	0	9,656	336	0	0,828	0	0	0,453
17	6265	1	3702,016	148	0	0,297	0	0	0,328
16	5212	1	4417,766	413	0	1,000	0	0	1,110
16	5293	3	4520,735	282	0	0,687	0	0	0,625
16	5315	2	4166,047	474	0	1,141	0	0	0,735
15				1807	0	4,641	0	0	4,328
15				5000	1	12,953	11	0	49,985
15				981	0	2,453	0	0	14,188
14				5000	3	12,766	5000	1	12936,891
14				5000	4	12,828	5000	2	12613,969
14				5000	3	12,781	5000	2	12768,422

Tabulka 5.4 - 100 vrcholů, hustota 0,5

6.3.2 Výsledky pro grafy s hustotou 0,1

V tabulce 5.5 jsou výsledky testů na náhodném grafu s 10 vrcholy a hustotou hran 0,1. Zde se žádné rozdíly mezi barvicími algoritmy neprojevily, všechny nalezly legální řešení s jakkoliv nízkým počtem barev (až do hodnoty 2, barvit graf jednou barvou nemá smysl, ledaže by neobsahoval žádné hrany) a prakticky v nulovém čase. Tento graf je zřejmě příliš jednoduchý.

	ANTCOL			TABUCOL			GH		
barvy	iterace	konf. h.	čas [s]	iterace	konf. h.	čas [s]	iterace	konf. h.	čas [s]
4	0	0	0,000	2	0	0,000	0	0	0,000
4	0	0	0,000	1	0	0,000	0	0	0,015
4	0	0	0,000	1	0	0,000	0	0	0,000
3	0	0	0,000	4	0	0,000	0	0	0,000
3	0	0	0,000	2	0	0,000	0	0	0,000
3	0	0	0,016	2	0	0,000	0	0	0,000
2	0	0	0,000	4	0	0,015	0	0	0,000
2	0	0	0,000	5	0	0,000	0	0	0,000
2	0	0	0,000	12	0	0,000	0	0	0,015

Tabulka 5.5 - 10 vrcholů, hustota 0,1

V tabulce 5.6 jsou výsledky testů na náhodném grafu s 30 vrcholy a hustotou hran 0,1. Rozdíly v počtu barev se opět neprojevily, pouze lehké rozdíly v počtu zbylých konfliktních hran pro obarvení 2 barvami a GH opět trval nejdéle - skoro 3 minuty.

	ANTCOL			TABUCOL			GH		
barvy	iterace	konf. h.	čas [s]	iterace	konf. h.	čas [s]	iterace	konf. h.	čas [s]
4	0	0	0,000	10	0	0,000	0	0	0,000
4	0	0	0,000	12	0	0,000	0	0	0,000
4	0	0	0,000	11	0	0,000	0	0	0,000
3	5000	1	0,562	5000	1	0,266	5000	1	169,781
3	5000	1	0,578	5000	1	0,266	5000	1	169,562
3	5000	1	0,531	5000	1	0,281	5000	1	171,766
2	5001	7	0,453	5000	7	0,250	5000	7	149,734
2	6401	8	0,593	5000	7	0,250	5000	7	152,203
2	8965	8	0,781	5000	7	0,250	5000	7	151,203

Tabulka 5.6 - 30 vrcholů, hustota 0,1

V tabulce 5.7 jsou výsledky testů na náhodném grafu s 50 vrcholy a hustotou hran 0,1. Ani zde se rozdíly v počtu barev neprojevily, opět pouze rozdíly v počtu konfliktních hran a v čase. Algoritmus GH už se dostal skoro na 12 minut.

