Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta

Řešení optimální cesty svozu odpadů pomocí rojové inteligence

Bakalářská práce

Ladislav Vácha

Školitel: doc. Ing. Ladislav Beránek, CSc.

České Budějovice 2016

Vácha, L., 2015: Řešení optimální cesty svozu odpadů pomocí rojové inteligence. [Solving optimal ways of waste collection by swarm intelligence. Bc. Thesis, in Czech.] – 48 p., Faculty of Science, University of South Bohemia, České Budějovice, Czech Republic.

Anotace

Tato práce je zaměřena na řešení problémů třídy nedeterministicky polynomiální (NP) složitosti pomocí optimalizace mravenčí kolonie. Práce je rozdělena na tři bloky. V prvním je přiblížena výše zmíněná optimalizace spolu s některými modifikacemi. Druhá část je zaměřena na samotné problémy, v tomto případě problém obchodního cestujícího (TSP) a z něho vycházející vehicle routing problem (VRP). Závěr tvoří aplikace těchto nástrojů na svoz tříděného odpadu pro část Českých Budějovic.

Abstract

This work is focused on problem-solving nondeterministically polynomial (NP) complexity using ant colony optimization. The work is divided into three blocks. The first is approximated aforementioned optimization along with some modifications. The second part focuses on the problems themselves, in this case, the traveling salesman problem (TSP), from which the vehicle routing problem (VRP). The final part of this thesis describes the use of these tools for the collection of separated waste for district of the České Budějovice.

Prohlašuji, že svoji bakalářskou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury. Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne

Obsah

1. Úvod

Bakalářská práce se zabývá aplikací metod rojové inteligence při řešení problematiky svozu tříděného odpadu. Společnosti, které zabezpečují svoz tříděného odpadu, nemají vypracovaný nějaký systém sběrných cest, ale vozidla pro svoz odpadu mají vymezené určité území, ze kterého tříděný odpad sváží.

Předkládaná práce je určitou zjednodušenou studii, která by měla řešit problém optimálních cest svozu tříděného odpadu. Úkolem je použít metodu optimalizace mravenčí kolonie pro hledání optimálních cest. Metoda bude aplikována na menším vzorku dat (vybraná lokalita) s následnou možností aplikovat vypracovanou metodu i na větší lokalitu.

Problém svozu tříděného odpadu obdoba známého problému obchodního cestujícího, kdy sběrná místa představují města. Okružní jízda představuje jednotlivé trasy, po kterých je nutno sběrná místa objet. Problém je navíc rozšířen o nutnost odjezdů na skládku, pokud vozidlo pro svoz tříděného odpadu již naplní svoji kapacitu po určitém počtu navštívených míst. Naplnění kontejnerů se tříděným odpadem je zde modelován jako náhodná proměnná.

Problém obchodního cestujícího patří mezi nejstudovanější NP problémy, včetně z něho vycházející problém okružných jízd. Jedná se o problematiku, která má poměrně široké praktické využití v běžném životě, jedná se například o rozvoz balíků, zásobování obchodů a mnoha dalších.

V druhé kapitole bude popsán metodický postup. Následně v třetí kapitole přiblížím optimalizaci mravenčí kolonií. Od prvních pokusů převedení pravidel, kterými se mravenci rozhodují v přírodě, po sofistikované metaheuristické algoritmy řešící složité problémy. V kapitolách 4 a 5 budou popsány problémy obchodního cestujícího a okružních jízd. Zmíním některé další způsoby řešení a varianty těchto problémů. Zbývající část práce (kapitola 6) se bude zabývat vytvořením algoritmu pro nalezení optimální cesty svozu odpadu. Poslední částí je zhodnocení algoritmu.

2. Cíle práce a metodika

2.1 Cíl práce

Cílem práce je využít metody optimalizace mravenčí kolonie pro návrh postupu efektivního svozu tříděného odpadu ve vybrané lokalitě v Českých Budějovicích.

Dílčí cíle a úkoly:

1) Vytvořit aplikaci pro výpočet optimální cesty pomocí optimalizace mravenčí kolonií za podmínek, že známe dopředu zaplnění kontejnerů tříděným odpadem a kapacita sběrných vozidel je omezená.

Podcíle

2) Získat lokality kontejnerů tříděného odpadu, vytvořit graf z mapy dané lokality pro použití optimalizace mravenčí kolonie.
3) Navrhnout postup efektivního svozu tříděného odpadu za použití algoritmu optimalizace mravenčí kolonií, implementovat metodu ve vhodném prostředí a aplikovat na získaná data.

2.2 Použité nástroje a metody

Pro dosažení stanovených cílů je nejdříve využita dostupná odborná literatura a seznámení s problematikou optimalizace a problémem okružních jízd.

Získané teoretické znalosti a data budou následně aplikovány při tvorbě programu, který navrhne optimální trasu pro svoz tříděného odpadu v dané lokalitě. Program bude vytvořen v prostředí Microsoft Visual Studio 2010.

Účelem této studie bude zjištění základních problémů, které se při optimalizaci sběrných cest vyskytují, a navržení řešení. Algoritmus, který bude výstupem bakalářské práce, bude založen na využití metody optimalizace pomocí mravenčí kolonie. Algoritmus by měl být škálovatelný a použitelný i pro komunální odpad v jakémkoli městě.

3. Optimalizace mravenčí kolonií

3.1. Inspirace pro návrh algoritmu

Algoritmus optimalizace mravenčí kolonií je založen na stejných pravidlech a metodách, kterými se řídí mravenčí kolonie, přesněji na postupu od nalezení potravy až k následnému předání informace o její pozici.

Přestože jednotliví mravenci jsou poměrně jednoduší a sami o sobě toho moc nesvedou, mravenčí kolonie je proti nim velmi výkonný a efektivní celek. Mravenci řeší různá důležitá rozhodnutí, jako například nalezení co nejkratší cesty k potravě a zpět do hnízda, čímž ušetří energii.

Většina druhů má slabý zrak, přičemž někteří jsou zcela slepí, [3] z tohoto důvodu se mezi sebou dorozumívají pomocí *stigmergie*. Stigmergií nazýváme komunikaci, při které se modifikuje okolní prostředí. V tomto případě se jedná o vyměšování chemikálií, které nazýváme *feromony*. Mravenci jsou schopni cítit tuto látku a podle typu feromonu upravit své chování [1]. Hlavní feromon, jejž mravenci vylučují, je tzv. feromon stezky *(trail pheromone),* používaný pro označení cest, ať už je to cesta za potravou nebo materiály potřebné pro údržbu mraveniště. Tato nepřímá komunikace zásadním způsobem přispívá k samo-organizaci roje. To znamená, že není třeba, aby někdo dohlížel nebo kontroloval činnost kolonie. Veškerá komunikace probíhá pouze lokálně pomocí feromonů. [3]

Mravenci při hledání potravy nejprve náhodně prohledávají okolí. Když je některý z nich úspěšný a najde potravu, vrátí se stejnou cestou do kolonie. Jak při průzkumu tak i návratu vypouštějí za sebou feromonovou stopu, která následně slouží jako ukazatel – směrovka. Díky této informaci se vydá po stejné cestě, kterou se původně k potravě dostal, a je tím zesílena vrstva feromonu. Z tohoto důvodu bude tato již několikrát označená trasa přitažlivější než cesta mravenců, kteří se ještě nevrátili do hnízda. Noví průzkumníci, kteří se vydají pro ně touto atraktivnější cestou, dále přispějí k vytvoření silné feromonové stopy na kratší cestě. [5]. Po této stezce se časem vydávají téměř všichni mravenci. Tento způsob pozitivní zpětné vazby je ukázkou samo-organizace celého roje. [3]

Někteří mravenci se přesto nevydají po označené cestě, a i tak mohou při svém průzkumu nalézt tentýž zdroj potravy. Tím se vytvoří více feromonových cest. Feromon se postupem času vypařuje a díky tomu jej bude na kratší cestě více. Kvůli větší koncentraci látky na kratší cestě se po ni začnou časem pohybovat i ti mravenci, kteří doposud používali delší trasu.

Ovšem jak tomu v přírodě bývá, existují rozdíly mezi jednotlivými druhy, například některé vypouští feromon pouze při návratu nebo jiný druh (Lasius niger) není schopen objevit nově dostupné trasy k potravě, když je první cesta silně nasycená feromonem. [2]

Obrázek 1

A) Mravenci začnou náhodně prohledávat okolí.
B) Po návratu mravenců, kteří se vydali kratší cestou, je na této cestě větší koncentrace feromonu a tím je lákavější pro následující mravence.

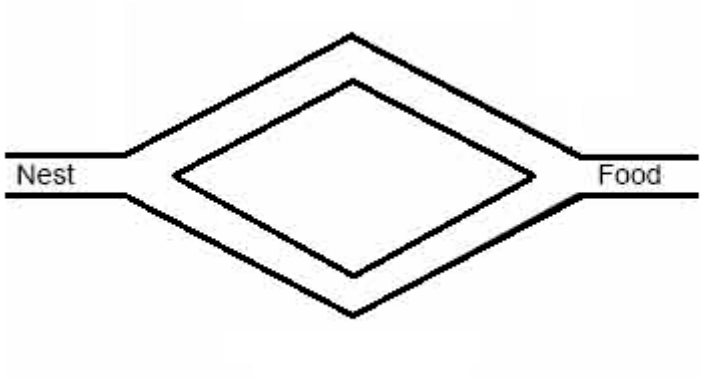
Komunikace roje pomocí feromonu byla testována různými experimenty. Mezi hlavní patří tzv. double bridge experiment pod vedením Jeana Louise Deneubourga. Ten spočíval ve spojení mravenčího hnízda a potravy dvojitým mostem a pozorování, jakou cestu si průzkumníci vyberou.

