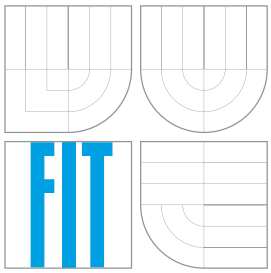
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

EXPERIMENTY S ROJOVOU INTELIGENCÍ (SWARM INTELLIGENCE)

EXPERIMENTS WITH THE SWARM INTELLIGENCE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. TOMÁŠ HULA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. LUKÁŠ GRULICH

BRNO 2008

Abstrakt

Práce se zabývá rojovou inteligencí jako podoborem umělé inteligence. Stručně popisuje biologické pozadí problematiky a zabývá se také principy hledání cest v mravenčích koloniích. Představena je i oblast kombinatorické optimalizace a detailně jsou definovány úlohy Travelling Salesman Problem a Quadratic Assignment Problem. Hlavní část práce sestává z popisu metod rojové inteligence pro řešení uvedených problémů a zhodnocení experimentů, které byly na těchto metodách provedeny. Konkrétně jde o algoritmy Ant System, Ant Colony System, Hybrid Ant System a Max-Min Ant System. V rámci práce byla také navržena a otestována vlastní metoda Genetic Ant System, která obohacuje základní Ant System mimo jiné o vývoj parametrů jednotek na základě genetických principů. V rámci obou řešených úloh jsou porovnány výsledky popisovaných metod společně s výsledky metod klasické umělé inteligence.

Klíčová slova

rojová inteligence, umělá inteligence, mravenčí kolonie, kombinatorická optimalizace, Travelling Salesman Problem, Quadratic Assignment Problem, Ant Colony Optimization, Ant System, Ant Colony System, Max-Min Ant System, Hybrid Ant System, Genetic Ant System

Abstract

This work deals with the issue of swarm intelligence as a subdiscipline of artificial intelligence. It describes biological background of the dilemma briefly and presents the principles of searching paths in ant colonies as well. There is also adduced combinatorial optimization and two selected tasks are defined in detail: Travelling Salesman Problem and Quadratic Assignment Problem. The main part of this work consists of description of swarm intelligence methods for solving mentioned problems and evaluation of experiments that were made on these methods. There were tested Ant System, Ant Colony System, Hybrid Ant System and Max-Min Ant System algorithm. Within the work there were also designed and tested my own method Genetic Ant System which enriches the basic Ant System i.a. with development of unit parameters based on genetical principles. The results of described methods were compared together with the ones of classical artificial intelligence within the frame of both solved problems.

Keywords

swarm intelligence, artificial intelligence, ant colony, combinatorial optimization, Travelling Salesman Problem, Quadratic Assignment Problem, Ant Colony Optimization, Ant System, Ant Colony System, Max-Min Ant System, Hybrid Ant System, Genetic Ant System

Citace

Tomáš Hula: Experimenty s rojovou inteligencí (swarm intelligence), diplomová práce, Brno, FIT VUT v Brně, 2008

Experimenty s rojovou inteligencí (swarm intelligence)

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Lukáše Grulicha. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Tomáš Hula
15. května 2008

Poděkování

Děkuji Ing. Lukáši Grulichovi za přínosné rady a pomoc při studiu této problematiky. Dále také Doc. Ing. Františku Zbořilovi, CSc. a Ing. Zdeňku Mazalovi za zapůjčení literatury.

Obsah

Kapitola 1

Úvod

Řešení úloh pomocí umělé inteligence je často prováděno tak, že je na daný problém aplikován určitý postup, který svými sekvenčními kroky vede k řešení úlohy. Jednotlivé kroky postupu jsou navrhovány s cílem, aby jeden po druhém transparentně přibližovaly výpočetní proces k nalezení správného řešení problému.

Rojová inteligence naproti tomu funguje na poněkud jiném principu, který je inspirován chováním společenství zvířat (především sociálního hmyzu). Problém je řešen více prvky s *jednoduchým chováním*, které spolu přímo či nepřímo interagují. Chování prvků však musí být navrženo tak, aby umožnilo vznik složitějšího chování při součinnosti prvků, a to bez jakéhokoliv centrálního řízení. Společná práce správně navržených jednotek při vhodném nastavení parametrů celého systému vede ke vzniku *kolektivního chování*, které převyšuje pouhé sjednocení jednoduchých chování jednotek. Získané komplexní chování potom představuje proces řešení problému.

Systémy postavené na principu rojové inteligence mají mnoho dobrých vlastností, které vyplývají především z decentralizovaného přístupu. Výhodou je zejména *robustnost* čili schopnost systému pokračovat v práci při selhání některé z jednotek a *flexibilita*, díky které je systém schopen správně pracovat i za měnících se podmínek. Nezanedbatelnou předností je také jednoduchá architektura jednotek.

Aplikace rojové inteligence nalezneme především v kombinatorické optimalizaci, rojové robotice a směrování v telekomunikačních sítích. Využívá se ale také v astronomii, filmovém průmyslu nebo pro modelování automobilového provozu.

Cílem této práce je právě zkoumání rojové inteligence. Kromě bližší charakterizace problematiky a nastínění jejího biologického pozadí popisuje použití této relativně moderní techniky pro řešení dvou vybraných problémů kombinatorické optimalizace, kterými jsou *Travelling Salesman Problem* a *Quadratic Assignment Problem*. Práce popisuje princip několika metod a hodnotí výsledky provedených experimentů. Záměrem je především zkoumání chování metod při řešení problémů, představení jejich různých mechanismů pro zlepšení kvality nalézaných řešení a odhalení vlivu parametrů. Všechny analyzované metody rojové inteligence i přidané metody klasické umělé inteligence jsou na závěr porovnány.

Kapitola 2 představuje umělou inteligenci a popisuje přírodní principy, které tvoří základ rojové inteligence. Vysvětluje také význam samoorganizace a stigmergie a princip fungování přírodních mravenčích kolonií. Dále je popsána podstata rojové inteligence, příslušné metody pro řešení úloh a praktické aplikace těchto technik. Kapitola 3 přibližuje kombinatorickou optimalizaci jako vybranou oblast aplikace rojové inteligence a detailně definuje Travelling Salesman Problem a Quadratic Assignment Problem. Kapitola 4 popisuje princip řešení úlohy Travelling Salesman Problem pomocí algoritmu Ant System, uvádí zpracované

výsledky experimentů a na závěr představuje porovnání použitých metod. Kapitola 5 podobně popisuje řešení úlohy Quadratic Assignment Problem, ale již pomocí více různých metod: Ant System, Ant Colony System, Max-Min Ant System, Hybrid Ant System a také metody Genetic Ant System, navržené v rámci této práce. U každé metody opět nechybí zhodnocení experimentů a závěrečné porovnání, a to i s metodami klasické umělé inteligence. Kapitola 7 stručně popisuje přínos celé práce, komentuje dosažené výsledky a uvádí některé náměty pro další možné zkoumání v této oblasti.

Práce navazuje na semestrální projekt. Přebírá z něj obecný úvod do rojové inteligence v podobě kapitoly 2 a přenesena a doplněna byla také kapitola o optimalizaci 3. Původní kapitola *Ant System*, která teoreticky popisovala řešení Travelling Salesman Problem a Quadratic Assignment Problem pomocí příslušné metody, byla rozdělena na samostatné kapitoly pro jednotlivé problémy. Do těchto nových částí práce byly doplněny další metody a výsledky provedených experimentů.

Kapitola 2

Rojová inteligence

Rojová inteligence je podoborem umělé inteligence, kterou si představíme v úvodu této kapitoly. Dále se budeme zabývat biologickým pozadím, jenž vedlo výzkumníky k úvahám o tom, jak využít poznatků o chování společenství zvířat pro počítačové řešení úloh. Blíže popíšeme chování mravenčních kolonií a nakonec se zaměříme na samotnou rojovou inteligenci a její aplikace.

2.1 Umělá inteligence

Existuje mnoho popisů, které se snaží co nejpřesněji a přitom obecně definovat umělou inteligenci. Marvin Minsky, významná postava na poli umělé inteligence, ji definoval následovně (citace z [8]):

> „Umělá inteligence je věda o vytváření strojů nebo systémů, které budou při řešení určitého úkolu užívat takového postupu, který – kdyby ho dělal člověk – bychom považovali za projev jeho inteligence.“

Takto definovaným cílem umělé inteligence je vlastně přiblížení se lidskému chování pomocí uměle vytvořeného systému. Umělou inteligenci dělíme na *slabou* (stroje se chovají jako inteligentní) a *silnou* (stroje skutečně přemýšlejí). Filosofickou otázkou zůstává, zda vůbec lze vytvořit stroj druhého typu. Pokoušíme se také stanovit kritérium, které by posuzovalo inteligenci stroje. Jedním z hodnotících postupů je např. *Turingův test*, kde se rozhodčí snaží rozeznat člověka od stroje.

Výsledky výzkumů umělé inteligence jsou úspěšně aplikovány v mnoha oblastech lidské činnosti. Vhodným příkladem je *robotika*, ve které se uplatňuje mnoho z umělé inteligence: principy počítačového vidění, mechanismy rozhodování, postupy pro plánování atd.

Rojová inteligence patří mezi relativně novou oblast umělé inteligence. Svým principem se dotýká především problematiky multiagentních systémů a můžeme zde také nalézt podobnost s některými rysy evolučních algoritmů.

2.2 Biologická inspirace

Řešení úloh na počítači, a s tím spojená oblast umělé inteligence, někdy z přírody přebírá evolucí ověřené postupy a rojová inteligence je toho příkladem. Pozdější úpravy algoritmů sice přinášejí zlepšující mechanismy za cenu vzdálení se původní biologické předloze, ale v základních myšlenkách je analogie s přírodou stále zřetelně viditelná.

Základem rojové inteligence je zkoumání živočichů, kteří žijí ve *společenstvích* (následujících informace byly získány především v [2] a [26]). Jde jak o ty nejmenší, jako jsou mravenci nebo včely, tak i o ryby, ptáky nebo savce. Důvodem jejich sdružování jsou výhody jak pro jednotlivce, tak i pro celou skupinu. Mohou se navzájem informovat o blížícím se nebezpečí, společně shánět potravu, spolupracovat na náročnějších úlohách apod.

Jedním z typických příkladů zvířat žijících v koloniích jsou *mravenci*. Přestože je jejich společenství často složeno z jedinců stejného druhu a stavby, rozdělují se bez centrálního řízení do skupin s vyhrazenými činnostmi. Jsou schopni stavět složitá hnízda, vyhledávat zdroje potravy a cestovat za nimi po efektivních cestách (viz sekci 2.2.3), stavět mosty z vlastních těl a mnoho dalšího. Jednotliví mravenci z kolonie provádějí jen jednoduché akce, ale každý z nich přispívá ke správnému běhu celého systému.

Popsané složité kolektivní chování jedinců vychází ze základů, které zároveň představují hlavní stavební kameny rojové inteligence. Jde především o interakci jednoduchých jednotek, samoorganizaci a stigmergii. Poslední dva zmíněné pojmy jsou detailněji popsány v následujících podkapitolách.

2.2.1 Samoorganizace

Samoorganizací (popis vychází z [23] a [2]) rozumíme proces, kdy organizace systému, složeného z jednoduchých částí, získává na složitosti vlivem interakcí částí podle jednoduchých pravidel, a to bez vnějšího řízení. Uvažujeme zde pravidla řízená pouze lokálními informacemi. Podobný proces nalezneme také v přírodě a důležitým následkem shlukování zvířat je potom vznik *kolektivního chování*. Jedinec často zvládne vykonávat pouze základní činnosti, ale vlivem vzájemných interakcí v rámci skupiny jedinců i interakcí s prostředím začne vznikat nové chování celku. Vzniklé chování může být výrazně složitější a možnosti skupiny jednotek rozsáhlejší. Popsaný princip vzniku společného chování skupiny jedinců je jedním ze základů *rojové inteligence*.

Samoorganizace přináší některé zajímavé výsledky. V původně stejnorodém prostředí vznikají dočasné struktury, přičemž některé z nich jsou udržovány a některé zaniknou. Některé z nich jsou zesilovány natolik, že se stanou dominantními a mezi takovými stavy pak systém balancuje, až nakonec jeden z nich převáží.

Vyjmenujme nyní čtyři základní aspekty, na kterých je samoorganizace založena:

Pozitivní zpětná vazba Jde o základní pravidla, která podporují samovolný vznik struktur. Řadíme zde *získávání* (recruitment) ostatních jednotek, kdy např. mravenci následují feromonové cesty vytvořené jinými mravenci. Kromě získávání sem patří i *odměna* (reinforcement).

Negativní zpětná vazba Tato zpětná vazba vyvažuje pozitivní zpětnou vazbu a udržuje tak stabilitu celého systému. Může být představována *soutěžením*, *nasycením* nebo *vyčerpáním*. Mezi zdroji potravy se například soutěží o to, který bude právě využíván.

Zesilování změn Změnami jsou myšleny chyby, náhodné procházky apod. Jsou klíčové pro hledání nových řešení a rozvíjení vznikající struktury. Příkladem je mravenec, který uhne z ověřené cesty. Tento (na první pohled špatný) krok může vést k nalezení zkratky k cíli cesty.

Vícenásobné interakce Jedinec je schopen vytvořit samoorganizující strukturu díky interakcím ostatních jedinců, kteří dokáží využívat práce své i ostatních a kteří tuto

strukturu udržují. Ilustrací principu je proces tvorby, využívání a posilování feromonových cest u mravenčích kolonií.

Samoorganizaci můžeme nalézt i v jiných oblastech, mezi které patří ekonomie, fyzika nebo matematické systémy, jako jsou celulární automaty.

2.2.2 Stigmergie

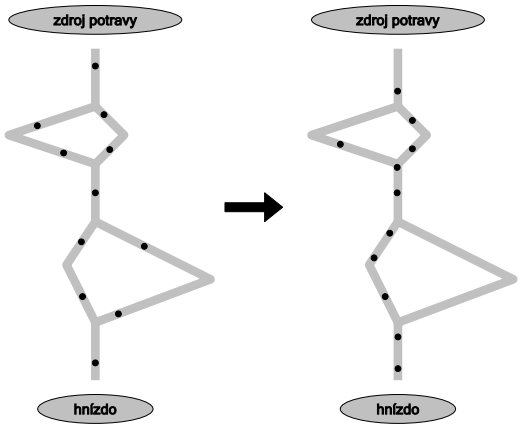
Stigmergie (viz [2]) je způsob komunikace v samoorganizujících systémech, kdy spolu jedinci interagují nepřímo přes prostředí, ve kterém se pohybují. Prováděné změny v prostředí jsou později jinými jednotkami detekovány a vedou k odpovídajícím akcím. Akce mohou opět měnit prostředí a vést tak k dalším akcím. Jedinci se tímto způsobem vzájemně ovlivňují, což umožňuje vznik kolektivního chování. Stigmergie vede k flexibilním systémům, kdy se prostředí postupně v krocích mění a jednotky se s těmito změnami vyrovnávají. Přínosem je také snížení četnosti komunikace a postačující jednoduché chování jedinců.

Příkladem stigmergie je využití feromonových stop pro získávání (recruitment) mravenců. Jeden mravenec mění prostředí tak, že zvyšuje stav feromonu na cestě, což přesvědčí jiného mravence k následování dané cesty.

2.2.3 Mravenčí kolonie

Tato práce se zaměřuje na optimalizaci inspirovanou mravenčími koloniemi. Nyní proto budou představeny v nich existující principy hledání výhodných cest ke zdrojům potravy (popis čerpá z [2] a [13]). Významnou roli zde hraje chemická látka zvaná *feromon*, kterou mravenci umí vylučovat i detekovat. Mravenec shánějící potravu totiž na zpáteční cestě od zdroje potravy pokládá feromonovou stopu. Jiný mravenec je tímto značkováním motivován a vybírá některý ze směrů se silnou feromonovou stopou. Při cestě zpět opět posiluje feromon na příslušné trase. Jde o příklad stigmergie, protože mravenci nepřímo komunikují pomocí prostředí.

Ze začátku je systém homogenní a mravenci cestují víceméně náhodně, ale později se začínají tvořit efektivní cesty, které jsou jejich využíváním dále posilovány. Celý systém je vyvažován *vypařováním* feromonu, které vede k postupnému zániku nepoužívaných, a tedy nepříliš vhodných tras. Vypařování však také omezuje sílu feromonových stop nejlepších



Obrázek 2.1: Cesty mravenců za zdrojem potravy (původní stav a stav po určité době)

cest, a umožňuje tak občasné náhodné vybočení z osvědčené cesty. Mravenec totiž nevybírá vždy směr s nejsilnější hladinou feromonu, což je klíčové pro dobré prohledávání prostoru. Výsledkem je samoorganizované kolektivní chování ilustrované na obrázku 2.1.

Prostředí s takovouto strukturou cest bylo reálně vytvořeno a byly na něm prováděny experimenty (viz [2]). Na počátku, ve feromonem neoznačkovaném prostředí, mravenci zkoušeli různé varianty cest. Rychleji se přitom vraceli ti, kteří vybírali krátké trasy ke zdroji potravy, a dříve tak ovlivnili rozhodování ostatních pomocí zanechané feromonové stopy. Postupem času se prosadila nejkratší cesta ke zdroji potravy, po které chodila většina mravenců. Jen někteří volili jiné alternativy, a podíleli se tak na prozkoumávání prostoru.

V jiném prostředí se dvěma stejně dlouhými cestami k cíli nebyly rovnoměrně využívány obě cesty. Přírodní systém mezi nimi nejdříve balancoval, avšak po čase jedna z nich vlivem náhodných odchylek převážila a cestovala po ní většina mravenců.

2.3 Podstata a využití rojové inteligence

Podle [5] můžeme uvažovat několik směrů výzkumu rojové inteligence. Jde o rozdělení na *přírodní* a *umělou* rojovou inteligenci podle toho, zda se zabýváme biologickými systémy nebo systémy uměle vytvořenými člověkem. Odlišujeme také *inženýrský* a *vědecký* přístup. Vědecký přístup spočívá v modelování rojových systémů a ve zkoumání vzniku kolektivního chování. Inženýrský přístup čerpá z výsledků vědeckých výzkumů a aplikuje je na reálné praktické problémy. Tato práce se řídí inženýrským přístupem k umělé rojové inteligencí.

Na rojovou inteligenci budeme proto pohlížet jako na podobor umělé inteligence, který je postaven na principech popsaných v předchozích sekcích. Základem je distribuovaný systém, složený z jednotek s jednoduchými pravidly chování. Jednotky mohou být vzájemně odlišné, ale žádná z nich systém centrálně neřídí. Jejich pravidla se řídí lokálními informacemi o okolí, a jednotky tak mají představu jen o určité části celého prostředí. Interagují jednak přímo mezi sebou, ale také nepřímo přes prostředí (viz sekci 2.2.2). U takového systému se za běhu projevuje samoorganizace (viz sekci 2.2.1) a interakce vedou ke vzniku kolektivního chování, které výrazně převyšuje schopnosti samostatných jednotek. Můžeme ho proto využít pro řešení různých úloh.

Mezi výhody rojových systémů patří jednoduché a nenáročné jednotky. Oproti např. inteligentnímu agentu se jedná o primitivní systémy s jednoduchými vzorci chování. Návrh jednotek však přesto může být poměrně obtížný. Řešení problému je nutné dosáhnout kombinací chování jednotek tak, aby vzniklo požadované kolektivní chování. Je tedy nutné správně definovat primitivní chování prvků, stanovit principy jejich interakce a nastavit vhodně parametry celého systému, abychom získali požadovaný efekt. Tato nepřímočarost návrhu je jedním z důvodů menší rozšířenosti systémů s rojovou inteligencí.

2.3.1 Významné metody

Nyní se zaměříme na nejvýznamnější metody a systémy založené na principech rojové inteligence.

Optimalizace pomocí mravenčích kolonií (Ant Colony Optimization, ACO)

Metoda, na niž je tato práce především zaměřena, vychází z chování mravenčích kolonií, které je blíže popsáno v sekci 2.2.3. Systém je postaven na větším počtu jednotek, jejichž

úkolem je procházení stavového prostoru problému. Využívá se virtuálního feromonu, který slouží k uchování kvalitních částí řešení. Podrobnosti o ACO lze nalézt v kapitole 3.4.1.

Optimalizace pomocí rojů částic (Particle Swarm Optimization, PSO)

Jde o stochastickou optimalizační techniku, jenž vychází z chování společenství ryb a ptáků (čerpáno z [7]). Proces začíná s množinou náhodných řešení (tzv. *částic*) a tyto částice se v prostoru řešení dále přesunují směrem k aktuálně optimálním částicím. Metoda je snadno implementovatelná, požaduje malé množství parametrů a je také silně odolná proti uváznutí v lokálním optimu. Slouží například k funkční optimalizaci nebo trénování neuronových sítí.

Na rojích založená správa sítí

Tato aplikace rojové inteligence (viz [2]) je zastoupena systémem *Ant-based Control* (ABC), adaptivním směrovacím algoritmem s jednoduchými agenty. Jejich úkolem je správa směrovacích tabulek tak, aby byly vyrovnávány změny zátěže a byl udržován co nejlepší výkon telefonní sítě. Vylepšený systém *AntNet* pro datové sítě (s přepínáním paketů) úspěšně konkuruje známým směrovacím algoritmům. Poskytuje dobrý výkon zejména při dynamických a náhodných zatíženích, kterými se vyznačuje i Internet.

Rojová robotika

Jde o kombinaci robotiky a rojové inteligence (popis vychází z [2]). Figuruje zde skupina jednoduchých robotů s převážně reaktivním chováním. Jednotky jsou levné, flexibilní a jejich chování není nutné často přeprogramovávat. Pro potřeby interakce se experimentuje např. s vůní nebo tepelnou stopou. Vzniklý distribuovaný systém může řešit složité úlohy a je robustní, protože porouchaný robot může být nahrazen ostatními. Důkladně navržené jednotky mají schopnost samoorganizace (viz sekci 2.2.1). Uplatnění roje robotů je orientováno např. na opravy motorů letadel, různé kontroly nebo pomoc při operacích.

2.3.2 Praktické aplikace

V této části bude krátce představeno několik společností, které komerčně využívají technik rojové inteligence.

- *AntOptima* (viz [1]) – Jde o švýcarskou společnost, která vznikla jako odnož výzkumného ústavu pro umělou inteligenci IDSIA. První skupinou nabízených produktů jsou aplikace pro plánování tras vozidel. Využívají metod založených na Ant Colony Optimization pro plánování primární a sekundární distribuce, tras donáškových služeb, obchůzek bezpečnostních služeb nebo rozvozu topné nafty. Další program slouží k plánování rozdělení zdrojů pomocí kombinace Ant Colony Optimization, tabu search a dalších metod. Společnost se zabývá také data miningem.

- *Southwest Airlines* (viz [13]) – Letecká společnost používá systém založený na rojové inteligenci pro plánování provozu na letišti. Systém hledá pro letadla co možná nejlepší využití přistávacích drah a hlavně bran. Nasazením nového systému bylo úspěšně dosaženo zadaného cíle, kterým bylo omezení doby čekání cestujících.

- Metody rojové inteligence lze také nalézt v produktech společností *NuTech Solutions*, *Eurobios* a dalších.

Kapitola 3

Optimalizace

V předchozích částech této práce bylo uvedeno množství různých aplikací rojové inteligence. Její počítačové využití je ve velké míře zaměřeno na oblast optimalizace, kde rojové systémy přinášejí užitečné a moderní přístupy k řešení úloh. Základem je zde samoorganizace velkého množství jednoduchých jednotek a jejich kolektivní chování reprezentuje proces řešení příslušné úlohy.

Rojová inteligence se uplatňuje zejména v kombinatorické optimalizaci, která bude popsána v následující sekci. Uvedeny jsou také definice dvou základních úloh tohoto typu optimalizace – Travelling Salesman Problem (úloha obchodního cestujícího) a Quadratic Assignment Problem, jejichž řešením pomocí rojových systémů se tato práce zabývá.

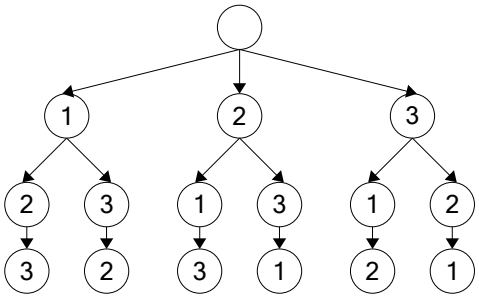
3.1 Kombinatorická optimalizace

Jedná se o optimalizační problémy s diskrétní množinou přípustných řešení (viz [9]). Chceme uspořádat zadané elementy tak, abychom dosáhli optimálního výsledku. Jednotlivá řešení hodnotíme účelovou (cenovou) funkcí. Například u úlohy obchodního cestujícího je touto funkcí délka cesty, jenž je dána posloupností měst.

Jednotlivá řešení úlohy jsou permutacemi vstupních elementů a prostor řešení je tvořen všemi možnými permutacemi. Počet těchto permutací s počtem elementů rychle roste a výsledný prostor řešení je u reálných problémů velmi široký. Konkrétně, máme-li n elementů, bude mít prostor řešení velikost $n!$. Hledání optimální permutace v tak velkém prostoru je velmi náročné a úlohy kombinatorické optimalizace často spadají do NP-těžkých problémů.

Základními přístupy, které prohledávají celý stavový prostor a jsou tak bez dalších úprav pro reálné systémy nepoužitelné, jsou *Breadth-first search* (BFS) a *Depth-first search* (DFS). Reprezentujeme-li stavový prostor kombinatorického problému jako strom (viz obrázek 3.1), pak BFS prohledává strom do šířky a vždy rozvine všechna částečná řešení na dané úrovni. Kompletní řešení jsou vytvořena až po sestrojení celého stromu. DFS (prohledávání do hloubky) oproti tomu jde vždy od kořene určitou cestou až k listovému uzlu a generuje tak postupným procházením větví jednotlivá kompletní řešení. Pro redukci stavového prostoru, a tím zrychlení jeho prohledávání, používáme *heuristiky*, které mohou vyloučit některá částečná řešení na základě jejich ohodnocení.

Mezi lepší metody patří *simulované žíhání*, *genetické algoritmy*, *tabu search* a další, které vesměs vycházejí ze zvoleného řešení a modifikacemi jej vylepšují. Optimalizace pomocí rojové inteligence souběžně prohledává prostor množstvím jednotek, které se řídí stopami zanechanými na cestách jinými jednotkami v předchozích krocích.



Obrázek 3.1: Stavový prostor všech možných permutací prvků 1, 2 a 3

3.2 Travelling Salesman Problem (TSP)

První úlohou, kterou si blíže charakterizujeme, je Travelling Salesman Problem neboli *úloha obchodního cestujícího* (čerpáno z [24]). Jde o dobře popsaný, snadno pochopitelný problém, jenž je možné řešit mnoha různými metodami. TSP je NP-těžký problém a není tedy znám algoritmus, který by jej dokázal řešit v polynomiálním čase.

Neformální definice zní následovně: máme zadaných n měst, kterými musí projet obchodní cestující. Každé město navštěvuje právě jednou a nakonec se vrací do výchozího města. Hledáme takovou posloupnost měst, která bude reprezentovat nejkratší možnou cestu.

Formálně lze problém popsat pomocí teorie grafů (teoretický základ následujícího popisu byl převzat z [28] a [27]). Uvažujeme úplný ohodnocený neorientovaný graf $G = (U, H, c)$ [1], kde $U = \{u_1, \ldots, u_n\}$ je množina uzlů, $H = \{\{u, v\} | u, v \in U \wedge u \neq v\}$ je množina hran (hranu spojující uzly u_i a u_j značíme zkráceně jako h_{ij}) a $c : H \to \mathbf{R}$ je cenová funkce, která určuje ohodnocení hran. V tomto grafu budeme hledat *hamiltonovskou kružnici* vyjádřenou permutací $\pi_{opt} = (u_1, \ldots, u_n) \in \Pi$, kde $u_i \in U$ pro všechna $i = 1, \ldots, n$ a Π je množina všech hamiltonovských kružnic daného grafu. Pro π_{opt} platí

$$\pi_{opt} = \underset{(u_1,\ldots,u_n)\in\Pi}{\operatorname{argmin}} \left(\sum_{i=1}^{n-1} c(h_{i\,i+1}) + c(h_{n\,1}) \right), \tag{3.1}$$

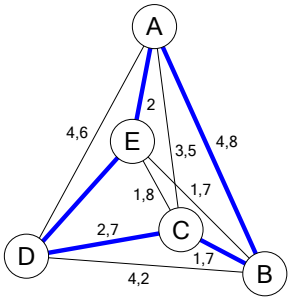
.

Uzly odpovídají městům, hrany představují cesty mezi městy a ohodnocení hran je dáno vzdálenostmi měst (viz obrázek 3.2). Obtížnost úlohy je dána velkým stavovým prostorem, protože kompletní graf s n uzly obsahuje $\frac{1}{2}(n-1)!$ hamiltonovských kružnic.

K řešení menších instancí TSP (s maximálně desítkami tisíc míst) můžeme využít některý z algoritmů pro hledání *přesného řešení*. Pro větší instance se však používají *heuristické algoritmy*. Tyto metody nezaručují nalezení skutečně optimálních řešení a jde tedy o kompromis mezi kvalitou řešení a výpočetní náročností. Získaná řešení se však často velmi blíží ideálním řešením při rozumných výpočetních časech. Mezi heuristické metody patří *lokální prohledávání*, kam řadíme i jednu z nejlepších metod pro řešení TSP – Lin-Kernighan algoritmus. Jiným přístupem jsou *konstruktivní heuristiky*, které postupně tvoří řešení (příkladem je algoritmus nejbližšího souseda). Dalšími vhodnými metodami jsou *si-*

[1] Existuje i *asymetrický TSP* (ATSP), kde nejsou vzdálenosti mezi městy symetrické, a bylo by proto nutné uvažovat orientovaný graf. Lze také řešit instance TSP, kde mezi některými městy nejsou cesty, což by odpovídalo neúplnému grafu.

mulované žíhání nebo *genetické algoritmy*. Travelling Salesman Problem jakožto problém hledání nejkratší cesty však také přímo směřuje k využití simulovaných *mravenčích kolonií*.



Obrázek 3.2: Ukázka zadání TSP úlohy pomocí symetrického grafu (modrou barvou je vyznačeno optimální řešení)

Základní aplikací úlohy obchodního cestujícího je obecně hledání nejkratšího okruhu mezi zadanými místy. Slouží proto k plánování rozvozu zboží, stanovení postupu pro vrtání děr do desek plošných spojů (jde o nalezení co nejkratší cesty, kterou musí absolvovat vrtací hlava), nalezení optimální sekvence zpracování úloh na jednom stroji atd.

3.2.1 Knihovna TSPLIB

K ověření výkonu různých algoritmů pro řešení Travelling Salesman Problem je vhodné vyzkoušet jejich práci na testovacích úlohách. Jedním ze souborů takových úloh je *TSPLIB* (viz [16]). V jednotném formátu jsou zde uvedeny instance symetrických i asymetrických TSP, ale i zadání pro ověření existence hamiltonovské kružnice v grafu, úlohy Sequential Ordering Problem a Capacitated Vehicle Routing Problem. U některých instancí problémů jsou uvedena optimální řešení, u jiných lze nalézt pouze nejlepší dosud známé řešení.

Podobnou knihovnou je *TSP Test Data* (viz [4]), která se zaměřuje především na větší instance TSP. Data jsou v souborech uložena ve formátu knihovny TSPLIB.

Formát testovacích souborů pro TSP a ATSP

Knihovna TSPLIB zavedla jednotný formát pro soubory s instancemi TSP úloh, který vypadá následovně:

`NAME` : <název problému>
`COMMENT` : <popis problému>
`TYPE` : <typ (TSP/ATSP)>
`DIMENSION` : <rozměr (počet měst)>
`EDGE_WEIGHT_TYPE` : <metrika pro výpočet vzdáleností>
<data>
`EOF`

Data jsou potom zadána v jedné ze dvou forem. První způsob uvádí souřadnice bodů a jejich vzdálenosti je před řešením problému nutné vypočítat podle uvedené metriky (`EUC_2D` například značí euklidovskou vzdálenost, `GEO` zeměpisnou vzdálenost). Pokud je metrika

zadána jako `EXPLICIT`, jedná se o druhý způsob definice dat, kdy jsou přímo zadány vzdálenosti mezi městy (a volitelně jejich souřadnice).

3.3 Quadratic Assignment Problem (QAP)

QAP (čerpáno z [2], [22] a [10]) můžeme popsat jako problém přiřazení n aktivit n lokacím tak, aby bylo přiřazení co nejvýhodnější. K dispozici přitom máme i informace o vzdálenostech lokací a síle toků mezi aktivitami. Vhodnost vybraného přiřazení je dána celkovou cenou přechodů mezi aktivitami, kde je cena přechodu vyjádřena jako součin toku mezi aktivitami a vzdálenosti příslušných lokací. Stejně jako v případě TSP jde o NP-těžký problém [2].

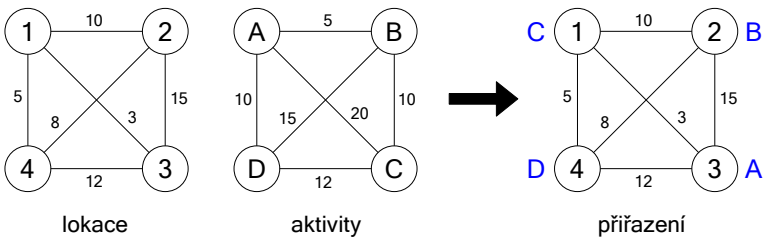
Formálně tedy máme n aktivit (množina A) a n lokací (množina L). Toky mezi aktivitami definujeme funkcí $f : A \times A \to \mathbf{R}$ a vzdálenosti mezi lokacemi funkcí $d : L \times L \to \mathbf{R}$. Uvedené funkce jsou často reprezentovány maticemi $\mathbf{F}$ a $\mathbf{D}$ o rozměrech $n \times n$. Přiřazení je dáno permutací $\pi \in \Pi(n)$ množiny $\{1, \ldots, n\}$, kde $\Pi(n)$ je množina všech možných permutací uvedené množiny. $\pi(i)$ potom určuje aktivitu, která je přiřazena lokaci i. Cílem je nalézt nejlepší přiřazení π_{opt}, tj. takové, u nějž je suma součinů toků a příslušných vzdáleností (čili hodnota účelové funkce C) nejmenší. Uvedený požadavek lze matematicky vyjadřit následovně:

$$C(\pi) = \sum_{i,j=1}^{n} d(i,j) f\left(\pi(i), \pi(j)\right) \tag{3.2}$$

$$\pi_{opt} = \underset{\pi \in \Pi(n)}{\operatorname{argmin}} C(\pi). \tag{3.3}$$

Protože je QAP příbuzný úloze obchodního cestujícího, řešíme jej podobnými algoritmy. Jde zejména o *genetické algoritmy*, *simulované žíhání*, *tabu search* nebo *GRASP* (Greedy Randomized Adaptive Search Procedure) a dobrých výsledků opět dosahujeme i využitím *simulovaných mravenčích kolonií*.

Jednou z aplikací tohoto problému je například nalezení vhodného rozvržení různých oddělení pro daný komplex staveb. Dalšími praktickými aplikacemi jsou efektivní rozmístění součástek na desku s plošnými spoji, návrh ovládacích panelů nebo plánování.



Obrázek 3.3: Ukázka zadání QAP úlohy pomocí symetrických grafů lokací a aktivit včetně optimálního přiřazení.

[2] TSP je totiž speciálním případem QAP.

3.3.1 Knihovna QAPLIB

Pro Quadratic Assignment Problem existuje knihovna testovacích příkladů (podobně jako TSPLIB v případě TSP) nazvaná QAPLIB (viz [3]). Mezi instance patří například problém přiřazování nemocničních oddělení budovám, návrh rozložení desky s plošnými spoji nebo některé uměle vygenerované problémy.

Instance v QAPLIB mají různý charakter a můžeme je rozdělit do čtyř základních typů (čerpáno z [21] a [19]):

Instance reálných problémů Jedná se o instance z reálného života, které reprezentují praktické aplikace QAP. Jejich matice toků mezi aktivitami jsou řídké a nenulové hodnoty jsou v maticích nepravidelně rozmístěny. Data jsou více strukturovaná než u jiných typů. Příklady instancí: `bur26*`, `els19`.

Instance podobné reálným problémům Zadání QAP tohoto typu byla uměle vygenerována tak, aby se charakterem co nejvíce blížila instancím předchozího typu. Důvodem jejich vzniku byly malé velikosti instancí skutečných problémů. Příklady instancí: `tai12b` až `tai150b`.

Nestrukturované náhodné instance Zcela náhodně vygenerovaná data, jejichž matice obsahují čísla získaná s rovnoměrným rozložením pravděpodobnosti, představují pro optimalizační algoritmy nejtěžší úlohy. Lokální optima jsou totiž rozprostřena ve stavovém prostoru. Příklady instancí: `tai12a` až `tai100a`.

Nestrukturované se vzdálenostmi na mřížce Matice toků opět obsahuje náhodně vygenerované hodnoty, ale v matici vzdáleností jsou uloženy Manhattanovské vzdálenosti bodů na mřížce. Instance mají více globálně optimálních řešení. Příklady instancí: `nug30`, `sko56`.

Formát testovacích souborů pro QAP

Instance v QAPLIB jsou uloženy v jednotném formátu, který je mnohem jednodušší než formát pro TSPLIB.

Data instancí jsou uložena v této podobě:

<n>
<matice vzdáleností>
<matice toků>

3.4 Metody kombinatorické optimalizace

3.4.1 Ant Colony Optimization

Jedná se o soubor optimalizačních technik. Prvotním a základním členem je algoritmus Ant System, jehož autorem je Marco Dorigo, jeden ze zakladatelů oboru rojové inteligence. Metody jsou inspirovány systémem přírodních mravenčích kolonií (konkrétně způsobem hledání cesty z hnízda ke zdroji potravy), který je blíže popsán v sekci 2.2.3.

Prostředí, ve kterém reální mravenci cestují, je zde nahrazeno *grafem* dané úlohy. V rámci grafu se pohybuje *velké množství jednotek* – jde vlastně o agenty s velmi *jednoduchým chováním*, kteří tak tvoří multiagentní systém. Rozhodování agentů (např. řízení cesty grafem u TSP) v každém kroku určují především dvě hlavní informace:

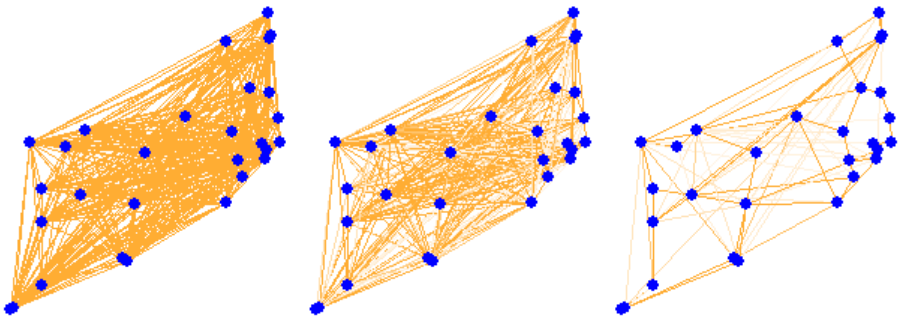
- *Heuristická vhodnost*: jde ve většině případů o statickou informaci, která může být předpočítána před začátkem řešení úlohy a během výpočtu zůstává neměnná. Např. u TSP tento aspekt představuje vzdálenost uzlů, čili ohodnocení příslušných hran.

- *Množství virtuálního feromonu*: úroveň hladiny feromonu se během řešení úlohy mění a využívá se jako dynamická informace o naučené vhodnosti daného kroku. Uvažujeme-li opět úlohu TSP, je hladina feromonu přiřazena ke každé z hran a je tím vyšší, čím více agenti danou hranu využívají.

Zatímco jsou heuristické znalosti při řešení optimalizačních problémů běžně využívány, je virtuální feromon jedním ze specifických základů optimalizace pomocí mravenčích kolonií. Rozložení feromonu v grafu úlohy tvoří společnou paměť, která uchovává části kvalitních řešení a umožňuje využívání předchozích zkušeností ostatních jednotek.

Virtuální mravenec tedy postupně prochází uzly jednak způsobem daným sémantikou úlohy, jednak na základě pravděpodobnostních funkcí, jejichž proměnné odpovídají zmíněným dvěma základním údajům [3]. Množství virtuálního feromonu je upravováno agenty. Děje se tak (podle konkrétní metody) po dokončení cesty grafem nebo i v jejím průběhu. Agenti tímto způsobem nepřímo komunikují, protože upravené množství feromonu ovlivňuje rozhodování dalších agentů. Vytváření feromonových stop umožňuje naplnit princip pozitivní zpětné vazby.

Důležitým principem, který také přebíráme od reálných mravenčích kolonií, je *vypařování feromonu*. Množství feromonu na hranách grafu tedy bude postupem času samovolně klesat. Kvalitní cesty budou neustále obnovovány, méně efektivní po určité době zmizí (viz obrázek 3.4). Vypařování také brání dominanci prvotních cest jednotek ve feromonových strukturách a obecně reprezentuje negativní zpětnou vazbu.