	ANTCOL			TABUCOL			GH		
barvy	iterace	konf. h.	čas [s]	iterace	konf. h.	čas [s]	iterace	konf. h.	čas [s]
4	0	0	0,000	41	0	0,000	0	0	0,000
4	7	0	0,000	31	0	0,000	0	0	0,000
4	0	0	0,000	31	0	0,000	0	0	0,000
3	5001	5	2,843	5000	4	0,688	5000	4	558,000
3	10891	6	6,000	5000	4	0,688	5000	4	566,390
3	9314	6	5,141	5000	4	0,688	5000	4	585,437
2	5003	33	1,593	5000	30	0,828	5000	30	710,046
2	5008	35	1,563	5000	31	0,828	5000	30	699,718
2	5002	32	1,562	5000	31	0,797	5000	30	696,953

Tabulka 5.7 - 50 vrcholů, hustota 0,1

V tabulce 5.8 jsou výsledky testů na náhodném grafu se 100 vrcholy a hustotou hran 0,1. Překvapivě ani zde se rozdíly v počtu barev neprojevily, přestože graf má už 100 vrcholů. Jsou zde ale jasné rozdíly v počtu konfliktních hran, například pro 4 barvy, kde algoritmus ANTCOL silně zaostává. A samozřejmě tvání algoritmu GH se opět prodloužilo, tentokrát až na 50 minut.

	ANTCOL			TABUCOL			GH		
barvy	iterace	konf. h.	čas [s]	iterace	konf. h.	čas [s]	iterace	konf. h.	čas [s]
5	18	0	0,031	100	0	0,062	0	0	0,031
5	0	0	0,015	81	0	0,047	0	0	0,078
5	0	0	0,000	74	0	0,031	0	0	0,110
4	5000	11	23,625	5000	2	2,890	5000	2	2840,156
4	5001	8	23,766	5000	4	2,906	5000	2	2814,593
4	5010	12	24,281	5000	2	2,875	5000	2	2797,500
3	5017	52	16,359	5000	35	3,094	5000	33	3005,656
3	5004	44	16,453	5000	36	3,235	5000	33	2936,219
3	5007	51	16,671	5000	34	3,031	5000	33	3020,109

Tabulka 5.8 - 100 vrcholů, hustota 0,1

Testy na grafech s hustotou hran 0,1 ukázaly pouze rozdíly v počtu konfliktních hran v případě nenalezení legálního řešení a rozdíly v časové náročnosti. V základní charakteristice, kterou je nejmenší počet barev, dopadly všechny tři algoritmy vždy stejně.

6.3.3 Výsledky pro rovinné grafy

V tabulkách 5.9 a 5.10 jsou výsledky testů na dvou rovinných grafech typů 1 a 2 (viz kapitola 4.8) s 49 vrcholy. Tyto grafy jsou opět velmi jednoduché, takže se na nich žádné rozdíly v počtu barev neprojevily, opět pouze rozdíly v počtu konfliktních hran a časová náročnost algoritmu GH (v tabulce 5.10 pro 2 barvy).

	ANTCOL			TABUCOL			GH		
barvy	iterace	konf. h.	čas [s]	iterace	konf. h.	čas [s]	iterace	konf. h.	čas [s]
4	0	0	0,000	20	0	0,015	0	0	0,000
4	0	0	0,000	17	0	0,000	0	0	0,000
4	0	0	0,015	14	0	0,000	0	0	0,000
3	0	0	0,031	33	0	0,000	0	0	0,015
3	0	0	0,000	27	0	0,000	0	0	0,000
3	0	0	0,015	16	0	0,000	0	0	0,000
2	0	0	0,000	44	0	0,015	0	0	0,000
2	0	0	0,000	38	0	0,015	0	0	0,000
2	0	0	0,000	34	0	0,000	0	0	0,000

Tabulka 5.9 - 49 vrcholů (7x7), typ 1

	ANTCOL			TABUCOL			GH		
barvy	iterace	konf. h.	čas [s]	iterace	konf. h.	čas [s]	iterace	konf. h.	čas [s]
4	0	0	0,000	46	0	0,015	0	0	0,015
4	0	0	0,000	35	0	0,000	0	0	0,015
4	0	0	0,000	41	0	0,000	0	0	0,015
3	0	0	0,000	66	0	0,015	0	0	0,015
3	0	0	0,015	86	0	0,016	0	0	0,015
3	0	0	0,000	88	0	0,031	0	0	0,015
2	5005	41	1,562	5000	37	0,671	5000	36	579,031
2	5003	39	1,578	5000	37	0,703	5000	36	575,421
2	5007	40	1,593	5000	38	0,734	5000	36	576,578