Zprvu byly obě větve stejně dlouhé (obr. 2). Když se mravenci začali vydávat za potravou, vybírali cestu náhodně, protože nebyl přítomen žádný feromon. Přestože obě cesty byly stejně dlouhé, postupem času se všichni mravenci pohybovali po jedné cestě (obr. 5a). Výsledné chování se přisuzuje náhodné složce při výběru cesty, přesněji řečeno, jednu z možností si vybere více mravenců, a tím bude na této větvi mostu silnější feromon, čímž bude lákavější pro následující průzkumníky. Jak je vidět z výsledků experimentu (obr. 2), výsledná trasa je náhodná a výběr je přibližně stejný pro obě trasy. [3]

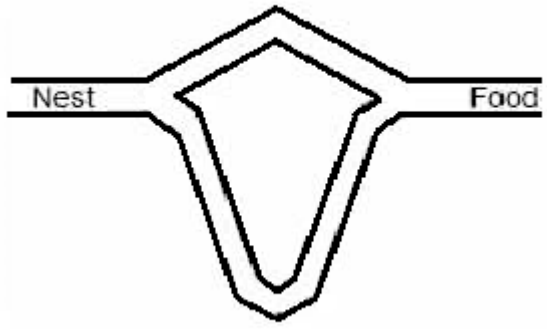
Při druhém experimentu byla jedna cesta mostu dvakrát delší než druhá (obr. 3). Při této variantě se po krátké chvíli téměř všichni pohybovali po kratším mostě. Stejně jako v předchozím případě si zpočátku mravenci vybrali cestu náhodně, avšak ti, kteří si vybrali tu kratší, se i rychleji dostali k potravě, a tím se při zpáteční cestě vraceli po silněji značené feromonové stopě, kterou po sobě zanechali. [5] Z výsledků experimentu (obr. 5b) je vidět

naprostá převaha mravenců, kteří se vydali kratší trasou. I přes silnou feromonovou stopu si stále někteří jedinci vybrali neoznačenou trasu v pokuse o další průzkum. [3]

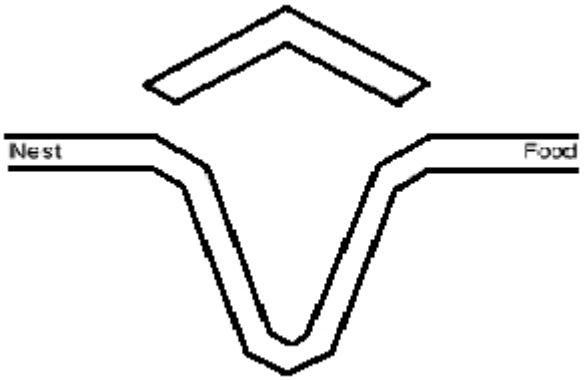
Při jiném experimentu byla zprvu mravencům poskytnuta jen dlouhá trasa a po půl hodině byla přidána druhá, kratší (obrázek 4). Tím se ukázalo, že i mezi jednotlivými druhy mravenců jsou rozdíly. Konkrétně průzkumníci druhu *Linepithaema humile* nebyli schopni upravit trasu po přidání kratší cesty. Přestože se někteří mravenci vydali nově přidanou trasou, nebylo jich dostatek, aby přemohli silnou feromonovou stopu na delší cestě a její pomalé odpařování, a tím uvázli v lokálním řešení. Kdežto zástupci druhu *Lasius niger* jsou schopni využít i později dostupné, výhodnější trasy. [4]



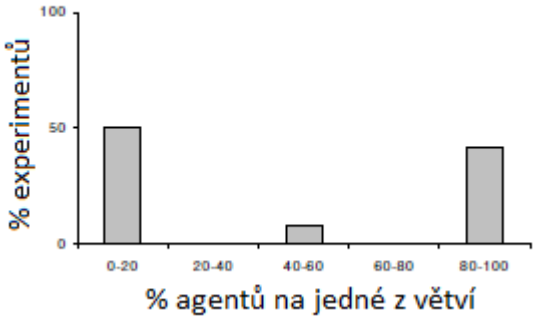
Obrázek 2: Schéma experimentu dvojitého mostu pro stejně dlouhé větve. [3]



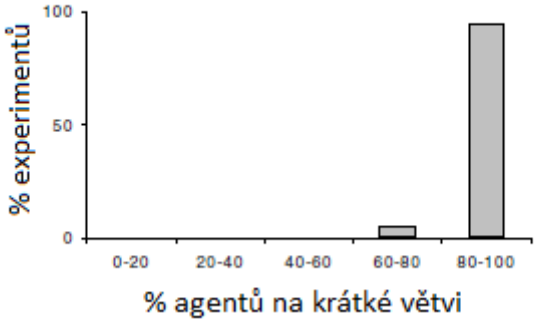
Obrázek 3: Schéma experimentu, kde je jedna větev dvakrát delší než druhá. [3]



Obrázek 4: Schéma experimentu, kde zprvu není připojena kratší větev. [3]



Obrázek 5a): Výsledek pro stejně dlouhé větve [3]



Obrázek 5b): Výsledek pro rozdílně dlouhé větve [3]

3.2 Popis algoritmu

3.2.1 Základní vlastnosti.

Optimalizace mravenčí kolonií mezi nejvýznamnější algoritmy ze skupiny tzv. optimalizačních algoritmů inspirované živou přírodou.

Jedná se o metaheuristickou techniku, která nehledá přesný výsledek, ale dostatečně dobré řešení. Jeho hlavní výhodou v porovnání s algoritmy, které hledají přesné nejlepší řešení, je znatelně zkrácená doba průběhu. V některých případech dokonce takový algoritmus ani nemusí být k dispozici. Algoritmus optimalizace mravenčí kolonií velmi flexibilní a jeho použití jsou rozsáhlá. Ať už se jedná o řešení rozvrhovacích problémů, kvadratický přiřazovací problém, sekvenční problém, problém barvení grafů nebo rozvržení studní pro kontrolu podzemní vody. [7]

3.3 Deneubourgův model

První pokusy o umělého mravence provedl tým Deneubourga v roce 1990. Tehdy ze svého pozorování a výsledků z double bridge experimentu vytvořili poměrně jednoduchý matematický model popisující chování průzkumníků. [3]

V tomto stochastickém modelu překročí ψ agentů most. Mravenci se pohybují v obou směrech konstantní rychlostí *v* (cm/s) a zároveň se zvedá hladinu feromonu o jedna. Každý mravenec se s pravděpodobností vydá buď krátkou (l_s) nebo dlouhou (l_l) větví. Agent, který se vydá krátkou stranou, dorazí do cíle za t_s=l_s/*v*. Pro delší variantu se vztah změní na $t_l = t_s \cdot r$, kde r=l_l / l_s . Množství feromonu na jednotlivých hranách se značí $\varphi_{ia}(t)$.

$$p_{is}(t) = \frac{(t_s+\varphi_{is}(t))^{\alpha}}{(t_s+\varphi_{is}(t))^{\alpha}+(t_s+\varphi_{il}(t))^{\alpha}} \tag{3.1}$$

Z toho vidíme, že pravděpodobnost v čase *t* pro kratší větev je dána množstvím feromonu a dobou přechodu větve k součtu těchto hodnot pro obě větve.

$$p_{is}(t) + p_{il}(t) = 1 \tag{3.2}$$

Tyto diferenciální rovnice popisují postupnou změnu feromonu na jednotlivých větvích.

$$d\varphi_{is}/dt = \psi p_{js}(t - t_s) + \psi p_{is}(t), \qquad (i=1, j=2;\ i=2, j=1), \qquad (3.3)$$

$$d\varphi_{il}/dt = \psi p_{jl}(t - t_s \cdot r) + \psi p_{il}(t), \qquad (i=1, j=2;\ i=2, j=1). \qquad (3.4)$$

Dají se přeložit jako změna feromonu ve větvi *s* za čas *t* v uzlu *i* se rovná toku mravenců (ψ) znásobenému pravděpodobností, že si agent vybere kratší cestu z uzlu *j* v čase $t\text{-}t_s$ plus mravenčí tok krát pravděpodobnost z uzlu *i* v čase *t*.
V tomto modelu se feromon neodpařuje a jeho intenzita je stejná jako počet mravenců, kteří si vybrali danou větev.

Při počítačové simulaci double bridge experimentu pro jeden tisíc agentů, byly výsledky velmi podobné těm, které byly zaznamenány při samotném experimentu s reálnými mravenci. [3]

3.4 Diskrétní model

Marco Dorigo se inspiroval výsledky modelu double bridge experimentu a vytvořil umělé mravence schopné pohybu po hranách grafů, a tím nalézt nejkratší cestu mezi dvěma vrcholy, hnízdem a potravou.

Tito mravenci se pohybují danou rychlostí mezi vrcholy a přitom zvyšují hladinu feromonu na hranách o jednu jednotku. Čas v tomto modelu, na rozdíl od modelu Deneubourga, je diskrétní. Stejně jako u předchozího řešení se jednotliví mravenci rozhodují o tom, na jaký následující vrchol se budou přesouvat pomocí pravděpodobností upravených podle množství feromonu na hranách k nim vedoucích.

Pravděpodobnosti pro tento model, když by se použil na double bridge experiment, by byli následující:

$$p_{is}(t) = \frac{[\varphi_{is}(t)]^{\alpha}}{[\varphi_{is}(t)]^{\alpha}[\varphi_{il}(t)]^{\alpha}} \qquad (3.5)$$

$$p_{il}(t) = \frac{[\varphi_{il}(t)]^{\alpha}}{[\varphi_{is}(t)]^{\alpha}[\varphi_{il}(t)]^{\alpha}} \qquad (3.6)$$

Zde vidíme, že nám přibyl parametr α, který pro double bridge bude mít hodnotu 2, stejně jako *r* ($r=l_l\ /\ l_s$), který je potřebný pro zjištění hladiny feromonu.