Obrázek 3.4: Vývoj hladiny feromonu v grafu při řešení úlohy TSP

Na práci algoritmů má významný vliv nastavení jejich *parametrů*. Mezi nejdůležitější parametry celého systému patří *počet virtuálních mravenců*, který musí být dostatečně velký pro vznik kolektivního chování (ovšem existují i upravené metody využívající třeba jen jedné jednotky). Dále se jedná například o *počáteční stav feromonu* na hranách nebo faktor *rychlosti vypařování*. Některé parametry však mohou mít u jednotlivých agentů různé hodnoty. Jde například o samostatné *váhy* heuristické znalosti a stavu feromonu pro výpočet pravděpodobnosti výběru následujícího kroku.

[3] Některé metody ale využívají jen informace o množství feromonu.

3.4.2 Genetické algoritmy

Genetické algoritmy (viz [17] a [25]) patří mezi *evoluční algoritmy*, společně např. s evolučními strategiemi nebo genetickým programováním. Tyto optimalizační metody jsou v základu inspirovány evolučními procesy v přírodě. Nejvýznamnější roli hraje populace jedinců, kteří představují jednotlivá řešení problému. Počáteční populace je složena z náhodně vygenerovaných řešení a v dalších krocích se vyvíjí podle principu upřednostňování kvalitních řešení. Kvalita celé populace, a tím i nalézaných řešení, se tímto způsobem postupně zlepšuje. S jednotlivými řešeními (jedinci) populace metoda pracuje v *zakódovaném tvaru*, a to nejčastěji ve formě řetězce symbolů. Obvyklé je binární zakódování, případně řetězce celých nebo reálných čísel.

Prvním krokem algoritmu je tedy vygenerování počáteční populace. Následuje *ohodnocení jedinců* pomocí zvolené fitness funkce, která každému jedinci přiřazuje číslo tím vyšší, čím kvalitnější řešení reprezentuje. Poté může dojít v závislosti na *podmínce ukončení* (např. dosažení maximálního počtu iterací nebo stagnace procesu řešení) k zastavení výpočtu a vrácení nejlepšího jedince jako výstupního řešení problému.

Algoritmus dále *vybírá jedince* pro křížení. Výběr je prováděn různými strategiemi – například strategie zvaná „ruleta“ vybírá na základě pravděpodobnostního rozložení, odvozeného od ohodnocení jedinců. Vybraní jedinci jsou dále po dvojicích *kříženi*, přičemž jsou vzájemně vyměňovány části jejich zakódovaného řešení. Následná *mutace* může u každého jedince provést malou náhodnou změnu v řešení. Způsob křížení a mutace je závislý především na podstatě problému a formě kódování řešení. Posledním krokem je *obnova populace*, ve kterém je vytvářena nová populace kombinací původní a právě vytvořené populace. Algoritmus se poté vrací ke kroku ohodnocení populace.

3.4.3 Simulované žíhání

Metoda simulovaného žíhání se zaměřuje na hlavní problém základních metod lokálního prohledávání stavového prostoru, kterým je časté uváznutí v lokálních optimech. V okolí aktuálního řešení totiž nemusí být nalezeno žádné lepší řešení. Aktuální řešení je tedy lokálním, ale ne nutně globálním optimem.

Klíčovou myšlenkou simulovaného žíhání je možnost přijetí řešení, které je méně kvalitní než aktuální řešení. Algoritmus díky tomu může vyváznout z lokálního optima a pokračovat dále v prohledávání stavového prostoru. Rozhodnutí, zda přijmout nový stav, je přitom ovlivněno pravděpodobnostní funkcí, která je závislá na proměnné označované jako *teplota*. Pravděpodobnostní funkce je sestavena tak, že při vysokých teplotách je s velkou pravděpodobností akceptováno i dočasné zhoršení kvality řešení a dochází k dobrému prohledávání prostoru řešení. Se snižující se teplotou klesá i daná pravděpodobnost a metoda již jen opatrně vylepšuje aktuální řešení. „Teplota“ se za běhu algoritmu snižuje a naplňuje tak popsaný princip.

Pravděpodobnostní funkcí pro akceptaci nového stavu, která splňuje uvedená kritéria, je například následující vztah (převzato a upraveno z [17]):

$$P = \min(e^{\frac{-(f(s')-f(s))}{T}}, 1), \tag{3.4}$$

kde s značí původní stav, s' nový stav, f funkci ohodnocení stavu (pro hledání minima), T teplotu a P pravděpodobnost přijetí nového stavu.

Kapitola 4

Travelling Salesman Problem

4.1 Ant System

V této části bude popsáno řešení TSP pomocí algoritmu Ant System. Popis uvažuje zadání úlohy grafem (viz sekci 3.2) a čerpá z [2]. Jde o vůbec první algoritmus ze souboru metod Ant Colony Optimization. Jeho výkon je (podle [2]) na malých instancích problému srovnatelný s klasickými metodami jako je simulované žíhání nebo genetický algoritmus.

4.1.1 Obecný pseudokód algoritmu

Základní princip je popsán následujícím pseudokódem:

Algoritmus 1 Pseudokód algoritmu Ant System

```
procedure AntSystem(τ0, ρ, Q, m, α, β, tmax)
    Inicializace
    while not Podmínka ukončení do
        Průchod mravenců grafem
        Aktualizace stavu feromonu
        if Nalezeno nové nejlepší řešení then
            Uložení nejlepšího řešení
        end if
    end while
    return Nejlepší řešení
end procedure
```

Význam vstupních parametrů je následující: τ_0 – počáteční hladina feromonu, ρ – koeficient vypařování feromonu, Q – koeficent pro přírůstek feromonu při aktualizaci stavu feromonu, m – počet mravenců, t_{max} – maximální počet iterací, α – váha feromonu a β – váha viditelnosti (viditelnost je odvozena z délky hrany) pro rozhodování mravence o přesunu do dalšího uzlu. Výstupem algoritmu je nejlepší nalezené řešení za celou dobu běhu algoritmu.

4.1.2 Fragmenty algoritmu

Nyní si postupně popíšeme důležité fragmenty pseudokódu.

Inicializace

Prvním úkolem inicializace algoritmu je nastavení počáteční hladiny feromonu všem hranám grafu na hodnotu τ_0. Algoritmus pro správnou funkci vyžaduje, aby platilo $\tau_0 > 0$. Na začátku procesu je také každý mravenec umístěn do náhodného uzlu grafu.

Podmínka ukončení

Algoritmus v základním tvaru postupně zvyšuje počítadlo iterací t a končí po t_{max} iteracích. Alternativně můžeme podmínku obohatit o test stagnace řešení problému.

Průchod mravenců grafem

V této fázi musí každý z m agentů absolvovat cestu grafem a vytvořit tak (vzhledem k definici TSP) hamiltonovskou kružnici. Pro každého z m agentů to znamená projít všech n uzlů grafu, přičemž musí být každý uzel navštíven právě jednou. Uvedenou podmínku zajišťujeme pomocí paměti označované jako *tabu list*, do které agent při své cestě postupně přidává navštívené uzly. Následující uzel agent vybírá z množiny kandidátských uzlů J_i^k (kde k je číslo agenta a i označuje aktuální uzel), která je odvozena z tabu listu a obsahuje zatím nenavštívená místa.

J_i^k tvoří pouze jakýsi omezený výběr, ale stále poskytuje několik různých uzlů, do nichž se agent při své cestě může dále přesunout. Výběr ze zbývajících uzlů je proto prováděn na základě pravděpodobnostní funkce, ve které hrají roli dvě důležité hodnoty spjaté s příslušnými hranami:

Viditelnost η_{ij} Tato veličina znamená vlastně blízkost uzlů i a j, protože se počítá jako $\eta_{ij} = \frac{1}{c(h_{ij})}$ ($c(h_{ij})$ je ohodnocení hrany). Při větší vzdálenosti uzlu se musí snižovat pravděpodobnost volby hrany, což použití viditelnosti naplňuje. Viditelnost představuje lokální informaci.

Množství feromonu τ_{ij} Hodnota vyjadřuje aktuální hladinu feromonu na hraně spojující uzly i a j (h_{ij}). Na rozdíl od viditelnosti je množství feromonu nestálé vlivem úprav ze strany jednotlivých agentů. Jde o globální informaci, která v sobě zahrnuje kombinace zkušeností mravenců.

Využitím výše popsaných veličin můžeme sestavit pravděpodobnost $p_{ij}^k(t)$, která představuje *pravděpodobnost přechodu* z uzlu i do uzlu j v iteraci t mravence k. Uvažujeme přechod do některého z kandidátských uzlů, tedy $j \in J_i^k$ [1]:

$$p_{ij}^k(t) = \frac{(\tau_{ij}(t))^\alpha \cdot (\eta_{ij})^\beta}{\sum_{l \in J_i^k} (\tau_{il}(t))^\alpha \cdot (\eta_{il})^\beta}. \tag{4.1}$$

Vztah definuje pravidlo, kterým se umělý mravenec řídí při výběru jednoho uzlu z množiny kandidátských uzlů J_i^k. Výběr je založen na náhodnosti a zvýhodněny jsou uzly s větší pravděpodobností přechodu. Vzhledem ke stochastickému řízení však může být vybrán jako následující i uzel, jehož příslušná pravděpodobnost je malá. Přestože se na první pohled může jednat o špatnou volbu, může vést k nalezení nového, lepšího řešení.

Způsob rozhodování agenta je ovlivněn parametry α a β. Zatímco α nastavuje míru důležitosti pro množství feromonu na příslušné hraně, β určuje míru důležitosti pro blízkost

[1] Pro uzly $j \notin J_i^k$ potom platí $p_{ij}^k(t) = 0$, protože již byly navštíveny a nesmí být znovu vybrány.

kandidátského uzlu. Pokud tedy bude platit $\frac{\alpha}{\beta} > 1$, bude se agent rozhodovat spíše podle množství feromonu než podle blízkosti uzlu a naopak při $\frac{\alpha}{\beta} < 1$. Při $\alpha = 0$ bude rozložení pravděpodobností pro jednotlivé uzly odpovídat greedy heuristice a zvýhodněny tedy budou bližší uzly bez ohledu na množství feromonu. Při opačné krajní situaci $\beta = 0$ nebudou mravenci vůbec uvažovat vzdálenosti uzlů a v několika prvních iteracích se prosadí nepříliš efektivní cesta.

Hrany, které agent k v iteraci t postupně používal při přesunech mezi uzly, tvoří cestu $T^k(t)$.

Aktualizace stavu feromonu

Poté, co všichni umělí mravenci dokončí své cesty, může algoritmus přejít do fáze aktualizace feromonu. Změny jsou dány úpravami množství virtuálního feromonu na cestách agentů a jeho vypařováním.

Každý agent zvyšuje hladinu feromonu na cestě, kterou absolvoval při procházení grafu. Velikost přírůstku je nepřímo úměrná délce cesty, čímž jsou upřednostňovány kratší a tedy lepší cesty. Přírůstek feromonu pro hranu z uzlu i do uzlu j *použitou* agentem k v iteraci t vyjadřujeme následujícím vztahem:

$$\Delta\tau_{ij}^k(t) = \frac{Q}{L^k(t)}, \tag{4.2}$$

kde $L^k(t)$ je délka cesty $T^k(t)$ agenta k v iteraci t a Q je již dříve zmíněný koeficient přírůstku feromonu. Pokud agent příslušnou hranu *nepoužil*, tedy $h_{ij} \notin T^k(t)$, potom bude přírůstek nulový ($\Delta\tau_{ij}^k(t) = 0$).

Celkové navýšení feromonu na hraně mezi uzly i a j bude dáno součtem příspěvků od jednotlivých agentů, čili

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta\tau_{ij}^k(t). \tag{4.3}$$

Pro vyhnutí se stagnaci, čili neustálému zesilování některé prvotní suboptimální cesty, je nutné do úprav rozložení feromonu zahrnout také vypařování feromonu. Feromon vyprchává v každém kroku na všech hranách a jeho novou hladinu po částečném vypaření v iteraci t můžeme popsat výrazem $(1 - \rho)\tau_{ij}(t)$, kde ρ je koeficient vypařování.

Konečně nový stav feromonu na hraně z uzlu i do uzlu j bude dán součtem stavu po vypaření části feromonu a přírůstků od agentů:

$$\tau_{ij}(t + 1) = (1 - \rho)\tau_{ij}(t) + \Delta\tau_{ij}(t). \tag{4.4}$$

Výstup algoritmu

Za běhu algoritmu je udržována nejlepší nalezená cesta T^+. Pokud v dané iteraci některý z agentů nalezne ještě efektivnější cestu, je nejlepší cesta patřičným způsobem aktualizována. Po skončení algoritmu je výsledná nejlepší cesta T^+ vrácena jako jeho výstup.

4.1.3 Verze s elitními mravenci

Jedna z vylepšených verzí algoritmu Ant System využívá tzv. *elitní mravence*. Úprava je postavena na následující úvaze: pokud v každé iteraci navíc posílíme dosud nejlepší řešení, bude toto relativně kvalitní řešení úlohy více využíváno při tvorbě nových řešení, čímž by se

měla obecně zlepšit kvalita řešení nalezených v další iteraci. Uvedený princip se v algoritmu promítá do fáze aktualizace stavu feromonu, kde je dočasně přidáno e elitních mravenců, kteří posilují celkově nejlepší cestu T^+. Jeden elitní mravenec tedy zvýší hladinu feromonu na hraně mezi uzly i a j, pokud hrana leží na nejlepší cestě ($h_{ij} \in T^+$), o následující přírůstek:

$$\Delta\tau_{ij}^{e}(t) = \frac{Q}{L^+}. \tag{4.5}$$

Příspěvek od elitního mravence pro hrany neležící na nejlepší cestě ($h_{ij} \notin T^+$) je nulový, čili pro takové hrany platí $\Delta\tau_{ij}^{e}(t) = 0$.

Výsledná rovnice pro aktualizaci feromonu, která vychází z původní rovnice 4.4 a přičítá příspěvek každého z e elitních mravenců daný vztahem 4.5, vypadá následovně:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t) + e \cdot \Delta\tau_{ij}^{e}(t). \tag{4.6}$$

4.1.4 Průměrný λ-faktor větvení

Pro posouzení charakteru struktur ve feromonovém rozložení je často používán *průměrný λ-faktor větvení* ($\overline{\lambda}$). λ-faktor větvení pro uzel i příslušného grafu počítáme jako počet uzlů j, pro které platí

$$\tau_{ij} > \tau_i^{min} + \lambda \cdot (\tau_i^{max} - \tau_i^{min}), \tag{4.7}$$

kde $\tau_i^{min} = \min_j \tau_{ij}$ (nejnižší úroveň feromonu pokud uvažujeme volbu všech možných uzlů j), $\tau_i^{max} = \max_j \tau_{ij}$ (nejvyšší úroveň feromonu) a $0 < \lambda < 1$ je parametr (často volíme $\lambda = 0,05$). Průměrný λ-faktor větvení počítáme přirozeně jako průměrou hodnotu λ-faktorů větvení pro všechny uzly grafu.

Tento ukazatel naznačuje míru dominance nebo naopak rovnoměrnosti v rozložení feromonu. Čím je nižší, tím méně relevantních voleb má v průměru agent v každém městě a tím více je tedy preferováno několik málo možných cest. Faktor větvení se používá i k detekci konvergence, kdy je hodnota $\overline{\lambda}$ blízká 1. To znamená, že v každém kroku přibližně jedna možnost volby výrazně převyšuje ostatní.

4.1.5 Úvod experimentů

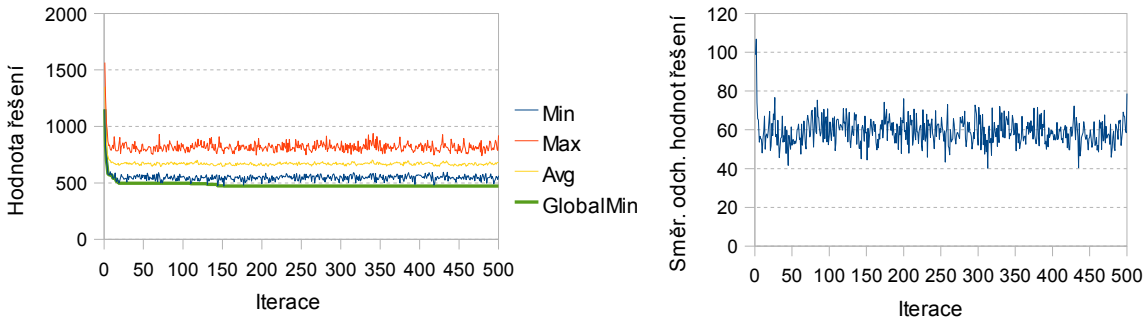
Následující části práce vycházejí z experimentování s algoritmem a pokouší se odhalit specifika chování algoritmu, stejně jako citlivost na jednotlivé parametry. Výchozí parametry pro provádění testů byly následující: $\tau_0 = 10^{-6}$, $Q = 1000$, $\rho = 0,7$, $\alpha = 2$, $\beta = 1$ a $e = 0$. Nebude-li uvedeno jinak, byly parametry daného testu nastaveny na tyto hodnoty.

Veškeré testy v této práci probíhaly na dvou počítačích. Prvním byl notebook IBM s Intel Core Duo T2300 1,66GHz a 1,5 GB RAM. Jako druhý (a hlavní) byl použit výzkumný server se Supermicro H8DME, 2× Dual Core AMD Opteron 2220 a 32 GB RAM. Měření časové složitosti probíhalo na prvním uvedeném stroji.

4.1.6 Charakter chování algoritmu

Zabývejme se nejdříve obecným charakterem chování algoritmu. Pro jeho ilustraci jsou uvedeny typické průběhy hodnot řešení a také vývoj standardní odchylky hodnot řešení v rámci celé skupiny jednotek (obrázek 4.1). Pokus byl proveden při tomto nastavení parametrů: $m = 50$, $t = 500$, $Q = 10$, $\rho = 0,9$, $\alpha = 1$ a $\beta = 1$ (zbylé měly výchozí hodnoty).

Jak je možné vidět na průběhu hodnot, v několika prvních iteracích algoritmus poměrně výrazně zlepšuje nacházená řešení a jejich hodnoty prudce klesají. Prudký sestup hodnot



Obrázek 4.1: Vlevo typický průběh algoritmu Ant System při řešení TSP úlohy `eil51`. Vpravo odpovídající průběh směrodatných odchylek.

však po úvodních iteracích ustává s tím, jak se vytvoří dominantní rozložení feromonu. Hodnoty nalézaných řešení se pohybují v přibližně stejných hranicích a algoritmus zkoumá sousedství dobrých, dříve nalezených cest. Nejlepší řešení je však překonáváno pouze v malých krocích vlivem náhodných odchylek při konstrukcích cest mravenců.

Směrodatná odchylka množiny řešení zůstává při výpočtu dostatečně velká a umožňuje tak nacházet nová řešení. Konvergenci, která sice snižuje diverzitu řešení, ale směřuje proces výpočtu k optimálnímu řešení, lze částečně navodit pomocí změn parametrů. Napříč prakticky všemi metodami optimalizace pomocí mravenčích kolonií se tedy vinou dvě protichůdné snahy: směřovat k optimálnímu řešení, ale nedosáhnout přitom předčasné konvergence (a tím uváznutí v lokálním optimu).

Algoritmus nedisponuje žádnými prostředky, které by v pozdějších iteracích detekovaly, že již není delší dobu překonáno nejlepší řešení. Výpočet v dalších iteracích proto musí spoléhat na miniaturní pokroky v delších časových intervalech.

4.1.7 Časová složitost

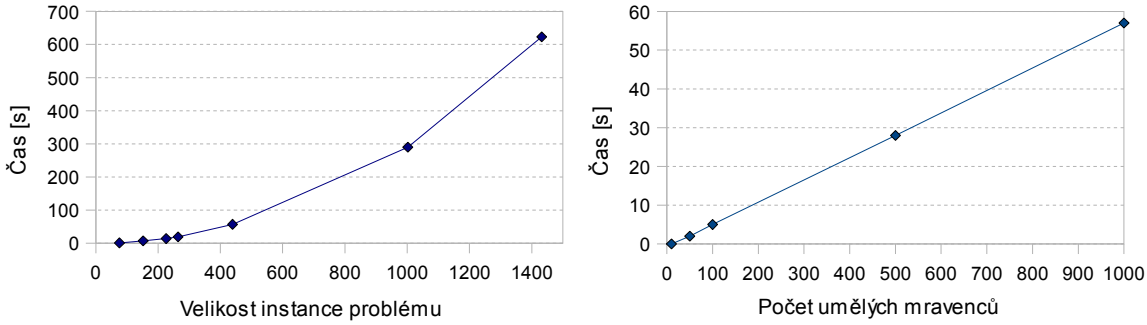
Nejdříve byl testován vliv velikosti řešené instance na časové nároky algoritmu. Jako základ časové analýzy bylo vybráno několik zadání z knihovny TSPLIB – konkrétně jde o symetrické instance `pr76`, `pr152`, `pr226`, `pr264`, `pr439`, `pr1002` a `u1432` (číslice v názvech instancí určují velikost problému). Počet umělých mravenců pro provedení testu byl stanoven na $m = 10$ a počet iterací na $t = 100$. Oproti výchozímu nastavení parametrů platilo $\rho = 0,5$. Výsledná závislost je zobrazena na obrázku 4.2 vlevo (průměr ze 3 testů).

Naměřená časová závislost je kvadratická a určuje ji agent, který prochází n měst a v každém se rozhoduje mezi $n - 1$ možnými následujícími městy, poté $n - 2$ městy atd.

Další proměnnou, která ovlivňuje dobu výpočtu, je počet umělých mravenců. Graf vpravo na obrázku 4.2 ukazuje, jak se mění doba výpočtu v závislosti na počtu agentů (měřeno pro počet mravenců 10, 50, 100, 500, 1000 a 5000 na úloze `eil50`, ostatní parametry byly stejné jako v předchozím pokusu).

Jak lze vidět, časová náročnost roste lineárně s počtem umělých mravenců. V knize [2] je uvedena teoretická časová složitost algoritmu Ant System jako $O(t \cdot n^2 \cdot m)$, kde t je počet iterací a m počet umělých mravenců. Vzhledem ke zjištěným časovým složitostem (a uvážíme-li, že časová závislost na počtu iterací je zjevně lineární), odpovídá experimentálně zjištěná složitost teoretické složitosti. Veškeré naměřené časy byly zapsány do tabulky A.1.

Vyšší nebo desetinné hodnoty parametrů α a β sice také mohou způsobit zpomalení

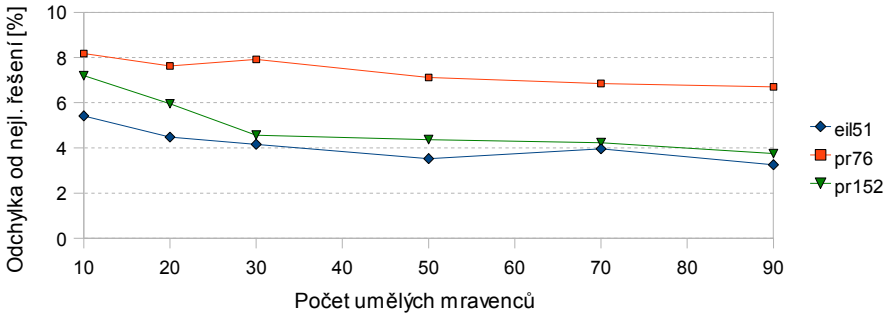


Obrázek 4.2: Vlevo závislost doby výpočtu na velikosti instance TSP. Vpravo závislost doby výpočtu na počtu umělých mravenců.

v důsledku výpočtu mocnin, jde však o implementační záležitost a mocniny lze částečně nahradit násobením.

4.1.8 Vliv počtu mravenců

V těchto experimentech byl zkoumán vztah počtu umělých mravenců a kvality nalezených řešení. Testy byly prováděny na instancích `eil51`, `pr76` a `pr266` a počet mravenců m byl nastavován na 10 až 90. Vypařování bylo oproti výchozím hodnotám parametrů nastaveno na $\rho = 0,9$. Výsledek je vidět na obrázku 4.3 a příslušná data jsou v tabulce A.2. Každý test zahrnoval 40 běhů algoritmu.



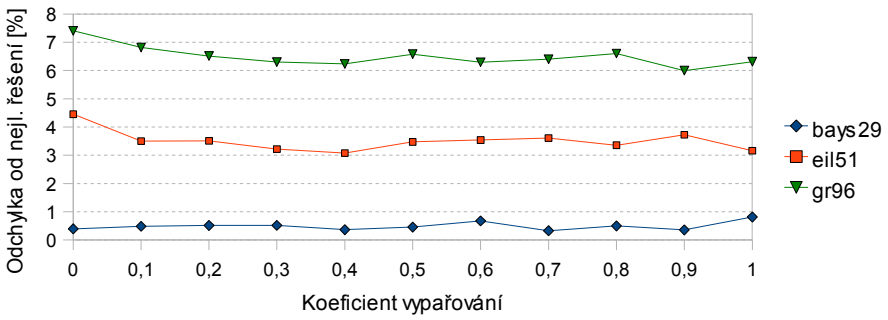
Obrázek 4.3: Závislost kvality řešení na počtu umělých mravenců.

Dle očekávání pomáhá větší počet agentů procesu řešení problému a klesá tak odchylka od nejlepšího řešení. Více agentů znamená důkladnější prohledávání stavového prostoru a tím větší šanci pro nalezení lepších řešení.

V literatuře (viz např. [2]) se často pro vhodný počet mravenců uvádí vztah $m = n$ a doporučuje se tedy použít tolik mravenců, kolik je měst v dané instanci problému. Testy však ukázaly, že další zvyšování počtu agentů může zlepšit výkon algoritmu. Přesto může být $m = n$ rozumným kompromisem mezi požadovanou kvalitou řešení a rychlostí jeho nalezení, protože s sebou vyšší počet agentů přirozeně nese větší časové nároky algoritmu (viz předchozí sekci 4.1.7).

4.1.9 Vypařování feromonu

Dalším z parametrů algoritmu Ant System je koeficient vypařování feromonu ρ. Vliv hodnot parametru od 0 do 1 byl testován na TSP instancích `eil51`, `gr96` a `pr152`. Každý z výsledků je průměrem ze 40 testů při parametrech $m = 51$, $t = 2000$ a $Q = 100$ (ostatní podle výchozích hodnot). Zjištěné závislosti ilustruje graf na obrázku 4.4 a konkrétní data jsou zanesena v tabulce A.3.



Obrázek 4.4: Závislost kvality řešení na koeficientu vypařování.

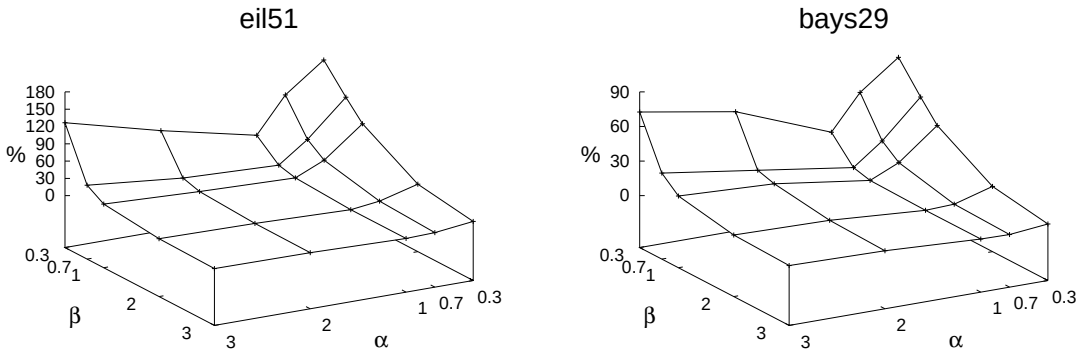
Prvotní test při omezení každého běhu na 1000 iterací nepřinesl jednoznačné výsledky a počet iterací byl proto zvýšen na 2000. Zde je viditelná nevhodnost nastavení $\rho = 0$ u instancí `eil51` a `gr96` a zlepšování výsledků až do $\rho = 0,4$. Dále je již průběh nepravidelný. Při použití lineární regrese bylo odhaleno nepříliš výrazné snižování odchylek od optimálního řešení při rostoucí hodnotě parametru ρ pro zmíněné dvě instance. Parametr při jejich řešení také ovlivňoval směrodatnou odchylku sady řešení v poslední iteraci, která vypovídá o diverzitě řešení na konci běhu algoritmu. Skupina řešení byla různorodější se vzrůstající hodnotou ρ, což může vysvětlovat mírně lepší výsledky pri vyšších hodnotách ρ.

4.1.10 Váhy viditelnosti a feromonu

Ant System je řízen i parametry α a β, které reprezentují váhy pro heuristickou a naučenou vhodnost volby hrany. Ovládáme jimi poměr důležitosti předpočítané viditelnosti a feromonové hladiny pro rozhodování umělého mravence. Druhou rolí každého z parametrů je ovládání síly rozlišení jednotlivých hodnot, protože parametry umocňují předpočítané vzdálenosti (resp. úrovně feromonu), viz vzorec 4.1. Má-li například agent na výběr mezi třemi hranami, kde jejich feromonové úrovně jsou $0,2$, $0,5$ a $0,3$, potom při parametru $\alpha = 1$ budou procentuální šance výběru hran 20%, 50% a 30%, zatímco při $\alpha = 2$ dostaneme $10,5\%$, $65,8\%$ a $23,7\%$.

První experiment opět testoval souvislost mezi nastavením těchto parametrů a kvalitou nalézaných řešení. Naměřená data jsou představena na obrázku 4.5 a také uvedena v tabulce A.4. Testy probíhaly na dvou různých instancích TSP pro všechny různé kombinace α a β při možných vahách $0,3$, $0,7$, 1, 2 a 3. Pro každou kombinaci byl vypočítán průměr ze 40 běhů algoritmu. Další parametry: $m = n$, $t = 2000$ a $Q = 100$ (zbylým parametrům byly nastaveny výchozí hodnoty).

Napříč všemi hodnotami váhy heuristiky se při testech na instanci `eil51` nejvíce osvědčila váha feromonu $\alpha = 1$. Větší či menší hodnoty tohoto parametru vždy přinesly (při konstantním β) zhoršení kvality nalézaných řešení. Vliv váhy heuristiky byl potom nejlepší



Obrázek 4.5: Závislost kvality řešení na vahách pro heuristickou informaci a hladinu feromonu.

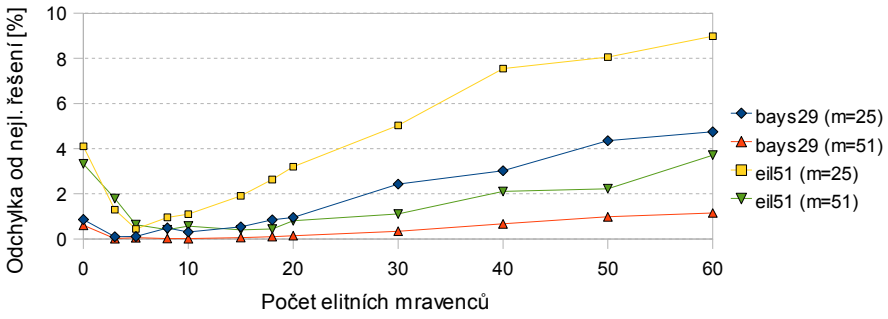
při dvou nejvyšších měřených hodnotách $\beta = 2$ a $\beta = 3$, zatímco nižší hodnoty obecně vedly k horším výsledkům. Jako nejlepší kombinaci vah tedy můžeme na základě měření stanovit $\alpha = 1$ a $\beta = 2$, protože toto nastavení zajistilo nejnižší průměrnou hodnotu nalezených řešení. Experimenty s menší instancí `bays29` přinesly podobné výsledky a trendy, jen se nejlepší kombinace vah přesunula na $\alpha = 0,7$ a $\beta = 3$. Výše uvedená nejlepší kombinace pro `eil51` ale přinesla druhý nejnižší průměr nalezených řešení. Algoritmus také pro `bays29` poskytoval řešení, která byla obecně bližší optimálnímu řešení a procentuální odchylky tak byly nižší než v případě `eil51`.

Hodnoty $\alpha > 1$ vedou ke zvýrazňování rozdílů mezi hladinami feromonových stop a preferují tak dominantní cesty. To však vede k méně důkladnému prohledávání stavového prostoru a v důsledku pak ke snižování diverzity řešení nalezených jednotlivými agenty. Dokonce se stává, že po určitém malém počtu iterací začnou všichni umělí mravenci konstruovat naprosto stejná řešení a v dalších iteracích již nedochází k žádnému zlepšení. Váha feromonu způsobuje výraznou preferenci počátečních vytvořených stop, které jsou takto dále posilovány, až se jedna z nich stane naprosto dominantní. Nízké hodnoty vah naopak nedostatečně využívají heuristických i naučených informací a málo směřují algoritmus k dobrým řešením.

4.1.11 Role elitních mravenců

V sekci 4.1.3 je popisována úprava původního algoritmu Ant System, která do změn rozložení feromonu zavádí i vliv elitních mravenců, kteří posilují nejlepší dosud nalezené řešení. Otázkou však je, jak vhodně nastavit parametr e, který označuje počet elitních mravenců. Vztah mezi hodnotami nalézaných řešení a nastavením parametru e zkoumaly experimenty na instancích TSP `eil51` a `bays29`, a to vždy pro počet běžných mravenců $m = 25$ a $m = 51$. Výsledný graf na obrázku 4.6 představuje průměrné odchylky od optimálního řešení pro hodnoty parametru 0 až 60 (přesná data lze nalézt v tabulce A.5). Každý běh absolvoval 1000 iterací a byl 40krát zopakován. Zbylé parametry měly výchozí hodnoty.

Uvedený graf ukazuje, že je nutné počet elitních mravenců nastavovat uváženě, protože jejich velká skupina již spíše degraduje výkon algoritmu. Kvalitní řešení byla nalézána při $e = 3$ až $e = 15$, přičemž výsledky naznačují, že pro menší počet obyčejných mravenců by se měl počet dodatečných elitních mravenců pohybovat spíše u spodní hranice uvedeného intervalu a naopak. Na základě experimentů tedy můžeme odhadnout, že vhodný počet



Obrázek 4.6: Závislost kvality řešení na počtu elitních mravenců.

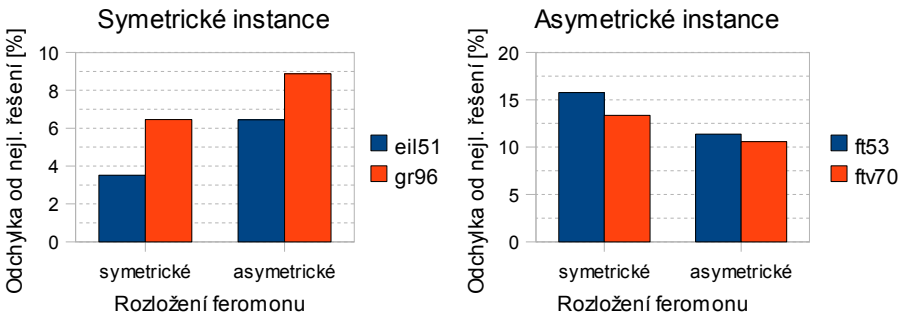
elitních mravenců odpovídá přibližně pětině počtu běžných umělých mravenců.

Elitní mravenci zvýhodňují dosavadně nejlepší řešení, které pak ostatní mravenci také více konstruují a opět je posilují. Použití přiměřeného počtu elitních mravenců proto znamená velmi mírné omezení diverzity feromonových stop, které v tomto případě přispívá k nalézání lepších řešení. Příliš velký počet elitních mravenců je spíše nežádoucí, protože tak velmi rychle vznikne několik málo dominantních cest, což způsobuje uváznutí algoritmu v suboptimálních stavech.

4.1.12 Symetrické a asymetrické rozložení feromonu

Pro každý přechod z uzlu i do uzlu j (kde uzly představují města) je v Ant System definována příslušná hodnota hladiny feromonu τ_{ij}. To ale znamená, že opačnému přechodu, tedy z uzlu j do ozlu i, náleží jiná hodnota τ_{ji}. Rozložení feromonu tedy zachycuje vhodnost hran s rozlišením směru jejich použití. Nabízí se ovšem i druhý přístup, kde platí $\tau_{ij} = \tau_{ji}$. Rozložení feromonu je potom symetrické a naučená vhodnost hrany nezávisí na směru, ve kterém po ní umělý mravenec přejde do dalšího uzlu.

Experimenty, které budou popsány níže, zkoumaly právě vliv způsobu uložení feromonových stop na výkon algoritmu. Pro tyto účely byly vybrány jak symetrické instance `eil51` a `gr96`, tak i asymetrické `ft53` a `ftv70`. U menších instancí (`eil51` a `fr53`) bylo při každém běhu provedeno 2000 iterací, zatímco u zbývajících větších instancí 1000 iterací. Výsledky testů (jde vždy o průměr z 15 běhů algoritmu) ilustruje graf na obrázku 4.7 a konkrétní odchylky od optimálního řešení jsou uvedeny v tabulce A.6.



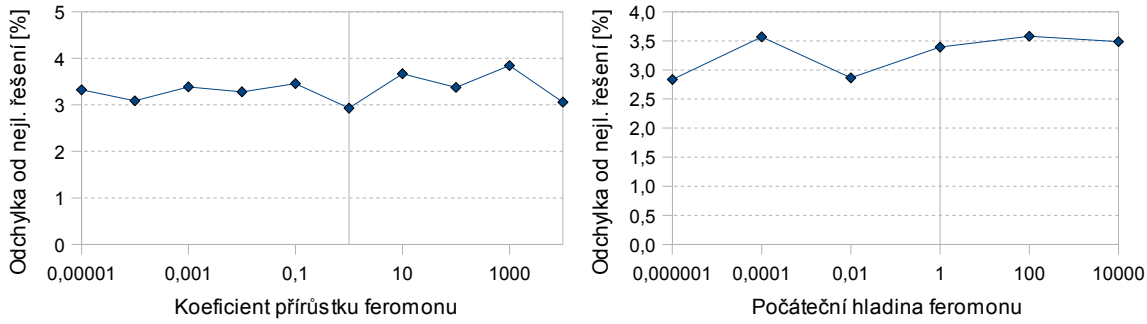
Obrázek 4.7: Závislost kvality řešení na způsobu uložení feromonu.

Výsledky experimentů jednoznačně ukazují, že je pro řešení symetrických problémů vhodné symetrické uložení feromonových hladin, zatímco u asymetrických se lépe osvědčilo asymetrické rozložení. Přestože se tento závěr zdá být poměrně očekávaným, o popsaném různém využití feromonu se použitá literatura nezmiňuje [2] a uvažuje tedy pravděpodobně asymetrické rozložení pro oba typy instancí TSP.

Za zmínku stojí také průměrná standardní odchylka řešení mravenců v poslední iteraci, která je vyšší u symetrické reprezentace feromonových stop nezávisle na typu problému. Porovnáním kvalit nalezených řešení také dojdeme k závěru, že je řešení asymetrických instancí pro Ant System (ale obecně zřejmě i pro další metody) obtížnější úlohou.

4.1.13 Další parametry

Experimenty s posledními dvěma parametry Q a τ_0 již budou představeny pouze ve zkrácené formě.



Obrázek 4.8: Vlevo závislost kvality řešení na koeficientu přírůstku feromonu. Vpravo závislost kvality řešení na počáteční hodnotě feromonu.

Koeficient přírůstku feromonu

Experimenty zkoumaly chování algoritmu pro několik možných hodnot parametru Q v rozsahu 10^{-5} až 10^4. Další parametry měly tyto hodnoty: $m = n$ a $t = 2000$ (ostatní parametry měly výchozí hodnoty). Bylo provedeno 40 opakování každého běhu a řešenou instancí byla `eil51`.

Na obrázku 4.8 vlevo lze vidět výsledné naměřené hodnoty, které jsou jinak přesně uvedeny v tabulce A.7. Výsledky měření potvrdily (viz např. [2]), že koeficient Q nemá zřetelný vliv na výkon algoritmu. Přestože byl testován poměrně velký rozsah hodnot parametru, odchylka od nejlepšího řešení pouze kolísá. Jen pokud proložíme graf křivkou logaritmické regrese (vzhledem k logaritmické ose x), zaznamenáme při zvyšující se hodnotě parametru velmi mírný, nepříliš významný nárůst hodnot řešení.

Počáteční hladina feromonu

Koeficient τ_0 určuje výchozí stav rozložení feromonu, který je v dalších iteracích upravován. Průměrná data ze vždy 40 běhů algoritmu ukazuje graf na obrázku 4.8 vpravo a konkrétní

[2] V [2] je například uvedeno, že typ problému neovlivňuje jeho řešení pomocí Ant System.

hodnoty jsou zaneseny v tabulce A.8. Pokusy byly prováděny na instanci `eil51` při hodnotách τ_0 postupně 10^{-6}, 10^{-4}, 10^{-2}, 1, 100 a 10^4. Zbývající parametry algoritmu Ant System byly nastaveny takto: $m = n$ a $t = 2000$ (ostatní jako ve výchozí sadě hodnot).