Tabulka 5.10 - 49 vrcholů (7x7), typ 2

6.4 Analýza výsledků

Tabulka 5.11 shrnuje pro každý graf a barvicí metodu nejlepší dosažený výsledek (tj. nejmenší počet barev, kterým se podařilo graf legálně obarvit), průměrný počet iterací, které k tomu byly potřeba, a průměrný čas, který to trvalo:

	ANTCOL			TABUCOL			GH		
graf	nej barvy	iterace	čas [s]	nej barvy	iterace	čas [s]	nej barvy	iterace	čas [s]
10; 0,5	4	0,000	0,000	4	6,667	0,000	4	0,000	0,000
30; 0,5	8	15,667	0,083	7	1138,667	0,166	7	0,000	0,151
50; 0,5	10	108,000	2,618	9	492,000	0,229	9	0,000	0,302
100; 0,5	17	61,000	9,656	15	1394,000	3,547	15	3,667	22,834
10; 0,1	2	0,000	0,000	2	7,000	0,005	2	0,000	0,005
30; 0,1	4	0,000	0,000	4	11,000	0,000	4	0,000	0,000
50; 0,1	4	2,333	0,000	4	34,333	0,000	4	0,000	0,000
100; 0,1	5	6,000	0,015	5	85,000	0,047	5	0,000	0,073
49; typ1	2	0,000	0,000	2	38,667	0,010	2	0,000	0,000
49; typ2	3	0,000	0,005	3	80,000	0,021	3	0,000	0,015

Tabulka 5.11 - Shrnutí všech grafů

6.4.1 Co bylo zjištěno

- Rozdíly mezi jednotlivými algoritmy (myšleno rozdíly v nejmenším počtu barev, kterými se podařilo graf obarvit) se projeví až na složitějších grafech, tj. grafech s větším počtem vrcholů a větší hustotou. Pro graf s 10 vrcholy a hustotou hran 0,5 skončily všechny algoritmy stejně. Pro grafy s hustotou 0,1 a pro rovinné grafy také skončily všechny algoritmy stejně. Tyto grafy byly zřejmě příliš jednoduché, než aby se na nich mohly projevit nějaké rozdíly.
- V algoritmu TABUCOL je graf na začátku obarven náhodně. Proto TABUCOL vždy projde alespoň několik iterací, než najde správné řešení. Naproti tomu v algoritmech ANTCOL a GH je graf na začátku obarven důmyslnějšími způsoby a může se stát, že je rovnou nalezeno správné řešení a vlastní cyklus algoritmu vůbec neproběhne. Pak je v tabulce u počtu iterací hodnota 0.
- Algoritmy TABUCOL a GH skončí nejpozději tehdy, když uběhne 5000 iterací od začátku algoritmu. Naproti tomu algoritmus ANTCOL skončí nejpozději tehdy, když uběhne 5000 iterací od posledního zlepšení řešení. Může se tedy stát, že celkový počet iterací, které ANTCOL potřeboval, je větší než 5000. Většinou ne o moc, ale někdy se dostane i k deseti tisícům.
- Všechny metody se velmi často chovaly tak, že buď nalezly řešení hned, nebo vůbec. Především algoritmus GH skoro nezná jiné hodnoty počtu iterací než 0 a 5000. To může být opět způsobeno jednoduchostí testovaných grafů, případně i špatným nastavením parametrů.
- Co se týče nejmenšího počtu barev, kterými algoritmus legálně obarvil graf, skončil ANTCOL vždy hůře nebo stejně jako TABUCOL a GH. TABUCOL vždy skončil stejně jako GH. Algoritmus GH ovšem na obarevní potřeboval více času.
- Pro daný počet barev, pro který se žádnému algoritmu nepodařilo obarvit graf bez konfliktů, skončil vždy TABUCOL s menším nebo stejným počtem konfliktních hran jako ANTCOL a GH skončil s menším nebo stejným počtem konfliktních hran jako TABUCOL. Zde se však

už naplno projevila velká časová náročnost algoritmu GH, který byl vždy nejpomalejší. ANTCOL pak byl pomalejší než TABUCOL.