Hladinu feromonu spočítáme z následujících vztahů:

(3.7)

$$\varphi_{is}(t) = \varphi_{is}(t-1) + p_{is}(t-1)m_i(t-1) + p_{js}(t-1)m_j(t-1), \text{ (i=1, j=2; i=2, j=1)}$$

(3.8)

$$\varphi_{il}(t) = \varphi_{il}(t-1) + p_{il}(t-1)m_i(t-1) + p_{jl}(t-r)m_j(t-r), \text{ (i=1, j=2; i=2, j=1)}$$

V těchto posledních rovnicích je nová proměnná m_i(t), která nám říká, kolik mravenců se nachází na vrcholu i v čase t. Tento počet je dán tímto vztahem:

$$m_i(t) = p_{js}(t-1)m_j(t-1) + p_{jl}(t-r)m_j(t-r), \text{ (i=1, j=2; i=2, j=1)} \quad (3.9)$$

Tyto rovnice nicméně platí pouze v případě, že mravenci zanechávají za sebou feromon jak při průzkumu, tak při návratu od zdroje potravy. Jakmile by zanechávali feromon pouze na cestě z nebo do hnízda, tím by kolonie ztratila schopnost nalezení nejkratší cesty. Stejně je tomu tak i v přírodě, kde druhy mravenců, kteří vypouštějí feromon pouze při návratu od potravy, nemají tuto výhodu. [3]

Díky tomuto modelu je možné vytvořit umělé agenty, kteří jsou schopni napodobit vlastnosti reálných mravenců, co se týče hledání nejkratší cesty k potravě. Ovšem nejsme limitováni pouze na délku cesty, ale můžeme nahradit hodnoty na hranách grafu, a tak vyhledávat například nejúspornější, nejlevnější nebo nejméně náročná řešení. [6]

Toto řešení je funkční na jednoduchých grafech jako je právě double bridge experiment, ale při nasazení na složitějších příkladech může v některých případech způsobit vytváření smyček.

Ty vznikají kvůli nanášení feromonu již při průzkumu a ne jen při se vrací od svého cíle. Kdybychom pouze odstranili tuto vlastnost, ztratili bychom vyhledávání nejkratší cesty, což je náš cíl, a učinili bychom tak algoritmus bezcenný. Ovšem pokud uděláme více úprav, je možné smyčkám předejít. [3]

3.5 Simple Ant Colony Optimization - S-ACO

Od předchozích algoritmů je zde velký krok vpřed. Je zde zohledněno odpařování feromonu na hranách, přičemž Deneubourg tento jev ve svém modelu nezohlednil, protože bylo příliš pomalé při použití na double bridge experiment a nemělo by žádný vliv. Další rozdíl spočívá v přidání paměti jednotlivým mravencům. Ta slouží k uložení jejich cesty pro odstranění smyček. [3]

Samotný algoritmus se tedy skládá se tří hlavních částí - průzkum, návrat a vypařování feromonu.

Na začátku je na každé hraně základní množství feromonu $\tau_0 = 1$, abychom předcházeli dělení 0. Dále se na hranách užívá proměnná *umělá feromonová stopa* $\tau = \tau_{ij}(\mathrm{t})$, která se používá pro výpočet pravděpodobnosti p_{ij}^k neboli pravděpodobnost, že mravenec *k*, který se nachází na uzlu *i*, použije tuto hranu, a tím se přesune na vrchol *j*. [8]

Samotný vzorec pro tuto pravděpodobnost je pak následující:

$$p_{ij}^k = \begin{cases} \frac{\tau_{ij}^{\alpha}}{\sum_{l \in N_i^k} \tau_{il}^{\alpha}} & \text{pro } j \in N_i^k \\ 0, & \text{pro } j \notin N_i^k \end{cases} \tag{3.10}$$

Kde N_i^k obsahuje všechny vrcholy, které jsou s *i* přímo spojené, kromě vrcholu, ze kterého se agent dostal na *i*. Tento předchozí vrchol je dostupný, pouze pokud je jinak tato množina prázdná, a tudíž se mravenec nachází na slepém vrcholu. Takto se průzkumník pohybuje po grafu, dokud nenarazí na cílový vrchol – potravu.

Dříve než se agent vydá zpět do hnízda, jsou odstraněny smyčky z jeho cesty. Vyhledávání těchto smyček se provádí iterativním skenováním [3], při němž se postupně prohledává cesta mravence od hnízda k potravě. Smyčky se odstraňují v pořadí, ve kterém byly nalezeny, což nezaručuje odstranění největší smyčky z řešení. V případě, že se smyčky kříží a odstraní se první (kratší) smyčka, může být konečné řešení delší, než pokud by se odstranila delší smyčka.

Po odstranění smyček se průzkumník vydá po upravené cestě nazpět do hnízda a na každé hraně, kterou teď bude procházet, upraví hladinu feromonu podle 3.11.

$$\tau_{ij} = \tau_{ij} + \Delta\tau^k \tag{3.11}$$

Hodnotu $\Delta\tau^k$ v nejjednodušším případě nastavíme pro všechny mravence stejnou. Při tomto nastavení však jediné, co hraje ve prospěch krátké trasy, je to, že se mravenci, kteří se jí vydali, budou dříve vracet, a tím zvýší feromon na hranách, které použili. Tento důsledek se nazývá efekt rozdílně dlouhé cesty (*differencial path lenght*). [3]

Jiné řešení je udělat z $\Delta\tau^k$ funkci kvality zvolené cesty. Neboli mravenec bude na své zpáteční cestě vypouštět $1/L^k$ jednotky feromonu, kde L^k je délka trasy, kterou se bude mravenec *k* vracet.

Poslední důležitou částí je vypařování feromonu. Podle Deneubourga bylo vypařování feromonu u reálných mravenců tak pomalé, že ho ve svém modelu neuvedl, ale u umělých mravenců může výrazně urychlit nalezení optimální cesty, zejména u složitějších grafů.

Samotné vypařování se provádí při každé iteraci algoritmu a na všech hranách. Úbytek je pak dán vztahem 3.12.

$$\tau = (1 - \rho)\tau \tag{3.12}$$

Nastavením parametru ρ upravujeme rychlost odpařování a je dán intervalem: $\rho \in (0,1]$

Pokud by byl nastaven na 0, feromon se neodpařuje. Na druhou stranu kdybychom ho nastavili na hodnotu 1 byl by algoritmus degradován na pouhé náhodné vyhledávání. [8]

3.6 Ant systém – AS

Ant systém je základní verzí mravenčích algoritmů a byl použit například na řešení TSP. [3] Přestože jeho základní verze nebyla tak účinná jako jiné tehdy dostupné algoritmy řešící tento problém, stal se inspirací pro jeho modifikace, které výrazně zlepšují jeho efektivitu. Tyto jiné verze si většinou ponechávají jak rozhodování na vrcholech grafů, tak odpařování, ale jsou rozdílné v metodách rozhodování, kolik feromonu bude agent nanášet na hrany.

Mravenec se na vrcholu rozhoduje, na který další vrchol se vydá podle pravděpodobnosti p_{ij}^k.

$$p_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in N_i^k}[\tau_{il}]^{\alpha}[\eta_{il}]^{\beta}} \quad (3.13)$$

Před zahájením algoritmu je vhodné nastavit na hranách přiměřenou hladinu feromonu, která bude o něco málo větší, než předpokládaná hodnota, kterou budou mravenci vypouštět. Pokud bude hladina moc nízká, hrozí, že řešení bude zkreslené prvními trasami agentů. V opačném případě, když bude výchozí množství moc vysoké, nebude mít feromon umístěn mravenci v prvních cyklech algoritmu žádný význam, dokud se tento prvotní feromon neodpaří. [3]

Přibyla tu veličina η_{ij}, která reprezentuje heuristickou a-priori informaci, jež značí vhodnost přechodu z vrcholu *i* do *j*. [1]

$$\eta_{ij} = \frac{1}{d_{ij}} \quad (3.14)$$

Délku hrany, která spojuje vrchol *i* s *j* se značíme d_{ij}.

Parametrem α upravujeme výraznost feromonu, ten bývá obvykle nastaven na velikost 1. Pokud bychom ho nastavili na 0, bude algoritmus přesouvat mravence vždy do nejbližšího možného města.

Druhým parametrem β nastavujeme význam heuristické informace. Pro většinu TSP příkladů je nastaven na velikost mezi 2 až 5. Při nastavení β=0 bude mít vliv na přesun mravenců pouze feromon, což bude mít za následek to, že se po nějaké uplynulé době budou všichni agenti pohybovat po stejné trase a nebudou generovat nová řešení i v případě, že to, kterým se budou řídit, nebude optimální.
To platí zejména v případě, že α>1. [3]

U tohoto algoritmu máme dvě možnosti jak nechat mravence vytvořit svá řešení. Buď paralelní implementaci, při níž všichni agenti vytváří svá řešení současně, nebo sekvenční řešení konstrukce naopak tvoří cesty pro jednotlivé mravence jednotlivě, neboli nejprve se vytvoří trasa prvního agenta, a teprve když je úplná, tak se začne s cestou pro následujícího.

Rozdíly v řešení se ve výsledku liší zanedbatelným způsobem až na variace, kde se mění hladiny feromonu během iterace, jako např. u ACS, kde se po použití hrany odebírá část stopy.

Úprava feromonu se provádí, jakmile každý mravenec vygeneruje své řešení. Nejprve

se hladina sníží odpařováním. To je, stejně jako u S-ACO, upravováno parametrem ρ. Jeho správné nastavení umožní algoritmu zapomenout špatná řešení, a zároveň zabrání hromadění feromonu.

$$\tau_{ij} = (1-\rho)\tau_{ij} \quad \forall(i,j) \in L \tag{3.15}$$

Následně se zjistí, kolik a na jakou hranu se nanese feromonu za tento krok. Změna hladiny je závislá na kvalitě řešení. Neboli čím lepší řešení agent objeví, tím více ovlivňuje množství látky, která bude na hranu nanesena.

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{1}{C^k}, & a_{ij} \in T^k \\ 0, & a_{ij} \notin T^k \end{cases} \tag{3.16}$$

kde:
C^k je suma délky tras cesty vygenerované mravencem *k*.
T^k je množina hran trasy agenta *k*.
L je množina všech hran v grafu.

Samotný konečný feromon, který na konci kroku bude na jednotlivých hranách, je dán následující změnou.

$$\tau_{ij} = \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \quad \forall(i,j) \in L \tag{3.17}$$

Veličina *m* označuje celkový počet mravenců.

Z toho vidíme, že hrany, které jsou v řešeních často na kvalitních trasách, budou mít značně více feromonu, a tím budou upřednostněni v následujících krocích algoritmu.

Ovšem AS v porovnání s ostatními meta-heuristickými algoritmy, má tendenci značně snižovat svůj výkon. Proto byla navržena některá určitá vylepšení, aby se tento problém podařilo minimalizovat. [3]

3.7 Další varianty algoritmu

3.7.1 Elitářský mravenčí systém (Elitist ant system - EAS)

První takovou úpravou byla implementace elitismu. Tato modifikace byla úspěšně použita i u jiných výpočetních inteligencí jako například u evolučních algoritmů. [1] Hlavní princip spočívá v posílení zatím nejlepšího nalezeného řešení, čímž se algoritmus zrychlí, zároveň se ovšem vystavujeme riziku uváznutí v lokálním optimu.