Kvalita nalézaných řešení při různých hodnotách τ_0 spíše kolísá stejně jako v případě předchozího parametru. Také další údaje, jako je střední odchylka hodnot řešení nebo diverzita rozložení feromonových stop mají poměrně vyrovnané hodnoty. Logaritmická regrese zde sice opět naznačuje degradaci kvality řešení při zvyšování hodnoty parametru, ovšem vztah, pokud existuje, zjevně není silný.

Drobný rozdíl najdeme až při detailnějším pohledu na průběhy hodnot řešení. Při vysoké počáteční hodnotě feromonu dochází v několika prvních iteracích k delšímu vypařování feromonu a průměrná hodnota řešení o to později sestoupí do rozumného lokálního optima, v jehož okolí jsou poté hledána další řešení. Tento jev však podle výsledků testů nemá významný vliv na výkon algoritmu.

4.2 Porovnání algoritmů

4.2.1 Popis testů a parametry algoritmů

Algoritmus Ant System bylo pro korektní zhodnocení nutné porovnat s dalšími algoritmy. Celkem tak bylo provedeno měření na instancích o velikosti 48 až 439 pro algoritmus Ant System (AS), Ant System s lokálním prohledáváním (AS + l. p.) [3], simulované žíhání (SA) a genetický algoritmus (GA). Zároveň je ale potřeba zohlednit časové nároky algoritmů, protože mohou stejný počet iterací provádět výrazně různou dobu. Běh všech algoritmů byl proto omezen na 5 minut, přičemž nastavení jejich parametrů bylo takové, aby tohoto časového limitu plně využily.

Každý experiment zahrnoval 20 běhů algoritmu, získané výsledky jsou průměrem těchto běhů. Na základě experimentů s algoritmy bylo zvoleno následující nastavení parametrů.

Ant System Koeficient vypařování $\rho = 0,9$, váha feromonu $\alpha = 1$, váha heuristiky $\beta = 2$, koeficient přírustku feromonu $Q = 1$ a počáteční hladina feromonu $\tau_0 = 10^{-6}$. Pro řešení nejmenší instance `att48` byl zvolen počet mravenců $m = 200$, který byl pro větší instance snižován, aby algoritmus mohl v daném časovém limitu absolvovat dostatečný počet iterací. Počet elitních mravenců e byl nastavován vždy na pětinu m.

Ant System s lokálním prohledáváním Parametry byly stejné jako u Ant System, jen počet umělých mranveců byl obecně nižší (pro nejmenší instanci `att48` $m = 50$).

Simulované žíhání Parametry byly nastavovány tak, aby byl v časovém limitu absolvován přibližně celý průběh od počáteční do koncové teploty. Pro instanci `att48` to například byla počáteční teplota 3000 a počet modifikací řešení pro jednu teplotu 110000. Pro všechny instance byla po provedení modifikací teplota vynásobena koeficientem 0,98.

Genetický algoritmus Společné parametry pro všechny instance: Pravděpodobnost mutace 0,9, pravděpodobnost křížení 0,1, výměna 10% staré populace, elitismus jako strategie výměny populace a výběr rodičů ke křížení pomocí ruletového výběru. Velikost populace pro `att48` byla 2000 a počet jedinců ke křížení 1600. Obojí bylo snižováno pro větší instance.

[3] Princip přidaného lokálního prohledávání je podrobněji popsán pro úlohu QAP v části 5.1.2. TSP se liší pouze modifikací řešení: zde jsou v posloupnosti vyměněna dvě náhodně vybraná města.

4.2.2 Výsledky porovnání

Naměřená data byla zapsána do tabulky na obrázku 4.9. δ označuje průměrnou odchylku od optimálního řešení, n_{opt} počet běhů, ve kterých bylo nalezeno optimální řešení (test na každé instanci zahrnoval vždy 20 běhů algoritmu), a t^* označuje průměrný počet iterací, po kterých bylo nalezeno nejlepší řešení běhu algoritmu. V horním řádku jsou názvy instancí, kde číslo v názvu vždy znamená velikost příslušné instance. Nejlepší výsledek pro každou instanci je označen kurzívou a modrou barvou.

		att48	eil51	st70	gr96	pr152	pr226	pr439	ft53	ftv70
AS	δ	0,20%	0,25%	1,58%	2,48%	1,39%	2,34%	14,95%	3,60%	4,37%
	n_{opt} | t*	10 | 779	5 | 656	0 | 1516	0 | 1238	0 | 879	0 | 582	0 | 371	0 | 1511	0 | 1056
AS + l. p.	δ	*0,00%*	*0,00%*	*0,00%*	*0,45%*	*0,23%*	1,23%	8,67%	*3,59%*	*3,82%*
	n_{opt} | t*	20 | 25	20 | 303	20 | 446	1 | 655	1 | 426	0 | 146	0 | 21	0 | 1463	0 | 679
SA	δ	0,20%	0,28%	0,42%	0,69%	0,46%	*1,00%*	*4,28%*	21,21%	43,68%
	n_{opt} | t*	9 | 244	2 | 394	8 | 404	1 | 225	3 | 291	0 | 276	0 | 320	0 | 238	0 | 291
GA	δ	6,00%	7,42%	23,16%	28,65%	161,61%	312,87%	256,08%	15,87%	33,67%
	n_{opt} | t*	0 | 3041	0 | 3005	0 | 3571	0 | 5828	0 | 7206	0 | 7708	0 | 3333	0 | 3242	0 | 3608

Obrázek 4.9: Porovnání výkonů algoritmů.

Jak lze vidět v tabulce, ve většině případů přinesl nejlepší výsledky algoritmus Ant System s lokálním prohledáváním. Oproti základní verzi je sice algoritmus časově náročnější, avšak potřebuje také k dosažení kvalitních výsledků méně iterací. Překonán byl pouze simulovaným žíháním v případě dvou největších měřených instancí. Simulované žíhání však i u jiných instancí (vyjma asymetrických) nalézalo velmi kvalitní řešení. Ant System bez lokálního prohledávání v porovnání se simulovaným žíháním řešil lépe pouze dvě nejmenší instance, u větších již byly jeho výsledky horší. Obecně se ale algoritmy založené na mravenčích koloniích osvědčily pro asymetrické instance, kde obě verze Ant System dokázaly asymetrická zadání řešit výrazně lépe než ostatní algoritmy. Jednoznačně nejméně kvalitní řešení nalézal genetický algoritmus. Symetrické instance řešily všechny ostatní algoritmy výrazně lépe. Markantní rozdíl je zejména u velkých instancí, kde jsou odchylky od optimálních řešení již velmi vysoké. Jen v řešení asymetrických instancí byl genetický algoritmus lepší než simulované žíhání. Jeho výsledky by však jistě bylo možné zlepšit volbou jiných parametrů nebo odlišnými strategiemi pro obnovu populace, výběr rodičů atd. V použité literatuře bohužel nebyly nalezeny vhodné parametry genetického algoritmu, podle kterých by bylo možné jej nastavit, a případně tak dosáhnout lepších výsledků.

Závěrem lze konstatovat, že zejména Ant System s lokálním prohledáváním je algoritmus, který lze s úspěchem použít pro řešení Travelling Salesman Problem. Verze bez lokálního prohledávání také přinesla poměrně dobré výsledky, ovšem je již překonávána simulovaným žíháním, které je přitom implementačně jednodušší. Simulované žihání jinak překonalo ostatní algoritmy při řešení velkých instancí. Ant System je však pouze základním algoritmem z rodiny algoritmů založených na simulovaných mravenčích koloniích. Lze proto očekávat, že by vylepšené verze dosahovaly ještě lepších výsledků.

Kapitola 5

Quadratic Assignment Problem

5.1 Ant System

V dřívější sekci 4.1 byl popsán algoritmus Ant System pro řešení TSP. V této sekci využijeme stejný algoritmus pro řešení jiné úlohy, a to *Quadratic Assignment Problem* (QAP). Úloha bude zadána (ve schodě se sekcí 3.3) pomocí matice vzdáleností lokací **D** a matice toků mezi aktivitami **F**. Budeme vycházet z podrobného popisu algoritmu Ant System pro TSP, a budeme se na něj často odkazovat. Základem je stejně jako u TSP pseudokód 1. Aplikace Ant System na QAP popisuje [2] a podrobně článek [11], ze kterých také vychází celá tato sekce.

5.1.1 Popis algoritmu

Inicializace

Ve fázi inicializace je opět nutné nastavit počáteční hladinu feromonu na hodnotu τ_0. Množství feromonu τ_{ij}, kde i je lokace a j je aktivita, tentokrát znamená naučenou vhodnost přiřazení aktivity j lokaci i. Pro všechna taková přiřazení tedy musí po inicializaci platit $\tau_{ij} = \tau_0$.

Dalším úkolem inicializace je předpočítat data, která budou nezbytná pro další kroky algoritmu. Při výpočtech budeme vycházet ze zadané matice vzdáleností lokací **D** a matice toků mezi aktivitami **F**. První potřebnou strukturou je *vektor vzdálenostních potenciálů* **d**, který vyjadřuje vzdálenostní potenciál pro každou lokaci tak, že sečte její vzdálenosti od všech sousedních lokací. Pro každou složku ($i = 1, \ldots, n$) vektoru **d** tedy platí

$$d_i = \sum_{j=1}^{n} d_{ij}, \tag{5.1}$$

kde d_{ij} je odpovídající prvek matice **D**. Menší vzdálenostní potenciál lokace znamená, že je v bližším okolí ostatních lokací

Odpovídajícím způsobem počítáme také *vektor potenciálů toků* **f**, který určuje potenciál toku pro každou z aktivit:

$$f_i = \sum_{j=1}^{n} f_{ij}, \tag{5.2}$$

kde $i = 1, \ldots, n$ a f_{ij} je odpovídající prvek matice **F**. Čím větší je potenciál toků dané aktivity, tím více komunikuje s ostatními aktivitami.

Na základě uvedených vektorů je vypočítána *matice spojení* $\mathbf{B}$, kde pro každý její prvek platí $b_{ij} = d_i \cdot f_j$ (tedy $\mathbf{B} = \mathbf{d} \cdot \mathbf{f^T}$). Prvek b_{ij} představuje heuristickou předpokládanou vhodnost přiřazení aktivity j lokaci i. Využíváme totiž tzv. *min-max spojovacího pravidla*, podle kterého lokacím s malým vzdálenostním potenciálem přiřazujeme aktivity s velkým potenciálem toku a naopak. Snažíme se tak stanovit vyrovnané optimální přiřazení. Pokud tedy budeme vybírat lokace postupně podle jejich vzdálenostního potenciálu od nejnižšího, budeme chtít lokaci i přiřadit aktivitu j s co nejvyšším potenciálem toku – chceme tedy, aby b_{ij} bylo co nejvyšší. Čím vyšší bude b_{ij}, tím vyšší bude pravděpodobnost přiřazení dané aktivity a pro algoritmus Ant System tedy uvažujeme $\eta_{ij} = b_{ij}$.

Podmínka ukončení

Stejně jako v případě TSP můžeme algoritmus ukončit po dosažení maximálního počtu iterací nebo v případě stagnace průběhu řešení.

Průchod mravenců grafem

Úkolem každého umělého mravence je cestovat v grafu lokací po hamiltonovské kružnici a navštívit tak každou lokaci právě jednou. Lokacím jsou postupně v pořadí jejich návštěv přiřazovány vhodné aktivity. Postupný výběr lokací se řídí vektorem vzdálenostních potenciálů $\mathbf{d}$. Cestování grafem začíná v lokaci s nejnižším potenciálem a končí v lokaci s nejvyšším potenciálem. Přiřazované aktivity vybíráme z množiny J_i^k, která obsahuje dosud nepoužité aktivity.

Mravenec se v každé lokaci rozhoduje mezi kandidátními aktivitami (všechny $j \in J_i^k$) podle jejich pravděpodobnosti. Uvažujeme-li feromonovou stopu τ_{ij} pro přiřazení aktivity j lokaci i a odpovídající předpokládanou vhodnost η_{ij} přiřazení, bude pravděpodobnost tohoto přiřazení pro mravence k v iteraci t vyjádřena naprosto stejně jako v případě TSP, tedy vztahem 4.1. Parametr α ovlivňuje váhu naučené (dynamické) vhodnosti přiřazení na základě množství feromonu a β nastavuje důležitost předem vypočítané (statické) vhodnosti přiřazení.

Agent tedy prochází jednotlivé lokace, na základě pravděpodobnosti jim přiřazuje aktivity a tvoří výslednou permutaci π, která reprezentuje více či méně kvalitní řešení QAP.

Aktualizace stavu feromonu

Tato fáze algoritmu je téměř stejná jako při řešení TSP (viz sekci 4.1.2). Změna stavu feromonu je opět dána jak vypařováním, tak i příspěvky od agentů. U všech přiřazení tedy v každé iteraci klesá hladina feromonu podle koeficientu vypařování ρ a naopak stoupá vlivem agentů, kteří dané přiřazení použili. Vztah pro výpočet nového stavu feromonu spojeného s přiřazením aktivity j lokaci i proto je stejně jako u TSP vyjádřen rovnicí 4.4.

Jediným rozdílem je způsob výpočtu příspěvků feromonu od agentů, který musíme upravit vzhledem k sémantice QAP. Kvalita řešení agenta je posuzována účelovou funkcí QAP (viz rovnici 3.2). Pro příspěvek od agenta k v iteraci t k hladině feromonu τ_{ij} tedy platí

$$\Delta\tau_{ij}^k(t) = \frac{Q}{C^k(t)}, \tag{5.3}$$

pokud agent použil přiřazení aktivity j lokaci i. V opačném případě platí $\Delta\tau_{ij}^k(t) = 0$. Lepší řešení, tzn. s nižší účelovou funkcí $C^k(t)$, jsou vzhledem k uvedenému vztahu více posilována.

Výstup algoritmu

Za běhu algoritmu Ant System se uchovává a pravidelně aktualizuje přiřazení s nejnižší účelovou funkcí π_{opt}. Toto nejlepší přiřazení je po splnění podmínky ukončení vráceno jako výstup algoritmu.

5.1.2 Lokální prohledávání

Algoritmus Ant System, stejně jako další algoritmy založené na umělých mravenčích koloniích, může být obohacen o lokální prohledávání okolí řešení (čerpáno z obsáhlejšího popisu v [19]). Takto vylepšené algoritmy se nazývají hybridní. Po dokončení konstruktivního procesu je na nalezené řešení aplikována některá z metod lokálního prohledávání stavového prostorou, která má dané řešení vylepšit. Využívají se jednoduché metody, ale například i simulované žíhání nebo tabu search. Složitější algoritmy jsou však náročnější na výpočet, což hraje významnou roli vzhledem k většímu počtu umělých agentů, kteří tohoto prohledávání využívají.

Za okolí řešení QAP mohou být považovány stavy, které vzniknou z aktuálního stavu vzájemnou výměnou aktivit, přiřazených dvěma vybraným lokacím. Důležitou částí je potom ohodnocení přínosu této výměny. Bylo by sice možné zjistit kvalitu obou řešení pomocí základní účelové funkce (viz rovnici 3.2), ale časová složitost by byla $O(n^2)$. Proto využíváme jiného vztahu, který je uveden v upravené formě rovnice z [6]:

$$\Delta(\pi,i,j) = (d_{ii}-d_{jj})(f_{\pi_j\pi_j}-f_{\pi_i\pi_i})+(d_{ij}-d_{ji})(f_{\pi_j\pi_i}-f_{\pi_i\pi_j})+ \sum_{k=1,k\neq i,j}^{n}(d_{ki}-d_{kj})(f_{\pi_k\pi_j}-f_{\pi_k\pi_i})+(d_{ik}-d_{jk})(f_{\pi_j\pi_k}-f_{\pi_i\pi_k}), \quad (5.4)$$

kde $\Delta(\pi,i,j)$ značí rozdíl účelových funkcí při výměně aktivit pro lokace i a j v přiřazení π. Uvedený vztah, jehož složitost je snížena na $O(n)$, slouží k porovnání kvalit řešení v rámci jednoduchých metod jako je *iterative improvement* (nazývaných také 2-opt). Metody prohledávají okolí a přecházejí na lepší řešení buď ihned při jeho nalezení, nebo až po prozkoumání celého okolí.

V této práci je způsob lokálního prohledávání inspirován v [6], kde jsou postupně náhodně vybírány dvojice lokací pro výměnu přiřazených aktivit a tato výměna je okamžitě provedena, pokud vede k lepšímu řešení. V průběhu procesu jsou vyzkoušeny všechny možné výměny a je tak prozkoumáno celé okolí původního stavu. V rámci tohoto procesu však není zaručeno nalezení lokálního optima.

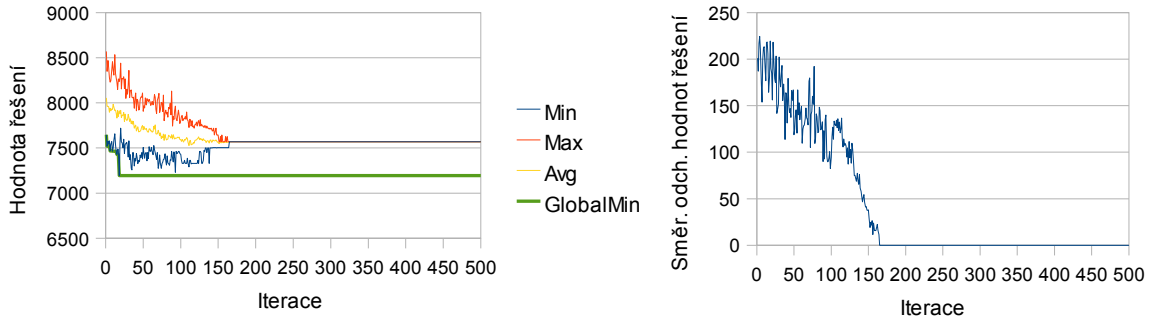
5.1.3 Úvod experimentů

S algoritmem Ant System již sice bylo experimentováno při řešení TSP (viz sekci 4.1.5), ovšem jeho chování může být odlišné vzhledem k jinému řešenému úkolu. Po zpracování výsledků testů proto můžeme mimo jiné zjistit, v čem se změní práce daného algoritmu při řešení dvou různých typů úloh.

V rámci testů budeme uvažovat *výchozí nastavení parametrů* $\tau_0 = 10^{-6}$, $Q = 100$, $\alpha = 1$, $\beta = 1$, $\rho = 0,5$, $e = 0$ (počet elitních mravenců) a $m = 50$. Případné jiné hodnoty parametrů budou u konkrétních testů explicitně uvedeny.

5.1.4 Charakter chování algoritmu

První analýza se týká celkového chování algoritmu při hledání řešení. Charakter průběhu sice samozřejmě závisí na nastavení parametrů, ale přesto můžeme nalézt jeho určité základní rysy. Podkladem pro další popis bude typický průběh hodnot řešení, který je zobrazený na obrázku 5.1 vlevo a který byl získán při výchozích hodnotách parametrů.



Obrázek 5.1: Vlevo typický průběh algoritmu Ant System při řešení QAP úlohy `nug30`. Vpravo odpovídající průběh směrodatných odchylek.

Nejlepší nalezené řešení je poměrně dost vzdáleno optimálnímu řešení (6124 pro instanci `nug30`). Nejvýraznějším prvkem chování je silná konvergence, která v určitém bodě přejde v naprostou stagnaci, kdy všichni mravenci konstruují stejná řešení (což pěkně ilustruje strmý pokles standardní odchylky nalézaných řešení, viz obrázek 5.1 vpravo).

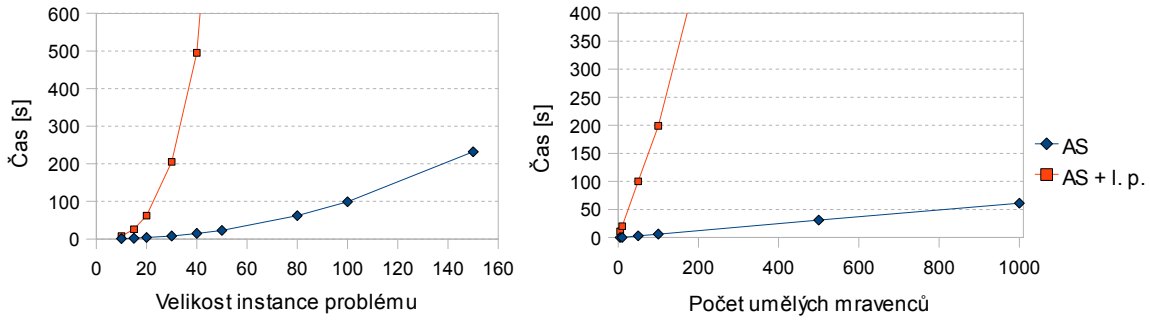
Algoritmus se navíc nedrží minulých dobrých řešení a konverguje k některému z dříve nalezených horších lokálních optim. Příslušné řešení je z hlediska feromonových stop stále více posilováno, zatímco všechny ostatní cesty jsou zeslabovány až do stavu, kdy jednotlivé mravence vedou feromonové stopy vždy po stejné cestě. Jde přitom o cestu poměrně nekvalitní.

Ant System byl průkopníkem v použití mravenčích kolonií pro optimalizaci QAP instancí a nebyl vybaven žádnými nástroji, které by byly užitečné pro překonání výše popsaných obtíží. Jedinou možností, jak omezit stagnaci a nasměrovat konvergenci směrem k dobrým řešením, je tedy použití vhodné sady parametrů. Experimenty přitom ukázaly, že výhodné hodnoty parametrů skutečně existují a silná konvergence algoritmu je za jejich použití výrazně omezena.

5.1.5 Časová složitost

Nejdříve se zaměříme na časové nároky algoritmu vzhledem k velikosti instance problému. Testy probíhaly na zadáních `tai10a`, `tai15a`, `tai20a`, `tai30a`, `tai40a`, `tai50a`, `tai80a`, `wil100` a `tho150` z QAPLIB. Výsledky zobrazuje graf na obrázku 5.2 vlevo, který byl získán na základě průměrných dat ze 3 běhů algoritmu. Pokus se zaměřil jak na původní algoritmus Ant System (AS), tak na jeho verzi s přidaným lokálním prohledáváním (AS + l. p.), kde však vzhledem k časovým nárokům měření proběhla pouze pro instance o velikosti 10 až 40. Počet iterací algoritmu byl nastaven na $t = 500$.

Zjištěná časová závislost je kvadratická, stejně jako v případě řešení TSP. Je totiž sice složitější vypočítat hodnotu nalezeného řešení (u TSP $O(n)$, u QAP $O(n^2)$), ovšem horní časovou hranici i tak tvoří proces konstrukce řešení – pouze procházíme lokace a přidělujeme



Obrázek 5.2: Vlevo závislost doby výpočtu na velikosti úlohy QAP. Vpravo závislost doby výpočtu na počtu umělých mravenců.

aktivity místo procházení měst a vybírání dalšího města. Pokud však integrujeme lokální prohledávání, které je aplikováno na každé nalezené řešení, vzroste složitost na kubickou ($O(n^3)$). Horní časovou hranici tak nyní tvoří právě lokální prohledávání.

Časové nároky jsou také závislé na počtu umělých mravenců. V následující analýze proto bylo na řešení instance `tai40a` použito 5, 10, 50, 100, 500 a 1000 agentů (pro verzi s lokálním prohledáváním mimo dvě nejvyšší hodnoty). Zbývající parametry byly nastaveny stejně jako při minulém experimentu. Zjištěné výsledky jsou zaneseny v grafu na obrázku 5.2 vpravo. Časová závislost je v obou případech lineární, lze však vidět nápadně vyšší náročnost pro variantu s lokálním prohledáváním. Doba běhu algoritmu roste vlivem přidaných výpočtů, které provádí každý mravenec v každé iteraci, s počtem mravenců výrazně strměji.

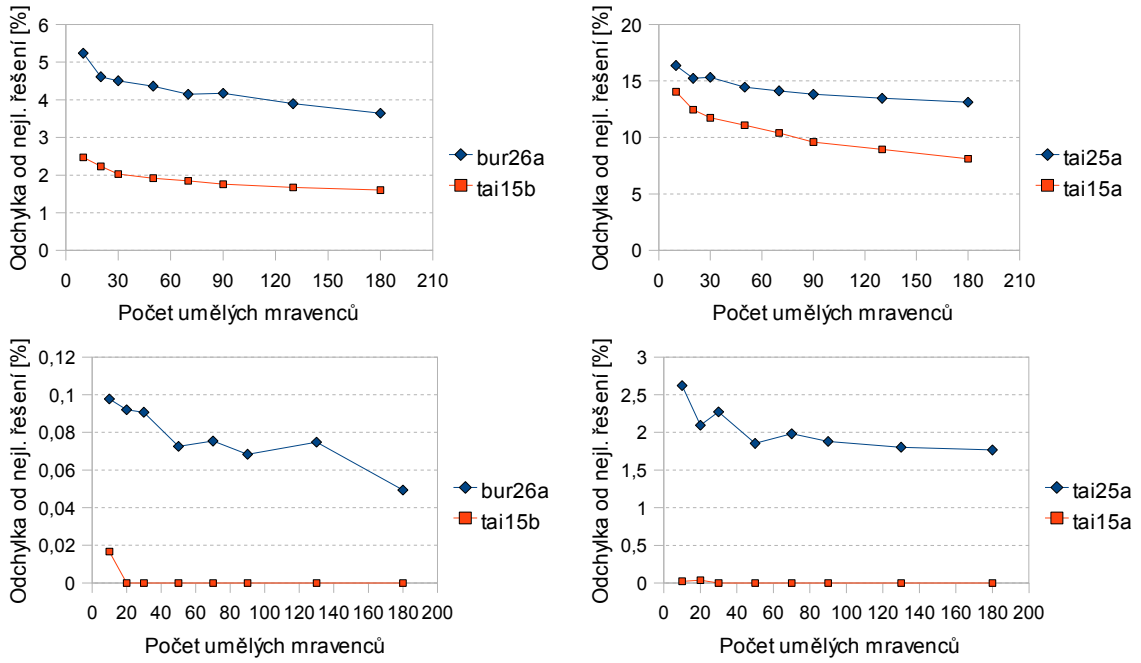
Změna řešené úlohy z TSP na QAP tedy nezměnila časovou složitost algoritmu. Jak však bylo ukázáno výše, obohacení algoritmu o lokální prohledávání značně zvyšuje jeho časové nároky. Veškerá naměřená data jsou uvedena v tabulkách na obrázku A.9.

5.1.6 Vliv počtu mravenců

Nyní budeme zkoumat, jak počet použitých umělých mravenců ovlivňuje kvalitu nalézaných řešení. Testy probíhaly na instancích náhodného charakteru `tai15a` a `tai25a`, reálném problému `bur26a` a instanci blížící se reálnému problému `tai15b`. Počet mravenců se pohyboval v rozmezí 10 až 180 a pokusy se opět zaměřily jak na klasický Ant System (kde bylo $t = 1000$ a 40 běhů na každý pokus), tak i na variantu s lokálním prohledáváním (vzhledem k časové náročnosti $t = 300$ a 15 běhů). Data, která byla naměřena na obou verzích, jsou přesně uvedena v tabulkách A.10 a A.11.

Nejdříve se zaměřme na variantu bez lokálního prohledávání. V grafech na obrázku 5.3 nahoře lze vidět jednoznačný nárůst průměrné kvality nalézaných řešení při zvyšování počtu umělých mravenců. Trend je zřetelný v celém rozsahu testovaných počtů mravenců, i když na instancích `tai15b` a `tai25a` můžeme pozorovat zmírňování závislosti při větším počtu mravenců. Podle velikosti odchylek lze také vypozorovat obecně vyšší náročnost instancí náhodného charakteru, přestože rozměrově odpovídají problémům z druhé skupiny.

Důsledkem většího počtu agentů je i pozdější stagnace procesu řešení. Podle naměřených hodnot průměrného λ-faktoru větvení vede zvyšováním počtu agentů k oddalování momentu, kdy začnou všichni agenti konstruovat stejná přiřazení (vlivem dominantních hladin feromonu pro dané přiřazení). Větší skupina jednotek tedy znamená pozdější dobu ustálení rozložení feromonu a tím více času na prozkoumávání stavového prostoru. Je to



Obrázek 5.3: Závislost kvality řešení na počtu umělých mravenců (nahoře bez lokálního prohledávání, dole s lokálním prohledáváním).

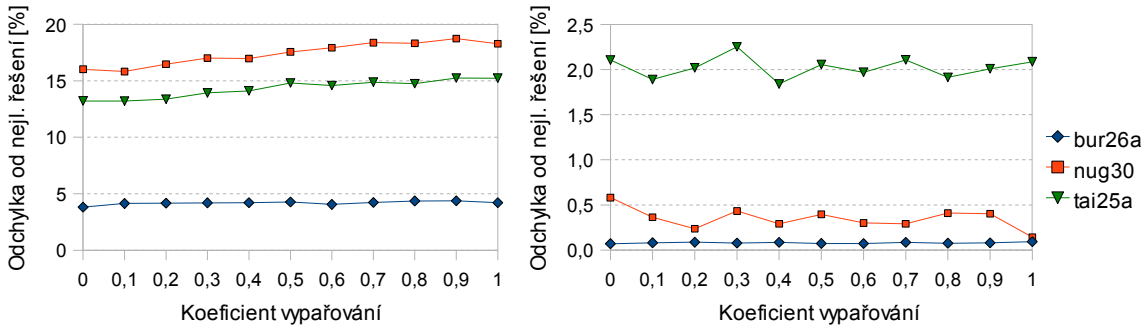
způsobeno zejména rozsáhlejším prohledáváním prostoru v prvních iteracích, kdy je zvýhodněno více hran, které jsou v dalších iteracích dále vyžívány a udržovány. Při daném nastavení parametrů se však přesto průměrná hodnota řešení mravenců brzy ustálí v okolí některého z nepříliš kvalitních lokálních optim.

Testy pro přidané lokální prohledávání opět potvrdily zdokonalování nalézaných řešení při růstu počtu agentů (viz grafy na obrázku 5.3 dole), průběhy však mají vlivem měření menších čísel více kolísavý charakter. U malých instancí (`tai15b` a `tai15a`) stačí i poměrně nízký počet mravenců pro dokonalou funkci algoritmu, tedy nalézání pouze optimálních řešení.

5.1.7 Vypařování feromonu

Předmětem dalších experimentů byl vliv koeficientu vypařování ρ, který byl nastavován na hodnoty od 0 do 1 s krokem $0,1$. Testy probíhaly na instancích `bur26a`, `tai30b` a `nug30`. Algoritmus bez lokálního prohledávání absolvoval vždy 40 běhů po 1000 iteracích, s lokálním prohledávání 15 běhů při 300 iteracích.

Výkon samotného algoritmu Ant System ilustruje graf na obrázku 5.4 vlevo a zdrojová data lze nalézt v tabulce A.12. Výsledky testů na všech třech instancích ukázaly, že k lepším nalézaným řešením vedou nízké hodnoty ρ. Odchylky od optima jsou nejnižší pro $\rho = 0$ až $\rho = 0,1$, což znamená žádné nebo velmi slabé vypařování. Čím vyšší je totiž vypařování feromonu, tím více jsou oslabována řešení, která se v daném čase jeví podle četnosti jejich využití agenty jako méně vhodná. Při vysoké míře vypařování tak v některé z počátečních iterací začně dominovat suboptimální cesta, která nakonec naprosto převáží všechny ostatní. Jde vlastně o stejný problém jako při malém počtu agentů, viz sekci 5.1.6. Při ma-



Obrázek 5.4: Závislost kvality řešení na koeficientu vypařování (vlevo bez lokálního prohledávání, vpravo s lokálním prohledáváním).

lém (nebo dokonce nulovém) vypařování algoritmus zachovává díky pomalému vyřazování nevýhodných cest diverzitu nalézaných řešení a umožňuje tak vyváznutí z lokálního optima. Tento závěr potvrzují standardní odchylky hodnot řešení v poslední iteraci (při silnějším vypařování jsou nulové) i hodnoty λ-faktoru větvení.

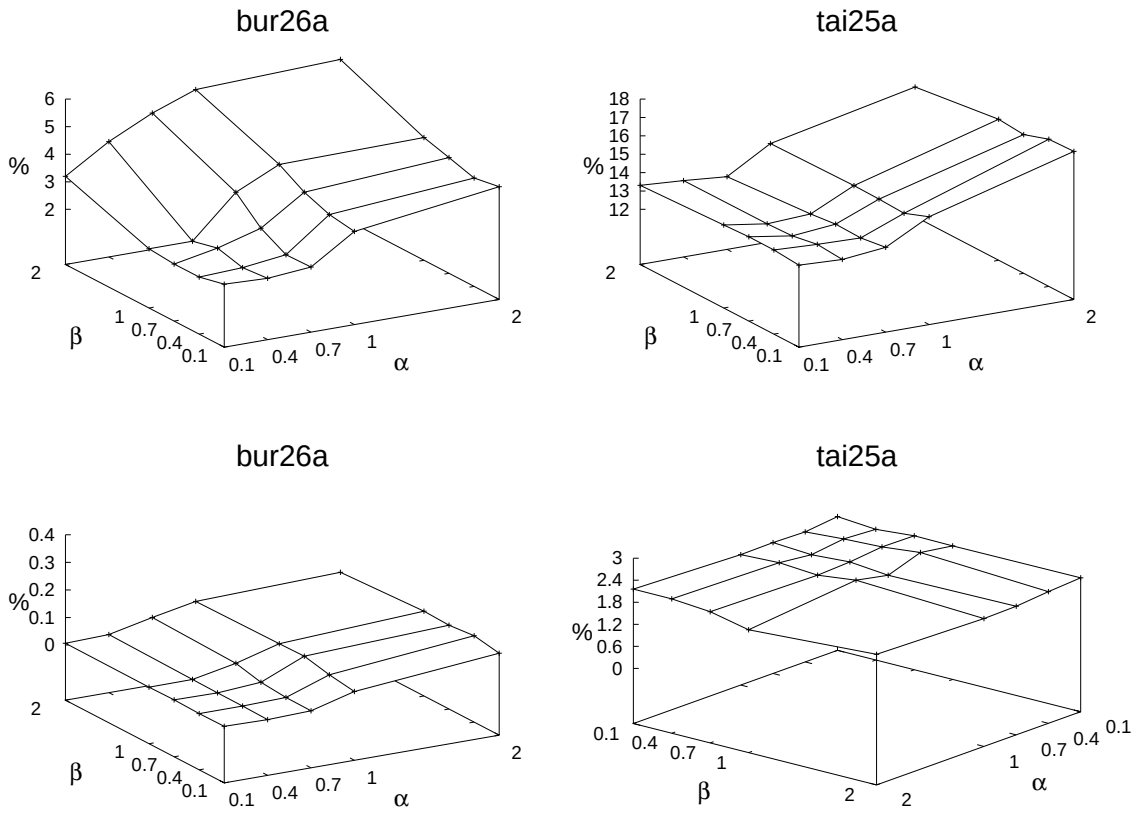
Graf na obrázku 5.4 vpravo ukazuje, že zmíněné vztahy již nejsou tak evidentní, pokud algoritmus vylepšuje nalezená řešení pomocí lokálního prohledávání. Odchylky se pohybují v okolí určité střední hodnoty a výkyvy je (bez existence přesnějších měření) možné přičíst vlivu náhodných odchylek. Mechanismus lokálního prohledávání představuje poměrně výrazný zásah do činnosti algoritmu a může tak, jak ukázaly testy, omezit vliv některých parametrů. Naměřená data jsou uvedena v tabulce A.13.

5.1.8 Váhy heuristiky a feromonu

V této části je analyzována role parametrů α a β, které reprezentují důležitost hodnot feromonu a heuristické informace pro rozhodování umělého mravence. Na instancích `bur26a` a `tai25a` byl testován vliv všech možných kombinací hodnot $\alpha = 0,1; 0,4; 0,7; 1; 2$ a $\beta = 0,1; 0,4; 0,7; 1; 2$. Každý běh absolvoval 1000 iterací a každý pokus byl pro následné zprůměrování 40krát zopakován, pro verzi s lokálním prohledáváním to bylo 300 iterací a 15 opakování.

První sada grafů (obrázek 5.5 nahoře) ukazuje výsledky Ant System bez lokálního prohledávání. Příslušná data jsou uvedena v tabulce A.14. Již na první pohled je vidět, že mají váhy jiný vliv při řešení reálných instancí (zde `bur26a`) a náhodných instancí (`tai25a`). Nejdříve se zaměřme na první uvedený typ. Pro algoritmus jsou nevýhodné vysoké hodnoty obou vah a při jejich snižování získáváme lepší výsledky. Váha feromonu α ovlivňuje průběh změn rozložení feromonu za běhu algoritmu, a to zejména rychlost ustálení několika dominantních cest. Čím vyšší je tento parametr, tím rychlejší je ustálení daných tras a jejich počet je také nižší. Naopak váha heuristiky β ovlivňuje téměř výhradně jen počet dominantních feromonových tras a charakter průběhu změn feromonu je při stejné hodnotě α zachován. Při zvyšování β je využíváno méně cest a oba parametry mají v důsledku podobný účinek – snižování míry prohledávání prostoru řešení. Podle výsledků testů je nejlepší kombinací vah pro tuto reálnou instanci $\alpha = 0,1$ a $\beta = 0,4$.

U náhodné instance `tai25a` můžeme pozorovat podobný vliv váhy feromonu α, ale jeho optimální hodnota se (uvažujeme-li konstantní β) pohybuje spíše v rozmezí 0,4 až 0,7. Slabší váha má již na výkon algoritmu negativní vliv. Rozložení feromonu je tímto parametrem



Obrázek 5.5: Závislost kvality řešení na vahách pro viditelnost a hladinu feromonu (nahoře bez lokálního prohledávání, dole s lokálním prohledáváním).

ovlivněno naprosto stejně jako v předchozím případě. Zajímavý rozdíl však najdeme u váhy heuristiky β. Pro konstantní α je téměř vždy nejvýhodnější hodnotou 2, čili volba nejvíce posílené role heuristiky. Vliv na rozložení feromonu je pak, opět v kontrastu s předchozím případem, neznatelný. Pro náhodné instance tedy vysoká váha heuristiky zřejmě směřuje algoritmus k výhodným řešením a nesnižuje přitom míru prohledávání stavového prostoru. Jako vůbec nejlepší kombinace vah se podle testů ukázala být volba $\alpha = 0,7$ a $\beta = 2$.

Dodatečná kontrolní měření při slabším vypařování $\rho = 0,1$ potvrdila zjištěné poznatky, pouze se u obou typů instancí ukázaly být vyhodnější mírně vyšší váhy feromonu (tedy nejbližší vyšší měřené hodnoty).

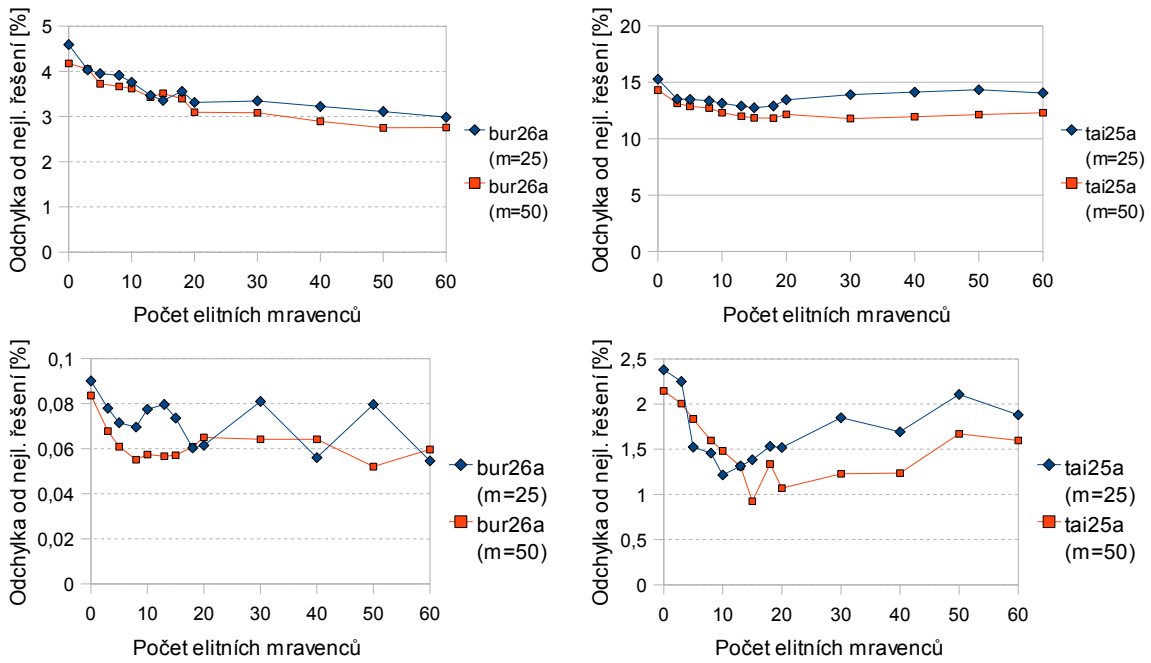
Výsledky testů na verzi s lokálním prohledáváním obsahuje tabulka A.15 a grafy na obrázku 5.5 dole. Zaměříme-li se na reálnou instanci `bur26a`, projevily se v experimentech stejné vztahy jako bez použití lokálního prohledávání. Snižování obou vah zlepšuje výkon algoritmu a vede k lepším nalézaným řešením. Nejvýhodnějších kombinací vah je nyní více a vedou k nalezení optimálního řešení v každém běhu algoritmu. Můžeme použít například váhy, které se osvědčily i pro variantu algoritmu bez lokálního prohledávání, a to $\alpha = 0,1$ a $\beta = 0,4$. Zvyšování hodnoty parametru α má opět za následek rychlejší ustalování feromonových cest, jejichž počet se také snižuje. β se ale nepodílí na tak výrazném snížení počtu cest v absolutní míře jako v předchozím případě. Snížení je velmi malé a při největší hodnotě $\alpha = 2$ dochází dokonce ke zvýšení.