- Shrnutí předchozích dvou bodů:
  Nejmenší počet barev: ANTCOL > TABUCOL = GH
  Nejmenší počet konfliktních hran: ANTCOL > TABUCOL > GH
  Uplynulý čas: GH > ANTCOL > TABUCOL

6.4.2 Srovnání s oficiálními výsledky

V publikaci o algoritmu GH ([8]) jsou tabulky poronávající algoritmy TABUCOL a GH. (Ty jsou zde ovšem nazvány TS a HCA.) Z těchto tabulek vyplývá, že GH vždy nalezl řešení s menším počtem barev než TABUCOL. Pro daný počet barev, pro který nalezly řešení oba algoritmy, potřeboval GH menší počet iterací než TABUCOL. Tyto výsledky se od výsledků v mé práci poněkud liší, neboť zde skončil GH vždy stejně jako TABUCOL. Toto je pravděpodobně způsobeno tím, že algoritmy byly testovány na jednodušších grafech, než na kterých je testovali autoři GH. Ti použili grafy s počty vrcholů od 250 do 1000.

Dále je třeba podotknout, že i když GH proběhne v méně iteracích než TABUCOL, tak ve skutečnosti v každé iteraci GH proběhne určitý nemalý počet iterací TABUCOLU. Ve výsledku je pak GH nepochybně časově náročnější.

V publikaci o algoritmu ANTCOL ([6]) je tabulka porovnávající mimo jiné metody ANTCOL, TABUCOL a GH. Vyplývá z ní, že v nejmenším počtu barev skončil TABUCOL vždy lépe než ANTCOL a GH skončil lépe nebo stejně jako TABUCOL. Rozdíl mezi TABUCOLem a GH, se však projeví až na grafu s 500 a více vrcholy, což souhlasí s mými výsledky a potvrzuje to, že rozdíly mezi některými algoritmy se projevují jen na složitějších grafech. Zde se tedy moje výsledky s jejich shodují.

Stejné závěry vyplývají i ze srovnání algoritmů TABUCOL a GH v publikacích o VNS ([11]) a VSS ([13]), kde je GH v počtu barev lepší než TABUCOL, ale na jednoduchých grafech se to neprojeví a oba skončí stejně.

6.4.3 Vyjádření časové složitosti

Jedním z bodů zadání původně bylo zjistit, jakou má ten který algoritmus časovou složitost a toto vyjádřit vzorcem. Toho bohužel nebylo dosaženo.

Asymptotická časová složitost vyjadřuje, jakým způsobem se změní časová náročnost algoritmu při změně počtu prvků na vstupu. Například lineární složitost $O(n)$ znamená, že když se počet prvků na vstupu změní, tak se stejným způsobem změní i časová náročnost výpočtu. [15][16]

Podle Wikipedie je časová složitost barvicích algoritmů $O(2^n n)$ [1]. To je ovšem pouze hrubý odhad pro všechny algoritmy. U algortimů, které jsou v této práci popsány, žádný vzorec složitosti zadán není. V publikacích, které jednotlivé algoritmy poprvé představují ([6][8][10][11][12][13]), není o časové nebo prostorové složitosti ani zmínka. Jsou zde vždy jen experimentální výsledky a naměřené časy. Konkrétní vzorec však žádný. To je způsobeno podstatou funkce těchto algoritmů, které mohou řešení nalézt hned nebo taky vůbec a mohou tedy potenciálně běžet až do nekonečna. Proto jsou omezeny maximálním počtem iterací a časová složitost závisí spíše na něm než na počtu prvků na vstupu. Kromě toho v sobě tyto algoritmy mají zabudovanou jistou náhodnost, takže dávají různé výsledky dokonce i pro stejná vstupní data. Časová složitost se tedy případ od případu liší a nelze ji vyjádřit vzorcem jako u deterministických algoritmů.