Tohoto posílení nejlepšího řešení dosáhneme úpravou nanášeného feromonu, kde se na hrany, po kterých vede zatím nejlepší řešení (T^{bs}), nanese větší množství, jehož velikost je funkce kvality trasy.

$$\Delta\tau_{ij}^{bs} = \begin{cases} \frac{1}{C^{bs}}, & a_{ij} \in T^{bs} \\ 0\ , & a_{ij} \notin T^{bs} \end{cases} \tag{3.18}$$

C^{bs} je velikost celé trasy T^{bs}.

Nanášení feromonu je dále upraveno o přidání této změny znásobené parametrem e. Správným nastavením tohoto parametru dosáhneme lepších výsledků za kratší dobu. Pokud je nastaven na velikost 0, dostáváme tím základní AS. [3]

$$\tau_{\mathrm{ij}} = \tau_{\mathrm{ij}} + \sum_{\mathrm{k=1}}^{\mathrm{m}} \Delta\tau_{\mathrm{ij}}^{\mathrm{k}} + \mathrm{e}\Delta\tau_{\mathrm{ij}}^{\mathrm{bs}} \tag{3.19}$$

3.7.2 Rank-base ant system

Toto je další příklad použití elitismu v AS. Hlavní rozdíl od předešlé modifikace je zejména ten, že feromon pokládá pouze *w* mravenců, kteří nalezli nejlepší řešení. Množství feromonu, které bude následně agent nanášet na hranu, je dáno jeho pořadím. [1]
Stejně jako u EAS se i zde nanáší více feromonu na T^{bs}. Samotná úprava hran je pak následovná:

$$\tau_{ij} = \tau_{ij} + \sum_{r=1}^{w-1} (w - r)\, \Delta\tau_{ij}^{r} + w\Delta\tau_{ij}^{bs} \tag{3.20}$$

Pořadí mravence je označeno *r*.

Hodnotu $\Delta\tau_{ij}^{r}$ dostaneme použitím vzorce 6.4 a úprava feromonu pro nejlepší řešení je stejná jako u EAS.

Podle Bullnheimera má tento algoritmus mírně lepší výsledky než EAS, ale je znatelně výhodnější než pouze AS. [3]

3.7.3 Min-max ant system

Poslední úprava AS, kterou zmíním je MMAS. Liší se od předchozích zejména tím, že feromonová stopa je omezena intervalem $<\tau_{min}, \tau_{max}>$. Počáteční hodnoty feromonu na hranách se blíží horní hranici intervalu, což zároveň s pomalým odpařováním podporuje rychlé prozkoumání grafu. [1]

Feromon pokládá pouze jeden agent. Tento mravenec je buďto zatím nejlepším nalezeným řešením (T^{bs}) nebo nejlepším řešením nalezeným v této iteraci (T^{ib}). [3]

$$\tau_{ij} = \tau_{ij} + \Delta\tau_{ij}^{best} \tag{3.21}$$

$$\Delta\tau_{ij}^{best} = \frac{1}{C^{bs}} \tag{3.22}$$

$$\Delta\tau_{ij}^{best} = \frac{1}{C^{ib}} \tag{3.23}$$

Pokud se feromon nanáší podle T^{bs}, feromonová stopa na této trase se stane rychle velmi silnou a pro následující agenty neodolatelnou. Zatímco když se upravuje feromon podle nejlepšího agenta z iterace (T^{ib}), je jeho hladina rovnoměrněji rozprostřena po grafu, než aby se vytvořila jedna velmi silná stopa.

Většinou se používají oba způsoby, které se postupně střídají. Experimenty ukázali, že pro menší instance TSP je nejvýhodnější upravovat feromon podle zatím nejlepší trasy, zatímco pro větší příklady o stovkách vrcholů je nejlepší postupně zvyšovat zastoupení C^{bs} při průběhu algoritmu. [3]

Aby se zabránilo uváznutí v lokálním optimu, je zde navíc implementována ochrana, která při stagnování algoritmu, nebo pokud uplyne předem stanovený počet iterací bez změny T^{bs}, tak se hodnoty feromonu na všech hranách reinicializují do původních hodnot. [1]

3.7.4 Systém mravenčí kolonie - ACS

Tento algoritmus vychází z AS, ale na rozdíl od něj a jeho úprav, které byly víceméně orientované okolo pokládání feromonu a jeho množství, v ACS se provedli změny i v jiných částech algoritmu.

Hlavní rozdíly se dají shrnout do těchto tří částí. Zaprvé používá pseudonáhodné proporční pravidlo. Za druhé nanášení feromonu se provádí pouze lokálně na rozdíl od globální úpravy u AS. Změny na konci iterace se tedy týkají pouze T^{bs}. A nakonec, když mravenec zvolí hranu z vrcholu i do j, tak je z této hrany odebrána část feromonu. [1] [3]

Pseudonáhodné proporční pravidlo

Jedná se o kompromis mezi pseudonáhodným výběrem a rozhodováním mravenčího systému. Jsou zde dvě možné varianty, podle nichž se agent rozhodne o následujícím vrcholu.

Zneužití (*exploitation*) je dáno pravděpodobností q_0 a znamená, že mravenec se vydá po nejvíce pravděpodobné hraně. Jelikož se zde feromon ukládá pouze na T^{bs}, znamená to, že při této situaci se vydá právě po hraně náležící do této trasy, pokud je dostupná. Hodnota q_0 je z intervalu <0,1>, většinou bývá nastaven na 0,9. [9]

Průzkum (*exploration*) je rozhodování pomocí AS (7.1), neboli náhodný vrchol podle pravděpodobnosti závislé na hodnotách η_{ij} a τ_{ij}.

$$j = \begin{cases} algmax_{l \in N_i^k} \left\{ [\tau_{ij}]^{\alpha} [\eta_{ij}]^{\beta} \right\}, & q \leq q_0 \ (exploitation) \\ J, & q > q_0 \ (exploration) \end{cases} \tag{3.24}$$

Nastavením q_0 tedy rozhoduje o tom, zda bude algoritmus více prozkoumávat graf, a tím hledat nové možné trasy, nebo jestli se bude držet zatím nejlepšího řešení. J značí náhodně vybraný vrchol podle mravenčího systému pro parametry α=1 a β=2. [9]

Samotný výběr následujícího vrcholu tedy začíná porovnáním hodnot q a q_0. Proměnná q se náhodně vybere z intervalu <0,1> a podle výsledku porovnání s q_0, se rozhodne, zda se použije nejsilnější feromonová stopa, nebo jestli se použije náhodný výběr z dostupných vrcholů. [1]

Nanášení feromonu

Jak již bylo zmíněno, jedná se pouze o lokální úpravu. Takže místo úpravy feromonu na všech hranách grafu se upravují pouze hrany patřící do T^{bs}. To snižuje výpočetní složitost z O (n^2) na O (n), kde n je velikost instance. [3]

$$\tau_{ij} = (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij}^{bs}, \qquad \forall(i,j) \in T_{ij}^{bs} \tag{3.25}$$

Nanášené množství je upravováno parametrem ρ.
Hodnotu τ_{ij}^{bs} dostaneme stejně jako u EAS ze vztahu 3.18.

Úprava feromonové stopy po použití

Kromě odpařování se feromonová stopa oslabuje při každém použití. Tato hrana se stane méně atraktivní pro následující agenty, čímž se zvyšuje šance na průzkum a možnost nalezení nové T^{bs}.

Toto pravidlo se provádí vždy hned po překročení hrany při konstrukci trasy. [3]

$$\tau_{ij} = (1-\xi)\tau_{ij} + \xi\tau_0 \tag{3.26}$$

Přibyly tu dva parametry τ_0 a ξ.

Velikost ξ je dána intervalem (0,1), pomocí experimentů bylo zjištěno, že je výhodné nastavit jej na velikost 0,1.

Druhý parametr je závislý na grafu, ve kterém vyhledáváme trasu.

$$\tau_0 = \frac{1}{nC^{nn}} \tag{3.27}$$

Proměnná n je počet vrcholů v grafu a C^{nn} je délka trasy vygenerovaná pomocí nejbližšího souseda (nearest-neighbor tour). [9] Generování této cesty probíhá tak, že se vždy vybere hrana, která vede k vrcholu s nejkratší cestou. Algoritmus se nevrací do vrcholu, kterým již prošel kromě původního města. Tento proces probíhá, dokud nejsou navštívena všechna města, přičemž se vrátí do startujícího vrcholu. [12]

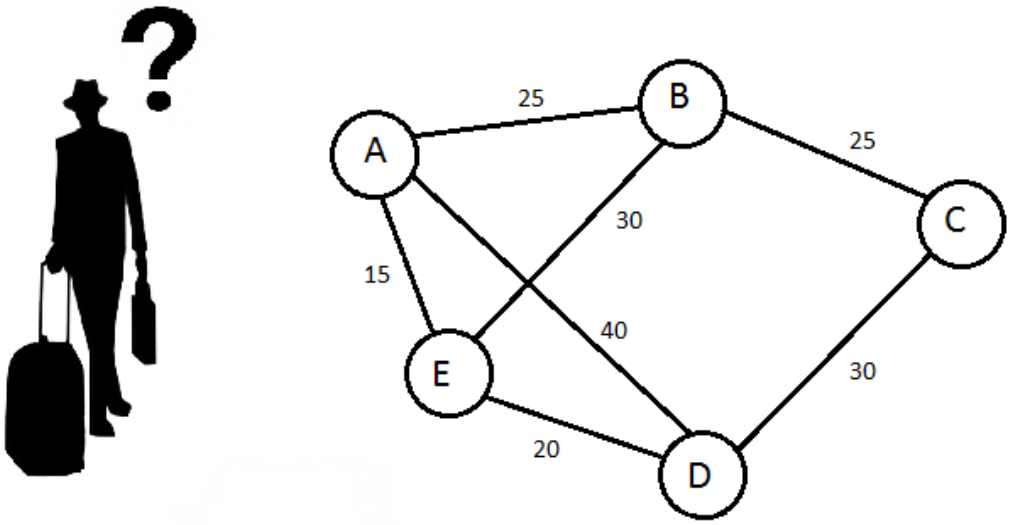
Toto pravidlo zabraňuje stagnaci algoritmu, a tím se nemusí přímo řešit jako např. u MMAS. Zároveň způsobuje rozdílné chování podle způsobu konstrukce řešení, protože se

feromonová stopa mění během iterace algoritmu, a proto paralelní a sériový průzkum probíhá jinak. Většinou se používá paralelní provedení, ale zatím nebyla experimenty prokázána výhoda jedné varianty nad druhou. [3]

4. Problém obchodního cestujícího

4.1 Formulace problému

Zadáním úlohy je vyhledat nejkratší možnou cestu mezi městy s návratem do výchozího bodu, přičemž je každé město navštíveno právě jednou (viz obrázek …). Výsledkem úlohy by měl být vypsání měst v pořadí, v jakém by měly být navštíveny, aby byly splněny podmínky zadání.