Data, která se vztahují k náhodné instanci `tai25a`, ukazují poměrně vyrovnané vý-

sledky algoritmu pro různá nastavení vah. Z hlediska rozložení feromonu má výrazný vliv až nastavení $\alpha = 2$, kde dochází v počátečních iteracích k rychlému ustálení dominantní cesty. Při ostatních hodnotách této váhy je po celou dobu běhu algoritmu udržován vysoký a prakticky neměnný počet využívaných cest. Parametr β neměl na feromonovou strukturu viditelný vliv. Podle výsledků testů je nejlepší kombinací vah $\alpha = 2$ a $\beta = 1$.

5.1.9 Role elitních mravenců

Další experimenty na algoritmu Ant System testovaly vliv elitních mravenců. Jejich množství e bylo nastavováno od 0 do 60 a naměřené hodnoty byly zapsány do tabulky A.19 (verze bez lokálního prohledávání) a A.20 (verze s lokálním prohledáváním). Bylo použito 25 a 50 umělých mravenců, v první verzi algoritmus běžel vždy 1000 iterací a každý běh byl opakován 40krát, pro druhou bylo nastaveno 300 iterací a 15 opakování.



Obrázek 5.6: Závislost kvality řešení na počtu elitních mravenců (nahoře s lokálním prohledáváním, dole bez lokálního prohledávání).

Měření varianty bez lokálního prohledávání reprezentují grafy na obrázku 5.6 nahoře. V rámci reálné instance `bur26a` vedlo v celém testovaném rozsahu zvyšování počtu elitních mravenců ke zlepšování kvality nalézaných řešení. Elitní mravenci se připojují k ostatním pouze při aktualizaci feromonu a nepodílejí se tedy na konstrukcích řešení. Zvyšování jejich počtu proto navíc nepředstavuje významnou časovou zátěž. Tyto speciální jednotky také ovlivňují rozložení feromonu. Veškeré jejich neprázdné skupiny měly za následek udržování těsně více než jednoho dominantního řešení a přinášely tak potřebnou diverzitu řešení i v pozdějších iteracích.

Na náhodné instanci `tai25a` bylo zaznamenáno odlišné chování. Zde již existovala hranice, od které zvyšování počtu elitních mravenců vedlo k horším výsledkům. Hranici není možné jednoznačně stanovit, ale výhodnou hodnotou parametru e by mohla být přibližně

polovina počtu běžných mravenců m. Elitní mravenci v tomto případě nezabránili rychlému ustálení jednoho dominantního přiřazení ve feromonových stopách a příliš velký počet elitních mravenců vedl k příliš rychlé konvergenci k jednomu řešení, které konstruovaly všechny jednotky.

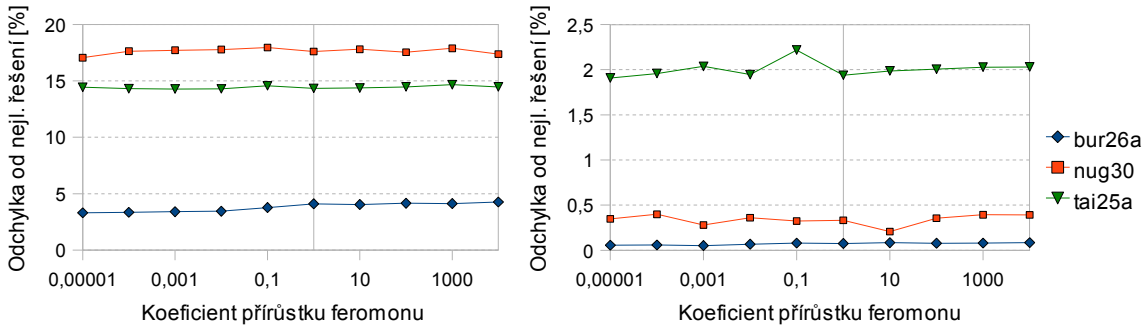
Jak lze vidět v grafech na obrázku 5.6 dole, odchylky od optima byly při použití lokálního prohledávání již poměrně nízké a značně se zde projevují náhodné odchylky. Z grafu pro instanci `bur26a` lze za využití aproximace lineární regresí opět vidět zlepšování výsledků algoritmu při zvyšování počtu elitních mravenců, ale jisté jsou pouze špatné výsledky pro nízké hodnoty e. Běh algoritmu sice nekončí na jedné dominantní cestě a alternativních cest je obecně více než u verze bez lokálního prohledávání, ale rozšiřování skupiny těchto mravenců i tak způsobuje postupné omezování množství relevantních feromonových cest.

Měření na náhodné instanci `tai25a` rovněž potvrzuje výsledky testů na předchozí verzi, a to existenci určité hranice, po kterou je zvyšování počtu elitních mravenců výhodné. Tato hranice se však oproti minulé verzi mírně posunula směrem k menším e. K stanovení přesnějších závěrů by bylo nutné provést další měření (např. s více běhy algoritmu v jednom pokusu), která však vyžadují poměrně značné časové nároky. Pokud jde o rozložení feromonu, zvyšování počtu elitních mravenců vede postupně až k ustalování na jedné dominantní feromonové trase.

5.1.10 Další parametry

Koeficient přírůstku feromonu

V této části se budeme zabývat působením různých hodnot koeficientu Q na činnost algoritmu. Parametr byl nastavován na hodnoty 10^{-5} až 10^{4}.



Obrázek 5.7: Závislost kvality řešení na koeficientu přírůstku feromonu (vlevo bez lokálního prohledávání, vpravo s lokálním prohledáváním).

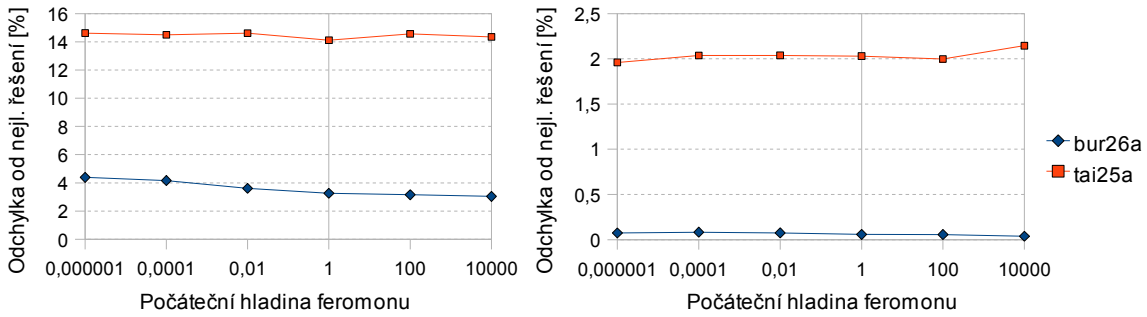
Algoritmus bez lokálního prohledávání absolvoval vždy 1000 iterací a každý test je průměrem ze 40 běhů. Jak ukazuje výsledný graf na obrázku 5.7 vlevo (příslušná data je možné nalézt v tabulce A.16), u náhodných instancí `tai25a` a `nug30` kvalita řešení sice kolísá, ale pravděpodobně především vlivem náhodných odchylek. Při malých hodnotách parametru lze zaznamenat větší počet cest, které mravenci prohledávají v několika počátečních iteracích, a také trvá mírně kratší dobu proces ustalování na jednom řešení. Vzhledem ke kvalitě nalézaných řešení by také bylo možné doporučit nízké hodnoty parametru, ale vliv zřejmě není nijak významný. Naproti tomu experimenty s instancí `bur26a` jednoznačně prokázaly, že čím nižší je hodnota Q, tím lepší řešení získáme. Nízké hodnoty opět vedou k většímu

počtu využívaných feromonových cest v počátečních iteracích, což má pravděpodobně významný dopad na kvalitu nalézaných řešení.

Měření probíhala i na algoritmu s lokálním prohledáváním a zjištěná data ilustruje graf na obrázku 5.7 vpravo, který čerpá z dat uvedených v tabulce A.17. Testy potvrdily výše uvedené poznatky získané na algoritmu bez lokálního prohledávání. Jednoznačnou závislost můžeme pozorovat pouze v rámci experimentů na reálné instanci `bur26a`, kde jsou opět řešení degradována se zvyšováním hodnoty parametru Q (závislost je však výrazně slabší). Jeho nízké hodnoty také vedou k více relevantním feromonovým cestám v počátečních iteracích. Oproti předchozí verzi algoritmu zde však nedochází k postupnému ustálení na jedné dominantní cestě – až do konce běhu algoritmu je tedy zachována diverzita nalézaných řešení.

Počáteční hladina feromonu

Další experimenty se zabývaly parametrem τ_0. Hodnota parametru byla postupně 10^{-6} až 10^4 s krokem exponentu 2. Pro variantu bez lokálního prohledávání platilo $t = 1000$ a 40 opakování každého běhu, s lokálním prohledáváním $t = 300$ a 15 opakování. Naměřená data lze nalézt v tabulce A.18.



Obrázek 5.8: Závislost kvality řešení na počáteční hladině feromonu (vlevo bez lokálního prohledávání, vpravo s lokálním prohledáváním).

Výsledky experimentů pro Ant System bez lokálního prohledávání jsou uvedené na obrázku 5.8 vlevo. Experimentování s reálnou instancí `bur26a` ukázalo, že zvyšování počáteční hladiny feromonu vede k lepším řešením. Vůbec nejvýhodnější byla nejvyšší testovaná hodnota parametru 10^4. Zajímavé je, že má parametr velmi podobný vliv na feromonové stopy jako koeficient přírůstku z předchozího experimentu. K většímu počtu silných cest v počátečních iteracích však vedou tentokrát vyšší hodnoty parametru τ_0. Vypařování počáteční hladiny totiž trvá delší dobu a pro umělé mravence jsou tak déle relevantní i hrany, které budou později z hlediska feromonu potlačeny. U náhodné instance `tai25a` byl nalezen stejný vliv na feromonové stopy, který se však již zřetelně neprojevil na kvalitě nalézaných řešení.

Chování verze s lokálním prohledáváním je ukázáno na obrázku 5.8 vpravo. Rozdíly v kvalitě řešení jsou zde pro různé hodnoty parametru mnohem menší. Přesto však u reálné instance `bur26a` nalezneme obdobný vztah jako v předchozím případě a nejvýhodnější tak byla nejnižší testovaná hodnota τ_0. Vliv na feromonové struktury je v počátečních iteracích stejný jako v předchozí verzi, na další již změny parametru nemají účinek. U náhodné instance `tai25a` je kvalita nalézaných řešení poměrně vyrovnaná, výjimkou jsou pouze výsledky pro obě krajní hodnoty parametru. Podle nich můžeme odhadnout, že by pro

náhodné instance mohlo být nejvýhodnějším nastavením $\tau_0 = 10^6$. Vzhledem ke skokovým krajním hodnotám by však pro potvrzení bylo nutné provést další testy. Vliv na průběh změn feromonových struktur byl minimální.

5.2 Ant Colony System

5.2.1 Popis algoritmu

Následující popis algoritmu Ant Colony System (ACS) vychází z [2], kde je použit na řešení Travelling Salesman Problem. ACS vychází z původního algoritmu Ant System (viz pseudokód 1), ale přináší zároveň podstatné změny. Mezi nejdůležitější patří lokální změny feromonu prováděné během konstrukce řešení agentem. Hladiny feromonu jsou pro zvolené kroky snižovány a ACS se takto snaží udržovat přiměřenou diverzitu množiny řešení. Globální změny feromonu také provádí pouze nejlepší agent a upravené je i rozhodování o dalších krocích mravence.

Průchod mravenců grafem

Umělý mravenec opět cestuje grafem od lokace s nejmenším vzdálenostním potenciálem postupně k lokacím s vyššími potenciály (je však vhodné připomenout, že toto není jediný možný způsob výběru lokací). Zatímco v Ant System agent v každé lokaci vždy vybíral vhodnou volnou aktivitu na základě pravděpodobnostní funkce, v ACS jsou mu nabízeny dvě alternativy. Rozhoduje se mezi nimi na základě pravděpodobnosti q_0 (parametr navíc oproti metodě Ant System) podle náhodné hodnoty q v rozsahu 0–1:

Prozkoumávání (exploration), $q > q_0$ Případ prozkoumávání odpovídá principu volby aktivity pro danou lokaci v Ant System, tedy na základě pravděpodobnostního rozložení jednotlivých volných aktivit. Rozložení je proto dáno vztahem 4.1 v nezměněné formě. Výhodou pravděpodobnostního výběru je možnost volby z lokálního pohledu horšího kroku, který však může přispět k nalezení globálně lepšího řešení.

Využívání (exploitation), $q \leq q_0$ Pokud je náhodně vybrána tato možnost, využívá se v maximální možné míře znalostí jak heuristických, tak naučených (hladiny feromonu). Je proto vybrána ta aktivita j, která je podle všech aktuálních měřítek pro danou lokaci i nejvhodnější. Vzorec pro výběr aktivity j tedy vypadá následovně:

$$j = \underset{u \in J_i^k}{\operatorname{argmax}} \left((\tau_{iu}(t))(\eta_{iu})^\beta \right), \tag{5.5}$$

kde J_i^k je množina zatím nepřiřazených aktivit pro lokaci i v kroku k a τ a η mají stejný význam jako v algoritmu Ant System. Jak je možné vidět, vyvážení důležitosti heuristiky a množství feromonu se nyní děje pouze pomocí jednoho parametru – β.

S nastavením hodnoty parametru q_0 je možné experimentovat a nebo dokonce použít stejný postup jako u simulovaného žíhání, což je jeden z námětů pro vylepšení Ant Colony System uvedený v [2]. Myšlenka je následující: ze začátku nastavit malé q_0 pro podporu rozsáhlého průzkumu stavového prostoru a poté q_0 postupně snižovat a umožňovat tak jakési „doladění“ řešení.

Aktualizace stavu feromonu

Významných změn doznal oproti Ant System i způsob úprav rozložení feromonu. Jiný přístup představují dva hlavní rozdíly: hladina feromonu je lokálně měněna již během průchodu agenta grafem úlohy a globální změna, na kterou přichází řada po dokončení cest všech umělých mravenců, je prováděna pouze podle nejlepšího řešení v rámci celého běhu algoritmu.

Lokální změny feromonu, které agent provádí při postupné konstrukci řešení, mají především ovlivnit další agenty při tvorbě jejich vlastních řešení. Tento účel sice splňuje i globální aktualizace feromonu, která ale působí na chování agentů pomaleji po jednotlivých iteracích a je určena spíše k posilování kvalitních řešení. Lokální úpravy naproti tomu mají ovlivnit hned následující jednotky v rámci stejné iterace. A to zejména tak, aby další agenti konstruovali odlišná řešení.

Popisovanou lokální změnu feromonu provede agent vždy, když přiřadí vybranou aktivitu j aktuální lokaci i. Příslušná hladina feromonu τ_{ij} je změněna podle následujícího vztahu:

$$\tau_{ij}(t) = (1 - \rho)\tau_{ij}(t) + \rho\tau_0, \tag{5.6}$$

kde ρ je opět koeficient vypařování a τ_0 je zvolená počáteční hladina feromonu. Dané přiřazení je tedy oslabeno, čímž metoda nabádá ostatní agenty k volbám jiných přiřazení. Podle [12] je tímto způsobem hladina feromonu všech přiřazení zespoda omezena na hodnotu τ_0 a přiřazení, která se nikdy za běhu algoritmu nestala součástí některého z nejlepších řešení, jsou stále spojena s touto minimální hodnotou.

Po průchodu všech agentů grafem přichází na řadu *globální změny feromonu*. Uvažujeme nyní pouze nejlepší řešení celého běhu algoritmu π^+ a feromon měníme jen u všech jeho přiřazení aktivit j lokacím i podle vzorce:

$$\tau_{ij}(t+1) = (1 - \rho)\tau_{ij}(t) + \rho\frac{1}{C(\pi^+)}, \tag{5.7}$$

kde $C(\pi^+)$ je hodnota účelové funkce pro uvedené nejlepší přiřazení π^+. Globální změna feromonu je tedy zaměřena na posilování kvalitního řešení a nabádá mravence k prohledávání jeho okolí. Feromonové hladiny ostatních možných přiřazení jsou těmito globálními změnami nedotknuté a efekt vypařování feromonu je uplatněn pouze v omezené míře u lokálních změn.

Využití struktury candidate list

V [2] i [12] se doporučuje využití struktury *candidate list* pro větší TSP instance. Je zde uloženo několik nejbližších měst, ze kterých agent primárně vybírá. Pouze pokud jsou již využity všechny položky seznamu, jsou brána v úvahu i další města. U Quadratic Assignment Problem za použití stávající heuristiky pro odhad vhodnosti přiřazení (viz 5.1.1) je možnost využití candidate listu poněkud omezená. Vzhledem k metodě řešení problému a způsobu průchodu grafem postupně od lokalit s nejmenším vzdálenostním potenciálem je totiž množina několika nejvýhodnějších aktivit pro všechny lokace stejná.

5.2.2 Úvod experimentů

V rámci experimentů s metodou Ant Colony System byly použity následující *výchozí parametry*: $\tau_0 = 10^{-5}$, $\rho = 0,1$, $\beta = 2$ a $q_0 = 0,9$. Základní verze algoritmu byla nastavena na

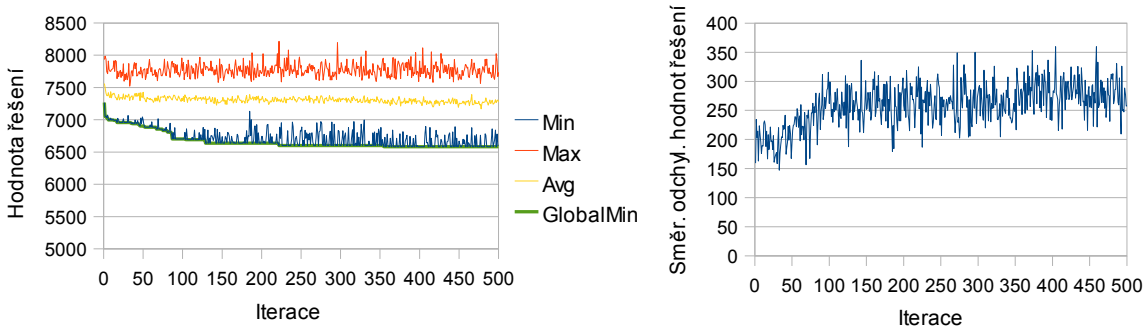
$t = 1000$, $m = 100$ a 40 opakování každého pokusu, zatímco verze s lokálním prohledáváním na $t = 500$, $m = 50$ a 20 opakování. Vliv struktury candidate list nebyl v rámci experimentů sledován a jeho velikost cl byla vždy nastavena na maximum (což odpovídá vyřazení této struktury).

Experimenty se již (oproti testům na Ant System) nezabývaly časovou složitostí algoritmu a zaměřily se spíše na jednotlivé nastavitelné parametry.

5.2.3 Charakter chování algoritmu

Pro ukázku typického průběhu řešení úlohy `nug30` pomocí Ant Colony System byla použita výchozí sada parametrů. Vývoj hodnot řešení a příslušné směrodatné odchylky skupiny řešení je ukázán na obrázku 5.9.

Hlavním rozdílem oproti Ant System je zachování diverzity nalézaných řešení po celou dobu běhu algoritmu. V Ant System toho sice také můžeme dosáhnout vhodnou volbou parametrů, ovšem Ant Colony System potřebné rozptýlení řešení ve stavovém prostoru již přímo podporuje pomocí lokálních úprav rozložení feromonu. Lze také vidět, že se minimální hodnoty řešení pohybují v oblasti globálně nejlepšího nalezeného řešení. Průměrná hodnota řešení v každé iteraci pozvolna klesá a kvalita řešení, která jsou nalézána skupinou agentů, se tak stále zlepšuje.

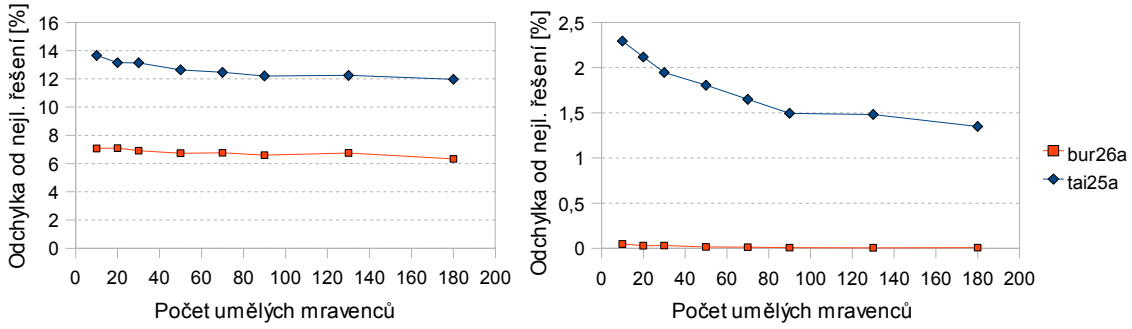


Obrázek 5.9: Vlevo typický průběh algoritmu Ant System při řešení QAP úlohy `nug30`. Vpravo odpovídající průběh směrodatných odchylek.

5.2.4 Vliv počtu mravenců

První testy měly za úkol odhalit, zda zvyšování počtu umělých mravenců (parametr m), které s sebou samozřejmě nese zpomalení chodu algoritmu, přináší zlepšení kvality nalézaných řešení. Testovaný rozsah počtu jednotek byl 10 až 180. Výsledky testů pro obě verze jsou uvedeny v tabulkách A.21 a A.22. Pro názornost jsou také zaneseny v grafech na obrázku 5.10.

Verze bez lokálního prohledávání přinesla na větších instancích `bur26a` a `tai25a` očekávaný účinek zvětšování počtu umělých mravenců, tedy snižování průměrné odchylky hodnot řešení od optimálního řešení. Přitom nebyl ovlivněn vývoj rozložení feromonu, které jednotkám ve všech případech po celou dobu běhu algoritmu umožňovalo využívání drtivé většiny možných přiřazení. Jiná situace nastala u menších instancí (nejsou v grafech uvedeny). Pro pseudoreálnou instanci `tai15b` přineslo nejlepší výsledky použití 70 agentů a náhodná instance `tai15a` dokonce preferovala nejnižší testovanou hodnotu $m = 10$. Větší počet



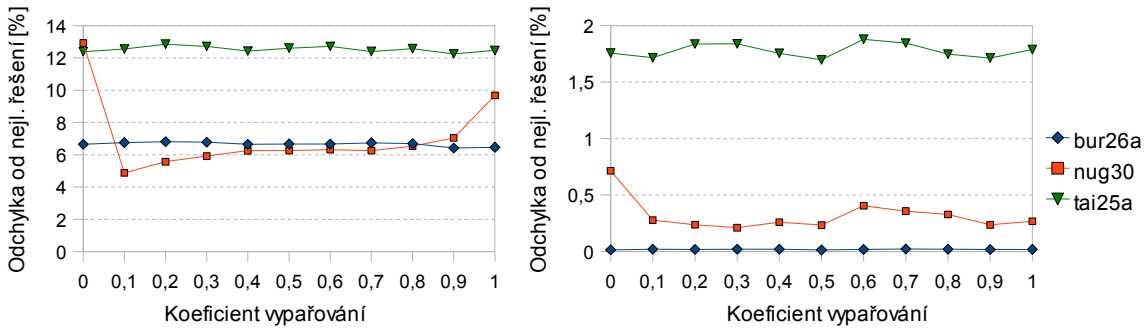
Obrázek 5.10: Závislost kvality řešení na počtu umělých mravenců (vlevo bez lokálního prohledávání, vpravo s lokálním prohledáváním).

umělých mravenců zde přitom neměl na sledované ukazatele výraznější vliv (neuvažujeme-li samotné hodnoty řešení).

Přidané lokální prohledávání vedlo k dokonalé práci algoritmu na menších instancích obou typů a ve všech bězích byla nalézána optimální řešení. Pro větší instance `bur26a` a `tai25a` mělo zvyšování počtu umělých mravenců pozitivní dopad na kvalitu nalézaných řešení, i když u první jmenované (reálné) instance již tento trend při velkých hodnotách m spíše stagnoval. Rozložení feromonu u těchto instancí opět umělým mravencům umožňovalo prozkoumávat stavový prostor téměř v plném rozsahu. Další ukazatele nebyly změnami m významně ovlivněny.

5.2.5 Vypařování feromonu

Experimenty, které budou nyní představeny, se zaměřily na parametr ρ. Jeho hodnoty byly nastavovány na 0 až 1 s krokem 0,1.



Obrázek 5.11: Závislost kvality řešení na koeficientu vypařování (vlevo bez lokálního prohledávání, vpravo s lokálním prohledáváním).

Data, která byla naměřena na základní verzi algoritmu, jsou uvedena v tabulce A.23 a zanesena do grafu na obrázku 5.11 vlevo. V případě instancí `bur26a` a `tai25a` se změny koeficientu vypařování ve výsledcích algoritmu významněji neprojevily a neumožnily tak odhadnout souvislost míry vypařování s kvalitou nalézaných řešení. O to zajímavější data však byla zjištěna pomocí experimentů na `nug30`, což je náhodná instance, ale se vzdále-

nostmi na mřížce. Obě krajní testované hodnoty parametru ρ se projevily jako výrazně nevýhodné. Jinak platilo, že silnější vypařování (vyšší hodnota parametru) vedlo k průměrně horším nalézaným řešením. Při volbě $\rho = 0$ nedochází k žádnému vypařování a rozložení feromonu je neustále shodné s rozložením po inicializaci. Jednotky se tak mohou spolehnout pouze na heuristickou informaci. Z hlediska feromonových stop se instance `nug30` lišila také tím, že bylo preferováno jen okolí jednoho dominantního přiřazení. U ostatních testovaných instancí byl virtuální feromon rozprostřen rovnoměrně mezi všechna přiřazení.

Výsledky měření na verzi s lokálním prohledáváním jsou uvedeny v tabulce A.24 a ilustrovány grafem na obrázku 5.11 vpravo. Jednoznačně se potvrdila pouze nevýhodnost deaktivace vypařování ($\rho = 0$) u instance `nug30`. Zajímavé je, že u `nug30` i `tai25a` byla shodně nejhorší řešení (nevažujeme-li předchozí případ) nalezena při $\rho = 0,6$, v blízkosti téměř nejlepších řešení při $\rho = 0,5$. Potvrzení tohoto jevu by ale vyžadovalo dodatečné přesnější testy.

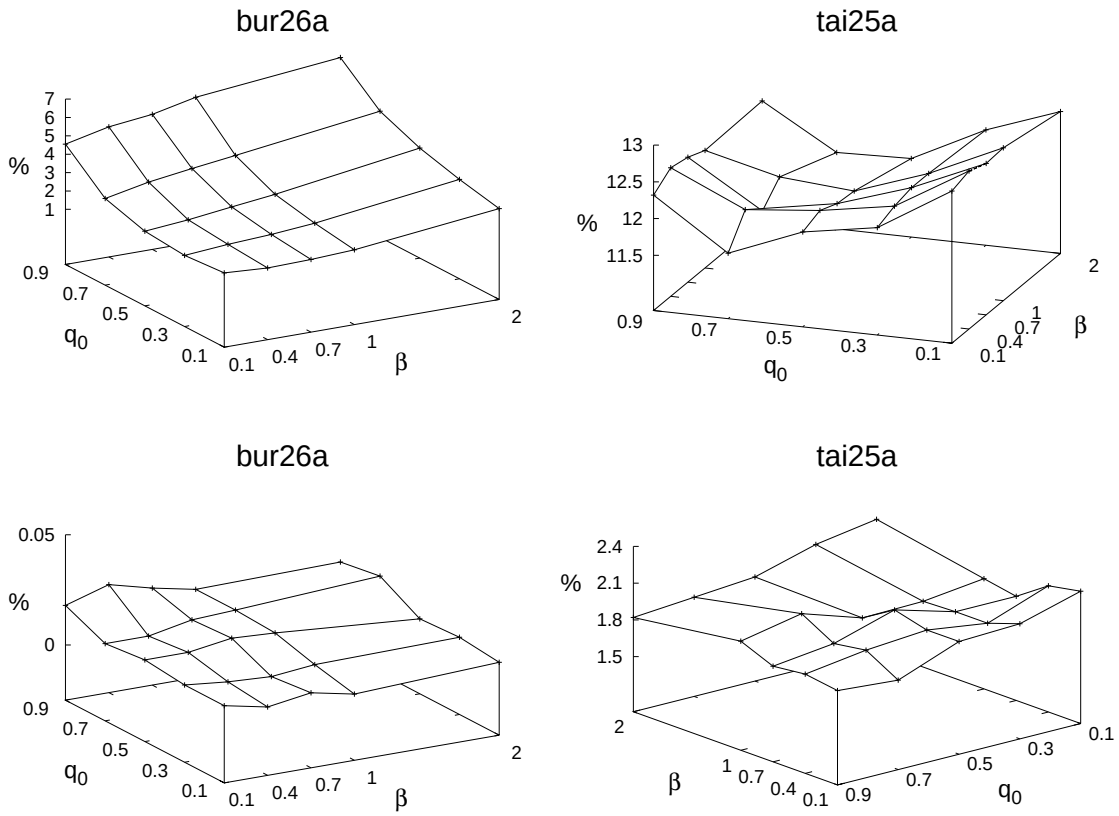
5.2.6 Váha heuristiky a vyvážení prohledávání a využívání

Další experimenty se zabývaly zároveň nastavením váhy heuristické informace β a parametru q_0. Tento koeficient určuje, zda budou agenti při stanovování dílčích přiřazení vybírat nejlepší možnost nebo budou spíše volit na základě pravděpodobností. Váha heuristické informace byla nastavována na hodnoty $\beta = 0,1; 0,4; 0,7; 1; 2$, druhý testovaný parametr potom na $q_0 = 0,1; 0,3; 0,5; 0,7; 0,9$. U verze s lokálním prohledáváním byl oproti výchozí konfiguraci počet iterací $t = 400$. Výsledná data jsou uvedena v tabulkách A.25 a A.26.

Verze bez lokálního prohledávání opět vykazovala jiné chování na reálné a náhodné instanci, viz obrázek 5.12 nahoře. Výsledky algoritmu na reálné instanci `bur26a` jednoznačně ukazují zlepšování kvality řešení při snižování hodnot obou testovaných parametrů. Preferováno je tedy spíše pravděpodobnostní prozkoumávání různých voleb a snížení rozdílů mezi heuristickou výhodností jednotlivých voleb. Při všech nastaveních bylo přitom rozložení feromonu poměrně rovnoměrné a umožňovalo tak rovnocenně konstruovat všechna možná přiřazení. Zvyšování váhy heuristiky dále vedlo k nacházení nejlepších řešení běhu v dřívějších iteracích. V průměru nejlepší řešení přineslo nastavení $\beta = 0,1$ a $q_0 = 0,3$.

U náhodné instance `tai25a` bylo výhodnější nastavení vyšší míry prohledávání, a to $q_0 = 0,5$ až $q_0 = 0,7$. Heuristickou informaci bylo opět vhodné potlačit a pro většinu nastavení q_0 algoritmus předvedl nejlepší výsledky při nejnižší váze heuristiky $\beta = 0,1$. Jak však lze vidět v příslušném grafu, závislost na váze heuristiky je zde nyní mnohem slabší než u reálné instance a více se projevují náhodné odchylky. Další sledované ukazatele nebyly různými nastaveními obou parametrů významně ovlivněny. Feromon je opět rovnoměrně rozložen mezi jednotlivá dílčí přiřazení.

Grafy na obrázku 5.12 dole jsou výsledkem zpracování měření na algoritmu s lokálním prohledáváním. Řešení reálné instance `bur26a` se již v průměru velmi blíží optimálním řešením. K dobré práci algoritmu vedlo nastavení nízkých hodnot q_0, vliv β se ale jednoznačně neprojevil. Nejlepší řešení bylo nalezeno pro $q_0 = 0,1$ a $\beta = 1$. V rámci náhodné instance `tai25a` byly výsledky relativně ještě vyrovnanější. Mírně lepší řešení algoritmus nalézal při vyšších hodnotách parametru q_0. Význam β opět nebylo možné jednoznačně stanovit. Nejmenší průměrnou odchylku od optima tentokrát přinesla kombinace $q_0 = 0,7$ a $\beta = 0,1$, ovšem jak již bylo řečeno výše, výsledky jsou obecně při lokálním prohledávání poměrně vyrovnané.



Obrázek 5.12: Závislost kvality řešení na váze heuristické informace a vyvážení prozkoumávání a využívání (nahoře bez lokálního prohledávání, dole s lokálním prohledáváním).

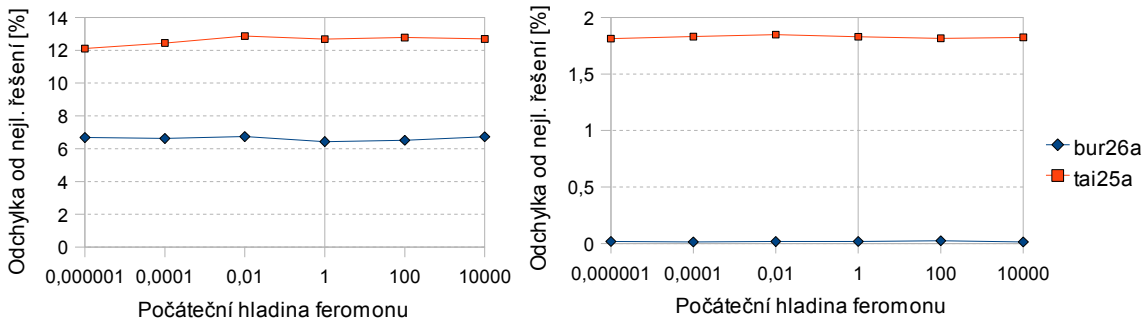
5.2.7 Další parametry

Počáteční hladina feromonu

Poslední experimenty na algoritmu Ant Colony System měly za úkol zjistit, jaký vliv má počáteční hodnota feromonu τ_0 na fungování a výsledky algoritmu. Rozsah nastavovaných hodnot byl 10^{-6} až 10^4 s krokem exponentu 2. Výsledné odchylky od optima jsou uvedeny v tabulce A.27.

Graf 5.13 vlevo ukazuje vývoj odchylek od optima pro Ant Colony System bez lokálního prohledávání. Testy na reálné instanci `bur26a` dopadly nejlépe při nastavení $\tau_0 = 1$. Při zkoumání rozdílů v chování algoritmu oproti nejnižší hodnotě $\tau_0 = 10^{-6}$ bylo nalezeno pouze velmi mírné postupné omezování relevantních přiřazení z hlediska feromonu během práce algoritmu. Pro zmíněnou nejnižší hodnotu bylo zachováváno feromonové rozložení po celou dobu běhu tak, že agenti významně nepreferovali některé z dílčích přiřazení. U náhodné instance `tai25a` se nejlépe osvědčila právě spodní testovaná hranice $\tau_0 = 10^{-6}$. Vliv parametru na feromonové stopy byl podobný jako v minulém případě.

Verze s lokálním prohledáváním, jejíž výkony jsou prezentovány grafem na obrázku 5.13 vpravo, opět nepřinesla jednoznačnou informaci o vlivu parametru. Přestože se ale v rámci instance `tai25a` kvalita řešení pro různé hodnoty τ_0 liší naprosto minimálně, opět přineslo nejlepší výsledky nastavení $\tau_0 = 10^{-6}$. U `bur26a` to byla naopak nejvyšší testovaná hranice $\tau_0 = 10^4$. Obecně by bylo vhodné rozložit celý testovaný rozsah hodnot τ_0 na menší úseky a



Obrázek 5.13: Závislost kvality řešení na počáteční hodnotě feromonu (bez lokálního prohledávání).

testovat vliv parametru v jejich rámci. Na základě poměrně velkého rozsahu vyzkoušených hodnot τ_0 lze však odhadnout, že tento parametr nemá velký vliv na práci algoritmu. To potvrzuje i relativní vyrovnanost ostatních sledovaných údajů.

5.3 Max–Min Ant System

5.3.1 Popis algoritmu

Max–Min Ant System (MMAS) je dalším z řady algoritmů, které rozšiřují původní algoritmus Ant System a snaží se různými úpravami dosáhnout lepších výsledků při řešení úloh. V následujících podsekcích proto budou popsány pouze odlišnosti od Ant System. Popis čerpá z obecného výkladu algoritmu v [20] a [21], ale také z [18], kde je představeno použití MMAS na řešení QAP.

Průchod mravenců grafem

Postupná konstrukce řešení daného umělého mravence je podobná jako u algoritmu Ant Colony System. V každém kroku je aplikována strategie *prozkoumávání* nebo *využívání* v závislosti na výsledku porovnání parametru q_0 s náhodným číslem q (z rozsahu 0–1). Ve shodě s [18] budeme při stanovování dílčího přiřazení aktivity j lokaci i uvažovat pouze informaci o příslušné hodnotě feromonu τ_{ij}. Lokace je dána a aktivitu vybíráme podle aktuálně zvolené strategie (samozřejmě s omezením na zatím neuplatněné aktivity, čili $j \in J_i^k$).

Prozkoumávání, uplatněné při $q > q_0$, staví na náhodném výběru aktivity pro danou lokaci i podle pravděpodobností $p_{ij}^k(t)$ volných aktivit. Pro mravence k a iteraci t počítáme pravděpodobnost přiřazení aktivity j takto:

$$p_{ij}^k(t) = \frac{\tau_{ij}(t)}{\sum_{l \in J_i^k} \tau_{il}(t)}. \tag{5.8}$$

Uvedený vztah platí pro dosud nepoužitou aktivitu j; pokud naopak $j \notin J_i^k$, potom $p_{ij}^k(t) = 0$. Strategie prozkoumávání opět vede k dobrému využití stavového prostoru, protože lze díky ní zvolit z lokálního pohledu méně výhodné dílčí přiřazení.

Strategie *využívání* je zvolena pokud $q \leq q_0$. Potom hledáme z hlediska feromonu nejvýhodnější aktivitu j pro danou lokaci i v iteraci t následovně:

$$j = \underset{u \in J_i^k}{\operatorname{argmax}} \tau_{iu}(t). \tag{5.9}$$

Omezení hladiny feromonu

Jednou z nejvýraznějších změn oproti Ant System je použití *horního* (τ_{max}) a *dolního* (τ_{min}) omezení hladiny feromonu. Při každé manipulaci s hodnotou feromonu τ_{ij} pro dané přiřazení musí být tyto hranice respektovány a musí tedy platit $\tau_{min} \leq \tau_{ij} \leq \tau_{max}$. Pokud nastavovaná hodnota feromonu překračuje některou z mezí, zarovnáme ji na hodnotu dané meze, aby byly úrovně feromonu vždy korektní.

Hlavním cílem představeného přístupu je omezení *stagnace* procesu řešení. Pokud průběh hledání optimálního přiřazení stagnuje, je hladina feromonu pro málo využívaná dílčí přiřazení blízká nule a naopak aktuálně nejlepší řešení je v rozložení feromonu zastoupeno tak silně, že jej konstruuje valná většina umělých mravenců. Pokud respektujeme dolní mez hladiny feromonu, mají i nepoužívaná dílčí přiřazení přiměřenou hodnotu feromonu a mohou tak být vybírána při konstrukci celkového přiřazení. Horní mez naopak zabraňuje příliš silné dominanci aktuálně nejlepšího řešení.

Aby meze co nejlépe korespondovaly s procesem řešení úlohy, jsou *dynamické* a mění se za běhu algoritmu. Horní mez počítáme (viz [21]) za použití nejlepšího řešení celého běhu algoritmu π^+ a koeficientu vypařování ρ takto:

$$\tau_{max} = \frac{1}{\rho \cdot C(\pi^+)}. \tag{5.10}$$

Výpočet dolní meze je v literatuře uveden v několika různých podobách, které jsou navíc pro praktické aplikace často výrazně zjednodušovány. Pro potřeby této práce byl vybrán vztah z [21]:

$$\tau_{min} = \frac{\tau_{max} \cdot (1 - \sqrt[n]{p_{best}})}{(avg - 1) \cdot \sqrt[n]{p_{best}}}, \tag{5.11}$$

kde n je rozměr problému (počet lokací / aktivit), p_{best} je pravděpodobnost nalezení nejlepšího řešení v případě konvergence algoritmu (podle [21] zvolena hodnota 0,05) a $avg = n/2$ je průměrný počet rozhodnutí, které musí agent udělat v každém kroku konstrukce řešení.

Protože je výpočet horní meze (a tím pádem i dolní meze) závislý na ohodnocení celkově nejlepšího nalezeného řešení, je nutné přepočítat meze při každém překonání tohoto řešení. Pokud dojde k situaci $\tau_{min} > \tau_{max}$, pak je nutné nastavit $\tau_{min} = \tau_{max}$.