6.4.4 Časová náročnost a změny v průběhu barvení

Nyní bude na základě pokusů alespoň slovně popsáno, na čem závisí časová náročnost konkrétních algoritmů a jak se případně mění v průběhu výpočtu.

Rychlost algoritmu ANTCOL závisí na počtu vrcholů grafu, počtu hran, především však na počtu barev, protože na začátku je na každém vrcholu jeden mravenec od každé barvy. Máme-li tedy

v vrcholů a *k* barev, pak počet mravenců je $m = v*k$. Se vzrůstajícím počtem barev časová náročnost algoritmu strmě stoupá.

ANTCOL v každé iteraci prochází v cyklu všechny vrcholy, ale dále pátrá jen na těch, které jsou konfliktní. Ostatní přeskočí. Rychlost ANTCOLu je tedy na začátku menší, protože konfliktních vrcholů je hodně. Jak se řešení postupně zlepšuje a konfliktních vrcholů ubývá, algoritmus se zrychluje.

Rychlost algoritmu TABUCOL závisí především na počtu vrcholů a na velikosti okolí, ve kterém se hledá řešení (ta ale rovněž závisí na počtu vrcholů). Rychlost TABUCOLu nezávisí na počtu barev.

V každé iteraci TABUCOLu se hledá lepší řešení než dosud nejlepší nalezené řešení a pokud se nalezne, cyklus se přeruší a pokračuje se další iterací. Z toho vyplývá, že TABUCOL je na rozdíl od ANTCOLu rychlý na začátku, kdy je řešení nekvalitní (tím spíš, že TABUCOL začíná od náhodného obarvení) a není těžké nalézt lepší. S postupným zlepšováním řešení se TABUCOL zpomaluje, protože konfliktních vrcholů ubývá a nalézt lepší řešení je čím dál tím těžší.

Rychlost algoritmu GH závisí na počtu vrcholů grafu, velikosti populace, především však na počtu iterací TABUCOLu volaného z GH. GH totiž v každé své iteraci jednou volá metodu TABUCOL. Pokud bude mít GH nastaven maximální počet iterací na 5000 a TABUCOL volaný z GH na 1000, pak ve výsledku může proběhnout až 5 000 000 iterací TABUCOLu, čímž se GH stává velice časově náročným (přestože TABUCOL je poměrně rychlý). Podle mých výpočtů by obarvení grafu s 1000 vrcholy a hustotou hran 0,5 při nastavení GH na 5000 iterací a TABUCOLu na 1000 iterací trvalo asi 202 hodin. Pokud bychom snížili počet iterací TABUCOLu na 250, pak by stejné obarvení trvalo asi 15,5 hodiny, což je významný posun. (Pro testování je to však stále příliš velká hodnota, vzhledem k tomu, že barvicí algoritmus se musí pro každý graf spustit několikrát.)

Rychlost algoritmu GH se v průběhu barvení příliš nemění.

6.5 Modifikovaný ANTCOL

Při programování ANTCOLu byla náhodou objevena modifikace, která jistým způsobem mění jeho vlastnosti.

ANTCOL má fungovat tak, že se v každé iteraci snaží změnit barvu některého konfliktního vrcholu. Toho docílí tak, že z něj odstraní všechny mravence jeho barvy (protože barvicí algoritmus může vrcholu přiřadit pouze takovou barvu, kterou má alespoň jeden mravenec na něm). Algoritmus tedy pro každý konfliktní vrchol projde v cyklu všechny mravence a pokud mají barvu svého vrcholu, tak pro ně dál zjišťuje, jak pro ně bude ten který pohyb výhodný. Nakonec pak vybere mravence s nejvýhodnějším přesunem.

Výše zmíněná modifikace spočívá v tom, že cyklus prověřuje všechny mravence na vrcholu, nejen ty se správnou barvou. Modifikovaný ANTCOL pak začne dávat lepší výsledky v počtu barev. Zároveň se ale stane pomalejším, protože cyklus musí prověřit všechny mravence na vrcholu, nejen ty se stejnou barvou jakou má vrchol.