Obrázek 6: Ilustrace problému obchodního cestujícího

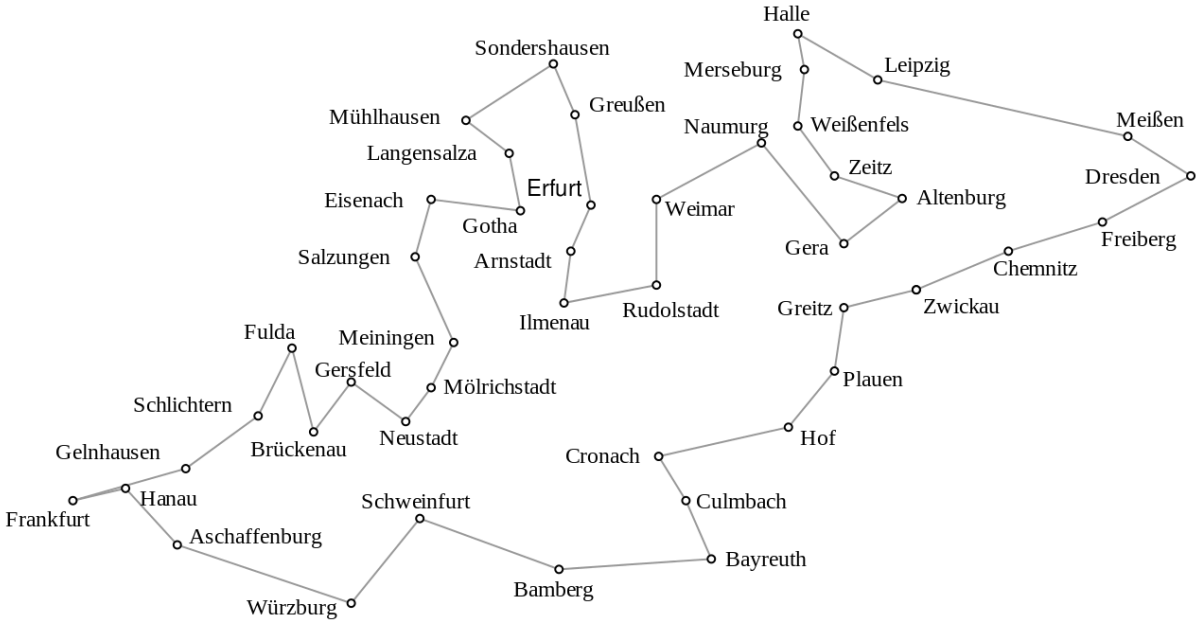
V tomto problému rozlišujeme dva základní typy podle grafu. Buď je symetrický, nebo asymetrický.

Asymetrický znamená, že vzdálenost z vrcholu A do B není stejná jako vzdálenost z B do A. Matematicky lze toto zapsat $d_{AB} \neq d_{BA}$.

První matematický výzkum tohoto problému uskutečnil roku 1934 matematik Hassler Whitney z Princetonovy univerzity. Ovšem problém obchodního cestujícího (TSP) byl již popsán předtím v německé příručce pro cestující prodejce roku 1832 pod názvem *Der*

Handlungsreisende wie er sein soll und was er zu thun hat, um Aufträge zu erhalten und eines glücklichen Erfolgs in seinen Geschäften gewiß zu sein - von einem alten Commis-Voyageur. V ní bylo uvedeno pět tras, z nichž čtyři byly organizované takovým způsobem, že některá města představovala základny, do kterých se obchodník vracel po návštěvě okolních lokací. [11]

Pátá cesta popisovala právě řešení problému obchodního cestujícího pro města v okolí Frankfurtu.



Obrázek 7: trasa z roku 1832 okolo města Frankfurt [11]

Autor kladl také důraz na plánování cesty pro tyto prodejce, protože díky důkladné přípravě trasy mohl takto obchodník ušetřit náklady a čas, a tím se cesta stávala výnosnější. [11]

Hledání nejoptimálnější cesty se nevztahuje pouze na obchodníky, ale uplatnění je velmi široké. O to rozsáhlejší je užití TSP, protože vzdálenost na hranách můžeme zaměnit za jiné veličiny, a tím hledat nejefektivnější nebo nejrychlejší řešení. Můžou se také pomocí tohoto problému analyzovat data z psychologie, rentgenové krystalografie a opravy vzduchových turbín. [27]

Jedna z nejvíce studovaných aplikací je pořadí prací pro sekvenční stroje. U tohoto problému je na hranách místo vzdálenosti cena C_{ij} , která představuje náročnost přenastavení stroje z úkonu *i* na *j*. To se využívá zejména u různých linek, kde při přecházení na výrobu jiného produktu se musí pozměnit některé části výrobního procesu nebo vyměnit materiál. Po skončení všech prací se stroj vrací do původního nastavení, takže je třeba přidat cenu této

změny (C_{j1}). Cílem pak je, aby se linka při přechodu na jiný výrobek měnila co nejméně, a tím se ušetřil čas. [14]

V dopravních firmách může správná optimalizace tras znamenat významnou výhodu nad konkurencí na trhu. Ať už se jedná o různé rozvozy zboží, svozy matriálu, poštu, trasy autobusů, tak i další podobná užití. Nejenže za stejný čas je řidič schopen vyřídit více práce, ale zároveň se při správně navržené cestě ušetří pohonné hmoty, a tím se stává toto řešení pro zaměstnavatele výnosnější.

Některé překážky, které firmy řeší pomocí optimalizace se od TSP liší ve specifických úpravách, a tím vznikají nové matematické problémy, jako například problém čínského listonoše a problém okružních jízd.

4.2 Hledání řešení

Problém obchodního cestujícího řadíme do skupiny NP - těžkých problémů. To znamená, že neexistuje přesný algoritmus, který by tento problém dokázal vyřešit pro větší instance v rozumném čase. Nejlepší algoritmus pro přesné řešení s garancí vyvinuli v roce 1962 Held a Karp s výpočetní složitostí O ($n^2 2^n$). [10]

Kvůli velké časové náročnosti pro větší objem dat se používají heuristické algoritmy, ty ovšem nemusí najít nejlepší řešení, ale poskytují dostatečně dobré výsledky za značně kratší dobu. Mezi níže zmíněné možné způsoby řešení patří i optimalizace mravenčí kolonií.

4.2.1 Metoda nejbližšího souseda

Funkce tohoto algoritmu byla zmíněna již výše v kapitole o ACS. Jedná se o rychlý a jednoduchý algoritmus (O (n^2)). [12] Přestože se jedná o způsob optimalizace, velmi často uvázne v lokálním optimu.

4.2.2 Genetický algoritmus

Tento algoritmus patří mezi časově náročnější algoritmy. Jedná se o metaheuristiku. Stejně jako v případě optimalizace mravenčí kolonií je i tento algoritmus odvozen z přírodních dějů. Patří do evolučních algoritmů, které se snaží reprodukovat evoluční procesy z biologie.

Hlavními znaky tohoto algoritmu jsou křížení a mutace, pomocí kterých vznikají nová řešení. Na začátku procesu se začíná s náhodně vygenerovanými jedinci populace (řešení), u těch se zjistí jejich kvalita neboli fitness, pomocí které se určí jedinci, kteří se

budou dále reprodukovat a tím vytvářet potomky. [12]

Samotní potomci vznikají křížením. To probíhá tak, že řetězec rodičů se rozdělí na náhodně vybraném místě a část se prohodí s řetězcem jiného rodiče, a tím se vytvoří dva potomci. Toto rozdělení může proběhnout na jednom (*single-point crossover*) nebo více místech. Experimentovalo se hlavně se dvěma body křížení (*two-point crossover*), protože rozdělení pouze v jednom bodě nemůže vytvořit všechny možné variace, a zároveň schémata s dlouhou délkou mají tendenci být zničena tímto způsobem křížení. Další možností je uniformní křížení (*uniform crossover),* kde každý bit řetězce má pravděpodobnost *p*, že se použije u vytvoření potomka. Tím může vzniknout jakákoliv kombinace z rodičů, ale tento způsob je velmi rušivý pro všechny vzory v datech. [13]

Jakmile jsou vytvořeni potomci, je u každého malá šance na mutaci. Hlavní cíl mutace je zabránění stagnování algoritmu v bitech, které by se křížením neměnily. U těchto nových potomků se vypočítá fitness a vše se opakuje, dokud nedostaneme dostatečně dobré řešení (potomka, který bude odpovídat předem dané kvalitě), po určité době nebo po předem daném počtu generací.

Stejně jako u ACO se i zde zanedlouho začal používat elitismus. To znamená přidání rodiče s nejvyšší způsobilostí fitness s jeho potomky do vytváření nových potomků. Tím se zabezpečilo, že algoritmus nemohl postupem času generovat horší výsledky. [13]

4.3 Varianty úlohy obchodního cestujícího

Varianty původní úlohy vznikly ze situací z reálného života a potencionálních aplikací. Většinou se liší od TSP pouze drobnými úpravami.