Inicializace hladiny feromonu

MMAS pro inicializaci feromonových stop využívá vyšších hodnot, aby bylo při startu algoritmu co nejvíce uplatněno prohledávání stavového prostoru. Doporučuje se začít s výpočtem při hladině feromonu odpovídající horní mezi τ_{max}, kterou však vzhledem k absenci nejlepšího řešení nemůžeme vypočítat. Problém lze řešit například tak, že nastavíme obě meze na zvolenou počáteční hodnotu feromonu. Po provedení první iterace již není problém meze korektně nastavit a také zarovnat (před jeho dalšími úpravami) aktuální rozložení feromonu na hodnotu horní meze.

Algoritmus tedy začíná s vyššími hodnotami feromonu pro všechna τ_{ij} a postupně snižuje hladinu u nevyužívaných dílčích přiřazení. Kvalitní části řešení si naopak udržují hladinu feromonu blízkou horní hranici τ_{max}.

Aktualizace stavu feromonu

Po dokončení konstruktivní fáze všech umělých mravenců dojde, stejně jako u Ant Colony System, k aktualizaci stavu feromonu na základě řešení pouze jednoho mravence. ACS ovšem přesně stanovoval, že mají být úpravy feromonu provedeny podle nejlepšího řešení vzhledem k celému běhu algoritmu. U MMAS máme na výběr – nabízí se buď globálně nejlepší řešení jako u ACS, nebo nejlepší řešení v rámci aktuální iterace. První způsob by měl více preferovat kvalitní řešení a algoritmus by měl rychleji konvergovat, ovšem často k suboptimálnímu řešení. Druhý způsob zase více podporuje prohledávání prostoru.

V literatuře (viz např. [21]) jsou popsány i složitější kombinace obou možností. Jedním z rozvinutějších přístupů je výchozí použití optimálního řešení v rámci iterace, přičemž je stanoven určitý počet iterací t_{gb}, po kterém je vždy aplikováno celkově nejlepší řešení. Právě popsaný postup pro aktualizaci stavu feromonu je uvažován i pro použití MMAS v rámci této práce.

Vyhlazování feromonových stop

Pomocí horní a dolní meze pro úrovně feromonu sice omezujeme stagnaci, ale nezbavujeme se jimi podobného jevu, zvaného *konvergence* (viz [21]). Stagnace je stav, kdy všechny jednotky konstruují stejná řešení. Tento stav se v MMAS vzhledem ke spodní mezi feromonu nevyskytuje. Konvergence nastává, pokud je pro danou lokaci úroveň feromonu jedné z aktivit blízká horní mezi τ_{max} a všech ostatních aktivit blízká dolní mezi τ_{min}. Bude-li tedy umělý mravenec volit v každém kroku možnost s nejvyšším feromonovým ohodnocením, zkonstruuje dříve nalezené globálně nejlepší řešení.

Vyhlazování feromonových stop (popsané v [21] i [20]) je prostředkem pro vyrovnání se s jevem konvergence, který detekujeme na základě kontroly průměrného λ-faktoru větvení. Pokud proces řešení úlohy konverguje, dojde k navýšení stavu všech feromonových úrovní τ_{ij}. Zvýšení je přímo úměrné vzdálenosti od horní meze, jak ukazuje příslušný vzorec:

$$\tau'_{ij}(t) = \tau_{ij}(t) + \delta \cdot (\tau_{max}(t) - \tau_{ij}(t)), \tag{5.12}$$

kde $\tau'_{ij}(t)$ představuje novou hodnotu feromonu pro přiřazení aktivity j lokaci i a $0 < \delta < 1$ je parametr. Uvedená úprava zvyšuje hladinu feromonu málo používaných fragmentů řešení a snižuje dominanci aktuálně nejlepšího řešení. S rostoucím parametrem δ se zvyšuje míra zahlazení naučených informací a pro $\delta = 1$ dochází ke kompletnímu resetování stavu feromonu.

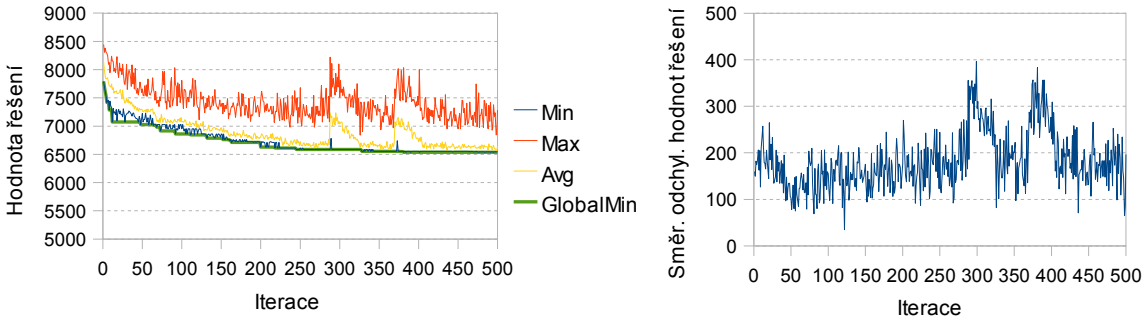
5.3.2 Úvod experimentů

V této části se opět zaměříme na experimentování s algoritmem a na testování vlivu jeho různých parametrů. Jako *výchozí hodnoty parametrů* byly zvoleny $\rho = 0,1$, $q_0 = 0,9$, $t_{gb} = 20$, $\delta = 0,5$ a $\tau_0 = 100$. Dále bude parametr t_{conv} označovat počet iterací, po kterých je vždy zjišťována případná konvergence (pro následné možné provedení vyhlazování). Jeho výchozí hodnota byla $t_{conv} = 40$. Pro variantu algoritmu bez lokálního prohledávání dále platilo $t = 1000$, $m = 100$ a každý test byl průměrem ze 40 běhů. U varianty s lokálním prohledáváním bylo nastavení následující: $t = 500$, $m = 50$ a 20 opakování každého běhu.

5.3.3 Charakter chování algoritmu

Chování algoritmu Max-min Ant System bude předvedeno na řešení instance `nug30` při výchozím nastavení parametrů, výjimkou je pouze $m = 30$. Průběhy hodnot řešení a smě-

rodatné odchylky skupiny řešení jsou zobrazeny v grafu 5.14.



Obrázek 5.14: Vlevo typický průběh algoritmu Max–Min Ant System při řešení QAP úlohy `nug30`. Vpravo odpovídající průběh směrodatných odchylek.

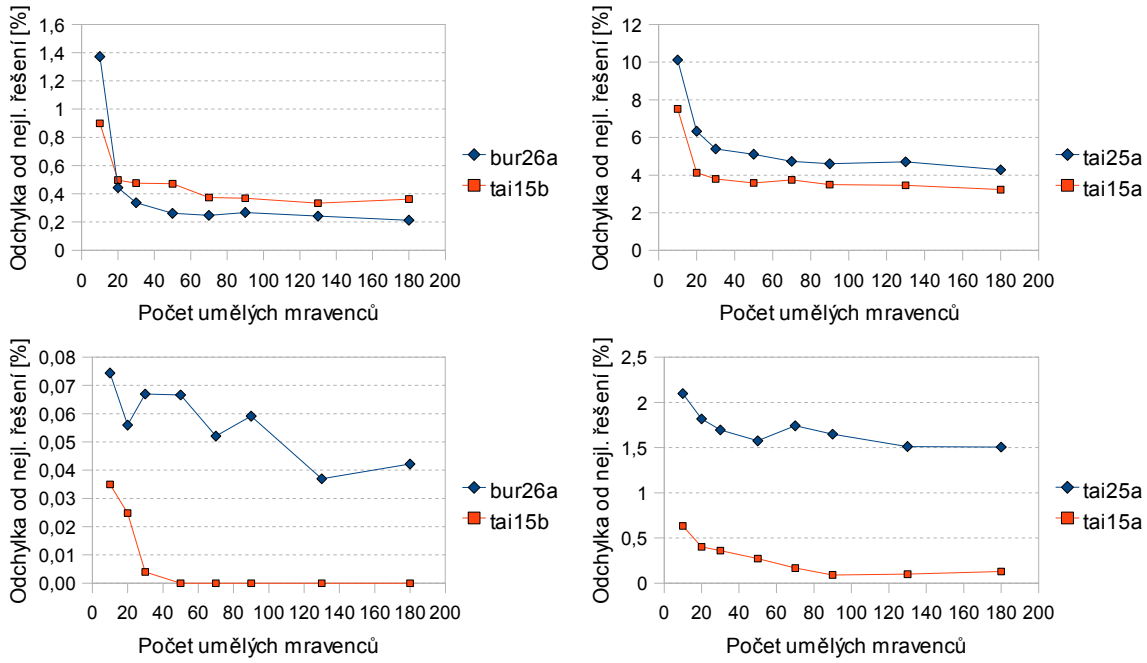
V průbězích lze vidět, že se řešení v jednotlivých iteracích kontinuálně zlepšují a postupem času klesají jejich hodnoty. Algoritmus udržuje množinu řešení v určitém přiměřeném rozmezí hodnot a i přes směřování k nižším hodnotám je zachována diverzita skupiny řešení. Přibližně v 290. iteraci lze pozorovat dopad provedeného vyhlazování feromonových stop. Po jeho provedení hodnoty řešení skokově vzrostou, ovšem jsou v dalších iteracích opět snižovány, až je překonáno dosavadní nejlepší řešení. Účinek vyhlazování se promítl také do hodnot směrodatných odchylek, které jsou rovněž dočasně zvýšeny.

5.3.4 Vliv počtu mravenců

Nejdříve budeme zjišťovat, nakolik počet použitých umělých mravenců ovlivňuje kvalitu nalézaných řešení. Parametr m byl proto nastavován na hodnoty 10 až 180, a to v rámci různých typů instancí. Výsledky testů lze nalézt v tabulkách A.36 (verze bez lokálního prohledávání) a A.37 (s lokálním prohledáváním).

Grafy na obrázku 5.15 nahoře ilustrují výsledky základní verze algoritmu. Zvyšování počtu jednotek má výrazný dopad zejména u spodní hranice testovaného intervalu hodnot m. Po překročení hranice přibližně 25 jednotek se s dalším zvyšováním m zlepšovala kvalita řešení v mnohem menší míře. Z hlediska feromonu omezují větší skupiny jednotek velmi mírně rozsah relevantních přiřazení. Jinak také znamenají větší diverzitu řešení a rychlejší zlepšování řešení v počátečních iteracích. Zajímavé je, že bylo řešení menší pseudoreálné instance `tai15b` náročnější než větší reálné instance `bur26a`. První instance má zřejmě méně strukturované rozložení lokálních optim.

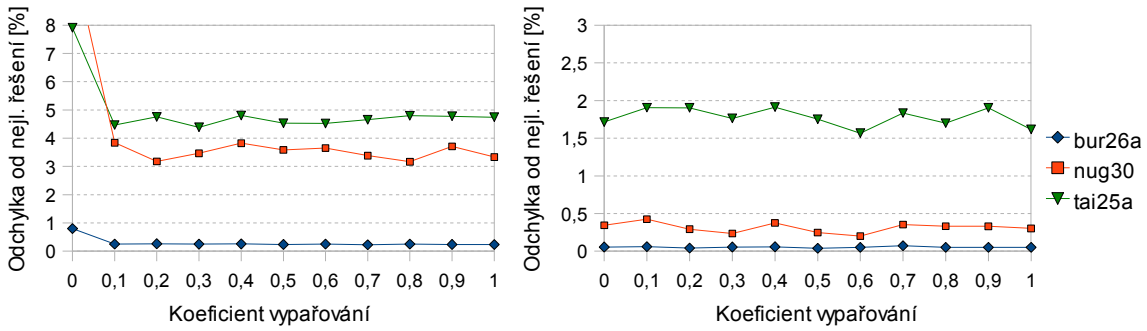
Ačkoliv jsou průběhy v grafech na obrázku 5.15 dole (verze s lokálním prohledáváním) značně rozkolísané, lze opět pozorovat kladný dopad rozšiřování skupiny umělých mravenců na kvalitu nalézaných řešení. Pokud ale vezmeme v úvahu podobná meření u jiných metod, pak lokální prohledávání méně pomáhá při řešení menších testovaných instancí (`tai15b` a zejména `tai15a`). Ty bývají jinými metodami řešeny s nulovými odchylkami i s využitím poměrně malého počtu jednotek. Výkon algoritmu však může být zdokonalen vhodnější volbou parametrů.



Obrázek 5.15: Závislost kvality řešení na počtu umělých mravenců (nahoře bez lokálního prohledávání, dole s lokálním prohledáváním).

5.3.5 Vypařování feromonu

Následující experimenty měly za úkol zjistit, jakou roli hraje parametr ρ, tedy jak ovlivňuje síla vypařování práci algoritmu. Parametr byl nastavován na hodnoty 0 až 1 s krokem 0,1. Naměřené odchylky pro verzi bez / s lokálním prohledáváním jsou uvedeny v tabulkách A.38 a A.39.



Obrázek 5.16: Závislost kvality řešení na koeficientu vypařování (vlevo bez lokálního prohledávání, vpravo s lokálním prohledáváním).

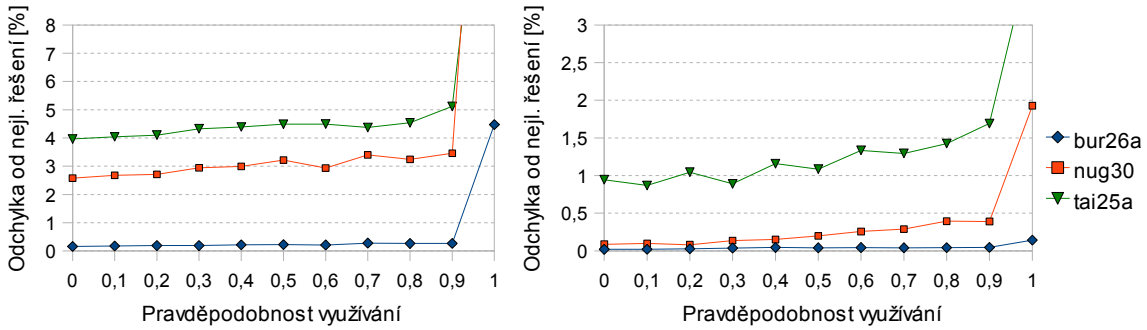
Graf na obrázku 5.16 vlevo má omezen rozsah osy y, protože byly odchylky od optimálního řešení pro nulové vypařování velké a nebylo možné pozorovat změny v odchylkách při zvyšování hodnoty parametru ρ. Výsledky testů ovšem naznačují, že kromě uvedené krajní hodnoty nemá parametr ρ významný vliv na kvalitu nalézaných řešení. Spíše lze od-

hadnout, že se odchylky pohybují kolem určité střední hodnoty. Zvyšování ρ mělo jinak za následek omezování počtu relevantních přiřazení, které jsou preferovány z hlediska rozložení feromonu.

Výsledky verze s lokálním prohledáváním byly použity k vytvoření grafu na obrázku 5.16 vpravo, který ale opět neukazuje přesvědčivý trend. Ani spodní krajní hodnota parametru již nemá tak významný výsledek, jako v předchozím měření. V průbězích pro instanci `tai25a` a `nug30` je sice viditelná jistá podobnost a také bylo nejlepších výsledků dosaženo pro nastavení přibližně $\rho = 0,5$ až $\rho = 0,6$, ale je možné, že by opakované měření mohlo tyto hodnoty popřít. Vliv parametru opět není výrazný.

5.3.6 Vyvážení využívání a prozkoumávání

Další testy se zaměřily na parametr q_0, který určuje, zda budou umělí mravenci spíše využívat naučených informací a vybírat nejvhodnější možnosti nebo zda budou více prozkoumávat na základě pravděpodobnostního rozložení. Protože parametr představuje pravděpodobnost využívání, byly hodnoty parametru nastavovány na 0 až 1 s krokem 0,1. Zjištěné odchylky od optimálních řešení jsou uvedeny v tabulce A.40 pro verzi bez lokálního prohledávání a v tabulce A.41 pro vylepšenou verzi.



Obrázek 5.17: Závislost kvality řešení na vyvážení využívání a prozkoumávání (vlevo s lokálním prohledáváním, vpravo bez lokálního prohledávání).

q_0 je případem parametru, který algoritmus výrazně a jednoznačně ovlivňuje. Z grafu na obrázku 5.17 vlevo lze odvodit jeho vztah ke kvalitě nalézaných řešení: zvyšováním q_0 jsou řešení degradována. Obzvláště nevýhodné je nastavení $q_0 = 1$ (tedy povolení pouze využívání), kde jsou odchylky od optimálního řešení velmi vysoké. Nejlepší volbou bylo naopak nastavení $q_0 = 0$, tedy maximálního prozkoumávání a nulového využívání [1]. Parametr měl také vliv na feromonové struktury. Pro jeho malé hodnoty algoritmus považoval v počátečních iteracích za relevantní vždy přibližně polovinu dílčích přiřazení a v pozdějších se dařilo narušovat jedno dominantní přiřazení pomocí vyhlazování. Při zvyšování q_0 však docházelo obecně k omezování počtu těchto relevantních řešení, až pro $q_0 = 1$ podporovaly feromonové struktury jediné dominantní přiřazení v celém průběhu řešení. To mělo vliv i na diverzitu řešení, která se snižovala.

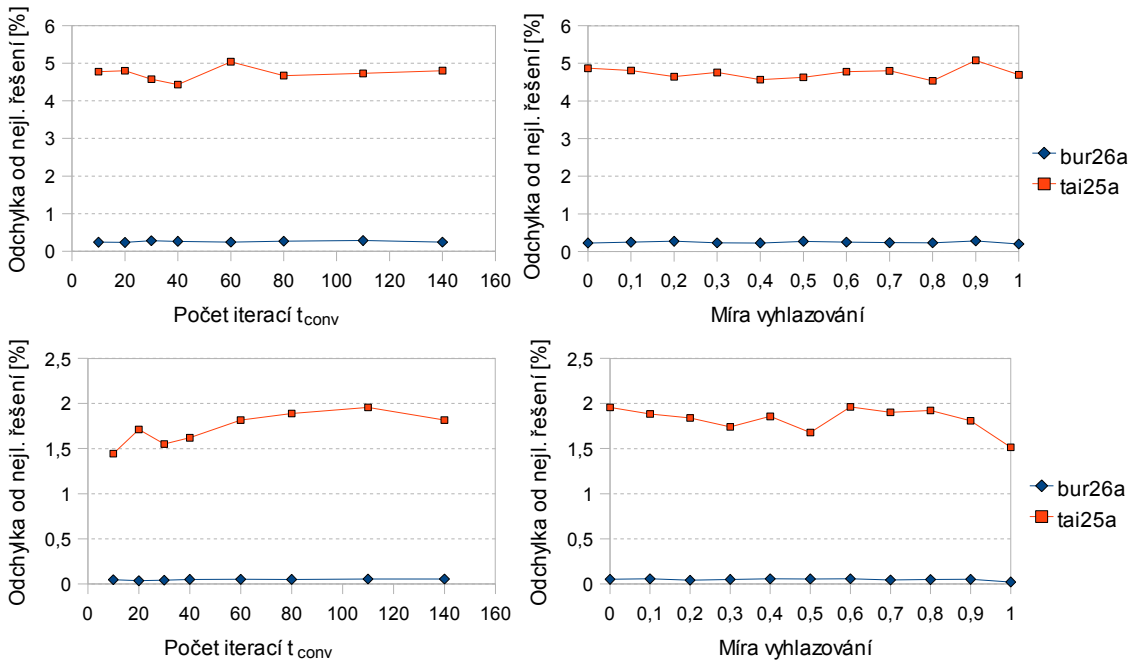
Graf v vpravo na obrázku 5.17 pro verzi s lokálním prohledáváním je velmi podobný předchozímu a potvrzuje vliv zkoumaného parametru. Opět je zde nejvýhodnější volbou

[1] Na tomto místě můžeme zmíni výsledky experimentů na Hybrid Ant System, kde byl odhalen přesně opačný vliv hodnoty q_0 na kvalitu nalézaných řešení.

$q_0 = 0$ a i ostatní sledované údaje byly parametrem podobně ovlivněny.

5.3.7 Vyhlazování feromonových stop

Dalšími testovanými parametry byly t_{conv}, čili počet iterací, po kterých je vždy zkontrolována míra konvergence, a δ, který určuje sílu případného vyhlazení rozložení feromonu. Protože spolu souvisí, bude uvedeno zhodnocení příslušných experimentů dohromady. t_{conv} nabýval v rámci testů hodnot 10 až 140 a výsledné odchylky od optimálních řešení byly zapsány do tabulky A.44. Parametr δ byl nastavován na hodnotu 0 až 1 s krokem 0,1 a příslušné výsledky testů lze nalézt v tabulce A.45.



Obrázek 5.18: Závislost kvality řešení na počtu iterací pro kontrolu konvergence / síle vyhlazení feromonu (nahoře bez lokálního prohledávání, dole s lokálním prohledáváním).

Grafy na obrázku 5.18 nahoře ilustrují výsledky testů obou parametrů v rámci algoritmu bez lokálního prohledávání. Pokud jde o vliv t_{conv}, po aproximaci získaného průběhu lineární regresí byla odhalena mírná degradace řešení při zvyšování hodnoty tohoto parametru u instance `tai25a`. Vztah však zřejmě není silný. Vzhledem k častějším kontrolám konvergence je také algoritmus při vyšších t_{conv} častěji restartován.

Odchylky od optima při změnách hodnot δ kolísaly kolem určité střední hodnoty a nebyl tak viditelný zřetelný vliv parametru. Vyšší hodnoty δ po provedení vyhlazování způsobily očekávanou větší diverzitu řešení a také rovnoměrnější rozložení feromonu.

Měření na algoritmu s lokálním prohledáváním přinesla výsledky, které lze vidět v grafu na obrázku 5.18 dole. t_{conv} potvrdil závislost naznačenou u předchozí verze algoritmu pro instanci `tai25a`. Vliv zde byl silnější, a je proto možné doporučit nejnižší testovanou hodnotu $t_{conv} = 10$. Nízké hodnoty parametru byly výhodné i pro reálnou instanci `bur26a`. Častější kontroly konvergence měly na průběhy podobné účinky jako u minulé verze algoritmu.

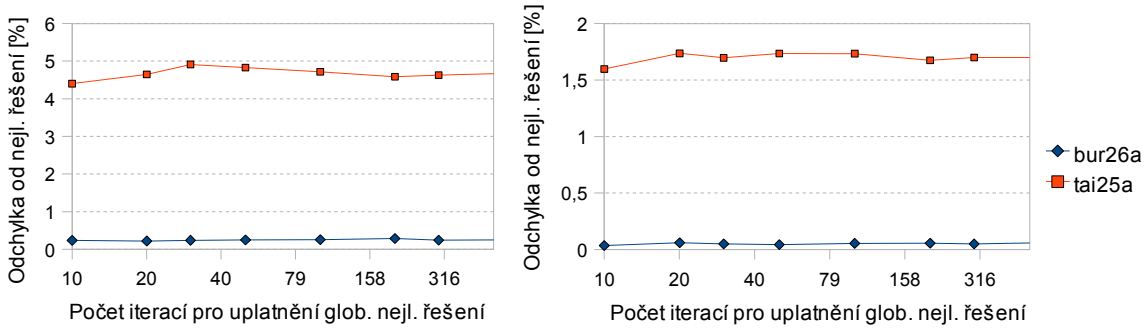
Nejlepších výsledků nyní algoritmus dosáhl při nastavení $\delta = 1$, a to pro obě testované

instance. Vliv na diverzitu řešení po provedené vyhlazování i na feromonové struktury zůstal zachován.

5.3.8 Další parametry

Počet iterací mezi použitím globálně nejlepšího řešení

Tyto experimenty zkoumaly chování algoritmu při různých hodnotách parametru t_{gb}, který určuje počet iterací, po kterých je periodicky při úpravách rozložení feromonu použito globálně nejlepší řešení. Parametr byl v rámci testů nastavován na hodnoty 10 až 500 a výsledky byly zapsány do tabulky A.42.



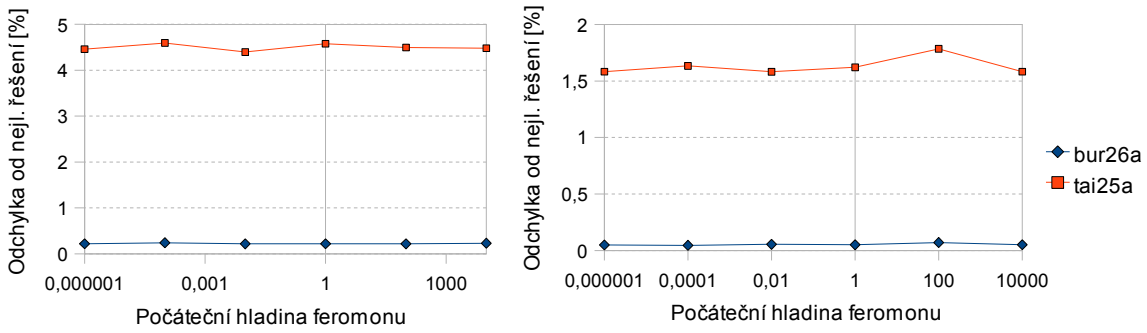
Obrázek 5.19: Závislost kvality řešení na počtu iterací pro aplikaci globálně nejl. řešení při úpravě feromonu (vlevo bez lokálního prohledávání, vpravo s lokálním prohledáváním).

Výsledky testů základní verze ilustruje graf na obrázku 5.19 vlevo s logaritmickým měřítkem v ose x. Odchylky naznačují, že jsou výhodnější nižší hodnoty t_{gb}, kdy je globální řešení aplikováno častěji. Pro `bu26a` přineslo nejlepší výsledky nastavení $t_{gb} = 20$, pro náhodnou instanci `tai25a` potom $t_{gb} = 10$. Rozdíly v odchylkách však nejsou velké. I ostatní sledované ukazatele nebyly významněji ovlivněny. Měření na verzi s lokálním prohledáváním opět vede k doporučení nejnižší testované hodnoty parametru t_{gb}. Odchylky lze názorně vidět na uvedeném obrázku vpravo.

Počáteční hladina feromonu

V této části budou popsány experimenty, které zkoumaly roli parametru τ_0. K tomuto účelu byl nastavován na hodnoty 10^{-6} až 10^4 s krokem exponentu 2. Naměřená data obsahuje tabulka A.43.

Jak ukazuje graf na obrázku 5.20 vlevo, neměly změny hodnot parametru významný dopad na kvalitu nalézaných řešení. Přestože byl testován poměrně velký rozsah možných hodnot τ_0, pohybují se odchylky v malém okolí určité střední hodnoty. Také v rozložení feromonu a dalších údajích nebyly zaznamenány změny. Stejně tak je tomu i v případě verze Max–Min Ant System s lokálním prohledáváním, viz graf 5.20 vpravo. Počáteční stav feromonu totiž slouží prakticky jen pro nastavení spodní a horní meze feromonových hladin před startem algoritmu, aby bylo možné vytvořit první sadu řešení.



Obrázek 5.20: Závislost kvality řešení na počáteční hodnotě feromonu (vlevo bez lokálního prohledávání, vpravo s lokálním prohledáváním).

5.4 Hybrid Ant System

5.4.1 Popis algoritmu

Tento algoritmus, jehož následující popis vychází z [2] a [6], se podstatným způsobem liší od předchozích algoritmů. Nejvýraznějším rozdílem je naprosto odlišná práce s jednotlivými řešeními. Zatímco u dříve uvedených algoritmů byla tato řešení (čili přiřazení u QAP) v každé iteraci každým mravencem postupně konstruktivně vytvářena, zde je každému mravenci na počátku přiřazeno počáteční řešení, které je v dalších iteracích upravováno. Představme si nejdříve pseudokód algoritmu, který je vzhledem k odlišné a složitější funkci metody uveden v detailnější formě (algoritmus 2).

Pseudokód algoritmu

Parametry algoritmu mají následující význam: ρ je koeficient vypařování feromonu, q_0 parametr vyvažující prohledávání a využívání při modifikaci řešení, m počet mravenců, Q koeficient pro inicializaci feromonu, R počet výměn aktivit při modifikaci řešení, S maximální počet iterací bez překonání dosavadního nejlepšího řešení před provedením diverzifikace a t_{max} maximální počet iterací algoritmu.

Inicializace algoritmu

Hlavním úkolem inicializace je nyní vygenerování náhodných *počátečních řešení* pro jednotlivé umělé mravence. Každé takové řešení bude s daným mravencem během výpočtu spjato a bude postupně vyvíjeno a zlepšováno. Ještě ve fázi inicializace je každé z řešení vylepšeno pomocí lokálního prohledávání, například způsobem popsaným v části 5.1.2. Získáváme tak sadu kvalitních počátečních řešení, které jsou dobrým výchozím stavem pro následující výpočty.

Do fáze inicializace opět patří i *inicializace feromonu*. Její výchozí hladina není tentokrát dána vstupním parametrem, ale je vypočítána podle hodnoty nejlepšího řešení π^+ z počáteční skupiny řešení:

$$\tau_0 = \frac{1}{Q \cdot C(\pi^+)}. \tag{5.13}$$

Algoritmus 2 Pseudokód Hybrid Ant System algoritmu

```
procedure HybridAntSystem(ρ, q0, m, Q, R, S, t_max)
    Vygenerování počátečních řešení mravenců
    Vylepšení poč. řešení pomocí lokálního prohledávání
    Inicializace feromonu
    Zapnutí intenzifikace
    while not Podmínka ukončení do
        Modifikace řešení mravenců pomocí R výměn
        Vylepšení řešení pomocí lokálního prohledávání
        Proces intenzifikace (je-li aktivní)
        if Žádné řešení se od minulé iterace nezměnilo then
            Vypnutí intenzifikace
        end if
        if Nalezeno nové nejlepší řešení then
            Uložení nejlepšího řešení
            Zapnutí intenzifikace
        end if
        Aktualizace stavu feromonu
        if Během S iterací nebylo překonáno nejlepší řešení then
            Proces diverzifikace
        end if
    end while
    return Nejlepší řešení
end procedure
```

Modifikace jednotlivých řešení

Jak bylo uvedeno výše, každý umělý mravenec si nese od fáze inicializace své vlastní řešení dané QAP instance. Modifikace toho řešení jsou prováděny pouze na základě rozložení hodnot feromonu (tedy naučené informace) a spočívají ve výměně aktivit u celkem R (parametr metody) vybraných dvojic lokací. První lokace i je vždy vybrána náhodně. Druhá lokace j je vybrána podle toho, zda je v závislosti na parametru q_0 vybrána modifikace řešení pomocí strategie *prozkoumávání* nebo *využívání* (podobně jako v případě ACS nebo MMAS, viz 5.2.1). Jednotlivé strategie jsou voleny prostřednictvím náhodně generované hodnoty q v rozsahu 0–1:

Prozkoumávání, $q > q_0$ Tato možnost pracuje s pravděpodobnostním rozložením, odvozeným od feromonové informace. Pravděpodobnost výměny lokace i a j v řešení mravence k, podle které náhodně vybíráme lokaci j, je stanovena takto:

$$p_{ij}^k = \frac{\tau_{i\pi_j^k} + \tau_{j\pi_i^k}}{\sum_{a=1,a\neq i}^{n} \tau_{i\pi_a^k} + \tau_{a\pi_i^k}}, \tag{5.14}$$

kde π^k je řešení daného mravence. Prozkoumávání má podobný cíl, jako u předchozích algoritmů: pokrýt co nejlépe různé možnosti a vyzkoušet tak i horší výměny, které ale mohou ve výsledku přinést lepší výsledky.

Využívání, $q \leq q_0$ Strategie využívání naopak hledá druhou lokaci j způsobem, který se zaměřuje na momentálně (z hlediska feromonových informací) nejlepší možnou vari-

antu. Lokace tedy bude zvolena podle vztahu

$$j = \underset{1 \leq a \leq n, a \neq i}{\operatorname{argmax}} \left(\tau_{i\pi_a^k} + \tau_{a\pi_i^k} \right). \tag{5.15}$$

Intenzifikace

Pseudokód 2 pro Hybrid Ant System ukazuje, že je intenzifikace zapnuta ve chvíli (kromě počáteční aktivace při startu algoritmu), kdy je po aktualizaci řešení mravenců rozpoznáno nové, globálně nejlepší řešení. Proces intenzifikace potom toto kvalitní řešení preferuje, umožňuje vnést jeho vliv do feromonových struktur a vede algoritmus k prozkoumávání jeho okolí. Princip je jednoduchý: po stanovení nového řešení každého agenta (tzn. po fázi modifikace a lokálním prohledávání) je porovnán původní a nový získaný stav. Pokud je nové řešení méně kvalitní, agent je nepřijme a ponechá si výchozí řešení. Sousedství aktuálních výhodných přiřazení je díky intenzifikaci řádně využito, protože dané přiřazení není umělým mravencem opuštěno při první, třeba i degradující úpravě.

Po určité době jsou však odmítány veškeré změny řešení mravenců jako nevyhovující. Za tohoto stavu je nutné deaktivovat intenzifikaci a umožnit tak přecházení na nová řešení pro širší prohledávání stavového prostoru.

Aktualizace stavu feromonu

Změny rozložení feromonu probíhají podle dvou hlavních pravidel: nejdříve je stav všech feromonových hladin τ_{ij} oslaben (vypařování feromonu) a poté je posíleno dosud nejlepší nalezené řešení (přiřazení) π^+. Pokud uvedené změny uvažujeme v kontextu úprav feromonu v předchozích představených algoritmech, zjistíme, že první změna má svoji obdobu v algoritmu Ant System, zatímco druhá je typická pro Ant Colony System. Proces aktualizace feromonu je opět řízen koeficientem vypařování ρ a novou hladinu feromonu pro přiřazení aktivity j lokaci i vypočítáme podle vzorce:

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \rho \cdot \Delta\tau_{ij}(t), \tag{5.16}$$

kde $\Delta\tau_{ij}(t)$ je příspěvek od nejlepšího řešení π^+. Tzn. pokud π^+ obsahuje přiřazení aktivity j lokaci i, bude velikost přírůstku dána vztahem $\Delta\tau_{ij}(t) = \frac{1}{C(\pi^+)}$. V opačném případě platí $\Delta\tau_{ij}(t) = 0$.

Diverzifikace

Diverzifikace je dobrým a potřebným nástrojem v situaci, kdy proces řešení problému uvázne v některém z lokálních optim. Hybrid Ant System takový stav detekuje ve chvíli, kdy během S předchozích iterací nedošlo k překonání globálně nejlepšího řešení π^+. HAS reaguje na uvedenou situaci vykonáním procesu diverzifikace, který významným způsobem mění rozložení feromonu a řešení spjatá s mravenci. Jedinému mravenci je přiděleno nejlepší řešení π^+ a všem ostatním jsou jejich řešení náhodně inicializována. Dochází také k inicializaci feromonu, a to podle stejného vzorce jako při startu algoritmu (viz vztah 5.13).

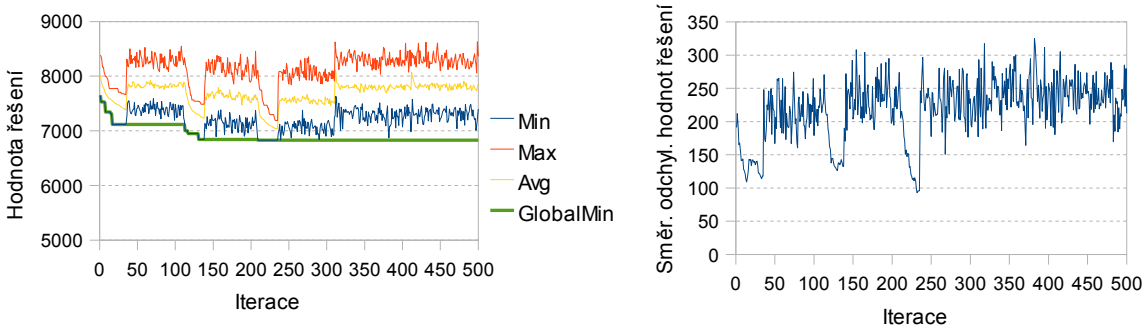
5.4.2 Úvod experimentů

Na algoritmu Hybrid Ant System byly opět prováděny experimenty, které především sledovaly vliv hodnot parametrů na kvalitu nalézaných řešení, ale vyhodnocovány byly také další údaje.

Jako *výchozí nastavení parametrů* byly použity následující hodnoty: $\rho = 0,1$, $S = 15$, $R = 10$ a $q_0 = 0,9$. U dalších se rozlišovalo mezi verzí s a bez lokálního prohledávání. V rámci první verze platilo $t = 1000$, $m = 100$ a bylo provedeno 40 opakování každého pokusu, druhé verzi bylo nastavováno $t = 500$, $m = 50$ a každý pokus byl 20krát zopakován.

5.4.3 Charakter chování algoritmu

Tato část bude prezentovat základní vlastnosti algoritmu na jednom z jeho zaznamenaných běhů, který je ukázán na obrázku 5.21 vlevo (hodnoty řešení) a vpravo (směrodatná odchylka hodnot). Algoritmu byly nastaveny výchozí parametry, počet agentů $m = 30$ a řešenou instancí byla `nug30`.



Obrázek 5.21: Vlevo typický průběh algoritmu Hybrid Ant System při řešení QAP úlohy `nug30`. Vpravo odpovídající průběh směrodatných odchylek.

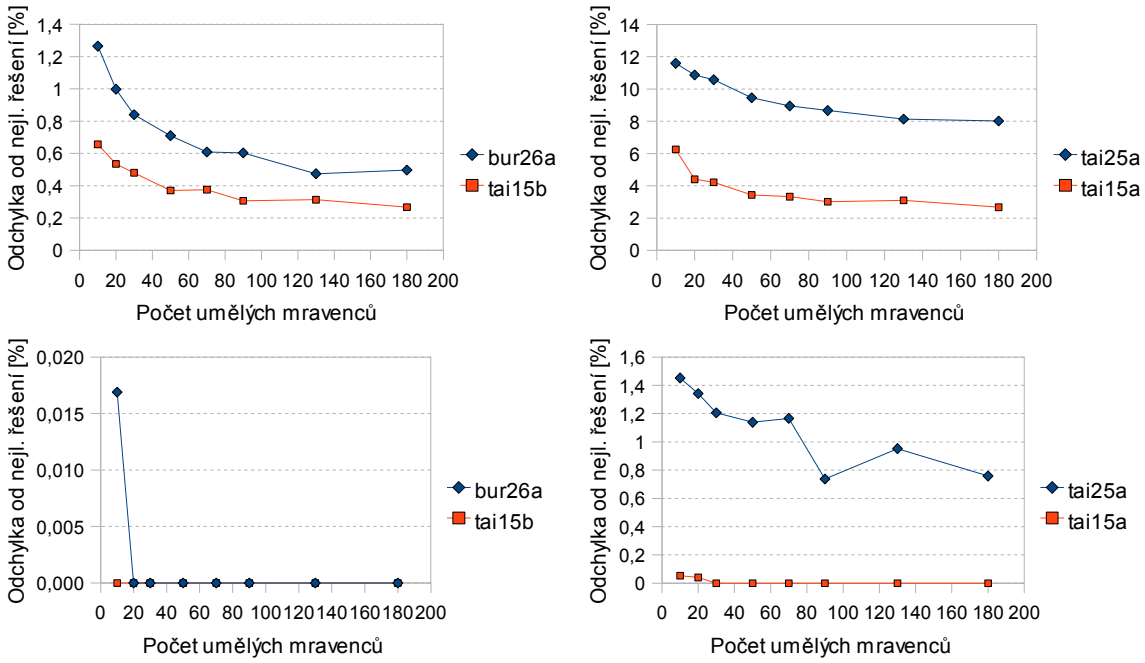
V průběhu hodnot řešení lze pěkně vidět jednotlivé fáze algoritmu. Na začátku je aktivován proces intenzifikace, a hodnoty řešení proto klesají. Každá jednotka totiž přijme modifikované řešení jen pokud je lepší než stávající. Časem však dojde k situaci, kdy není přijato žádné modifikované řešení, a tak je v přibližně 40. iteraci intenzifikace deaktivována. Řešení potom mají výrazně větší diverzitu a algoritmus se snaží vyváznout z lokálního optima. Toho dosáhne v přibližně 110. iteraci, opět je zapnut proces intenzifikace a hodnoty řešení klesají. Kolem 310. iterace lze pozorovat provedení diverzifikace. Nejlepší řešení nebylo během 100 iterací překonáno, a tak jsou znovu inicializována řešení spjatá s mravenci a rozložení feromonu je uvedeno do rovnoměrného stavu (vytvořené struktury jsou tedy smazány).

V průběhu směrodatelné odchylky skupiny řešení je vidět postupné omezování diverzity řešení během intenzifikace a naopak její opětovné navýšení po skončení tohoto procesu i po provedení diverzifikace.

5.4.4 Vliv počtu mravenců

Tyto testy se zaměřují na chování algoritmu při různě velké skupině agentů. Pro reálnou instanci `bur26a`, náhodnou `tai25a` a náhodnou se vzdálenostmi na mřížce `nug30` bylo nastavováno $m = 10$ až $m = 180$. Naměřená data obsahují tabulky A.28 a A.29.