Následují výsledky testů modifikovaného ANTCOLu a jejich srovnání s původním ANTCOLem.

6.5.1 Výsledky testů

V následujících tabulkách jsou výsledky testů na grafech s hustotou hran 0,5. Tyto grafy byly zvoleny, protože při předchozích testech bylo zjištěno, že právě na nich se nejlépe projevují rozdíly mezi algoritmy.

V tabulce 5.12 jsou výsledky testů na náhodném grafu s 10 vrcholy a hustotou hran 0,5. Zde se žádné rozdíly neprojevily, patrně vlivem nízkého počtu vrcholů.

	ANTCOL			modifikovaný ANTCOL		
barvy	iterace	konf. h.	čas [s]	iterace	konf. h.	čas [s]
4	0	0	0,000	0	0	0,000
4	0	0	0,000	0	0	0,000
4	0	0	0,000	0	0	0,015
3	5001	1	0,422	5000	1	0,546
3	5001	1	0,438	5002	1	0,531
3	5000	1	0,469	5000	1	0,562
2	5000	6	0,235	5002	6	0,265
2	5064	6	0,218	5008	6	0,250
2	5014	6	0,250	5002	6	0,250

Tabulka 5.12 - 10 vrcholů, hustota 0,5

V tabulce 5.13 jsou výsledky testů na náhodném grafu s 30 vrcholy a hustotou hran 0,5. Zde již byl modifikovaný ANTCOL schopen nalézt legální řešení s počtem barev o 1 nižším než ANTCOL původní. V případě neúspěchu ale trval skoro třikrát déle.

	ANTCOL			modifikovaný ANTCOL		
barvy	iterace	konf. h.	čas [s]	iterace	konf. h.	čas [s]
8	2	0	0,015	3	0	0,078
8	0	0	0,000	16	0	0,281
8	45	0	0,234	6	0	0,140
7	5000	1	36,094	2445	0	48,671
7	5000	1	37,000	5000	1	102,593
7	5743	1	41,578	4765	0	92,000
6	5906	4	50,156	5065	4	120,671
6	5008	4	42,015	5013	4	116,156
6	6393	4	53,015	5334	4	122,906

Tabulka 5.13 - 30 vrcholů, hustota 0,5

V tabulce 5.14 jsou výsledky testů na náhodném grafu s 50 vrcholy a hustotou hran 0,5. Zde modifikovaný ANTCOL nenašel řešení s menším počtem barev než původní ANTCOL. V případě neúspěchu byl modifikovaný ANTCOL o trochu lepší v nejmenším počtu konfliktních hran, ale trvalo mu to i přes 25 minut, zatímco původnímu ANTCOLu ne více než 10 minut.

	ANTCOL			modifikovaný ANTCOL		
barvy	iterace	konf. h.	čas [s]	iterace	konf. h.	čas [s]
11	0	0	0,000	0	0	0,000
11	0	0	0,000	0	0	0,000
11	0	0	0,015	0	0	0,015
10	5000	1	219,594	268	0	23,578
10	0	0	0,000	161	0	9,046
10	216	0	5,235	154	0	12,969
9	9053	1	507,234	5065	1	1050,641
9	5039	3	277,671	5157	1	986,437
9	10005	1	590,609	7524	1	1528,141

Tabulka 5.14 - 50 vrcholů, hustota 0,5

V tabulce 5.15 jsou výsledky testů na náhodném grafu se 100 vrcholy a hustotou hran 0,5. Modifikovaný ANTCOL nalezl řešení s počtem barev o 1 menším než původní ANTCOL. V počtu konfliktních hran při neúspěchu byl opět lepší a ve spotřebovaném čase opět horší - skoro 5,5 hodiny. Tím překonal i algoritmus GH.