4.3.1 Hledání maximální cesty obchodního cestujícího

Na rozdíl od standardní úlohy obchodního cestujícího, kde se hledá nejkratší cesta mezi všemi vrcholy, se v této variantě hledá nejdelší možnou trasa, kterou se může obchodník vydat. Toho lze dosáhnout negací velikosti hran. [14]

4.3.2 Zúžený problém obchodního cestujícího

V tomto případě hledáme trasu grafem G tak, aby nejdelší hrana v řešení, byla co nejkratší. Jednou z možností jak toho dosáhneme je umocněním velikosti hran před zahájení algoritmu. [14] Využití může mít například u aplikací, kde jsme omezeni maximální cenou,

kterou je agent schopen zaplatit. Jeden z příkladů jsou rozvozy zboží, které má omezenou trvanlivost. [28]

4.3.3 Úloha kurýrní služby

Známí také jako *wandering salesman problem*. Tento problém hledá nejkratší hamiltonovskou cestu z vrcholu A do B. To získáme pomocí TSP úpravou ceny hrany (A,B) na velkou zápornou hodnotu (–M). Pokud nemáme dané vrcholy a pouze chceme vypočítat nejkratší hamiltonovu cestu v grafu, můžeme k tomu také použít TSP, ale nejprve se musí upravit graf přidáním nového vrcholu, který bude spojen se všemi ostatními vrcholy a cena spojujících hran je –M. Pokud je graf orientovaný, musíme pro každý vrchol připojit dvě hrany s opačným směrem. Z optimálního řešení takto modifikovaného grafu lze dostat řešení původního problému. [14]

4.3.4 Úloha obchodního cestujícího s možností víckrát navštívit města (TSPM)

V této variantě musí výsledná trasa procházet každým městem alespoň jednou a skončit stejně jako u standardní úlohy TSP v prvním vrcholu. Úprava grafu pro takovýto způsob spočívá v přidání nejkratších tras vypočítaných podle jiných algoritmů. [14]

5. Problém okružních jízd

5.1 Problém

Problém okružních jízd řeší optimalizaci trasy rozvozu a svozu pomocí většího počtu vozidel tak, aby nebyly porušeny žádné předem stanovené konstanty, přičemž vozy vždy vyjíždí z jednoho nebo více dep. Mezi tyto konstanty patří zejména velikost, váha, typ a kapacita vozidel. Ty jsou důležité kvůli možným omezením na cestách, ať už se jedná o různá omezení v centrech měst, nemožnost překročení limitů mostů, podjezdů, tunelů a podobně. [7]

Kapacita nám udává, jaké množství je vozidlo schopno přepravit do nebo z depa. Žádná zakázka nemůže být rozdělena na více vozů. Při řešení tohoto problému na velké vzdálenosti je vhodné brát v potaz maximální pracovní dobu řidiče.

Vozový park, který obsahuje pouze jeden typ vozidla, se nazývá homogenní, naopak pokud se alespoň dvě vozidla od sebe liší, jedná se o park heterogenní.

Dále je možné, aby doba potřebná k dodání, byla ovlivněna hodinou a dnem. Donášky, které mají být provedeny během dopravní špičky v centru města ke konci pracovního týdne, budou vyžadovat více času, než donášky do té samé destinace mimo dopravní špičku.

Pokud nakládání a vykládání může zabírat nezanedbatelný čas, je možné ho také zohlednit při výpočtu trasy. Tento čas se nazývá *service time*. [7]

Graf pro tento problém má *n*+1 vrcholů, vrchol 1 je depo a *n* je počet zákazníků. Zákazníci jsou pak zbylé vrcholy $S_c = \{2,3, \dots, n+1\}$. Každý zákazník *i* má předem známý objem zboží q_i. Pokud je vozový park homogenní, tak počet vozidel je *m* $S_v = \{1,2, \dots, m\}$. Každé takové vozidlo má kapacitu Q. Objem objednávek zákazníků nemůže překročit maximální objem, který je možný vozidly rozvést (*m*Q). Protože zásilka má být vyřízena jedním vozidlem při jedné návštěvě vrcholu, je nutné, aby platil vztah $q_i \leq Q,\ i \in S_c$. [15]

Tento problém byl poprvé představen v práci autorů G. B. Dantziga a J. H. Ramsera roku 1959, v němž řešili nejoptimálnější trasu pro auta rozvážející benzín. [22]

5.2 Aplikace

Jelikož se jedná o specifičtější problém, než jakým byl předešlý problém obchodního cestujícího, nachází uplatnění zejména u firem zaměřených na transport.

Jen v Evropské unii dopravní sektor generuje více než 10% HDP a zaměstnává přes 10 miliónů lidí.

V Norsku roku 2002 existovalo 17 000 dopravních firem s obratem 44 miliard norských korun, ale využito bylo pouhých 46,7% kapacity. Jedním z hlavních důvodů tak malé efektivity je geologická nerovnoměrnost zavážek, přičemž velkou roli hrála i nesprávná optimalizace. [19]

Z těchto čísel vidíme, že nesprávná volba tras může mít nemalé důsledky, nejen pro dodavatele, ale ovlivňuje též jeho ceny, a tím pádem postihuje všechny ostatní zákazníky, ať již přímo (cena jízdenek) nebo nepřímo (cena zboží v obchodech).

Mezi hlavní problémy, jež spadají pod VRP, patří rozvozy jídel, novin, balíků, pošty, materiálů pro výrobní závody, zboží do supermarketů, doplňování výdejních automatů, svoz odpadu, vytváření tras pro autobusy a vlaky, svoz dětí pomocí školních autobusů a mnohé další. [7][17][18]

5.3 Hledání řešení problému okružních jízd

Stejně jako problém obchodní cestujícího i problém okružních jízd je NP - těžký problém. Přestože máme k dispozici postupy pro přesné řešení, nejsme schopni dostat výsledek v rozumném čase pro větší instance dat. Mezi metody pro přesné řešení patří například generování sloupců, větvení a řezů a Lagrangeovské relaxace. V závislosti na metodě můžeme tyto postupy použít na 50 – 100 zákazníků. [19]

Kvůli tomuto omezení se pro praktické aplikace používají metaheuristické algoritmy jako zakázané prohledávání, optimalizace mravenčí kolonií, evoluční a genetické algoritmy a heuristiky, zejména lokální vyhledávání. [7][19][21] Pokud máme malý objem zákazníků, které je třeba navštívit, je možné použít některé z přesných řešení. [19]

Termín metaheuristika byl definován ve stejné práci jako zakázané vyhledávání. Jedná se o metodu, která upravuje a navádí jinou heuristickou metodu, a tím může dosáhnout lepších výsledků. [25]

5.3.1 Lokální vyhledávání

Tento typ algoritmu nevytváří řešení, ale prohledává množinu možných řešení s cílem nalézt dostatečně dobré řešení. Vždy prohledává sousedy momentálního řešení.

Definice souseda závisí na řešeném problému, u problému obchodního cestujícího a problému okružních jízd se jedná o změnu použitých hran grafu, jinde se může jednat o přesouvání, prohazování nebo nahrazování částí řešení u jiných problémů. [23] Pomocí tohoto způsobu lze nalézt jak nejkratší, tak nejdelší trasu.

Pokud změníme pouze dvě hrany, jedná se o takzvaného souseda dvou změn (*2-change neighborhood)*. [24] Řešení A, B jsou sousedi, pokud můžeme z řešení A předem definovanou změnou dostat řešení B. U souseda dvou změn je tato změna definována jako změna dvou hran.

Kritérium, podle kterého se určuje kvalita řešení, se stejně jako u definice souseda liší problém od problému, pro VRP se využívá suma délek hran.

Algoritmus prohledává jednotlivé sousedy momentálního řešení, dokud nenajde lepší řešení, a tím to stávající nahradí. Tento způsob se nazývá gradientní algoritmus, známý také jako horolezecký algoritmus (*hill climbing algorithm*). Velkým rizikem při používání takového postupu je uvíznutí v nedostatečném lokálním optimu. [23] To se dá do jisté míry

omezit povolením přesouvání na horší řešení nebo vícero běhů algoritmu s jinou počáteční hodnotou.

Algoritmus se opakuje, dokud nedostane dostatečně dobré řešení, nebo dokud neproběhne předem stanovený maximální počet kroků. [23]

5.3.2 Zakázané prohledávání

Jedná se o metaheuristiku, která navádí lokální vyhledávání a umožňuje mu tak prohledávat i za hranice lokálního optima. Dosahuje toho implementací paměti a responzivního průzkumu. [26]

V paměti nejsou zaznamenávána celá řešení, ale pouze nedávné změny. Této paměti se také říká tabu list. [25] Jak ze jména vyplývá, tyto listy obsahují nemyslitelná řešení – nedávno použitá. Následující řešení se tedy nevybírá z množiny sousedů řešení x ($N_{(x)}$), ale z upravené množiny $N^*_{(x)}$, ve které se nevyskytují sousedi, kterých by se dosáhlo úpravami uloženými v tabu listu. V této nové množině se mohou vyskytovat i taková řešení, která nebyla v původní $N_{(x)}$. Mezi nově zařazené možnosti patří dříve použitá řešení, popřípadě dobře hodnocení sousedi těchto starších řešení. To znamená, že sousedi řešení x nejsou statičtí, ale dynamičtí. [26]

Klíčovým aspektem je správné využívání paměti mezi vyhledáváním nových řešení a využíváním dobrých řešení nalezených v dřívějším průzkumu algoritmu. [26]

5.4 Varianty

Protože VRP řeší zejména problémy dopravy pro různé firmy a většina těchto problémů je specifická pro daný obor, bylo navrhnuto velké množství modifikací. Některé tyto úpravy se dají kombinovat pro řešení komplexnějších problémů, které jsou blíže praktickému použití. Někdy jsou označované jako „rich VRPs“. Většinou se jedná o zařazení více dep, vozidla musejí provést více tras, rozdílné kapacity vozidel a jiné. [21]

5.4.1 Kapacitní VRP

Základní verze VRP je CVRP (*capacitated* VRP**)**. Doručení je omezeno pouze kapacitou vozů. K dispozici je jen jedno depo a vozový park je homogenní. Najít přesné řešení v rozumném čase jsme schopni najít pouze do počtu 50 zákazníků. [7]

5.4.2 VRP s časovými okny

Časová okna určují časový úsek, kdy má být předmět dopraven. Okno může být jedno nebo více pro jednu lokaci. Rozdělujeme je na dva typy. První představuje takové časové okno, které není možné nedodržet. Ta nazýváme tvrdá (hard) okna. Pokud je možné okno ignorovat, jedná se o měkká (soft). Pokud se měkké časové okno nedodrží, bude nastavena sankce. V reálném světě tuto situaci představuje například rozvoz pizzy, pokud nebude doručena do 30 minut, bude jídlo zdarma a jiné.