Verze bez lokálního prohledávání je zastoupena grafy na obrázku 5.22 nahoře. V rámci všech instancí byla nalézána kvalitnější řešení při větším počtu umělých mravenců. Závislost je podobná u všech měřených instancí. Při zvyšování počtu umělých mravenců je možné



Obrázek 5.22: Závislost kvality řešení na počtu umělých mravenců (nahoře bez lokálního prohledávání, dole s lokálním prohledáváním).

pozorovat, že se více udržují dobrá řešení, která se pohybují v blízkosti nejlepšího řešení dosavadního běhu algoritmu.

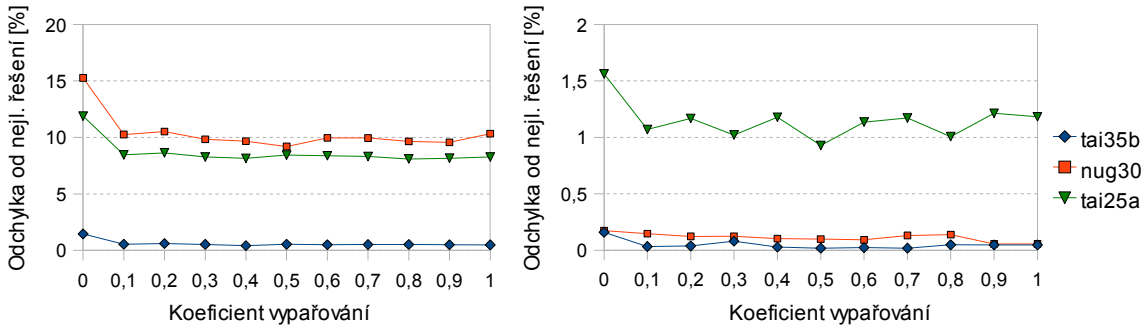
Průběhy změn v rozložení feromonu nebyly změnou počtu agentů nijak významně ovlivněny. Experimentování ale ukázalo, že se feromonové struktury obecně poměrně brzy ustálí na jednom dominantním přiřazení. Z toho stavu jsou jen krátkodobě vychýleny při provedení diverzifikace, ovšem takřka ihned se vrací zpět. U algoritmů, kde je feromonová informace jedním z vodítek pro konstrukci přiřazení, to většinou znamená významné snížení diverzity nalézaných řešení. U Hybrid Ant System však feromonová informace pouze pomáhá při modifikaci řešení, a vytvoření dominantní struktury proto nevede k tak silnému sjednocení řešení.

Grafy na obrázku 5.22 dole pro Hybrid Ant System s lokálním prohledáváním ukazují, že byly velmi úspěšně řešeny zejména reálné instance. Již 20 umělých mravenců stačilo k dokonalému řešení instance `bur26a`, pro menší (pseudoreálnou) `tai15b` stačil i nejnižší testovaný počet jednotek. Náhodné instance jsou obecně obtížněji řešitelné, ovšem u menší `tai15a` bylo dosaženo nulové průměrné odchylky od optima již s velmi malou skupinou umělých mravenců. Náročnější instanci `tai25a` opět prospívalo, zanedbáme-li odlehlé hodnoty, zvyšování počtu mravenců.

5.4.5 Vypařování feromonu

Další experimenty zkoumaly vliv míry vypařování feromonu, tedy nastavení parametru ρ. V rámci testů nabýval hodnot 0 až 1 s krokem 0,1. Získané odchylky od optimálních řešení jsou zaneseny v tabulkách A.30 a A.31.

Základní verzi algoritmu reprezentuje graf na obrázku 5.23 vlevo. Evidentní je nevhodné



Obrázek 5.23: Závislost kvality řešení na koeficientu vypařování (vlevo bez lokálního prohledávání, vpravo s lokálním prohledáváním).

nastavení $\rho = 0$, tedy prakticky vyřazení vypařování feromonu. Nejen že neubývá feromonu pro všechna dílčí přiřazení, ale také se nezvyšují feromonové hladiny příslušné k nejlepšímu nalezenému kompletnímu přiřazení. Rozložení feromonu se v důsledku toho vůbec nemění. Při zvyšující se hodnotě ρ dochází k omezování počtu relevantních přiřazení (z hlediska feromonu) v rámci počátečních iterací. Kvalita řešení pro různé hodnoty ρ je jinak poměrně vyrovnaná. U instance `nug30` lze pozorovat nejlepší výsledky při nastavení $\rho = 0,5$, pro vyloučení odlehlé hodnoty by však bylo nutné provést další testy. U ostatních instancí se odchylky pohybují v okolí určité střední hodnoty a ani další sledované údaje nenaznačují, že by měl koeficient vypařování na kvalitu nalézaných řešení výrazný vliv.

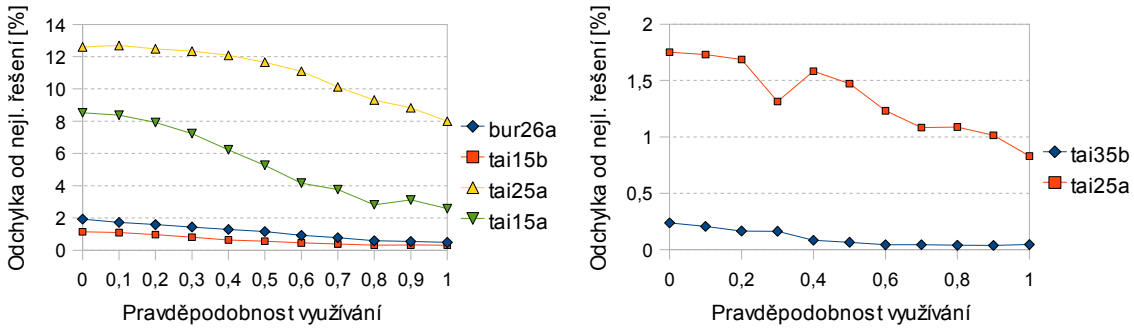
Grafy na obrázku 5.23 vpravo potvrzují, že úplné vyřazení vypařování ($\rho = 0$) má negativní dopad i na verzi s lokálním prohledáváním. Na základě výsledků pro instanci `nug30` lze odhadnout sestupnou tendenci odchylek od optima při zesilování vypařování, zatímco u náhodné instance `tai25a` se vzhledem ke kolísajícím hodnotám nepodařilo odhalit jednoznačný vliv vypařování. Měření na reálné instanci `bur26a` vedlo k dokonalé funkci algoritmu, a byla proto provedena dodatečná měření na pseudoreálné instanci `tai35b`. Dobré výsledky zde přineslo nastavení ρ na hodnotu 0,5 až 0,7, ale průběh jinak opět není pravidelný.

5.4.6 Vyvážení využívání a prozkoumávání

Hodnota parametru q_0 určuje, zda se budou umělí mravenci při modifikaci řešení naprosto spoléhat na feromonové informace (velké hodnoty parametru) nebo se budou rozhodovat na základě pravděpodobnostního rozložení (malé hodnoty parametru). Parametr byl nastavován na hodnoty 0 až 1 s krokem 0,1. Oproti výchozí konfiguraci pro provádění testů zde bylo pro verzi s lokálním prohledáváním použito pouze 15 opakování každého běhu. Naměřené průměrné odchylky od optimálních řešení jsou uvedeny v tabulkách A.32 a A.33.

Zjištěné odchylky pro algoritmus bez lokálního prohledávání jsou uvedeny v grafu na obrázku 5.24 vlevo. Závislost je zde napříč všemi měřenými instancemi naprosto jednoznačná: zvyšování pravděpodobnosti využívání vede k poměrně podstatnému zlepšování nalézaných řešení. Nejlepších výsledků bylo dosaženo při $q_0 = 1$, kdy je naprosto vyřazeno prozkoumávání a umělý mravenec vybírá vždy nejvýhodnější z uvažovaných modifikací. Při větším využívání se řešení s minimálními hodnotami pohybují blíže u globálně nejlepšího nalezeného řešení. Práce s rozložením feromonu se významně neměnila.

Graf na obrázku 5.24 vpravo ukazuje, že zjištěný vliv parametru q_0 platí také pro verzi s lokálním prohledáváním. Experimenty probíhaly také na instancích `bur26a`, `tai15a` a



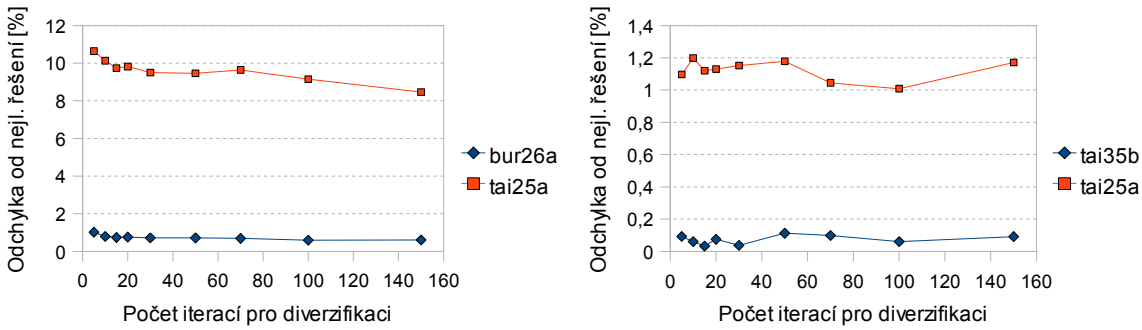
Obrázek 5.24: Závislost kvality řešení na vyvážení využívání a prozkoumávání (vlevo s lokálním prohledáváním, vpravo bez lokálního prohledávání).

`tai15b`, které byly dokonale řešeny při všech nastaveních testovaného parametru.

5.4.7 Další parametry

Počet iterací do provedení diverzifikace

Další experimenty se zaměřily na parametr S, který určuje, kolik proběhne iterací bez překonání dosud nejlepšího nalezeného řešení před provedením diverzifikace. Parametr byl testován v rozsahu hodnot 5 až 150 a naměřená data byla vložena do tabulky A.34.



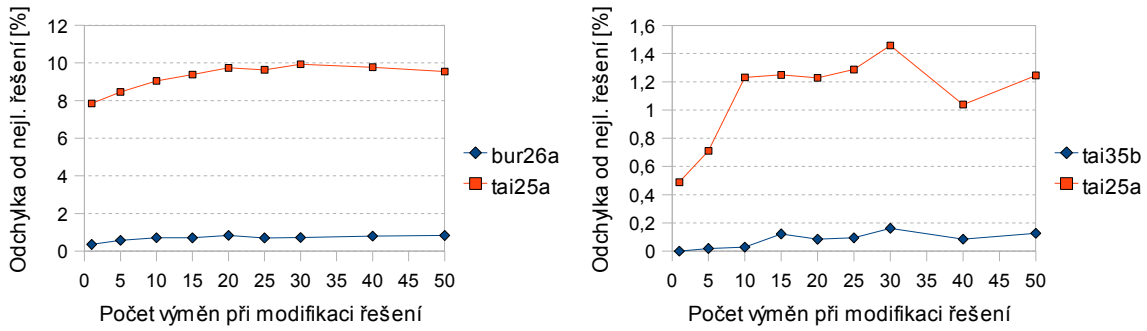
Obrázek 5.25: Závislost kvality řešení na počtu iterací pro diverzifikaci (vlevo bez lokálního prohledávání, vpravo s lokálním prohledáváním).

Graf na obrázku 5.25 vlevo pro verzi bez lokálního prohledávání ukazuje jednoznačný trend – zvyšování parametru S má na hledání řešení pozitivní vliv a snižuje se tak odchylka od optimálního řešení. Při malých hodnotách S má totiž algoritmus velmi málo času na překonání dosavadního nejlepšího řešení a jeho činnost je velmi často restartována. V opačném případě může déle využívat naučených informací a spíše nalezne lepší řešení.

Podobně jednoznačně se již vliv parametru S neprojevil na verzi s lokálním prohledáváním (viz graf na obrázku 5.25 vpravo). Pro pseudoreálnou instanci `tai35b` bylo nejvýhodnější nastavení $S = 15$, zatímco pro `tai25a` $S = 70$. Tyto údaje však, vzhledem ke kolísání odchylek, zřejmě nejsou příliš spolehlivé. Měření na instanci `tai35b` bylo provedeno dodatečně, protože byla `bur26a` řešena dokonale při všech nastaveních.

Počet výměn v rámci modifikace řešení

Parametr R algoritmu Hybrid Ant System slouží k nastavení počtu výměn, které se provedou při každé modikaci řešení daného agenta. Hodnota R byla nastavována na 1 až 50 a naměřené odchylky od optimálních řešení jsou uvedeny v tabulce A.35.



Obrázek 5.26: Závislost kvality řešení na počtu výměn při modifikaci řešení (vlevo bez lokálního prohledávání, vpravo s lokálním prohledáváním).

Z grafu na obrázku 5.26 vlevo lze jednoznačně vyčíst, že jsou pro kvalitu výsledků nejlepší nízké počty výměn a s jejich navyšováním jsou nacházena méně kvalitní řešení. Nejvýhodnější bylo nastavení nejnižšího testovaného počtu výměn, tedy $R = 1$. Malé hodnoty R vedly k delším procesům intenzifikace, během kterých také bylo vícekrát překonáno dosavadní nejlepší řešení.

Měření na algoritmu s lokálním prohledáváním potvrdily vliv parametru R. Průběhy v grafu na obrázku 5.26 vpravo jsou pouze méně pravidelné, ale trend odpovídá.

5.5 Genetic Ant System

V rámci této práce byla implementována vlastní metoda, která přináší oproti předchozím algoritmům některé nové prvky. Základem je prvotní algoritmus Ant System, který byl rozšířen o prvky genetického algoritmu. Využití skupiny jednotek přímo napovídá, že lze jejich parametry měnit dynamicky za běhu a dosáhnout tak jejich adaptování nejen na dosavadní průběh výpočtu, ale také na různé typy instancí. Podobný námět je rozpracován například i v práci [15], která však staví na Ant Colony System.

5.5.1 Popis algoritmu

Integrace genetického principu

Umělí mravenci v této metodě pomocí genetických principů ladí dva základní parametry algoritmu Ant System, a to váhu feromonu α a váhu heuristiky β. Váhy jsou každé jednotce na počátku náhodně vygenerovány. Chromozom, který přísluší ke každému z agentů, tedy obsahuje tyto dvě hodnoty jako reálná čísla. Hodnota fitness funkce chromozomu je odvozena z průměrné hodnoty řešení nalezených daným agentem. Čím lepší řešení agent nacházel, tím vyšší fitness je přiřazena jeho kombinaci vah. Vždy po provedení určitého počtu iterací, nastaveného parametrem t_{gen}, je aplikováno jedno kolo genetického algo-

ritmu. Výsledné chromozomy jsou potom základem pro novou skupinu agentů, jejichž váhy odpovídají hodnotám v chromozomech.

Obě váhy mají definovanou společnou maximální hodnotu, která je určena parametrem w_{max}. Výsledný rozsah hodnot vah od 0,1 (konstanta metody) do w_{max} je respektován při vytváření počáteční skupiny agentů i při úpravách chromozomů v rámci genetického algoritmu.

Aktualizace stavu feromonu

Původní algoritmus Ant System používal pro změny v rozložení feromonu trasy všech jednotek. V tomto algoritmu jsou oproti tomu preferováni agenti s dobrými výsledky, a feromonové rozložení proto upravuje pouze m_{upd} (parametr metody) nejlepších jednotek. Jinak je ale úprava rozložení feromonu prováděna stejně jako u Ant System. Výjimkou je pouze stav při rozrušování feromonových stop, popsaný v následující sekci.

Vyváznutí z lokálního optima

Jedním z podstatných problémů algoritmu Ant System je předčasná konvergence, doprovázená vytvořením dominantního přiřazení v rámci feromonové struktury. Genetic Ant System proto poskytuje prostředky, které pomáhají takové situace řešit.

Za běhu algoritmu je pravidelně kontrolován průměrný λ-faktor větvení, který dokáže detekovat přechod do stavu stagnace. Parametr λ_{min} určuje, jaká je požadovaná spodní mez tohoto ukazatele. Pokud je faktor větvení nižší, což představuje více dominantní struktury než je tolerováno, je spuštěn proces, který má za úkol narušit tyto struktury. Komplementární parametr λ_{max} určuje, kdy už je feromon dostatečně rozptýlen a popisovaný proces může být ukončen.

Popisovaný proces spočívá v odlišných úpravách rozložení feromonu. Dílčí přiřazení těch řešení, které byly zkonstruovány nejlepšími jednotkami, nyní nejsou posilovány. Jejich hodnota je naopak nastavována na minimální hodnotu (téměř rovnu nule). Tímto jsou dočasně vyřazena nejpoužívanější přiřazení, rozšíří se možnost výběru mezi relevantními dílčími přiřazeními a algoritmus tak prohledává jinou a větší část stavového prostoru.

Druhým prostředkem pro posílení procesu řešení je dočasné přidání pomocných agentů. Jejich počet je polovinou běžného počtu m. Algoritmus k tomu využívá parametru t_{res}. Pokud je vykonáno $t_{res}/2$ iterací od posledního překonání nejlepšího řešení, je skupina agentů rozšířena o speciální pomocné jednotky. Základem jejich konstruovaného řešení je nejlepší nalezené řešení. Poté je náhodně vybrána jedna z lokací, které je nově přiřazena některá z n_{best} nejlepších aktivit. Kvalita dílčích přiřazení je posuzována na základě aktuálního rozložení feromonu. Tito umělí mravenci tedy modifikují nejlepší řešení na základě naučených informací.

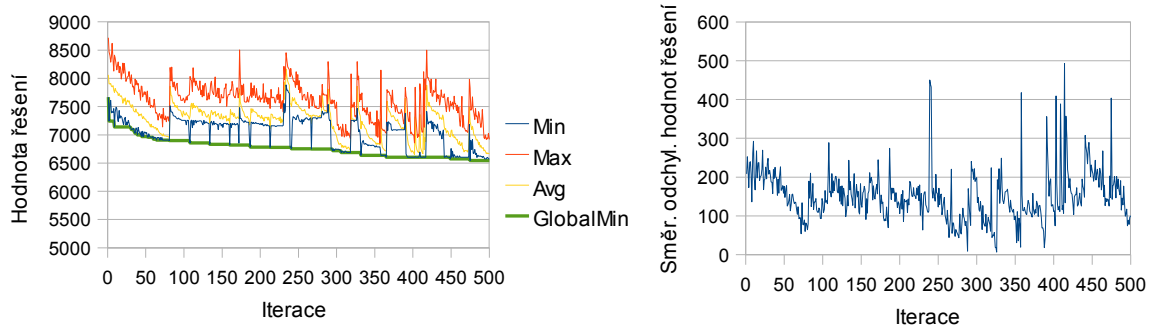
Pokud potom počet iterací bez nalezení lepšího řešení překoná hranici t_{res}, jsou pomocní agenti odstraněni a původní skupině jsou znovu náhodně inicializovány váhy.

5.5.2 Úvod experimentů

Uvedené experimenty využívaly následujících *výchozích hodnot parametrů*: $\tau_0 = 10^{-6}$, $\rho = 0,1$, $t_{gen} = 10$, $m_{upd} = 5$, $w_{max} = 1,5$, $t_{res} = 50$, $n_{best} = 10$, $\lambda_{min} = 1,3$ a $\lambda_{max} = 2$. Počet umělých mravenců a iterací se lišil podle použité verze algoritmu. Pro základní verzi to bylo $m = 80$ a $t = 1000$, pro verzi s lokálním prohledáváním $m = 40$ a $t = 300$.

5.5.3 Charakter chování algoritmu

Jako první bude opět představen typický průběh procesu řešení úlohy. Kromě výchozích parametrů bylo pro tento běh použito 30 agentů a řešena byla instance `nug30`. Průběh hodnot řešení a směrodatné odchylky hodnot řešení jsou zaneseny v grafech na obrázku 5.27.

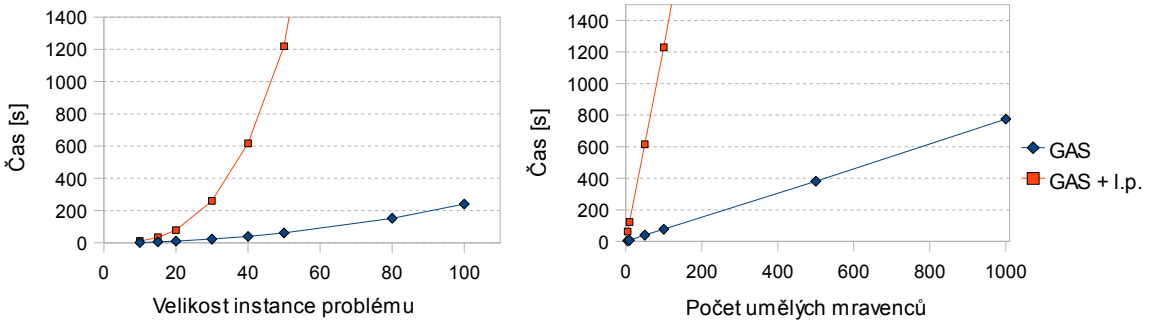


Obrázek 5.27: Vlevo typický průběh algoritmu Genetic Ant System při řešení QAP úlohy `nug30`. Vpravo směrodatná odchylka typického průběhu algoritmu.

Algoritmus na začátku běhu překonává nejlepší řešení a průměrná hodnota řešení se snižuje, ale snižuje se také diverzita řešení a zvyšuje dominance ve feromonových stopách. Přibližně v 80. iteraci je již dominance příliš vysoká a je tedy aktivován proces narušování feromonových struktur. Po absolvování tohoto procesu se již minimální řešení nemusejí pohybovat v blízkosti nejlepšího nalezeného řešení. Přesto je však toto řešení v určitých intervalech překonáváno prací pomocných mravenců, kteří jej modifikují s přihlédnutím k rozložení feromonu.

5.5.4 Časová složitost

Nejdříve se zaměříme na časové nároky algoritmu vzhledem k velikosti instance problému. Testy probíhaly na zadáních o velikosti 10 až 150 (stejné instance jako u Ant System, viz 5.1.5) a počet iterací algoritmu byl nastaven na $t = 500$. Výsledky zobrazuje graf na obrázku 5.28 vlevo.



Obrázek 5.28: Vlevo závislost doby výpočtu na velikosti úlohy QAP. Vpravo závislost doby výpočtu na počtu umělých mravenců.

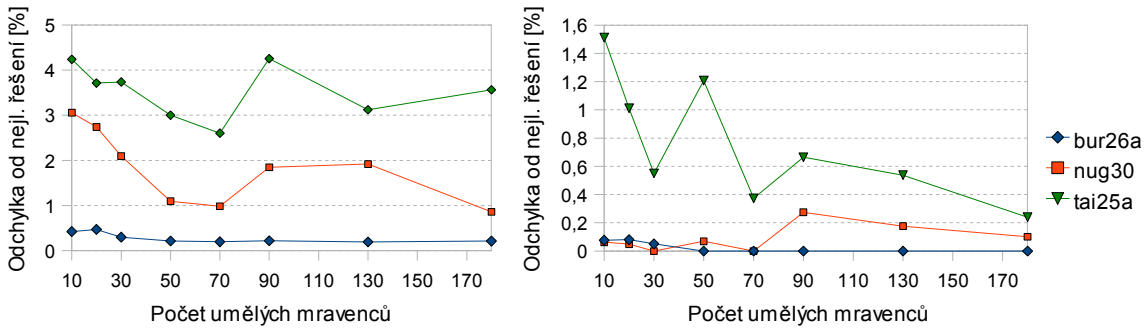
Časová závislost je, stejně jako u Ant System, kvadratická pro základní verzi a kubická pro verzi s lokálním prohledáváním. V absolutní míře je ovšem algoritmus náročnější a řešení stejně velké instance zabere delší dobu.

Závislost doby výpočtu na počtu umělých mravenců byla testována na řešení instance `tai40a` při 5, 10, 50, 100, 500 a 1000 agentech. Zbývající parametry byly nastaveny stejně jako u minulého experimentu. Zjištěné výsledky jsou zaneseny v grafu na obrázku 5.28 vpravo. Časová závislost je v obou případech lineární, lze ale vidět vyšší náročnost verze s lokálním prohledáváním. Časy jsou opět obecně vyšší než u Ant System.

Naměřené časy jsou také uvedeny v tabulce A.46. Jde o průměrná data ze tří běhů algoritmu, pouze data pro vysoké hodnoty n nebo m byla získána odhadem.

5.5.5 Vliv počtu mravenců

Tyto testy zjišťovaly vliv velikosti skupiny jednotek m, a to na instancích `bur26a`, `tai25a` a `nug30`. Počet jednotek byl nastavován na 10 až 180. Oproti výchozímu nastavení platilo pro verzi s lokálním prohledáváním $t = 400$. Výsledná data byla zanesena do tabulky A.47 (základní verze) a A.48 (verze s lok. prohledáváním) a jsou navíc ilustrována obrázkem 5.29.



Obrázek 5.29: Závislost kvality řešení na počtu umělých mravenců (vlevo bez lokálního prohledávání, vpravo s lokálním prohledáváním).

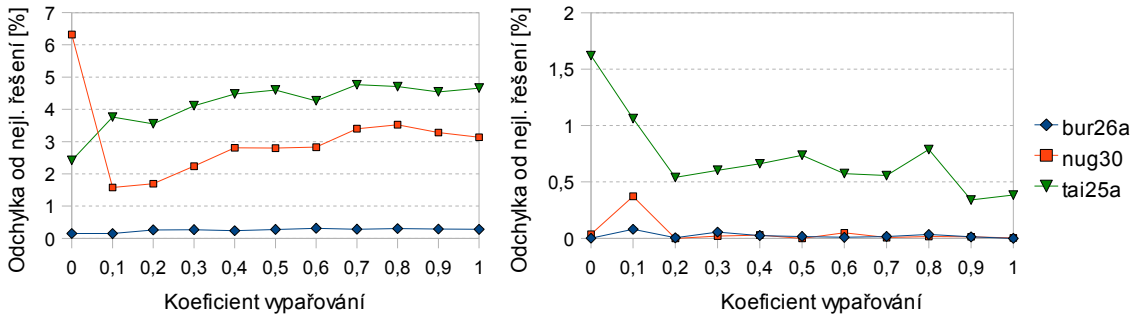
Průběhy pro základní variantu nejsou tak pravidelné jako u předcházejících metod. U všech instancí lze pozorovat zlepšování řešení při zvyšování m až do počtu 70 agentů, další zvyšování přineslo nepravidelné výsledky. Pouze u `bur26a` byla řešení při vyšších hodnotách m poměrně vyrovnaná. V případě instance `nug30` byly nalezeny nejlepší průměrné výsledky až při nejvyšší hodnotě $m = 180$. Tento jev by však, pro vyloučení odlehlé hodnoty, musela potvrdit další měření. U nízkého počtu mravenců se hodnoty řešení pohybovaly dále od hodnoty nejlepšího nalezeného řešení, ale také se méně často objevovala příliš vysoká dominance ve feromonových strukturách.

Pokud jde o verzi s lokálním prohledáváním, zde jsou výsledky také poměrně nepravidelné. U instance `bur26a` však lze vidět stejnou závislost jako u základní verze algoritmu a nastavení $m = 50$ a větší zajišťovalo dokonalou funkci algoritmu. Náhodná instance `nug30` byla dokonale řešena při $m = 30$ a $m = 70$ a vyšší počty agentů znamenaly horší řešení, která se ale při dalším zvyšování m zlepšovala. Výsledky měření na náhodné instanci jiného typu `tai25a` jsou do $m = 90$ velmi nevyrovnané. Poté je již zřetelný trend zlepšování řešení při rozšiřování skupiny jednotek.

Celkově se na zjištěné poznatky zřejmě nelze příliš spolehnout a přesnější vliv počtu jednotek by bylo vhodné zkoumat pomocí dalších měření.

5.5.6 Vypařování feromonu

Další experimenty se zaměřily na vliv vypařování feromonu, jehož síla je dána hodnotou parametru ρ. Parametr byla nastavován na hodnoty 0 až 1 s krokem 0,1 a měření na verzi s lokálním prohledáváním bylo prováděno oproti výchozímu nastavení při $t = 400$. Výsledná data pro základní a vylepšenou verzi jsou uvedena v tabulkách A.49 a A.50 a zobrazena v grafu 5.30.



Obrázek 5.30: Závislost kvality řešení na koeficientu vypařování (vlevo bez lokálního prohledávání, vpravo s lokálním prohledáváním).

Základní verze Genetic Ant System je, jak lze vidět v grafu na obrázku 5.30 vlevo, zřetelně ovlivněna nastavením míry vypařování. Se zvyšující se hodnotou ρ byla v rámci všech instancí nacházena horší řešení. Pro instance `bur26a` a `tai25a` bylo dokonce nejvýhodnější nulové vypařování, ovšem u instance `nug30` mělo toto nastavení výrazně záporný dopad. Nižší míra vypařování obecně přinesla dříve nalézaná nejlepší řešení v rámci běhu a také vyrovnanější výsledky opakovaných běhů. Malé vypařování znamenalo i vyšší míru dominance ve feromonových stopách.

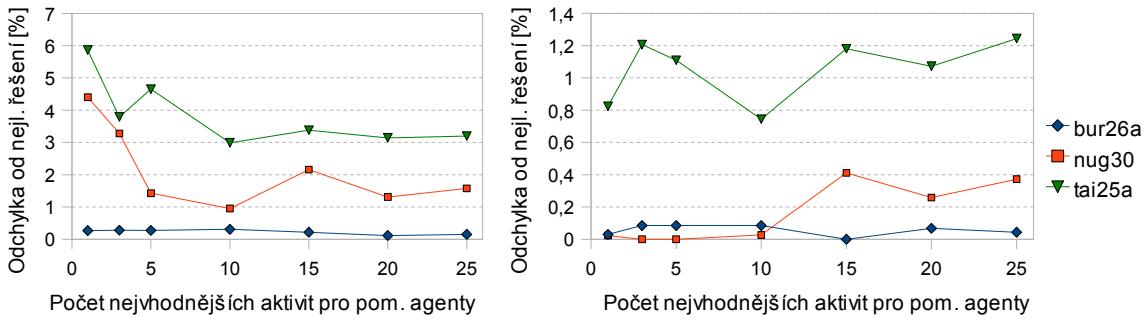
Verze s lokálním prohledáváním opět přiblížila hodnoty řešení optimálním hodnotám, v některých případech jich i dosáhla. Instance `tai25a` byla nejlépe řešena pro silné vypařování ($\rho = 0,9 - 1$). Hodnoty pro ostatní instance jsou již velmi nízké a průběh není pravidelný, ale uvedené nastavení lze opět doporučit. I když nejsou průběhy tak zřetelné, je zajímavé, že bylo dosaženo nízkých odchylek pro opačné hodnoty ρ než u základní verze algoritmu. Ostatní ukazatele jsou poměrně vyrovnané, pouze rozložení feromonu po procesu vymazání nejpoužívanějších cest je pro nízké hodnoty ρ vyrovnanější.

5.5.7 Počet nejvhodnějších aktivit pro pomocné agenty

Předmětem testů byl také parametr n_{best}, který slouží k nastavení počtu nejlepších aktivit, mezi kterými vybírá pomocný agent. Parametr nabýval hodnot 1 až 25. Výsledná data jsou uvedena v tabulkách A.51 a A.52 a byla také zanesena do grafů na obrázku 5.31.

Průběh odchylek je u obou náhodných instancí `nug30` a `tai25a` podobný a obě byly nejlépe řešeny při nastavení $n_{best} = 10$. Větší, ale zejména menší hodnoty n_{best} vedly k horším výsledkům. Reálná instance `bur26a` preferovala ještě vyšší počet nejlepších aktivit $n_{best} = 20$. Toto nastavení navíc oproti nízkým hodnotám parametru vedlo k dřívějšímu nalézání nejlepšího řešení běhu algoritmu.

Pokud jde o algoritmus s lokálním prohledáváním, lze vidět podobnost v průbězích odchylek pro náhodné instance v intervalu hodnot parametru 10 až 25. Rozdíly mezi výsledky

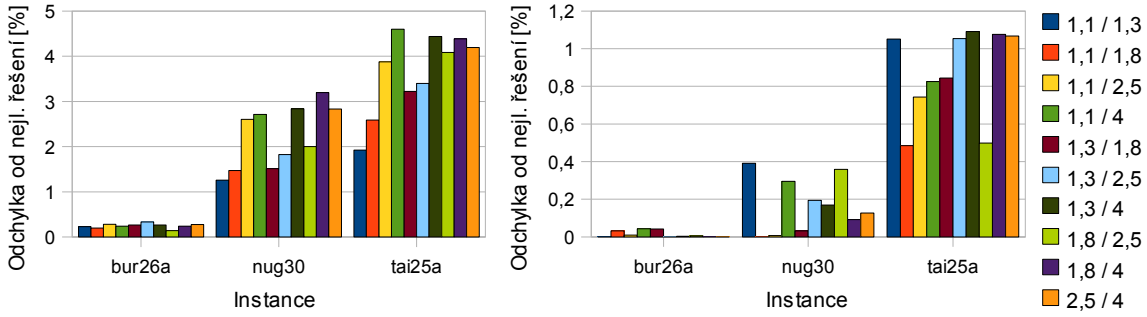


Obrázek 5.31: Závislost kvality řešení na počtu nejlepších aktivit pro pomocné mravence (vlevo bez lokálního prohledávání, vpravo s lokálním prohledáváním).

však již nejsou velké a nejlepší nastavení n_{best} se u různých instancí lišilo. Opět lze ale pro náhodné instance doporučit $n_{best} = 10$.

5.5.8 Hranice λ-faktoru větvení

Nyní se zaměříme na spodní (λ_{min}) a horní (λ_{max}) hranice průměrného λ-faktoru větvení, podle kterých je řízen proces narušování dominance feromonových struktur. Obě hranice byly nastavovány v rozsahu 1,1 až 4 s tím, že byly vyzkoušeny všechny kombinace, kde je spodní hranice nižší než horní. Výsledky testů jsou uvedeny v tabulkách A.53 a také ilustrovány pomocí grafů na obrázku 5.32.



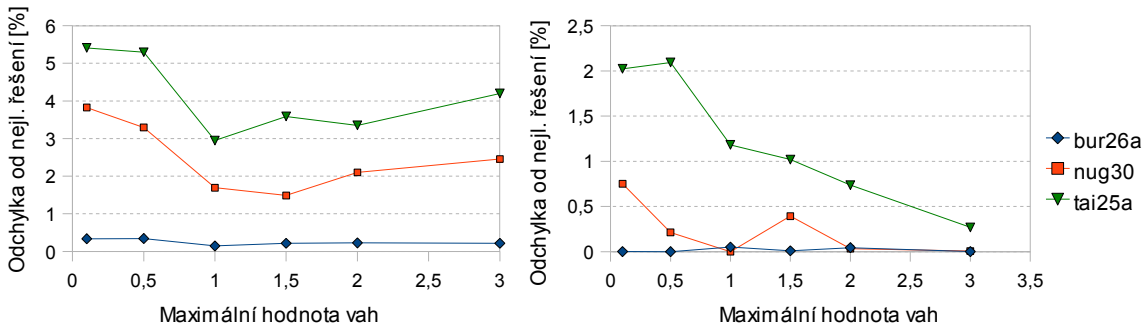
Obrázek 5.32: Závislost kvality řešení na hranici průměrného λ-faktoru větvení (vlevo pro základní verzi, vpravo pro verzi s lokál. prohledáváním). Notace legendy grafu: λ_{min} / λ_{max}.

U základní verze algoritmu je vliv obou parametrů zřetelně vidět v rámci náhodných instancí `nug30` a `tai25a`. Pro konstantní spodní hranici vede zvyšování horní hranice k degradaci nalézaných řešení. Kvalita řešení se také snižuje při zvyšování spodní hranice. Výhodná je tedy nízká spodní hranice a malý rozsah hranic, což odpovídá kombinaci $\lambda_{min} = 1,1$ a $\lambda_{max} = 1,3$. U reálné instance `bur26a` byly rozdíly mezi odchylkami malé a nebyl zde nalezen tak jednoznačný vztah. Nejlepší výsledky přineslo nastavení $\lambda_{min} = 1,8$ a $\lambda_{max} = 2,5$. Vyšší hodnoty horní hranice vedly obecně k menší míře dominance ve feromonových strukturách po provedení procesu jejich narušování. Při vyšších hodnotách spodní hranice je rozložení feromonu rovnoměrnější, protože je častěji spouštěn proces narušení dominance.

Lokální prohledávání uvedený vliv hodnot hranic víceméně odstraňuje, pouze u instance `tai25a` lze částečně rozpoznat podobný vztah k hodnotám řešení, jako v základní verzi. Nejlepší výsledky zde přineslo nastavení $\lambda_{min} = 1,1$ a $\lambda_{max} = 1,8$, stejně tak v případě `nug30`. Reálná instance `bur26a` byla dokonale řešena při více různých kombinacích hranic.

5.5.9 Maximální hodnoty vah

Následující testy se zaměřily na parametr w_{max}, který nastavuje povolený rozsah vah α a β na 0,1 až w_{max}. Hodnoty parametru byly nastavovány na hodnoty 0,1 až 3. Výsledky měření lze nalézt v tabulkách A.57 a A.58 a jsou také zobrazeny v grafu 5.33.



Obrázek 5.33: Závislost kvality řešení na maximální hodnotě vah (vlevo pro základní verzi, vpravo pro verzi s lokálním prohledáváním).

Pro základní verzi algoritmu nebyla výhodná ani příliš nízká, ani příliš vysoká hranice vah. Napříč testovanými instancemi se osvědčila hodnota $w_{max} = 1$, která byla také nejvhodnější pro `bur26a` a `tai25a`. Pro `nug30` přinesla nejlepší výsledek nejbližší vyšší testovaná hodnota $w_{max} = 1,5$. Při nastavení příliš malé maximální hodnoty vah jsou v nežádoucí míře stírány rozdíly mezi předpokládanými kvalitami přiřazení. Průměrná hodnota řešení je také za běhu výpočtu prakticky neměnná a vylepšení provádějí především pomocné jednotky. Aplikováním procesu narušování feromonových struktur přitom kolísá rozložení z dominantního do naprosto rovnoměrného. U nejvýhodnější hodnoty parametru je naopak velký počet relevantních přiřazení na začátku algoritmu a později se průměrný počet relevantních dílčích přiřazení ustálí na nízké hodnotě. Minimální hodnoty řešení se také pravidelně vracejí k nejlepší nalezené hodnotě.

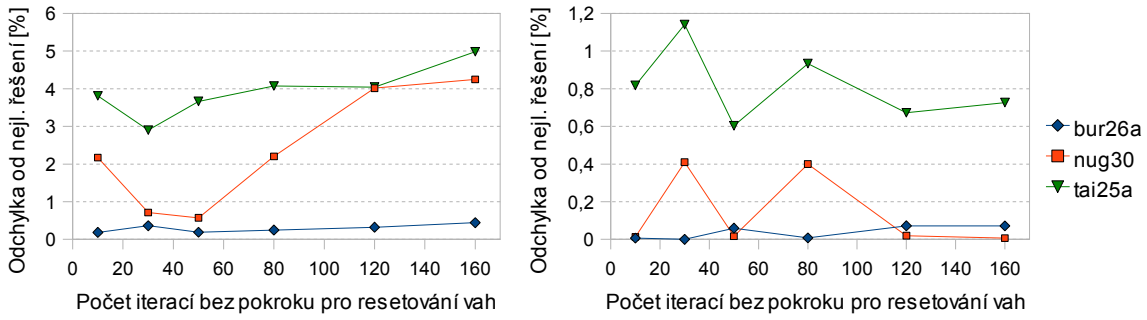
Verze s lokálním prohledáváním přinesla odlišné výsledky. Jednoznačný trend lze zaznamenat u náhodné instance `tai25a`, kde se s rostoucí hodnotou parametru zvyšuje kvalita nalézaných řešení a nejlepší hodnotou parametru je proto $w_{max} = 3$. Toto nastavení je výhodné i pro další instance, ačkoliv `nug30` mělo lepší (dokonalý) výsledek pro $w_{max} = 1$. Reálná instance `bur26a` byla dokonale řešitelná při více různých nastavení parametru.

5.5.10 Další parametry

Počet iterací bez pokroku pro resetování vah

Parametr t_{res} představuje počet iterací bez překonání nejlepšího nalezeného řešení, po kterém je provedeno náhodné naplnění vah mravenců, ale navíc jsou v polovině iterací t_{res}

do populace přidány pomocné jednotky. Parametr byl nastavován v rozsahu 10 až 160. Naměřené výsledky zobrazuje graf 5.34 a jsou zaneseny v tabulkách A.59 a A.60.