	ANTCOL			modifikovaný ANTCOL		
barvy	iterace	konf. h.	čas [s]	iterace	konf. h.	čas [s]
18	0	0	0,016	0	0	0,016
18	0	0	0,016	4	0	3,266
18	0	0	0,016	8	0	5,610
17	5131	1	3033,656	200	0	232,110
17	61	0	9,656	31	0	41,344
17	6265	1	3702,016	391	0	613,125
16	5212	1	4417,766	1938	0	4594,641
16	5293	3	4520,735	5117	1	16770,532
16	5315	2	4166,047	5224	1	19567,235

Tabulka 5.15 - 100 vrcholů, hustota 0,5

Tabulka 5.16 shrnuje pro každý graf a metodu nejmenší počet barev a průměrný čas na 1000 iterací:

	ANTCOL		modifikovaný ANTCOL	
graf	nej barvy	čas na 1000 it. [s]	nej barvy	čas na 1000 it. [s]
10; 0,5	4	0,068	4	0,081
30; 0,5	8	7,859	7	21,829
50; 0,5	10	54,596	10	197,001
100; 0,5	17	727,717	16	3239,207

Tabulka 5.16 - Shrnutí všech grafů

6.5.2 Analýza výsledků

Jak je vidět, tak modifikovaný ANTCOL dosáhl v počtu barev lepších výsledků než původní ANTCOL. Pro graf s 10 vrcholy se opět nic neprojevilo (kromě o trochu většího spotřebovaného času), ale už pro graf s 30 vrcholy dokázal modifikovaný ANTCOL nalézt řešení se 7 barvami, zatímco původní potřeboval nejméně 8. Pro graf s 50 vrcholy nalezl modifikovaný ANTCOL vždy řešení s 10 barvami (zatímco původní jenom někdy) a pro 9 barev mu zůstala vždy jedna konfliktní hrana (původnímu někdy i víc). Pro graf se 100 vrcholy modifikovaný ANTCOL opět zlepšil původní řešení, když nalezl řešení s 16 barvami (zatímco původní ANTCOL se 17 a to ještě ne vždy).

Na druhou stranu co se týče spotřebovaného času, byl modifikovaný ANTCOL ve všech případech o dost pomalejší než původní, což je v tabulkách vidět především tam, kde oba algoritmy řešení nenašly a prošly si všech 5000 nebo více iterací.

7 Závěr

Bylo popsáno šest různých metod barvení grafů a tři z nich byly implementovány. Tyto byly otestovány na různých grafech. Vzorce časových složitostí se zjisit nepodařilo, byl tedy proveden alespoň rozbor chování algoritmů na základě jejich testování.

Algoritmy byly porovnány vzhledem k úspěšnosti v nalezení obarvení s co nejnižším počtem barev a k celkovému času spotřebovanému k barvení. Žebříček celkové kvality těchto algoritmů vypadá takto:

- Na třetím místě je algoritmus ANTCOL, který se ukázal jako nejhorší co se týče počtu barev a ani časově nebyl nejrychlejší.
- Na druhém místě je algoritmus GH, který vždy skončil se stejným počtem barev jako TABUCOL, ale byl ze všech tří algoritmů časově nejnáročnější.
- Na prvním místě je pak algoritmus TABUCOL, protože dával nejlepší výsledky a zároveň byl nejrychlejší. A k tomu ještě připočtěme jeho jednoduchou definici a tím pádem i implementaci.

Ze srovnání zde získaných výsledků s oficiálními vyplývá, že tyto výsledky jsou podobné, ne však úplně stejné. Liší se v algoritmu GH, který by měl nacházet řešení s menším počtem barev než TABUCOL. Zde ale vždy vyšly oba tyto algoritmy stejně. To je způsobeno testováním na grafech s menším počtem vrcholů, rozdíl by se projevil až na větších a složitějších grafech.

Do budoucna by se dalo uvažovat o implementaci dalších barvicích metod a jejich porovnání s metodami již implementovanými. Také by bylo možné naprogramovat grafické rozhraní pro zobrazení a případně i tvorbu nových grafů. Toto by však mělo smysl pouze pro grafy s menším počtem vrcholů. Zobrazovat graf, který má 1000 vrcholů a 250 000 hran zřejmě nemá smysl. Rovněž by bylo možné provést testy algoritmů na grafech s větším počtem vrcholů, i když by to zabralo velké množství času.