Ke kapacitě z CVRP a samotným časovým oknům se při řešení tohoto problému zohledňuje i *service time*. Pro zákazníka *i* je interval dán $[a_i,\ b_i]$ a service time s_i. [7]

5.4.3 VRP s vyzvednutí a doručení

Pokud se musí balíky před dodáním vyzvednout, to znamená, že se nenakládají jako u předchozích verzí v depu, ale u jiných zákazníků, použije se tato modifikace. Vždy zároveň obsahuje časová okna. Ta se mohou vztahovat jak k vyzvednutí, tak k donášce, případně lze nastavit časy pro obě operace. [17]

5.4.4 Vehicle scheduling problems

Řešením tohoto problému je trasa, která se má opakovat vždy v daném intervalu. Jedná se zejména o plánování tras autobusů, vlaků a jiných dopravních prostředků, ať už se jedná o městkou, státní nebo mezinárodní dopravu.

5.4.5 Otevřené VRP

Od CVRP se liší pouze v tom, že se vozidlo po obsloužení posledního zákazníka nevrací do depa. [16] Využití nalezne například u firem, které nevyžadují, aby jejich zaměstnanci vraceli pracovní vozidla do firemních garáží (PPL, Essa s.r.o., a podobné).

5.4.6 VRP se zpětným sběrem

Jedná se o rozšíření CVRP, kde jsou vrcholy rozděleny do dvou skupin. První skupina (*linehaul* zákazníci) chtějí, aby jim bylo doručeno předem známé zboží. Naopak *backhaul* zákazníci chtějí, aby se od nich vyzvedlo. Než může vozidlo nakládat od *backhaul* zákazníků, musí nejprve vyložit všechny dodávky. [17]

5.4.7 VRP s heterogenním vozovým parkem

Hlavní cíl tohoto problému je nalézt co nejvhodnější vozidlo pro daný úsek z limitovaného vozového parku. [17]

5.4.8 VRP s rozdělenou dodávkou

Pokud je zásilka větší než kapacita dostupných vozidel, je třeba ji rozdělit. V některých případech může být rozdělení i malých zásilek, být výrazně znát na konečných nákladech. Podle Drora a Trudeaua je možné jakoukoliv zásilku rozdělit na více menších zásilek. [17]

5.4.9 Periodický VRP

Pokud zákazník chce, aby se mu dodávalo zboží opakovaně během vybraných dnů v týdnu, musí se takovéto plánování rozdělit na dvě části. Nejprve se navrhne tzv. rozvrh návštěv, který určuje, v jaké dny se provedou donášky pro jakého zákazníka. Poté se vyřeší trasa pro každý den zvlášť. Tato modifikace se používá u různých firem, které rozvážejí materiály pro výrobní závody a jiná zařízení. [20]

Poznámka k Periodickému VRP a Periodickému VRP with Service Choice

V případě VRP *with service choice* se jedná o rozšíření PVRP, kde se frekvence návštěv zákazníků získává pomocí modelu. Každý zákazník může mít jiné nároky, ať už jak časté mají dodávky být, či pokud jsou vůbec ochotní za častější dodávky doplácet. Správným modelem tedy můžeme zabránit zbytečnému zásobování lokací, které to nevyžadují a zároveň zamezit nedostatku zboží ve více vytížených obchodech. [20] Neboli frekvence zásobování se mění v závislosti na změnách poptávky. Tento problem se dá považovat za mezistupeň PVRP a IRP. [17]

Tyto problémy spadají pod kategorii opakovaného zásobování. Mezi podobné problémy patří *Inventory routing problem* (IRP), který se od PVRP liší zejména v tom, že zákazník nepodává objednávku. Funguje na principu doplňování zboží tak, aby cílová lokace nikdy nebyla bez zásoby. Sama společnost rozhoduje, jak velkou dodávku a kdy ji pošlou. Kvůli tomu musí být trasa navržena pro každý den zvlášť, přičemž závisí na zákaznících, kteří vyžadují doplnění zásob a na množství, které je třeba doplnit.[17]

Od toho se dostáváme k *vendor managed inventory* (VMI) neboli inventář řízený dodavatelem. Dodavatel je tedy zodpovědný za dostatek zboží na skladě, a zároveň za včasné doplnění zásob. Tím má více svobody a sám si určuje, které zákazníky a kdy

obslouží. Je tedy možné naplánovat trasy a objem zboží tak, aby se pokrylo více zákazníků poblíž sebe nebo vyslat plné vozidlo do jedné lokace, a tím snížit náklady na přepravu. Zákazník na druhou stranu ušetří náklady na uskladnění přebytečného zboží, protože si nastavuje kvóty, jaké má dodavatel dodržovat. Pro zajištění fungování systému potřebuje dodavatel přístup do záznamů zákazníka, aby mohl správně reagovat na změny v inventáři. Průkopníkem v tomto způsobu doplňování zboží do skladů byl Wal-Mart, dnes používaný například ve firmách Shell Chemical, Fruit of the Loom a jiných. [18] Hlavní uplatnění VMI nalézá v doplňování zboží v supermarketech, v udržování dostatku materiálu pro výrobní linky, v zásobování výdejních automatů a v dopravě paliv do benzínových pump. [17]

6. Aplikace Optimalizace mravenčí kolonií na svoz odpadu v Českých Budějovicích

Optimalizace mravenčí kolonií bude aplikována na část Českých Budějovic, přesněji na Suché Vrbné. Bude použita optimalizace systémem mravenčí kolonie (kapitola 3.7.4).

6.1 Analýza řešené problematiky

6.1.1 Stávající postup sběru tříděného odpadu

Úkolem sběrného vozidla je během týdne vyprázdnit všechny kontejnery na tříděný odpad. Řidič vozidla dostane k dispozici seznam sběrných míst. Na seznamu je napsaná ulice s čísly jednotlivých kontejnerů v této ulici. Po navštívení daného místa řidič položku ze seznamu odškrtne a pokračuje k dalšímu sběrnému místu.

Nezáleží tedy na pořadí navštívení jednotlivých sběrných míst. Jde pouze o to jednou za týden každé sběrné místo navštívit a kontejnery na tomto místě vyprázdnit. Seznam ulic a popisných čísel kontejnerů může být náročný na zvládnutí pro řidiče začátečníky. Zkušenější řidiči již znají většinu sběrných míst nazpaměť a mají vytvořený systém svozu. Systém svozu spočívá v rozdělení města do jednotlivých destinací. Po svezení odpadu z jedné destinace se postupuje k destinaci další. Obvykle se také nezabývají jakou cestou se dostat z jednoho sběrného místa do druhého a kde jak zacouvat. Řidiči většinou vycházejí ze zkušeností, které získali dlouhodobým objížděním stále stejných sběrných míst.

6.1.2 Konceptuální návrh metody zlepšení efektivnosti svozu

Obecně lze říci, že svoz odpadu je úloha podobná problému obchodního cestujícího. Sběrná místa odpadu představují jednotlivá města a vzdálenosti mezi sběrnými místy představují jednotlivé trasy. Úloha je navíc komplikovanější tím, že kapacita sběrných vozidel je omezená. To způsobuje nutnost odjezdu vozidel na skládku k vyprázdnění. Pokud by byla kapacita vozidla neomezená a bylo by možno posbírat všechny kontejnery během jedné cesty, jednalo by se pouze o problém obchodního cestujícího. V našem případě musíme do navrhovaného řešení přidat omezení plynoucí z omezené kapacity svozových vozidel. Snažíme se použít na svoz co nejmenší počet vozidel, takže při tvorbě řešení vozidlo po naplnění kapacity odjíždí na skládku a teprve pak vyjíždí další vozidlo, které ví, jaké kontejnery byly naloženy a vytváří trasu pro vyzvednutí zbývajících lokací.

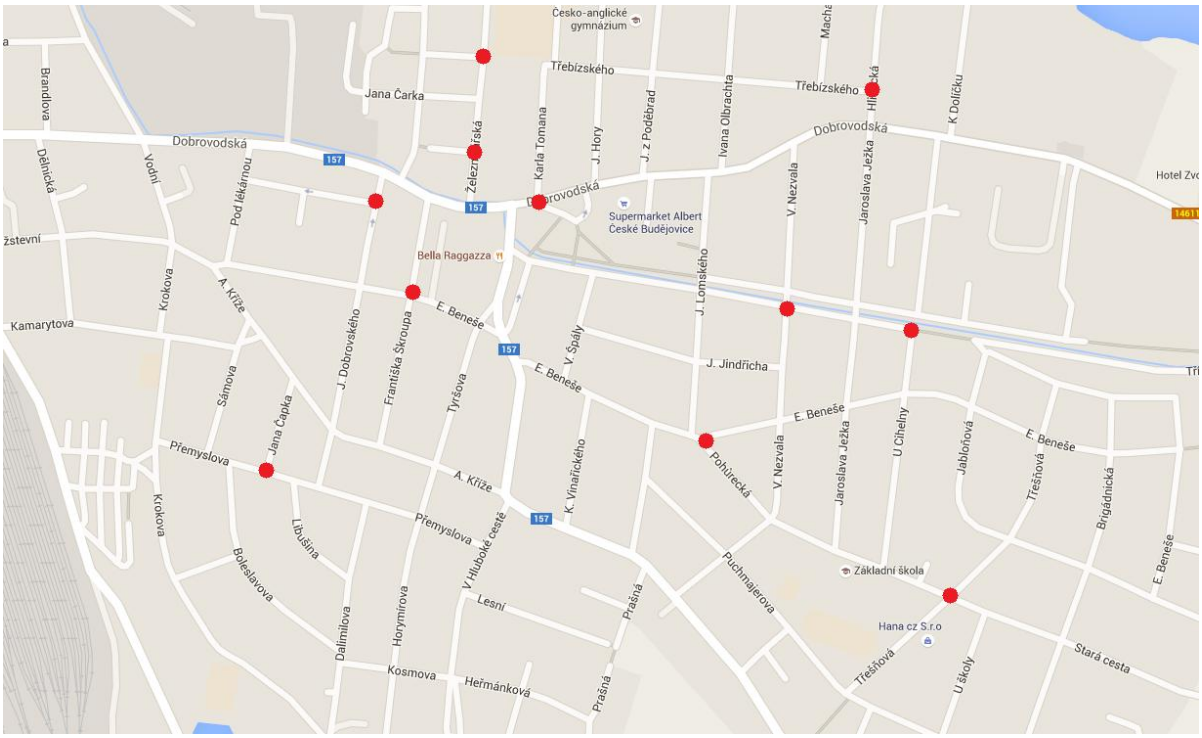
Navrhovaná metoda bude na základě předchozího teoretického rozboru založena na využití optimalizace mravenčí kolonií se zapracováním výše uvedených omezení. Výsledkem celé práce bude programu pro nalezení efektivního způsobu svozu tříděného odpadu.