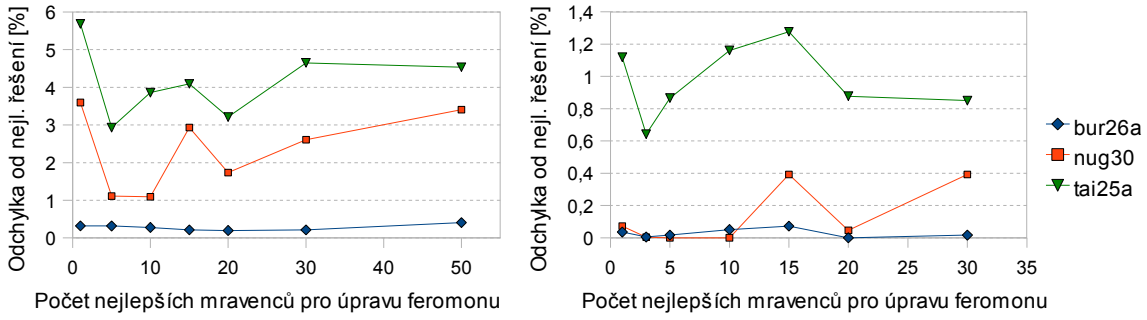
Obrázek 5.34: Závislost kvality řešení na počtu iterací pro resetování vah (vlevo pro základní verzi, vpravo pro verzi s lokálním prohledáváním).

Přibližně od hranice 50 iterací znamenalo pro základní verzi algoritmu zvyšování počtu iterací t_{res} degradaci kvality nalézaných řešení. Různé instance byly nejlépe řešeny při různém nastavení t_{res}: `bur26a` při nejnižších 10 iteracích, `nug30` při 50 iteracích a `tai25a` při 30 iteracích. Nízké hodnoty parametru vedly k odstupu minimálních řešení od nejlepšího nalezeného řešení a také k malé diverzitě skupiny řešení.

Vylepšená verze přinesla poměrně rozkolísané hodnoty pro náhodné instance `tai25a` a `nug30`, ale průběh odchylek je přitom pro různé hodnoty parametru velmi podobný. Instance `nug30` byla například dobře řešena při hodnotách parametru 10 a 160. Aproximace regresí přitom ukazuje zlepšování kvality řešení při zvyšování hodnoty parametru. Bez pokročilých řešení tedy nelze s jistotou stanovit jeho vliv. U reálné instance `bur26a` byl vztah viditelnější a nejlépe se osvědčilo nastavení $t_{res} = 30$.

Počet nejlepších mravenců pro úpravu feromonu

Dalším testovaným parametrem byl m_{best}. Nejnižší testovaná hodnota byla 1, nejvyšší 50 pro základní verzi a 30 pro verzi s lokálním prohledáváním. Naměřené výsledky jsou uvedeny v tabulkách A.61 a A.62 a ilustrovány obrázkem 5.35.



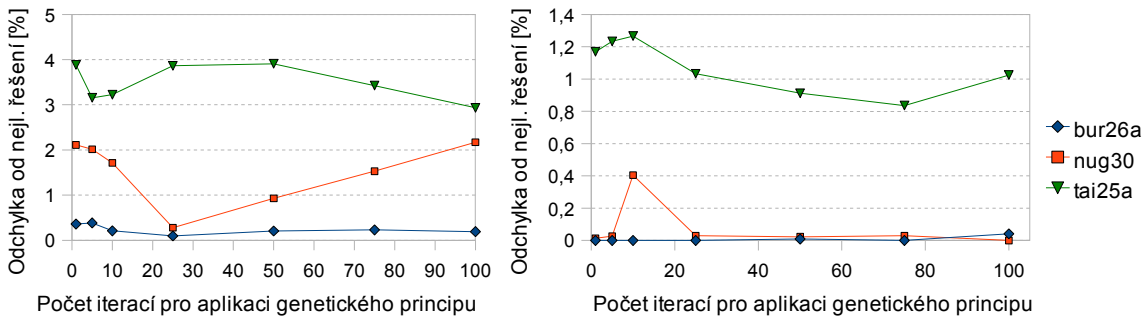
Obrázek 5.35: Závislost kvality řešení na počtu nejlepších mravenců pro úpravu feromonu (vlevo pro základní verzi, vpravo pro verzi s lokálním prohledáváním).

Měření na verzi bez lokálního prohledávání přineslo podobné výsledky pro náhodné instance `tai25a` a `nug30`. Přínosné zde byly nízké hodnoty m_{best} (kromě nejnižší), a obě instance tak byly dobře řešeny při $m_{best} = 5$. Reálná instance `bur26a` preferovala vyšší počet nejlepších mravenců $m_{best} = 20$, menší i větší hodnoty degradovaly kvalitu řešení. Vyšší hodnoty m_{best} vedly obecně k rovnoměrnějším rozložením feromonu, protože jej upravovalo více jednotek s různými řešeními.

U verze s lokálním prohledáváním jsou průběhy odchylek nepravidelné, ale v rámci všech instancí relativně podobné – např. hodnota $m_{best} = 15$ vedla vždy k nejméně kvalitním řešením. Pro náhodné instance se opět osvědčily nízké hodnoty parametru (kromě nejnižší). Reálná instance potvrzuje dobrá řešení při nízké hodnotě parametru, ale dokonale byla instance řešena při $m_{best} = 20$, což byla relativně dobrá volba i pro náhodné instance.

Frekvence aplikace genetického přístupu

Parametrem t_{gen} lze nastavit, po kolika iteracích budou periodicky pomocí genetického algoritmu přehodnoceny váhy spjaté s agenty. Počet těchto iterací byl nastavován v rozsahu 1 až 100 a výsledky byly zaneseny do tabulek A.55 a A.56. Data jsou také ilustrována grafem na obrázku 5.36.



Obrázek 5.36: Závislost kvality řešení na počtu iterací pro aplikaci genetického principu (vlevo pro základní verzi, vpravo pro verzi s lokálním prohledáváním).

U verze bez lokálního prohledávání se osvědčila stejná hodnota parametru pro instance `bur26a` a `nug30`, a to $t_{gen} = 25$. Okolní hodnoty vedly k významně horším řešením. Pouze náhodná instance `tai25a` měla dobré výsledky pro $t_{gen} = 5$ a nejlepší pro $t_{gen} = 100$. Pokud zde nepřevážil vliv odlehlých hodnot, mohla by ještě menší četnost aplikace genetického principu znamenat zvýšení kvality řešení. Lokální prohledávání opět významně zasahuje do chování algoritmu. Nejlepší výsledek na instanci `tai25a` byl získán při $t_{gen} = 75$. Pro `nug30` byla vhodná nejnižší hodnota parametru, zatímco `bur26a` byla dokonale řešena při většině, ale především malých hodnot t_{gen}.

5.6 Porovnání algoritmů

5.6.1 Popis testů a parametry algoritmů

Závěrečná sada experimentů spočívala v časově omezeném (5 minut) porovnání výkonů algoritmů. Předmětem porovnání byly všechny algoritmy z této kapitoly 5 a pro zastoupení klasické umělé inteligence bylo vybráno simulované žíhání a genetický algoritmus. Testovány

byly instance různých velikostí a typů. `bur26a` a `ste36a` zastupují reálné problémy, `tai25b` a `tai50b` jsou pseudoreálné problémy, `tai25a` a `tai50a` patří mezi náhodné instance a `nug30` a `sko56` jsou náhodné instance se vzdálenostmi na mřížce. Běh algoritmu byl pro další zprůměrování výsledků vždy 20krát zopakován.

Hodnoty parametrů pro algoritmy rojové inteligence přesně odpovídaly poznatkům z provedených testů a lišily se v závislosti na velikosti a typu konkrétní instance. Vliv některých parametrů nebyl testován pro náhodné instance se vzdálenostmi na mřížce – v takovém případě bylo nastavení stejné jako u běžných náhodných instancí. Podobně byla převzata některá nastavení pro pseudoreálné instance ze závěrů pro reálné instance. I počet umělých mravenců byl nastavován individuálně pro každý test. Pro instanci o velikosti $n = 26$ to bylo například $m = 500$ až $m = 2000$ (podle konkrétního algoritmu) a pro algoritmy s lokálním prohledáváním $m = 40$ až $m = 300$.

Simulovanému žíhání byla nastavována počáteční teplota na 5000, koncová teplota na 0, koeficient pro snižování teploty na 0,98 a počet modifikací pro jednu hodnotu teploty tak, aby byl dobře využit časový limit pro celý teplotní průběh algoritmu.

Genetický algoritmus pracoval s pravděpodobností mutace 0,9, pravděpodobností křížení 0,1, mírou obnovy populace 10%, ruletovým výběrem jedinců ke křížení s přepočítanými fitness podle pořadí a elitismem jako strategií obnovy populace. Velikost populace byla vhodně nastavována, aby genetický algoritmus vždy absolvoval dostatečný počet iterací (pro instanci o velikosti $n = 25$ šlo například o 1000 jedinců v populaci). Počet jedinců ke křížení byl tvořen polovinou počtu všech jedinců.

5.6.2 Výsledky porovnání

Naměřená data jsou uvedena v tabulce na obrázku 5.37. Symbol δ označuje průměrnou odchylku od optimálního řešení, n_{opt} je počet běhů, ve kterých bylo nalezeno optimální řešení, a t^* je průměrný počet iterací, po kterých bylo nalezeno nejlepší řešení běhu algoritmu. V záhlaví sloupců jsou názvy instancí, přičemž číslo obsažené v názvu instance odpovídá její velikosti n. Nejlepší výsledek pro každou instanci je označen kurzívou a modrou barvou. Pro některé instance není k dispozici optimální řešení, a bylo proto použito nejlepší známé řešení z [3]. Jedná se o `tai50b` (458821517), `sko56` (34458) a `tai50a` (4938796).

Základní algoritmus *Ant System* se příliš neosvědčil a jeho výsledky patřily napříč instancemi mezi nejhorší. Verze s lokálním prohledáváním již byla úspěšnější a překonávala simulované žíhání a v některých případech i genetický algoritmus. V porovnání s ostatními algoritmy rojové inteligence s lokálním prohledáváním však nejde o příliš dobré výsledky.

Zajímavé je, že blízké výsledky přináší i *Ant Colony System*, který obsahuje některá vylepšení oproti Ant System. Lepší byl pouze u poloviny instancí a jeho výkon obecně také není příliš dobrý. Verze s lokálním prohledáváním přinesla téměř vždy ještě horší výsledky než odpovídající verze Ant System. Přesto byly lepší než u simulovaného žíhání a v polovině případů překonaly i genetický algoritmus.

Základní verze *Max-Min Ant System* dokázala v porovnání s předcházejícími metodami instance řešit s někdy až řádově lepšími výsledky. Dokázala také ve všech případech překonat výsledky simulovaného žíhání. Přidané lokální prohledávání již umožnilo nalézání řešení s velmi malými odchylkami od optimálních řešení a i zde MMAS jednoznačně vítězil nad předchozími algoritmy. I genetický algoritmus našel u všech instancí horší řešení.

Verze *Hybrid Ant System* bez lokálního prohledávání konkurovala zejména předchozí metodě MMAS a byla lepší v polovině případů. Opět je významně lepší než simulované žíhání. Její verze s lokálním prohledáváním dokázala řešit instance tak dobře, že ji můžeme

		bur26a	ste36a	tai25b	tai50b	nug30	sko56	tai25a	tai50a
AS	δ	1,64%	50,44%	17,65%	36,31%	11,69%	14,08%	10,15%	13,54%
	n_{opt} | t^*	0 | 278	0 | 244	0 | 271	0 | 278	0 | 270	0 | 333	0 | 312	0 | 396
AS + l. p.	δ	*0,00%*	2,36%	0,07%	0,69%	0,19%	0,57%	1,10%	3,74%
	n_{opt} | t^*	20 | 63	0 | 97	0 | 154	0 | 101	9 | 128	0 | 127	0 | 181	0 | 98
ACS	δ	1,66%	43,27%	24,07%	37,20%	9,43%	12,70%	10,80%	13,38%
	n_{opt} | t^*	0 | 222	0 | 157	0 | 246	0 | 212	0 | 185	0 | 241	0 | 206	0 | 212
ACS + l. p.	δ	0,01%	3,00%	0,11%	0,99%	0,24%	0,92%	1,41%	3,70%
	n_{opt} | t^*	9 | 32	0 | 16	0 | 22	.0 | 17	2 | 18	0 | 19	0 | 65	0 | 24
MMAS	δ	0,11%	5,71%	0,37%	3,45%	1,87%	2,73%	3,69%	4,43%
	n_{opt} | t^*	1 | 538	0 | 479	0 | 639	0 | 485	0 | 696	0 | 473	0 | 624	0 | 489
MMAS + l. p.	δ	0,001%	0,22%	*0,00%*	0,38%	0,03%	*0,16%*	0,47%	1,90%
	n_{opt} | t^*	19 | 59	3 | 62	20 | 40	0 | 66	12 | 44	4 | 46	5 | 171	0 | 45
HAS	δ	0,08%	7,42%	0,18%	9,60%	1,40%	4,88%	2,68%	7,16%
	n_{opt} | t^*	6 | 407	0 | 306	0 | 403	0 | 144	0 | 493	0 | 202	0 | 490	0 | 246
HAS + l. p.	δ	*0,00%*	*0,17%*	*0,00%*	*0,15%*	*0,01%*	0,21%	*0,14%*	1,78%
	n_{opt} | t^*	20 | 4	6 | 39	20 | 7	0 | 11	18 | 50	0 | 48	11 | 108	0 | 60
GAS	δ	0,23%	8,19%	1,70%	12,82%	1,82%	3,83%	3,18%	4,37%
	n_{opt} | t^*	0 | 94	0 | 147	0 | 103	0 | 103	0 | 156	0 | 163	0 | 116	0 | 141
GAS + l. p.	δ	*0,00%*	0,48%	*0,00%*	0,39%	0,01%	0,27%	0,50%	*1,61%*
	n_{opt} | t^*	20 | 41	1 | 56	20 | 54	0 | 41	18 | 51	0 | 46	3 | 98	0 | 49
SA	δ	0,34%	16,17%	35,32%	19,61%	4,66%	5,25%	5,24%	8,15%
	n_{opt} | t^*	0 | 98	0 | 252	0 | 10	0 | 7	0 | 299	0 | 313	0 | 24	0 | 26
GA	δ	0,08%	0,92%	0,003%	0,54%	0,47%	1,02%	1,74%	3,13%
	n_{opt} | t^*	2 | 1406	1 | 2575	0 | 1286	0 | 4252	2 | 1849	0 | 4256	0 | 1470	0 | 3895

Obrázek 5.37: Porovnání výkonů algoritmů.

označit jako nejlepší metodu z celého testu. V osmi případech z deseti přinesla vůbec nejnižší odchylky od optimálních řešení a i ve zbývajících dvou jsou její výsledky velmi dobré.

V této práci navržená metoda *Genetic Ant System* byla v základní verzi vždy lepší než základní verze Ant System a Ant Colony System. Překonala také simulované žíhání, ale ne genetický algoritmus. Reálné instance řeší lépe předchozí metody HAS a MMAS, ale u náhodných instancí lze výkony GAS vždy zařadit mezi výsledky těchto metod. Verze s lokálním prohledáváním se svými výkony blíží odpovídající verzi metody MMAS, v některých případech ji i překonává. Ve třech případech přinesla nejlepší řešení dané instance, někdy ve shodě s jinými metodami.

Pokud jde o klasické metody, kvalitní řešení dokázal nalézat především *genetický algoritmus*. Jeho výsledky byly lepší než výsledky základních verzí algoritmů založených na simulovaných mravenčích koloniích. Verze s lokálním prohledáváním některých z těchto algoritmů ale genetický algoritmus dokázaly překonat. Výsledky *simulovaného žíhání* byly vždy horší než u genetického algoritmu a tento algoritmus tak zvítězil pouze nad základními verzemi horších algoritmů s umělými mravenci.

Dodatečná měření s limitem 3 minut (a upravenými parametry) potvrdila nejlepší výsledky algoritmu Hybrid Ant System. Kratší doba výpočtu měla negativní vliv zejména na genetický algoritmus. Naopak Max-Min Ant System a Ant Colony System, obojí bez lokálního prohledávání, nebyly zkrácením času výrazně ovlivněny a dosáhly podobných výsledků jako při předchozím měření.

Kapitola 6

Aplikace pro provádění experimentů

6.1 Popis aplikace

6.1.1 Úvodní informace

V rámci této diplomové práce byla vytvořena aplikace, s jejíž pomocí byly provedeny veškeré praktické testy a sběr jejich výsledků. Aplikace integruje řešení Travelling Salesman Problem a Quadratic Assignment Problem, a to pomocí různých metod. Na obrázku B.1 je zobrazeno hlavní okno programu s otevřeným TSP projektem.

Tato část práce uvádí pouze základní informace o testovacím programu. Práci s uživatelským prostředím, přípravu testů a další záležitosti přesněji popisuje uživatelský manuál, který je umístěn v samostatném dokumentu.

Program pro spuštění vyžaduje minimálně Java Runtime Environment verze 1.6, který je umístěn na přiloženém CD (viz přílohu C). Samotné spuštění lze provést pomocí příkazu

```
java -jar ACO.jar
```

Vzhledem k míře využití paměti je vhodné vyhradit při spuštění programu větší paměť například takto (uvedený příkaz vyhradí 500MB paměti):

```
java -Xmx500000000 -jar ACO.jar
```

6.1.2 Projekty

Aplikace pracuje s tzv. projekty, které obsahují zadání dané úlohy. Projekty mohou být různých typů. Každý typ je dán řešenou úlohou, tedy TSP nebo QAP, a způsobem reprezentace. Reprezentace problému může být pro obě úlohy grafová nebo tabulková, a lze tak pracovat s celkem čtyřmi druhy projektů. V rámci grafového vyjádření TSP program umožňuje vytvořit graf, kde uzly odpovídají městům a ohodnocení hran představuje vzdálenosti mezi městy. Graf je možné editovat, různě upravovat jeho zobrazení i jej doplňovat na kompletní včetně výpočtu ohodnocení hran. Tabulková reprezentace staví na jednoduché tabulce reálných čísel, která představuje vzdálenosti mezi městy. Pro QAP je situace odlišná jen tím, že se využívá dvou tabulek nebo grafů – definují se tak vzdálenosti mezi lokacemi a toky mezi aktivitami.

Lze vytvářet nové prázdné projekty nebo je načítat z XML souborů a také je ukládat. Výsledný soubor obsahuje typ projektu a zadání příslušného problému. Projekty lze

také vytvářet importem ze souborů, jejichž formát odpovídá formátu souborů s instancemi umístěnými v knihovnách TSPLIB a QAPLIB.

Program také umožňuje přepínání mezi základními dvěma módy. Mód *interactive* slouží k plnohodnotné práci s projektem, tedy k úpravě projektu formou editace grafu nebo tabulky. Druhým módem je *simple*, který uvedené změny neumožňuje a místo grafů a tabulek zobrazuje jen stručné informace o jejich velikosti. Mód *simple* slouží především pro práci s velmi velkými instancemi, u nichž vykreslování grafu (především) nebo tabulky může nepříjemně zdržovat práci s programem.

6.1.3 Metody

Zadání úloh, které jsou součástmi projektů, program umí řešit různými metodami. Protože se práce zaměřuje především na úlohu QAP, je tato řešitelná více různými metodami. Tabulka 6.1 ukazuje, které metody je možné použít pro různé typy úloh.

Metoda	TSP	QAP
Ant System	√	√
Ant Colony System	×	√
Hybrid Ant System	×	√
Max–Min Ant System	×	√
Genetic Ant System	×	√
simulované žíhání	√	√
genetický algoritmus	√	√

Tabulka 6.1: Podporované metody

Obrázek B.2 ukazuje jedno z oken pro řešení úlohy vybranou metodou. Na levé straně okna je oddíl pro nastavení parametrů vybrané metody, dále tlačítka pro ovládání procesu řešení úlohy a také informace o průběhu řešení. Volitelně zobrazitelná pravá část ukazuje aktuálně nejlepší řešení úlohy a přepínatelné grafy, které přehledně zobrazují různé průběhy (hodnot řešení, směrodatných odchylek atd.).

6.1.4 Provádění experimentů

Aplikace byla vytvořena především pro experimentování s vestavěnými metodami, a podporuje tak různými způsoby jejich zkoumání. Prvotní náhled na chování metody lze získat nastavováním různých hodnot parametrů a spouštěním procesu řešení dané úlohy. Po ukončení chodu metody získáme nejlepší nalezené řešení, ale můžeme také za běhu pozorovat její chování v grafech.

Testy jsou často prováděny opakovaně a výsledky jsou průměrovány, aby měly získané poznatky vyšší vypovídající hodnotu. Program tento přístup podporuje a umožňuje opakovat běhy algoritmu při nastavených parametrech. Výsledky jsou následně uloženy do zvoleného adresáře. Obsahují ve formě tabulek všechny průběhy hodnot, které jsou zobrazované v grafech (hodnoty řešení, průměrného λ-faktoru větvení apod.), ale také závěrečný textový report. Toto shrnutí obsahuje název řešené úlohy a identifikaci použité metody, dále hodnoty parametrů a také dodatečné informace, jako v kolika běžích bylo nalezeno optimální řešení, jaká je výsledná průměrná odchylka od optimálního řešení, kolik času trval průměrně jeden běh atd.

Program ale pro testování metod nabízí ještě více automatizované prostředí. Pomocí speciálního parametru příkazové řádky jej lze spustit jako konzolovou aplikaci, jejímž vstupem je XML soubor s definicemi testů. Tento soubor obsahuje jak základní nutné údaje (např. cesta k souboru s projektem, adresář pro uložení výsledků), tak i jednotlivé sady parametrů, pro které mají experimenty probíhat. Popisovaný XML soubor umožňuje dopředu naplánovat testy na různých instancích, pomocí odlišných metod i s různými hodnotami parametrů.

6.2 Implementace

Aplikace byla vytvořena v programovacím jazyce Java (verze 1.6.0), konkrétně v integrovaném vývojovém prostředí NetBeans (verze 6.1RC). Volba tohoto jazyka zajišťuje přenositelnost programu a měl by tak být spustitelný na většině významných operačních systémů.

Jedinou externí knihovnou, která byla použita při vývoji programu, je *JFreeChart* pro vykreslování grafů, která je šířena pod licencí LGPL. Příslušný balík souborů je umístěn na přiloženém CD (viz přílohu C).

6.2.1 Popis balíčků

Balíček `files`

Tento balíček obsahuje jedinou třídu `TextFileOperations`, která slouží k načítání a ukládání textových souborů. Používá se především pro práci s XML soubory.

Balíček `gui`

Zde jsou umístěny ty zdrojové kódy, které se týkají uživatelského rozhraní aplikace. Nejdůležitější je podbalíček `forms`, ve kterém jsou umístěny třídy s definicemi oken. Nalezneme zde mimo jiné třídu `FMain`, která popisuje hlavní okno aplikace, ale také důležitou třídu `FCompute`, která tvoří základ všech výpočetních oken. Zajišťuje funkci hlavních ovládacích prvků, komunikuje s výpočetním vláknem, vypisuje základní informace o průběhu výpočtu, spravuje provádění testů apod. Od tohoto základního okna jsou odvozena všechna výpočetní okna, například `FQapByAcs` pro řešení QAP pomocí Ant Colony System apod.

Balíček `methods`

V tomto balíčku jsou umístěny třídy, které obecně definují jednotlivé výpočetní metody. Systém byl navržen tak, že jsou metody abstrahovány od konkrétních řešených úloh. Každá metoda tedy sestává z vlastního popisu principu metody a rozhraní, kde jsou definovány operace, které metoda využívá. Toto rozhraní potom musí implementovat třída, která je již spojena s konkrétním řešeným problémem.

Základem je třída `AMethod`, ze které je odvozena každá třída popisující nějakou metodu. Tato základní třída udržuje nejlepší nalezená řešení, spravuje statistiky a vynucuje na podřízených třídách implementaci výpočetních metod tak, aby bylo později možné s třídami různých metod pracovat jednotným způsobem. Díky tomu bylo možné poměrně jednoduše vytvořit univerzální testovací systém pro příkazovou řádku. Základem všech rozhraní s hlavičkami operací výpočetních metod je rozhraní `IMethodOperations`.

Podbalíčky `aco`, `ga` a `sa` obsahují třídy a rozhraní pro definice výpočetních metod a také různé pomocné třídy (např. `PheromoneTable` pro udržování hladin feromonových stop). Jak

bylo nastíněno výše, každá metoda je v základu složena z hlavní třídy s popisem metody (např. `AntSystem` nebo `GeneticAlgorithm`) a rozhraní, které vynucuje implementace operací metody (např. `IAntSystemCompute` nebo `IGeneticAlgorithmCompute`).

Balíček `projects`

Jedná se o obsáhlý a důležitý balíček, který popisuje vše, co se týká správy projektů i jejich řešení. Možným typům projektů odpovídají podbalíčky `tsp` a `qap`. V nich najdeme třídy zastupující konkrétní typ projektu (např. `QapGraphProject`), jenž udržují zadání úlohy a dovedou je načítat ze souboru nebo do souboru ukládat. V podbalíčcích potom nalezneme třídy, které implementují rozhraní vybraných metod pro konkrétní úlohy. Třída `TspByGaCompute` tedy slouží k řešení TSP pomocí genetického algoritmu, třída `QapByAsCompute` (společně s definicí chování mravence v `QapByAsAnt`) definuje řešení QAP pomocí metody Ant System apod.

Balíček také obsahuje třídu pro výpočetní vlákna `CompThread`, pomocné třídy pro lokální prohledávání a další.

Balíček `stats`

Zde je umístěna třída `Statistics`, která slouží k ukládání naměřených hodnot a výpočtu statistických dat, a třída `Reporter` pro zpracování a ukládání výsledků testů.

Balíček `structs`

Tento balíček obsahuje definice dvou základních struktur pro uložení instance řešeného problému. Jedná se o třídy `Graph`, `Node` a další pro reprezentaci grafu a `Table` pro reprezentaci tabulky (matice) hodnot.

Balíček `tests`

Poslední balíček obsahuje třídy, které tvoří automatický testovací systém. Nastavení parametrů příslušného testu obsahuje třída `Test`. Testy jsou pak sdružovány a jsou k nim přidávány další údaje v rámci třídy `TestPack`. Jednotlivé výpočetní metody jsou spravovány třídou `TestMethod` a třída `TestMethodFactory` umožňuje vytváření jejich instancí podle zadání úlohy, typu problému atd. Samotné provádění testů konečně zajišťuje třída `TestSolver`.

Kapitola 7

Závěr

V této práci byly popsány základní principy rojové inteligence včetně nastínění biologického pozadí problematiky. Hlavní část práce je zaměřena na popis vybraných algoritmů pro řešení Travelling Salesman Problem a Quadratic Assignment Problem a zhodnocení provedených experimentů.

Pro řešení *Travelling Salesman Problem* byla vybrána pouze základní metoda *Ant System*. Na kvalitu nalézaných řešení měly vliv zejména váhy pro heuristickou informaci a informaci o množství feromonu a počet elitních agentů. Navíc se ukázalo, že pro symetrické instance TSP je vhodné použít symetrické struktury pro rozložení feromonu, a podobně pro asymetrické instance. Základní verze metody Ant System překonala obecně špatné výkony genetického algoritmu, ale simulované žíhání přineslo kvalitnější řešení. Výjimkou byly asymetrické instance, kde Ant System předvedl přesvědčivě nejlepší výkon. Ant System vylepšený o princip lokálního prohledávání nalezených řešení překonal všechny ostatní metody téměř na všech řešených instancích.

Quadratic Assignment Problem byl řešen více metodami. Algoritmus *Ant System* byl opět citlivý zejména na nastavení vah heuristické informace a feromonových hladin a počet elitních mravenců. *Ant Colony System* byl ovlivněn především vyvážením mezi prohledáváním prostoru řešení a využíváním informací, ale významný vliv měla také váha heuristické informace. Stejné vyvážení bylo důležité i pro *Max–Min Ant System* a *Hybrid Ant System*. Druhý uvedený algoritmus byl také významně ovlivněn počtem iterací do provedení diverzifikace a počtem výměn v rámci modifikace řešení. Vlastní metoda *Genetic Ant System*, která vznikla z Ant System přidáním genetických principů a dalších vylepšení, přinesla méně vyrovnané výsledky než ostatní metody. Na její výkon měly vliv téměř všechny parametry. Její nevýhodou je vyšší počet parametrů a také větší časové nároky. U většiny metod vedlo použití vyššího počtu agentů ke kvalitnějším řešením. Vhodné nastavení parametrů se někdy výrazně lišilo pro různé typy řešených instancí problému.

Instance Quadratic Assignment Problem byly dobře řešeny zejména algoritmy Max-Min Ant System a Hybrid Ant System, přičemž druhý uvedený algoritmus dosáhl ve verzi s lokálním prohledáváním vůbec nejlepších výsledků. Oba algoritmy ve vylepšené verzi dokázaly překonat simulované žíhání a i jinak velmi úspěšný genetický algoritmus. To platilo i pro Genetic Ant System, jehož výsledky se pohybují mezi dvěma zmíněnými nejlepšími algoritmy a relativně méně úspěšnými algoritmy Ant System a Ant Colony System.

Důležitým přínosem práce je zpracování popisu každé metody a ověření jeho správné funkce implementací. Textová část práce společně s příslušným programem tvoří základní přehled hlavních metod rojové inteligence pro kombinatorickou optimalizaci, se kterými je také možné experimentovat. Zajímavé poznatky přineslo i provedení a zhodnocení experi-

mentů na různých typech řešených instancí, a také závěrečné porovnání metod. V rámci práce byla navíc navržena vlastní metoda Genetic Ant System s integrací genetických přístupů a dalších mechanismů, které přispěly k poměrně dobré kvalitě nalézaných řešení. Tato metoda zahrnuje náměty na rozšíření stávajících metod, které vznikly během jejich zkoumání. Lze konstatovat, že se metody rojové inteligence (zejména některé vybrané) osvědčily a lze je doporučit k praktickému použití. Vzhledem k uvedeným přínosům byly splněny všechny body zadání.

Práce by dále mohla pokračovat provedením přesnějších experimentů pro parametry, jejichž vliv nemohl být podle stávajících testů jednoznačně určen. Bylo by také vhodné provést přesnější analýzu role genetického principu v Genetic Ant System a případně tuto metodu rozšířit o další prostředky pro dosažení lepších výsledků. Popsané metody by navíc mohly být použity i pro řešení odlišných problémů. Vzhledem k velmi dobrým výsledkům nejlepších algoritmů by také mělo smysl vyvinout aplikaci, která by pomocí metod rojové inteligence řešila některý konkrétní reálný problém (například rozvržení jednotlivých oddělení firmy v rámci komplexu budov).

Literatura

[1] AntOptima. [online], [cit. 2007-12-17].
URL http://www.antoptima.com/pdf/antoptima_en.pdf

[2] Bonabeau, E.; Dorigo, M.; Theraulaz, G.: *Swarm intelligence: From Natural to Artificial Systems*. Oxford University Press, 1999, ISBN 0-19-513159-2.

[3] Burkard, R. E.; Karisch, S. E.; Rendl, F.: QAPLIB — A Quadratic Assignment Problem Library. [online], [cit. 2007-12-15].
URL http://www.seas.upenn.edu/qaplib/

[4] Cook, B.: TSP Test Data. [online], [cit. 2007-12-15].
URL http://www.tsp.gatech.edu/data/index.html

[5] Dorigo, M.; Birattari, M.: Swarm Intelligence — Scholarpedia. 2007, [online], [cit. 2007-12-03].
URL http://www.scholarpedia.org/article/Swarm_Intelligence

[6] Gambardella, L.; Taillard, É.; Dorigo, M.: Ant Colonies for the QAP. 1998, [online], [cit. 2008-02-10].
URL http://ina2.eivd.ch/collaborateurs/etd/articles.dir/IDSIA-4-97.ps

[7] Hu, X.: Particle Swarm Optimization. 2006, [online], [cit. 2007-12-01].
URL http://www.swarmintelligence.org/

[8] Kapoun, J.: Marvin Minsky: Na cestě k umělé inteligenci. *Science World*, 2005, [online], [cit. 2007-12-05].
URL
http://www.scienceworld.cz/sw.nsf/ID/F659647FFB1E43B0C12570620050FE83

[9] Kennedy, J.; Eberhart, R. C.; Shi., Y.: *Swarm intelligence*. San Francisco: Morgan Kaufmann, 2001, ISBN 1-55860-595-9.

[10] Loiola, E. M.; de Abreu, N. M. M.; Boaventura-Netto, P. O.; aj.: An analytical survey for the quadratic assignment problem. 2004, [online], [cit. 2007-12-10].
URL http://www.seas.upenn.edu/qaplib/QAP_Survey_04.pdf

[11] Maniezzo, V.; Colorni, A.: The Ant System Applied to the Quadratic Assignment Problem. *Knowledge and Data Engineering*, ročník 11, č. 5, 1999: s. 769–778, [online], [cit. 2007-12-11].
URL http://www.csr.unibo.it/~maniezzo/papers/kde.ps

[12] Maniezzo, V.; Gambardella, L. M.; Luigi, F. D.: Ant Colony Optimization. In *New Optimization Techniques in Engineering*, editace G. C. Onwubolu; B. V. Babu, Springer-Verlag Berlin Heidelberg, 2004, [online], [cit. 2008-02-13].
URL http://www.cs.unibo.it/bison/publications/aco2004.pdf

[13] Miller, P.: Swarm Behavior. *National Geographic*, 2007, [online], [cit. 2007-12-12].
URL http://www7.nationalgeographic.com/ngm/0707/feature5/

[14] Němec, M.: *Optimalizace pomocí mravenčích kolonií*. Diplomová práce, ČVUT v Praze, Fakulta elektrotechnická, Praha, 2006, [online], [cit. 2008-02-10].
URL http://www.milosnemec.cz/download/dp.pdf

[15] Pilat, M. L.; White, T.: Using Genetic Algorithms to Optimize ACS-TSP. In *ANTS '02: Proceedings of the Third International Workshop on Ant Algorithms*, London, UK: Springer-Verlag, 2002, ISBN 3-540-44146-8, s. 282–287, [online], [cit. 2008-03-18].
URL http://www.scs.carleton.ca/~arpwhite/documents/ANTS2002.pdf

[16] Reinelt, G.: TSPLIB. [online], [cit. 2007-12-15].
URL http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/

[17] Schwarz, J.; Sekanina, L.: *Aplikované evoluční algoritmy (EVO) — studijní opora*. FIT VUT Brno, 2006.

[18] Stützle, T.: MAX-MIN Ant System for Quadratic Assignment Problems. Technická Zpráva AIDA–97–04, 1997, [online], [cit. 2008-02-15].
URL http://iridia.ulb.ac.be/~stuetzle/publications/AIDA-97-04.ps.gz

[19] Stützle, T.; Dorigo, M.: ACO algorithms for the quadratic assignment problem. 1999: s. 33–50, [online], [cit. 2008-02-12].
URL
http://www.intellektik.informatik.tu-darmstadt.de/Pubs/tom/NIO.ps.gz

[20] Stutzle, T.; Hoos, H.: Improvements on Ant-System: Introducing MAX-MIN Ant System. Technická Zpráva AIDA–96–12, 1996, [online], [cit. 2008-02-15].
URL http://www.cs.ubc.ca/~hoos/Publ/aida-96-12r.ps

[21] Stützle, T.; Hoos, H. H.: MAX-MIN Ant system. *Future Generation Computer Systems Journal*, ročník 16, č. 9, 2000: s. 889–914, ISSN 0167-739X, [online], [cit. 2008-02-15].
URL http://iridia.ulb.ac.be/~stuetzle/publications/FGCS.ps.gz

[22] Šlapal, J.: Minimální kostra — Přednášky do předmětu MAT na FIT VUT Brno. 2006, [online], [cit. 2007-12-01].
URL http://www.math.fme.vutbr.cz/download.aspx?id_file=1554

[23] Šlapal, J.: Obyčejné grafy — Přednášky do předmětu MAT na FIT VUT Brno. 2006, [online], [cit. 2007-12-01].
URL http://www.math.fme.vutbr.cz/download.aspx?id_file=1540

[24] Wikipedia: Quadratic assignment problem — Wikipedia, The Free Encyclopedia. 2007, [online], [cit. 2007-12-13].
URL http://en.wikipedia.org/wiki/Quadratic_assignment_problem

[25] Wikipedia: Self-organization — Wikipedia, The Free Encyclopedia. 2007, [online], [cit. 2007-12-11].
URL http://en.wikipedia.org/wiki/Self-organization

[26] Wikipedia: Travelling salesman problem — Wikipedia, The Free Encyclopedia. 2007, [online], [cit. 2007-12-13].
URL http://en.wikipedia.org/wiki/Traveling_salesman_problem

[27] Zbořil, F.: Genetické algoritmy — Přednášky do předmětu SFC na FIT VUT Brno. 2006.

[28] Zimmer, C.: From ants to people, an instinct to swarm. *International Herald Tribune*, 2007, [online], [cit. 2007-12-10].
URL http://www.iht.com/articles/2007/11/13/healthscience/13traff.php

Seznam příloh

Příloha A

Tabulky s výsledky testů

A.1 Travelling Salesman Problem

A.1.1 Ant System

	n						
	76	152	226	264	439	1002	1432
Čas [s]	1	7	14	19	57	290	623

	m					
	10	50	100	500	1000	5000
Čas [s]	0	2	5	28	57	284

Obrázek A.1: Závislost doby výpočtu na velikosti úlohy (vlevo) a počtu umělých mravenců (vpravo).

	m					
	10	20	30	50	70	90
eil51	5,42%	4,48%	4,16%	3,53%	3,96%	3,26%
pr76	8,18%	7,62%	7,92%	7,12%	6,85%	6,70%
pr152	7,20%	5,96%	4,57%	4,37%	4,23%	3,75%

Obrázek A.2: Odchylka průměrné hodnoty řešení od optima v závislosti na počtu použitých mravenců.

	ρ										
	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
bays29	0,40%	0,49%	0,51%	0,52%	0,37%	0,46%	0,68%	0,33%	0,50%	0,36%	0,81%
eil51	4,45%	3,50%	3,51%	3,22%	3,08%	3,47%	3,54%	3,61%	3,36%	3,73%	3,16%
gr96	7,41%	6,82%	6,51%	6,30%	6,24%	6,58%	6,29%	6,40%	6,60%	6,00%	6,31%

Obrázek A.3: Odchylka průměrné hodnoty řešení od optima v závislosti na koeficientu vypařování.

eil51

		β				
		0,3	0,7	1	2	3
α	0,3	157,34%	112,85%	81,85%	27,09%	12,39%
	0,7	108,45%	51,21%	30,23%	9,11%	4,48%
	1	47,09%	15,26%	8,15%	2,99%	3,08%
	2	83,96%	21,75%	13,32%	7,90%	7,31%
	3	126,57%	38,22%	20,36%	10,31%	8,51%

bays29

		β				
		0,3	0,7	1	2	3
α	0,3	81,01%	56,63%	39,43%	11,49%	3,95%
	0,7	56,35%	24,31%	13,03%	1,95%	0,61%
	1	26,15%	5,57%	1,87%	0,89%	1,07%
	2	58,41%	17,63%	13,41%	6,68%	5,31%
	3	72,59%	29,65%	17,16%	8,36%	6,89%

Obrázek A.4: Odchylka průměrné hodnoty řešení od optima v závislosti na vahách pro heuristickou informaci a hladinu feromonu.

		e											
		0	3	5	8	10	15	18	20	30	40	50	60
bays29	m=25	0,86%	0,10%	0,11%	0,49%	0,31%	0,54%	0,84%	0,95%	2,43%	3,02%	4,36%	4,75%
	m=51	0,61%	0,01%	0,06%	0,02%	0,01%	0,06%	0,10%	0,15%	0,34%	0,67%	0,98%	1,15%
eil51	m=25	4,10%	1,30%	0,45%	0,96%	1,10%	1,91%	2,63%	3,20%	5,03%	7,55%	8,06%	8,98%
	m=51	3,33%	1,79%	0,63%	0,41%	0,57%	0,40%	0,44%	0,80%	1,10%	2,11%	2,22%	3,71%

Obrázek A.5: Odchylka průměrné hodnoty řešení od optima v závislosti na počtu elitních mravenců.

Symetrické instance

	rozložení feromonu	
	symetrické	asymetrické
eil51	3,52%	6,45%
gr96	6,46%	8,88%

Asymetrické instance

	rozložení feromonu	
	symetrické	asymetrické
ft53	15,75%	11,36%
ftv70	13,36%	10,57%

Obrázek A.6: Odchylka průměrné hodnoty řešení od optima v závislosti na způsobu uložení feromonu.

	Q									
	10^{-5}	10^{-4}	10^{-3}	10^{-2}	10^{-1}	1	10	100	10^{3}	10^{4}
Odchylka	3,32%	3,09%	3,39%	3,28%	3,46%	2,93%	3,67%	3,37%	3,84%	3,06%

Obrázek A.7: Odchylka průměrné hodnoty řešení od optima v závislosti na koeficientu přírůstku feromonu (problém eil51).

	τ_0					
	10^{-6}	10^{-4}	10^{-2}	1	10^{2}	10^{4}
Odchylka	2,83%	3,57%	2,86%	3,39%	3,58%	3,49%

Obrázek A.8: Odchylka průměrné hodnoty řešení od optima v závislosti na počáteční hodnotě feromonu (problém eil51).