Literatura

[1] *Graph coloring (Wikipedia)* [online]. Poslední modifikace: 3. dubna 2009. [cit. 2009-04-19]. Dostupné na URL: <http://en.wikipedia.org/wiki/Graph_coloring>.

[2] Diestel, Reinhard. *Graph Theory.* 3. vydání. Springer-Verlag Heidelberg, New York, 2005.

[3] Kovár, Martin. *Diskrétní matematika.* Brno, 2003.

[4] *Four color theorem (Wikipedia)* [online]. Poslední modifikace: 13. dubna 2009. [cit. 2009-04-19]. Dostupné na URL: <http://en.wikipedia.org/wiki/Four_color_theorem>.

[5] *Frequency assignment* [online]. Poslední modifikace: 9. listopadu 2005. [cit. 2009-04-25]. Dostupné na URL: <http://www.cs.cmu.edu/afs/cs/project/jair/pub/volume24/samaras05a-html/node33.html>.

[6] Hertz, Alain, Zufferey, Nicolas. *A New Ant Algorithm for Graph Coloring*. Proceedings of the Workshop on Nature Inspired Cooperative Strategies for Optimization: 51-60, NICSO 2006, Granada, Spain, 2006. Dostupné na URL: <http://www.gerad.ca/~alainh/Ants.pdf>.

[7] *Sudoku (Wikipedie)* [online]. Poslední modifikace: 23. dubna 2009. [cit. 2009-04-25]. Dostupné na URL: <http://cs.wikipedia.org/wiki/Sudoku>.

[8] Galinier, Philippe, Hao, Jin-Kao. *Hybrid Evolutionary Algorithms for Graph Coloring*. Journal of Combinatorial Optimization 3: 379-397, Kluwer Academic Publishers, 1999.

[9] Devlin, Keith. *Problémy pro třetí tisíciletí - Sedm největších nevyřešených otázek matematiky.* Dokořán a Argo, Praha, 2005. ISBN 80-7363-016-8 (Dokořán). ISBN 80-7203-739-0 (Argo).

[10] Hertz, Alain, de Werra, Dominique. *Using Tabu Search Techniques for Graph Coloring*. Computing 39: 345-351, Springer-Verlag, 1987.

[11] Avanthay, Cédric, Hertz, Alain, Zufferey, Nicolas. *A variable neighborhood search for graph coloring*. European Journal of Operational Research 151: 379-388, 2003. Dostupné na URL: <http://www.gerad.ca/~alainh/VNSEJOR.pdf>.

[12] Galinier, Philippe, Hertz, Alain, Zufferey, Nicolas. *An Adaptive Memory Algorithm for the k-Colouring Problem*. Revize 2004. Discrete Applied Mathematics 156: 267-279, 2003. ISSN: 0711-2440. Dostupné na URL: <http://www.gerad.ca/~alainh/G-2003-35.pdf>.

[13] Hertz, Alain, Plumettaz, Matthieu, Zufferey, Nicolas. *Varibale Space Search for Graph Coloring*. Discrete Applied Mathematics 156 (13): 2551-2560, 2006. Dostupné na URL: <http://www.gerad.ca/~alainh/VSSSoumission.pdf>.

[14] *Clique and Coloring Problems Graph Format* [online]. Poslední modifikace: 8. května 1993. [cit. 2009-04-19]. Dostupné na URL: <ftp://dimacs.rutgers.edu/pub/challenge/graph/doc/ccformat.tex>.

[15] *Asymptotická složitost (Wikipedie)* [online]. Poslední modifikace: 2. srpna 2008. [cit. 2009-04-19]. Dostupné na URL: <http://cs.wikipedia.org/wiki/Asymptotick%C3%A1_slo%C5%BEitost>.

[16] *Big O notation (Wikipedia)* [online]. Poslední modifikace: 17. dubna 2009. [cit. 2009-04-19]. Dostupné na URL: <http://en.wikipedia.org/wiki/Big_O_notation>.

Seznam příloh

Příloha 1. CD se zdrojovými texty, programem a dokumentací