A.2 Quadratic Assignment Problem

A.2.1 Ant System

	n								
	10	15	20	30	40	50	80	100	150
AS	1	2	4	8	15	23	62	99	232
AS + l. p.	8	26	62	205	495	*1283*	*5448*	*11046*	*41835*

	m					
	5	10	50	100	500	1000
AS	0	0	3	6	31	61
AS + l. p.	11	20	100	199	*1312*	*2634*

Obrázek A.9: Závislost doby výpočtu na velikosti úlohy QAP a počtu umělých mravenců (v sekundách; kurzívou jsou uvedeny odhady po 5 iteracích výpočtu).

	m							
	10	20	30	50	70	90	130	180
bur26a	5,24%	4,61%	4,50%	4,36%	4,14%	4,17%	3,89%	3,64%
tai15b	2,47%	2,23%	2,03%	1,92%	1,85%	1,76%	1,67%	1,60%
tai25a	16,37%	15,23%	15,31%	14,45%	14,12%	13,81%	13,46%	13,12%
tai15a	14,04%	12,45%	11,73%	11,08%	10,40%	9,59%	8,94%	8,11%

Obrázek A.10: Odchylka průměrné hodnoty řešení od optima v závislosti na počtu použitých mravenců (bez lokálního prohledávání).

	m							
	10	20	30	50	70	90	130	180
bur26a	0,10%	0,09%	0,09%	0,07%	0,08%	0,07%	0,07%	0,05%
tai15b	0,02%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
tai25a	2,62%	2,10%	2,27%	1,85%	1,98%	1,88%	1,80%	1,77%
tai15a	0,02%	0,04%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%

Obrázek A.11: Odchylka průměrné hodnoty řešení od optima v závislosti na počtu použitých mravenců (s lokálním prohledáváním).

	ρ										
	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
bur26a	3,82%	4,14%	4,17%	4,19%	4,21%	4,27%	4,06%	4,23%	4,36%	4,37%	4,20%
nug30	16,02%	15,82%	16,46%	17,01%	16,97%	17,56%	17,93%	18,37%	18,32%	18,74%	18,28%
tai25a	13,22%	13,20%	13,37%	13,93%	14,11%	14,81%	14,58%	14,88%	14,75%	15,25%	15,24%

Obrázek A.12: Odchylka průměrné hodnoty řešení od optima v závislosti na koeficientu vypařování (bez lokálního prohledávání).

	ρ										
	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
bur26a	0,07%	0,08%	0,09%	0,08%	0,09%	0,07%	0,07%	0,09%	0,08%	0,08%	0,09%
nug30	0,58%	0,36%	0,24%	0,43%	0,29%	0,40%	0,30%	0,29%	0,41%	0,40%	0,14%
tai25a	2,10%	1,89%	2,02%	2,25%	1,84%	2,05%	1,97%	2,11%	1,91%	2,01%	2,09%

Obrázek A.13: Odchylka průměrné hodnoty řešení od optima v závislosti na koeficientu vypařování (s lokálním prohledáváním).

bur26a

		β				
		0,1	0,4	0,7	1	2
α	0,1	2,29%	2,07%	2,07%	2,15%	3,20%
	0,4	2,22%	2,14%	2,37%	2,80%	4,18%
	0,7	2,36%	2,33%	2,82%	3,65%	4,95%
	1	3,38%	3,52%	3,86%	4,39%	5,53%
	2	4,09%	3,93%	4,20%	4,45%	5,71%

tai25a

		β				
		0,1	0,4	0,7	1	2
α	0,1	13,46%	13,58%	13,58%	13,51%	13,31%
	0,4	13,36%	13,47%	13,23%	13,17%	13,16%
	0,7	13,61%	13,41%	13,45%	13,30%	12,96%
	1	14,86%	14,34%	14,41%	14,43%	14,35%
	2	17,06%	17,01%	16,55%	16,67%	16,06%

Obrázek A.14: Odchylka průměrné hodnoty řešení od optima v závislosti na vahách pro heuristickou informaci a hladinu feromonu (bez lokálního prohledávání).

bur26a

		β				
		0,1	0,4	0,7	1	2
α	0,1	0,01%	0,00%	0,01%	0,00%	0,01%
	0,4	0,00%	0,00%	0,00%	0,01%	0,01%
	0,7	0,01%	0,01%	0,02%	0,04%	0,05%
	1	0,05%	0,06%	0,08%	0,08%	0,08%
	2	0,10%	0,11%	0,10%	0,11%	0,09%

tai25a

		β				
		0,1	0,4	0,7	1	2
α	0,1	2,14%	2,06%	2,15%	2,13%	2,15%
	0,4	2,04%	2,11%	2,16%	2,27%	2,08%
	0,7	2,06%	1,99%	2,06%	1,96%	2,00%
	1	2,05%	2,09%	2,02%	2,14%	1,98%
	2	2,16%	2,16%	2,07%	1,84%	2,05%

Obrázek A.15: Odchylka průměrné hodnoty řešení od optima v závislosti na vahách pro heuristickou informaci a hladinu feromonu (s lokálním prohledáváním).

	Q									
	10^{-5}	10^{-4}	10^{-3}	10^{-2}	10^{-1}	1	10^{1}	10^{2}	10^{3}	10^{4}
bur26a	3,31%	3,35%	3,41%	3,45%	3,77%	4,11%	4,04%	4,15%	4,12%	4,26%
nug30	17,05%	17,63%	17,72%	17,78%	17,96%	17,61%	17,80%	17,54%	17,88%	17,37%
tai25a	14,44%	14,31%	14,28%	14,29%	14,57%	14,33%	14,38%	14,47%	14,67%	14,47%

Obrázek A.16: Odchylka průměrné hodnoty řešení od optima v závislosti na koeficientu přírůstku feromonu (bez lokálního prohledávání).

	Q									
	10^{-5}	10^{-4}	10^{-3}	10^{-2}	10^{-1}	1	10^{1}	10^{2}	10^{3}	10^{4}
bur26a	0,06%	0,06%	0,05%	0,07%	0,08%	0,08%	0,08%	0,08%	0,08%	0,08%
nug30	0,35%	0,40%	0,28%	0,36%	0,32%	0,33%	0,21%	0,36%	0,39%	0,39%
tai25a	1,91%	1,96%	2,04%	1,95%	2,22%	1,94%	1,99%	2,01%	2,03%	2,03%

Obrázek A.17: Odchylka průměrné hodnoty řešení od optima v závislosti na koeficientu přírůstku feromonu (s lokálním prohledáváním).

		τ_0					
		10^{-6}	10^{-4}	10^{-2}	1	10^{2}	10^{4}
AS	**bur26a**	4,39%	4,17%	3,62%	3,26%	3,16%	3,04%
	tai25a	14,62%	14,49%	14,61%	14,10%	14,56%	14,35%
AS + l. p.	**bur26a**	0,07%	0,08%	0,08%	0,06%	0,06%	0,04%
	tai25a	1,96%	2,04%	2,04%	2,03%	2,00%	2,14%

Obrázek A.18: Odchylka průměrné hodnoty řešení od optima v závislosti na počáteční hodnotě feromonu.

		e												
		0	3	5	8	10	13	15	18	20	30	40	50	60
bur26a	**m=25**	4,59%	4,03%	3,95%	3,91%	3,76%	3,47%	3,36%	3,56%	3,31%	3,35%	3,22%	3,11%	2,98%
	m=50	4,18%	4,05%	3,73%	3,67%	3,62%	3,43%	3,51%	3,40%	3,10%	3,08%	2,89%	2,75%	2,76%
tai25a	**m=25**	15,27%	13,51%	13,48%	13,36%	13,14%	12,89%	12,74%	12,90%	13,45%	13,90%	14,12%	14,35%	14,06%
	m=50	14,31%	13,13%	12,87%	12,71%	12,30%	11,98%	11,84%	11,83%	12,16%	11,79%	11,94%	12,13%	12,29%

Obrázek A.19: Odchylka průměrné hodnoty řešení od optima v závislosti na počtu elitních mravenců (bez lokálního prohledávání).

		e												
		0	3	5	8	10	13	15	18	20	30	40	50	60
bur26a	**m=25**	0,09%	0,08%	0,07%	0,07%	0,08%	0,08%	0,07%	0,06%	0,06%	0,08%	0,06%	0,08%	0,05%
	m=50	0,08%	0,07%	0,06%	0,06%	0,06%	0,06%	0,06%	0,06%	0,07%	0,06%	0,06%	0,05%	0,06%
tai25a	**m=25**	2,38%	2,25%	1,52%	1,46%	1,21%	1,32%	1,38%	1,53%	1,52%	1,85%	1,69%	2,11%	1,88%
	m=50	2,15%	2,00%	1,84%	1,60%	1,48%	1,31%	0,92%	1,34%	1,07%	1,23%	1,24%	1,67%	1,60%

Obrázek A.20: Odchylka průměrné hodnoty řešení od optima v závislosti na počtu elitních mravenců (s lokálním prohledáváním).

A.2.2 Ant Colony System

	m							
	10	20	30	50	70	90	130	180
bur26a	7,07%	7,09%	6,91%	6,73%	6,76%	6,59%	6,75%	6,33%
tai15b	2,53%	2,33%	2,25%	2,25%	2,20%	2,23%	2,27%	2,30%
tai25a	13,66%	13,15%	13,14%	12,64%	12,46%	12,21%	12,24%	11,97%
tai15a	6,18%	6,69%	6,46%	6,85%	7,13%	6,72%	7,04%	6,88%

Obrázek A.21: Odchylka průměrné hodnoty řešení od optima v závislosti na počtu použitých mravenců (bez lokálního prohledávání).

	m							
	10	20	30	50	70	90	130	180
bur26a	0,05%	0,03%	0,03%	0,02%	0,01%	0,01%	0,01%	0,01%
tai15b	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
tai25a	2,29%	2,12%	1,95%	1,80%	1,65%	1,49%	1,48%	1,35%
tai15a	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%

Obrázek A.22: Odchylka průměrné hodnoty řešení od optima v závislosti na počtu použitých mravenců (s lokálním prohledáváním).

	ρ										
	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
bur26a	6,65%	6,76%	6,81%	6,78%	6,65%	6,67%	6,66%	6,74%	6,69%	6,42%	6,47%
nug30	12,93%	4,88%	5,57%	5,92%	6,25%	6,26%	6,31%	6,26%	6,53%	7,04%	9,68%
tai25a	12,38%	12,55%	12,85%	12,72%	12,42%	12,61%	12,72%	12,40%	12,58%	12,25%	12,46%

Obrázek A.23: Odchylka průměrné hodnoty řešení od optima v závislosti na koeficientu vypařování (bez lokálního prohledávání).

	ρ										
	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
bur26a	0,01%	0,02%	0,02%	0,02%	0,02%	0,01%	0,02%	0,02%	0,02%	0,02%	0,02%
nug30	0,71%	0,28%	0,24%	0,21%	0,26%	0,23%	0,40%	0,36%	0,33%	0,24%	0,27%
tai25a	1,76%	1,72%	1,84%	1,84%	1,75%	1,70%	1,88%	1,84%	1,75%	1,71%	1,79%

Obrázek A.24: Odchylka průměrné hodnoty řešení od optima v závislosti na koeficientu vypařování (s lokálním prohledáváním).

bur26a

		q_0				
		0,1	0,3	0,5	0,7	0,9
β	0,1	2,04%	1,87%	2,06%	2,72%	4,55%
	0,4	1,90%	2,06%	2,27%	3,20%	5,08%
	0,7	1,95%	2,19%	2,56%	3,53%	5,35%
	1	2,07%	2,39%	2,84%	3,83%	5,87%
	2	2,94%	3,41%	3,99%	4,87%	6,66%

tai25a

		q_0				
		0,1	0,3	0,5	0,7	0,9
β	0,1	12,82%	12,21%	12,04%	11,64%	12,32%
	0,4	12,90%	12,31%	12,14%	12,04%	12,50%
	0,7	12,81%	12,37%	12,04%	11,83%	12,45%
	1	12,83%	12,37%	12,02%	12,10%	12,35%
	2	12,68%	12,32%	11,82%	11,79%	12,38%

Obrázek A.25: Odchylka průměrné hodnoty řešení od optima v závislosti na váze heuristické informace a vyvážení prozkoumávání / využívání (bez lokálního prohledávání).

bur26a

		q_0				
		0,1	0,3	0,5	0,7	0,9
β	0,1	0,010%	0,010%	0,012%	0,010%	0,018%
	0,4	0,006%	0,008%	0,012%	0,010%	0,024%
	0,7	0,009%	0,007%	0,015%	0,014%	0,019%
	1	0,005%	0,009%	0,014%	0,015%	0,015%
	2	0,008%	0,010%	0,009%	0,019%	0,016%

tai25a

		q_0				
		0,1	0,3	0,5	0,7	0,9
β	0,1	2,13%	1,99%	1,97%	1,78%	1,82%
	0,4	2,08%	1,90%	1,97%	1,93%	1,86%
	0,7	1,90%	1,90%	2,04%	1,89%	1,83%
	1	1,95%	1,89%	1,88%	2,04%	1,94%
	2	2,12%	2,04%	1,90%	1,86%	1,82%

Obrázek A.26: Odchylka průměrné hodnoty řešení od optima v závislosti na váze heuristické informace a vyvážením prozkoumávání / využívání (s lokálním prohledáváním).

		τ_0					
		10^{-6}	10^{-4}	10^{-2}	1	10^{2}	10^{4}
ACS	**bur26a**	6,67%	6,62%	6,73%	6,43%	6,51%	6,72%
	tai25a	12,10%	12,44%	12,86%	12,68%	12,77%	12,69%
ACS + l. p.	**bur26a**	0,018%	0,014%	0,017%	0,018%	0,024%	0,013%
	tai25a	1,81%	1,83%	1,85%	1,83%	1,82%	1,82%

Obrázek A.27: Odchylka průměrné hodnoty řešení od optima v závislosti na počáteční hodnotě feromonu.

A.2.3 Hybrid Ant System

	m							
	10	20	30	50	70	90	130	180
bur26a	1,26%	1,00%	0,84%	0,71%	0,61%	0,60%	0,47%	0,50%
tai15b	0,66%	0,54%	0,48%	0,37%	0,38%	0,31%	0,31%	0,27%
tai25a	11,59%	10,87%	10,57%	9,45%	8,95%	8,67%	8,14%	8,02%
tai15a	6,26%	4,40%	4,22%	3,43%	3,33%	3,02%	3,11%	2,67%

Obrázek A.28: Odchylka průměrné hodnoty řešení od optima v závislosti na počtu použitých mravenců (bez lokálního prohledávání).

	m							
	10	20	30	50	70	90	130	180
bur26a	0,02%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
tai15b	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%
tai25a	1,45%	1,34%	1,21%	1,14%	1,17%	0,74%	0,95%	0,76%
tai15a	0,05%	0,04%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%

Obrázek A.29: Odchylka průměrné hodnoty řešení od optima v závislosti na počtu použitých mravenců (s lokálním prohledáváním).

	ρ										
	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
bur26a	1,46%	0,55%	0,60%	0,53%	0,42%	0,54%	0,51%	0,51%	0,52%	0,51%	0,49%
nug30	15,27%	10,25%	10,52%	9,84%	9,67%	9,20%	9,96%	9,95%	9,65%	9,57%	10,33%
tai25a	11,88%	8,46%	8,63%	8,27%	8,14%	8,44%	8,38%	8,31%	8,09%	8,15%	8,27%

Obrázek A.30: Odchylka průměrné hodnoty řešení od optima v závislosti na koeficientu vypařování (bez lokálního prohledávání).

	ρ										
	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
tai35b	0,16%	0,03%	0,04%	0,08%	0,03%	0,02%	0,02%	0,02%	0,05%	0,05%	0,05%
nug30	0,17%	0,15%	0,12%	0,12%	0,10%	0,10%	0,09%	0,13%	0,14%	0,06%	0,06%
tai25a	1,56%	1,07%	1,17%	1,02%	1,18%	0,93%	1,14%	1,17%	1,01%	1,21%	1,18%

Obrázek A.31: Odchylka průměrné hodnoty řešení od optima v závislosti na koeficientu vypařování (s lokálním prohledáváním).

	q_0										
	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
bur26a	1,92%	1,72%	1,58%	1,44%	1,29%	1,16%	0,92%	0,78%	0,58%	0,54%	0,48%
tai15b	1,14%	1,09%	0,97%	0,81%	0,64%	0,56%	0,45%	0,39%	0,32%	0,32%	0,31%
tai25a	12,60%	12,69%	12,49%	12,34%	12,08%	11,65%	11,09%	10,12%	9,31%	8,83%	8,00%
tai15a	8,52%	8,37%	7,92%	7,23%	6,23%	5,26%	4,16%	3,76%	2,81%	3,12%	2,58%

Obrázek A.32: Odchylka průměrné hodnoty řešení od optima v závislosti na vyvážení využívání a prozkoumávání (bez lokálního prohledávání).

	q_0										
	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
tai35b	0,24%	0,21%	0,16%	0,16%	0,08%	0,06%	0,04%	0,04%	0,04%	0,04%	0,04%
tai25a	1,75%	1,73%	1,69%	1,31%	1,58%	1,47%	1,23%	1,08%	1,09%	1,01%	0,83%

Obrázek A.33: Odchylka průměrné hodnoty řešení od optima v závislosti na vyvážení využívání a prozkoumávání (s lokálním prohledáváním).

		S								
		5	10	15	20	30	50	70	100	150
HAS	**bur26a**	1,02%	0,79%	0,74%	0,76%	0,72%	0,71%	0,69%	0,59%	0,61%
	tai25a	10,64%	10,13%	9,73%	9,82%	9,49%	9,46%	9,64%	9,15%	8,46%
HAS	**tai35b**	0,09%	0,06%	0,03%	0,07%	0,04%	0,11%	0,10%	0,06%	0,09%
+ l. p.	**tai25a**	1,10%	1,20%	1,12%	1,13%	1,15%	1,18%	1,04%	1,01%	1,17%

Obrázek A.34: Odchylka průměrné hodnoty řešení od optima v závislosti na počtu iterací pro diverzifikaci.

		R								
		1	5	10	15	20	25	30	40	50
HAS	**bur26a**	0,35%	0,57%	0,71%	0,71%	0,84%	0,70%	0,72%	0,79%	0,83%
	tai25a	7,85%	8,46%	9,04%	9,39%	9,74%	9,63%	9,93%	9,77%	9,54%
HAS	**tai35b**	0,00%	0,02%	0,03%	0,12%	0,08%	0,09%	0,16%	0,08%	0,13%
+ l.p.	**tai25a**	0,49%	0,71%	1,23%	1,25%	1,23%	1,29%	1,46%	1,04%	1,24%

Obrázek A.35: Odchylka průměrné hodnoty řešení od optima v závislosti na počtu výměn při modifikaci řešení.

A.2.4 Max-Min Ant System

	m							
	10	20	30	50	70	90	130	180
bur26a	1,37%	0,44%	0,34%	0,26%	0,25%	0,27%	0,24%	0,21%
tai15b	0,90%	0,50%	0,48%	0,47%	0,37%	0,37%	0,33%	0,36%
tai25a	10,11%	6,33%	5,39%	5,11%	4,73%	4,60%	4,70%	4,28%
tai15a	7,51%	4,12%	3,79%	3,58%	3,75%	3,49%	3,45%	3,23%

Obrázek A.36: Odchylka průměrné hodnoty řešení od optima v závislosti na počtu použitých mravenců (bez lokálního prohledávání).

	m							
	10	20	30	50	70	90	130	180
bur26a	0,074%	0,056%	0,067%	0,067%	0,052%	0,059%	0,037%	0,042%
tai15b	0,035%	0,025%	0,004%	0,000%	0,000%	0,000%	0,000%	0,000%
tai25a	2,10%	1,82%	1,69%	1,57%	1,74%	1,65%	1,51%	1,50%
tai15a	0,63%	0,40%	0,36%	0,27%	0,17%	0,09%	0,10%	0,13%

Obrázek A.37: Odchylka průměrné hodnoty řešení od optima v závislosti na počtu použitých mravenců (s lokálním prohledáváním).

	ρ										
	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
bur26a	0,80%	0,25%	0,26%	0,25%	0,26%	0,24%	0,25%	0,23%	0,25%	0,24%	0,24%
nug30	10,46%	3,84%	3,18%	3,47%	3,82%	3,58%	3,65%	3,38%	3,17%	3,71%	3,33%
tai25a	7,91%	4,47%	4,75%	4,38%	4,80%	4,53%	4,52%	4,65%	4,79%	4,78%	4,74%

Obrázek A.38: Odchylka průměrné hodnoty řešení od optima v závislosti na koeficientu vypařování (bez lokálního prohledávání).

	ρ										
	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
bur26a	0,055%	0,059%	0,041%	0,055%	0,058%	0,038%	0,051%	0,072%	0,051%	0,051%	0,051%
nug30	0,34%	0,42%	0,29%	0,23%	0,37%	0,25%	0,20%	0,35%	0,33%	0,33%	0,30%
tai25a	1,71%	1,90%	1,90%	1,76%	1,91%	1,75%	1,57%	1,83%	1,70%	1,90%	1,62%

Obrázek A.39: Odchylka průměrné hodnoty řešení od optima v závislosti na koeficientu vypařování (s lokálním prohledáváním).

	q_0										
	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
bur26a	0,16%	0,17%	0,19%	0,19%	0,22%	0,23%	0,21%	0,28%	0,27%	0,27%	4,47%
nug30	2,58%	2,68%	2,71%	2,94%	3,00%	3,22%	2,94%	3,40%	3,24%	3,46%	24,50%
tai25a	3,97%	4,04%	4,10%	4,33%	4,39%	4,49%	4,49%	4,37%	4,54%	5,12%	17,91%

Obrázek A.40: Odchylka průměrné hodnoty řešení od optima v závislosti na vyvážení využívání a prozkoumávání (bez lokálního prohledávání).

	q_0										
	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
bur26a	0,020%	0,021%	0,027%	0,038%	0,045%	0,041%	0,044%	0,041%	0,043%	0,046%	0,143%
nug30	0,09%	0,10%	0,08%	0,14%	0,15%	0,20%	0,26%	0,29%	0,39%	0,39%	1,93%
tai25a	0,94%	0,87%	1,05%	0,89%	1,16%	1,08%	1,34%	1,29%	1,42%	1,69%	4,06%

Obrázek A.41: Odchylka průměrné hodnoty řešení od optima v závislosti na vyvážení využívání a prozkoumávání (s lokálním prohledáváním).

		t_{gb}							
		10	20	30	50	100	200	300	500
MMAS	**bur26a**	0,24%	0,22%	0,24%	0,25%	0,25%	0,29%	0,24%	0,25%
	tai25a	4,40%	4,65%	4,91%	4,83%	4,71%	4,58%	4,63%	4,66%
MMAS + l. p.	**bur26a**	0,035%	0,059%	0,050%	0,043%	0,055%	0,056%	0,050%	0,057%
	tai25a	1,60%	1,74%	1,70%	1,73%	1,73%	1,67%	1,70%	1,70%

Obrázek A.42: Odchylka průměrné hodnoty řešení od optima v závislosti na počtu iterací pro aplikaci globálně nejlepšího řešení při úpravě feromonu.

		τ_0					
		10^{-6}	10^{-4}	10^{-2}	1	10^{2}	10^{4}
MMAS	**bur26a**	0,22%	0,24%	0,22%	0,22%	0,22%	0,23%
	tai25a	4,46%	4,59%	4,40%	4,57%	4,49%	4,48%
MMAS + l. p.	**bur26a**	0,049%	0,046%	0,054%	0,050%	0,070%	0,051%
	tai25a	1,58%	1,63%	1,58%	1,62%	1,78%	1,58%

Obrázek A.43: Odchylka průměrné hodnoty řešení od optima v závislosti na počáteční hodnotě feromonu.

		δ										
		0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
MMAS	**bur26a**	0,23%	0,25%	0,27%	0,23%	0,23%	0,27%	0,25%	0,23%	0,23%	0,28%	0,20%
	tai25a	4,87%	4,81%	4,64%	4,76%	4,57%	4,63%	4,78%	4,80%	4,53%	5,08%	4,69%
MMAS + l. p.	**bur26a**	0,051%	0,057%	0,042%	0,051%	0,056%	0,055%	0,058%	0,044%	0,048%	0,051%	0,022%
	tai25a	1,96%	1,88%	1,84%	1,74%	1,86%	1,68%	1,96%	1,90%	1,92%	1,81%	1,51%

Obrázek A.44: Odchylka průměrné hodnoty řešení od optima v závislosti na síle vyhlazování feromonu.

		t_{conv}							
		10	20	30	40	60	80	110	140
MMAS	**bur26a**	0,24%	0,23%	0,28%	0,26%	0,24%	0,27%	0,28%	0,24%
	tai25a	4,78%	4,80%	4,58%	4,43%	5,04%	4,67%	4,73%	4,80%
MMAS + l. p.	**bur26a**	0,048%	0,036%	0,042%	0,051%	0,051%	0,051%	0,055%	0,055%
	tai25a	1,44%	1,71%	1,55%	1,62%	1,82%	1,89%	1,96%	1,82%

Obrázek A.45: Odchylka průměrné hodnoty řešení od optima v závislosti na počtu iterací pro kontrolu míry konvergence.

A.2.5 Genetic Ant System

	n							
	10	15	20	30	40	50	80	100
GAS	2	6	10	23	39	61	152	240
GAS + l. p.	11	34	79	260	617	1219	*5132*	*12150*

	m					
	5	10	50	100	500	1000
GAS	4	7	40	77	381	775
GAS + l. p.	62	123	615	1229	*6481*	*12681*

Obrázek A.46: Závislost doby výpočtu na velikosti úlohy (vlevo) a počtu umělých mravenců (vpravo). Údaje jsou v sekundách, kurzívou jsou uvedeny odhady po 30 iteracích výpočtu.

	m							
	10	20	30	50	70	90	130	180
bur26a	0,42%	0,47%	0,30%	0,21%	0,19%	0,22%	0,19%	0,21%
nug30	3,06%	2,74%	2,09%	1,09%	0,98%	1,84%	1,92%	0,86%
tai25a	4,24%	3,71%	3,74%	3,00%	2,60%	4,25%	3,12%	3,56%

Obrázek A.47: Odchylka průměrné hodnoty řešení od optima v závislosti na počtu použitých mravenců (bez lokálního prohledávání).

	m							
	10	20	30	50	70	90	130	180
bur26a	0,08%	0,08%	0,05%	0,00%	0,00%	0,00%	0,00%	0,00%
nug30	0,06%	0,05%	0,00%	0,07%	0,00%	0,27%	0,18%	0,10%
tai25a	1,51%	1,01%	0,55%	1,21%	0,37%	0,66%	0,54%	0,24%

Obrázek A.48: Odchylka průměrné hodnoty řešení od optima v závislosti na počtu použitých mravenců (s lokálním prohledáváním).

	ρ										
	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
bur26a	0,15%	0,15%	0,26%	0,27%	0,24%	0,28%	0,31%	0,28%	0,30%	0,29%	0,28%
nug30	6,32%	1,58%	1,69%	2,24%	2,81%	2,80%	2,83%	3,40%	3,52%	3,28%	3,14%
tai25a	2,42%	3,76%	3,55%	4,11%	4,48%	4,60%	4,26%	4,76%	4,71%	4,54%	4,66%

Obrázek A.49: Odchylka průměrné hodnoty řešení od optima v závislosti na koeficientu vypařování (bez lokálního prohledávání).

	ρ										
	0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1
bur26a	0,001%	0,080%	0,004%	0,055%	0,025%	0,017%	0,008%	0,017%	0,034%	0,013%	0,000%
nug30	0,036%	0,372%	0,000%	0,020%	0,029%	0,000%	0,049%	0,007%	0,016%	0,013%	0,003%
tai25a	1,62%	1,06%	0,54%	0,60%	0,66%	0,74%	0,57%	0,56%	0,79%	0,34%	0,38%

Obrázek A.50: Odchylka průměrné hodnoty řešení od optima v závislosti na koeficientu vypařování (s lokálním prohledáváním).

	n_{best}						
	1	3	5	10	15	20	25
bur26a	0,27%	0,28%	0,27%	0,31%	0,22%	0,11%	0,15%
nug30	4,40%	3,28%	1,43%	0,95%	2,16%	1,31%	1,57%
tai25a	5,86%	3,79%	4,65%	2,99%	3,38%	3,14%	3,20%

Obrázek A.51: Odchylka průměrné hodnoty řešení od optima v závislosti na počtu nejlepších aktivit pro pomocné agenty (bez lokálního prohledávání).

	n_{best}						
	1	3	5	10	15	20	25
bur26a	0,03%	0,08%	0,08%	0,08%	0,00%	0,07%	0,04%
nug30	0,02%	0,00%	0,00%	0,03%	0,41%	0,26%	0,37%
tai25a	0,82%	1,21%	1,11%	0,74%	1,18%	1,07%	1,24%

Obrázek A.52: Odchylka průměrné hodnoty řešení od optima v závislosti na počtu nejlepších aktivit pro pomocné agenty (s lokálním prohledáváním).

	λ_{min} / λ_{max}									
	1,1 / 1,3	1,1 / 1,8	1,1 / 2,5	1,1 / 4	1,3 / 1,8	1,3 / 2,5	1,3 / 4	1,8 / 2,5	1,8 / 4	2,5 / 4
bur26a	0,23%	0,20%	0,28%	0,24%	0,26%	0,34%	0,27%	0,14%	0,24%	0,28%
nug30	1,26%	1,47%	2,60%	2,71%	1,51%	1,82%	2,84%	2,00%	3,20%	2,83%
tai25a	1,93%	2,59%	3,88%	4,60%	3,22%	3,40%	4,44%	4,08%	4,39%	4,19%

Obrázek A.53: Odchylka průměrné hodnoty řešení od optima v závislosti na hranicích průměrného λ-faktoru větvení (bez lokálního prohledávání).

	λ_{min} / λ_{max}									
	1,1 / 1,3	1,1 / 1,8	1,1 / 2,5	1,1 / 4	1,3 / 1,8	1,3 / 2,5	1,3 / 4	1,8 / 2,5	1,8 / 4	2,5 / 4
bur26a	0,000%	0,032%	0,009%	0,043%	0,042%	0,000%	0,004%	0,006%	0,000%	0,000%
nug30	0,39%	0,000%	0,01%	0,30%	0,03%	0,19%	0,17%	0,36%	0,09%	0,13%
tai25a	1,05%	0,48%	0,74%	0,82%	0,84%	1,05%	1,09%	0,50%	1,08%	1,07%

Obrázek A.54: Odchylka průměrné hodnoty řešení od optima v závislosti na hranicích úrůměrného λ-faktoru větvení (s lokálním prohledáváním).

	t_{gen}						
	1	5	10	25	50	75	100
bur26a	0,36%	0,39%	0,21%	0,10%	0,21%	0,23%	0,19%
nug30	2,11%	2,01%	1,71%	0,28%	0,93%	1,53%	2,17%
tai25a	3,88%	3,15%	3,23%	3,86%	3,91%	3,42%	2,93%

Obrázek A.55: Odchylka průměrné hodnoty řešení od optima v závislosti na počtu iterací pro aplikaci genetického principu (bez lokálního prohledávání).

	t_{gen}						
	1	5	10	25	50	75	100
bur26a	0,000%	0,000%	0,000%	0,000%	0,008%	0,000%	0,041%
nug30	0,013%	0,026%	0,405%	0,029%	0,023%	0,029%	0,000%
tai25a	1,17%	1,23%	1,27%	1,03%	0,91%	0,84%	1,03%

Obrázek A.56: Odchylka průměrné hodnoty řešení od optima v závislosti na počtu iterací pro aplikaci genetického principu (s lokálním prohledáváním).

	w_{max}					
	0,1	0,5	1	1,5	2	3
bur26a	0,34%	0,34%	0,14%	0,22%	0,23%	0,22%
nug30	3,83%	3,29%	1,69%	1,49%	2,10%	2,46%
tai25a	5,41%	5,29%	2,95%	3,59%	3,35%	4,20%

Obrázek A.57: Odchylka průměrné hodnoty řešení od optima v závislosti na maximální hodnotě vah (bez lokálního prohledávání).

	w_{max}					
	0,1	0,5	1	1,5	2	3
bur26a	0,00%	0,00%	0,05%	0,01%	0,04%	0,00%
nug30	0,75%	0,21%	0,00%	0,39%	0,03%	0,01%
tai25a	2,02%	2,09%	1,18%	1,02%	0,74%	0,27%

Obrázek A.58: Odchylka průměrné hodnoty řešení od optima v závislosti na maximální hodnotě vah (s lokálním prohledáváním).

	t_{res}					
	10	30	50	80	120	160
bur26a	0,18%	0,36%	0,19%	0,25%	0,32%	0,44%
nug30	2,17%	0,71%	0,57%	2,20%	4,01%	4,24%
tai25a	3,81%	2,90%	3,66%	4,07%	4,04%	4,98%

Obrázek A.59: Odchylka průměrné hodnoty řešení od optima v závislosti na počtu iterací bez pokroku pro resetování vah (bez lokálního prohledávání).

	t_{res}					
	10	30	50	80	120	160
bur26a	0,006%	0,000%	0,060%	0,008%	0,072%	0,071%
nug30	0,01%	0,41%	0,02%	0,40%	0,02%	0,01%
tai25a	0,82%	1,14%	0,60%	0,93%	0,67%	0,73%

Obrázek A.60: Odchylka průměrné hodnoty řešení od optima v závislosti na počtu iterací bez pokroku pro resetování vah (s lokálním prohledáváním).

	m_{upd}						
	1	5	10	15	20	30	50
bur26a	0,32%	0,32%	0,28%	0,21%	0,20%	0,21%	0,41%
nug30	3,59%	1,11%	1,09%	2,93%	1,73%	2,61%	3,40%
tai25a	5,69%	2,93%	3,86%	4,09%	3,21%	4,65%	4,53%

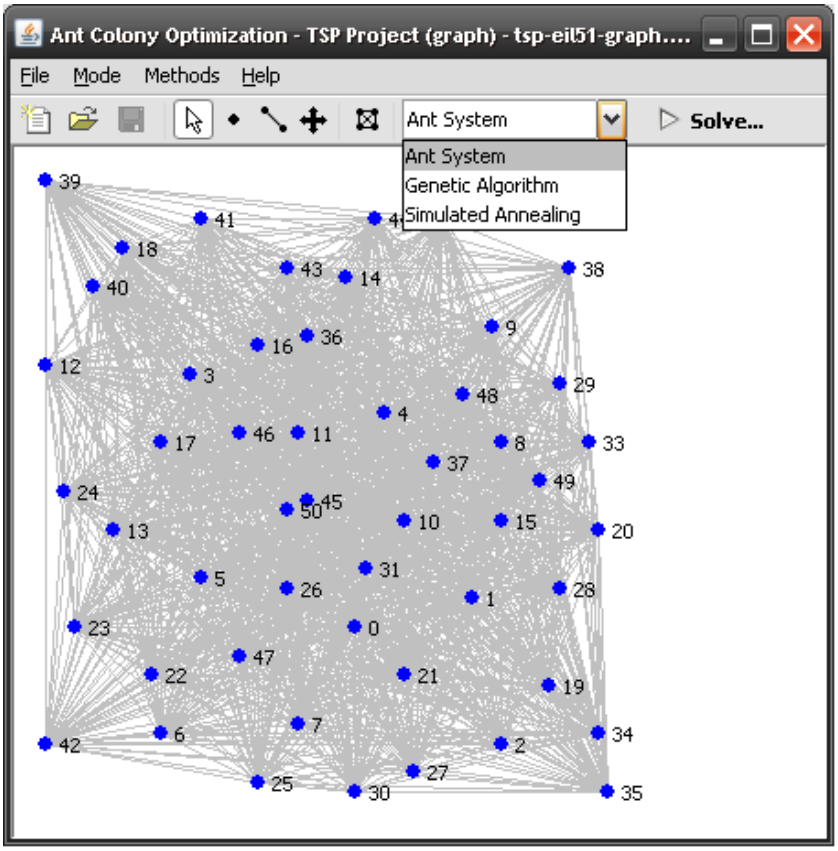
Obrázek A.61: Odchylka průměrné hodnoty řešení od optima v závislosti na počtu mravenců pro úpravu feromonu (bez lokálního prohledávání).

	m_{upd}						
	1	3	5	10	15	20	30
bur26a	0,035%	0,004%	0,017%	0,051%	0,072%	0,000%	0,016%
nug30	0,072%	0,003%	0,000%	0,000%	0,392%	0,046%	0,392%
tai25a	1,12%	0,64%	0,87%	1,16%	1,28%	0,88%	0,85%

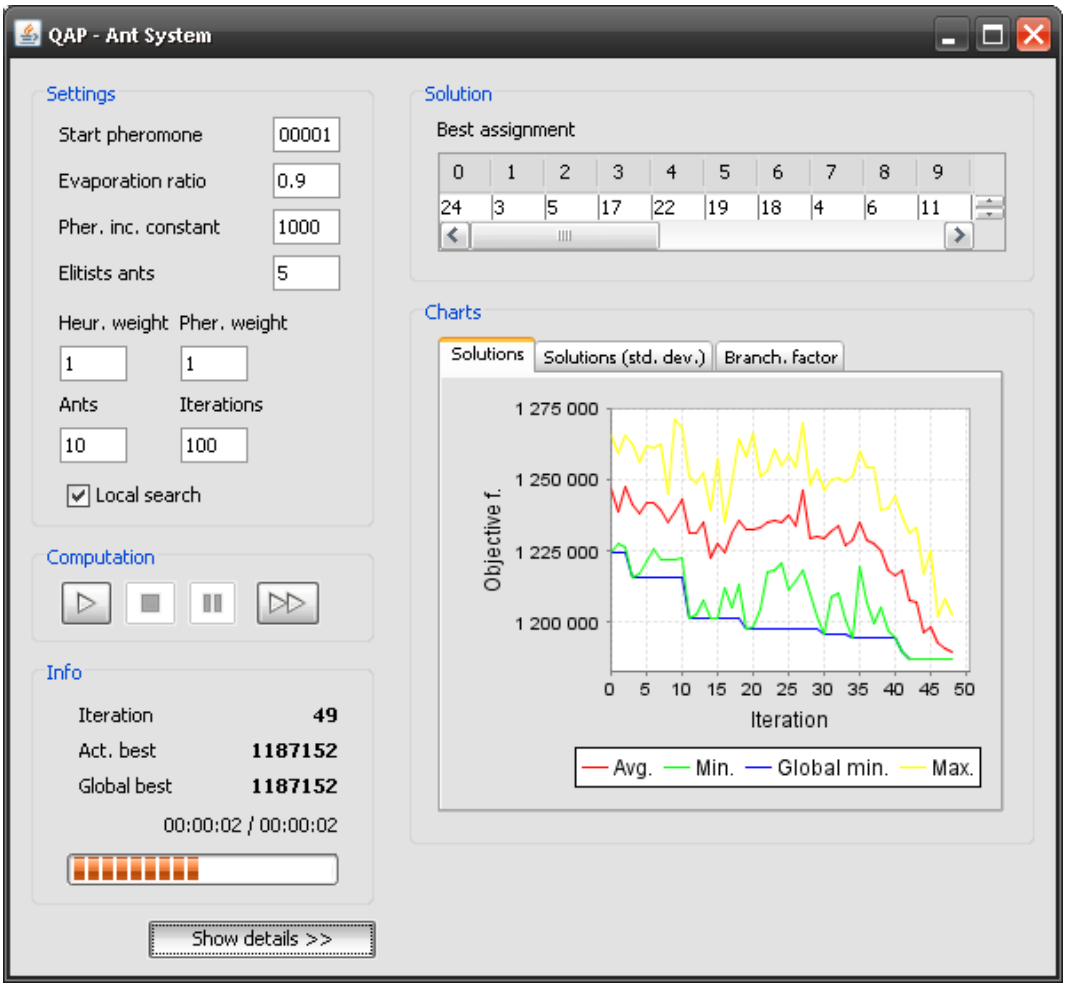
Obrázek A.62: Odchylka průměrné hodnoty řešení od optima v závislosti na počtu mravenců pro úpravu feromonu (s lokálním prohledáváním).

Příloha B

Uživatelské rozhraní aplikace



Obrázek B.1: Hlavní okno programu



Obrázek B.2: Okno pro řešení úlohy

Příloha C

Obsah přiloženého CD

Na CD, které je přiloženo k této práci, mají adresáře následující význam:

`dokumentace` Jsou zde dva podadresáře: v `programova_dokumentace` je programová dokumentace vygenerovaná pomocí nástroje JavaDoc a v `uzivatelsky_manual` je kompletní uživatelský manuál, který popisuje ovládání programu.

`instance` Zde jsou umístěny uložené soubory projektů pro program, který byl vyvinut v rámci této práce. Obsahují instance TSP (podadresář `tsp`) a QAP (podadresář `qap`) úloh importované z knihoven TSPLIB a QAPLIB.

`ostatni` Podadresář `jfreechart` obsahuje Javovskou knihovnu JFreeChart, která slouží k zobrazování grafů. V `JRE16` je Java Runtime Environment verze 1.6 (instalace je nutná, pokud již nebyla dříve provedena, pro běh programu).

`program` Zde lze nalézt samotný program, který je možné spustit pomocí souboru `ACO.jar`. Alternativou je dávka `run_aco.bat`, která při spuštění vyhradí programu paměť o velikosti 500MB.

`text` V podadresáři `zdrojove_soubory` jsou všechny potřebné soubory pro přeložení práce v systému LaTeX. `pdf` obsahuje výsledný text zprávy v podobě PDF souboru.

`zdrojove_soubory` Tento adresář obsahuje zdrojové soubory programu jako projekt vývojového nástroje NetBeans (verze 6.1RC).