Tento program bude mít dva možné způsoby zadání zaplnění kontejnerů. Buď může být nastavené pro všechny kontejnery stejné, a to v maximálním zaplnění, nebo přesné zaplnění pro každý zvlášť. První možnost je vhodná pro naplánování statické trasy, protože nemůže nastat situace, při které by tato trasa nebyla platná, avšak málokdy jsou všechny kontejnery zaplněné, a tím se plýtvá nepoužitý nákladový prostor. Druhý případ je použitelný, pokud máme nějaký způsob zjištění zaplnění kontejneru předtím, než se začne navrhovat trasa. Jednou z možností jsou tzv. chytré koše, které mají v sobě čidlo, jenž pokud bude koš zaplněný, vyšle signál se žádostí o vysypání. Koše, které nebudou vysílat tento signál, se budou jevit jako prázdné (zaplnění je nulové) a nebudou brány v potaz při tvorbě trasy. Tato technologie je využívána například v centru Prahy. Nevýhodou je zde pořizovací cena těchto odpadkových košů v porovnání s běžnými koši, navíc pokud chceme využít výhody, které poskytují, je nutné generovat nové trasy pro každý den zvlášť. Další možností je dlouhodobě zjišťovat zaplnění kontejnerů při nakládání a vyhotovit pravděpodobné zaplnění a to použít pro následující sběry. Tím můžeme předejít zbytečnému plýtvání nákladového prostoru. Hlavními nevýhodami této metody jsou možné výchylky od dosud získaných dat a nutností získávání nových dat při jakékoliv změně sběrných míst, ať už se jedná o přidání nebo odstranění kontejneru či změně lokace některého ze stávajících.

Naším cílem bude tedy vypracovat program, který se dá aplikovat nejen na svoz separovaných odpadů a odpadů komunálních, ale s drobnými úpravami i na ostatní podobné úlohy, které se zabývají svozem či rozvozem.

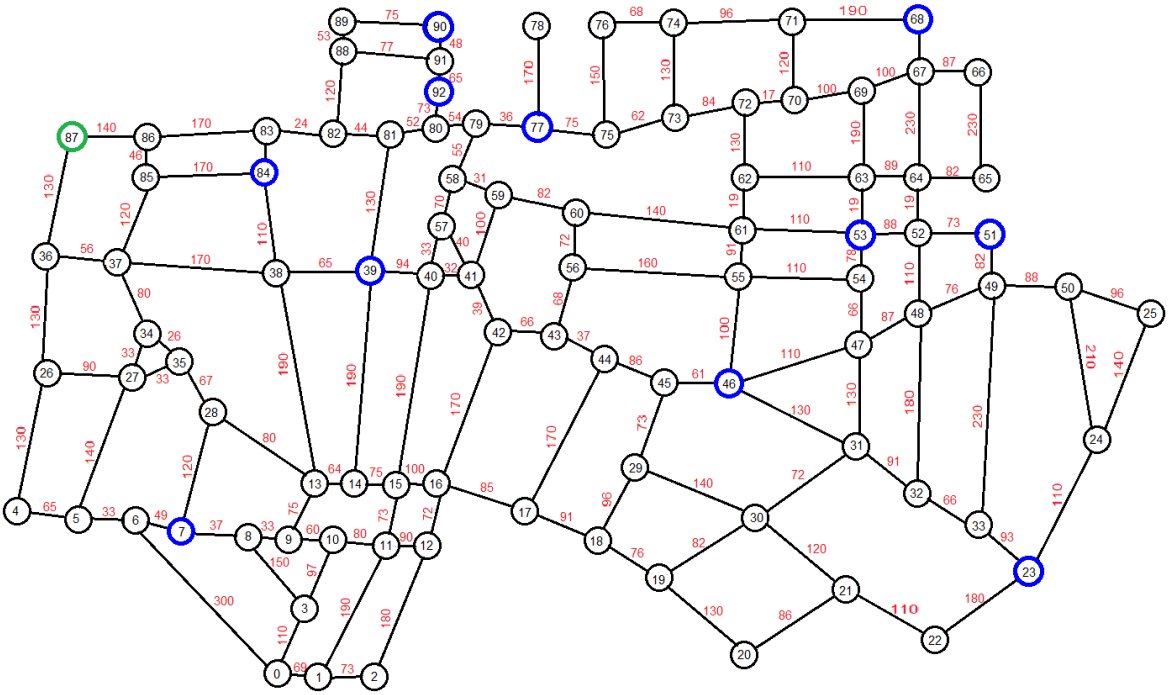
6.2 Získání dat

Prvním úkolem byl sběr dat o přibližné geografické poloze jednotlivých sběrných míst a zaznamenání kapacity těchto míst. Sběr dat byl zaměřen na veškerý tříděný odpad v lokalitě Suché Vrbné v Českých Budějovicích. Názvy ulic a vzdálenosti mezi křižovatkami byly získána z internetu a to pomocí údajů z mapy ze stránky https://www.google.cz/maps/@48.9689523,14.5063383,16.25z. Lokality kontejnerů na tříděný odpad má na svých stránkách dostupný magistrát města České Budějovice. http://www.c-budejovice.cz/cz/magistrat/odbory/osvs/stranky/nadoby-na-trideny-odpad.aspx GPS souřadnice jednotlivých křižovatek a sběrných míst byly získány z http://mapa.cz/gps-souradnice-m41.
Po získání umístění jednotlivých kontejnerů, jsem je namapoval na lokalitu, kde budu provádět optimalizaci svozu.



Obrázek 8: Mapa vybrané lokality (Suché Vrbné v ČB a umístění kontejnerů

Z této mapy získáme přehled o cestách a vzdálenostech mezi jednotlivými křižovatkami. Zároveň slouží jako předloha pro graf, který budeme následně použit pro výpočet – viz obrázek 9.

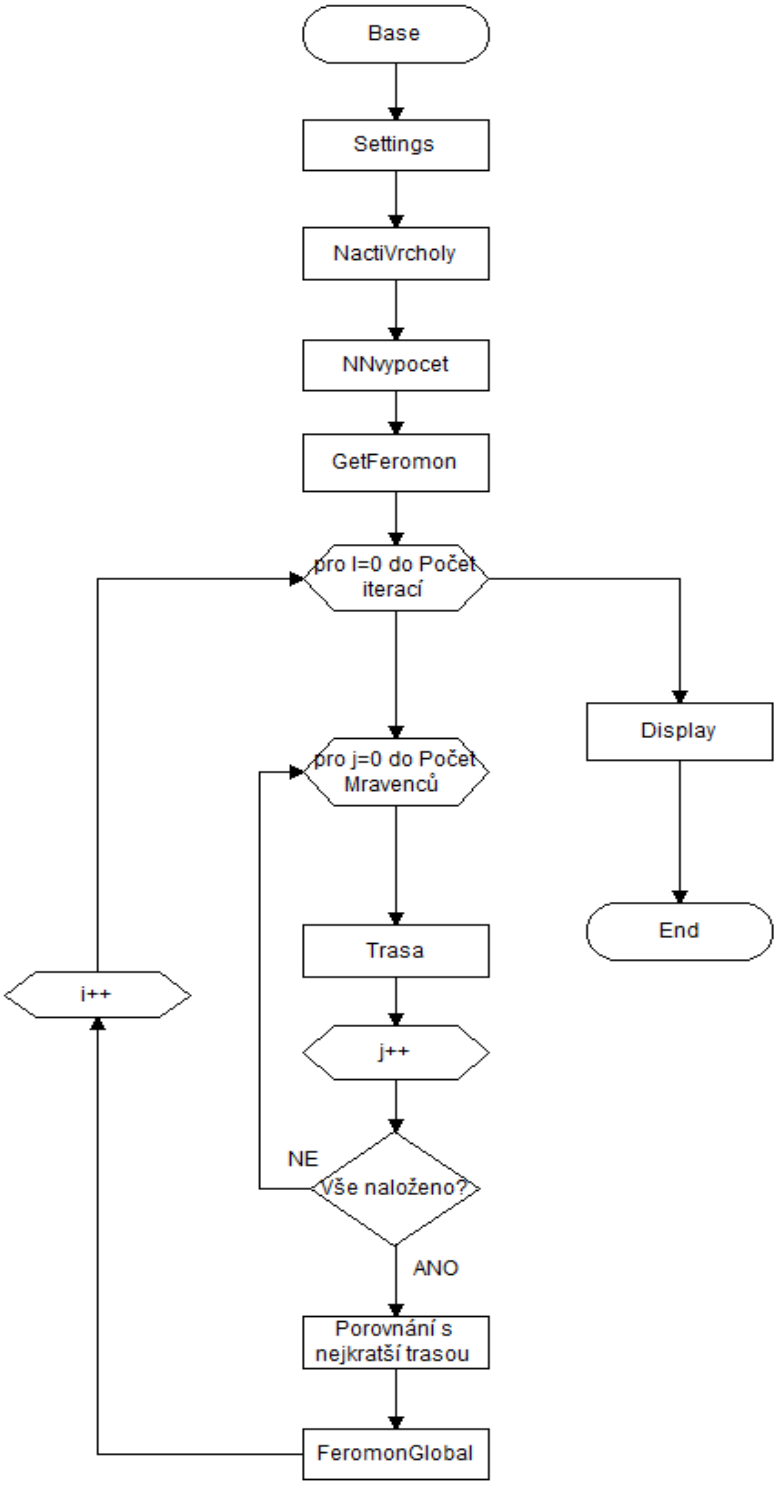


Obrázek 9: Schéma ulic a umístění kontejnerů – výsledný graf pro výpočet svozu

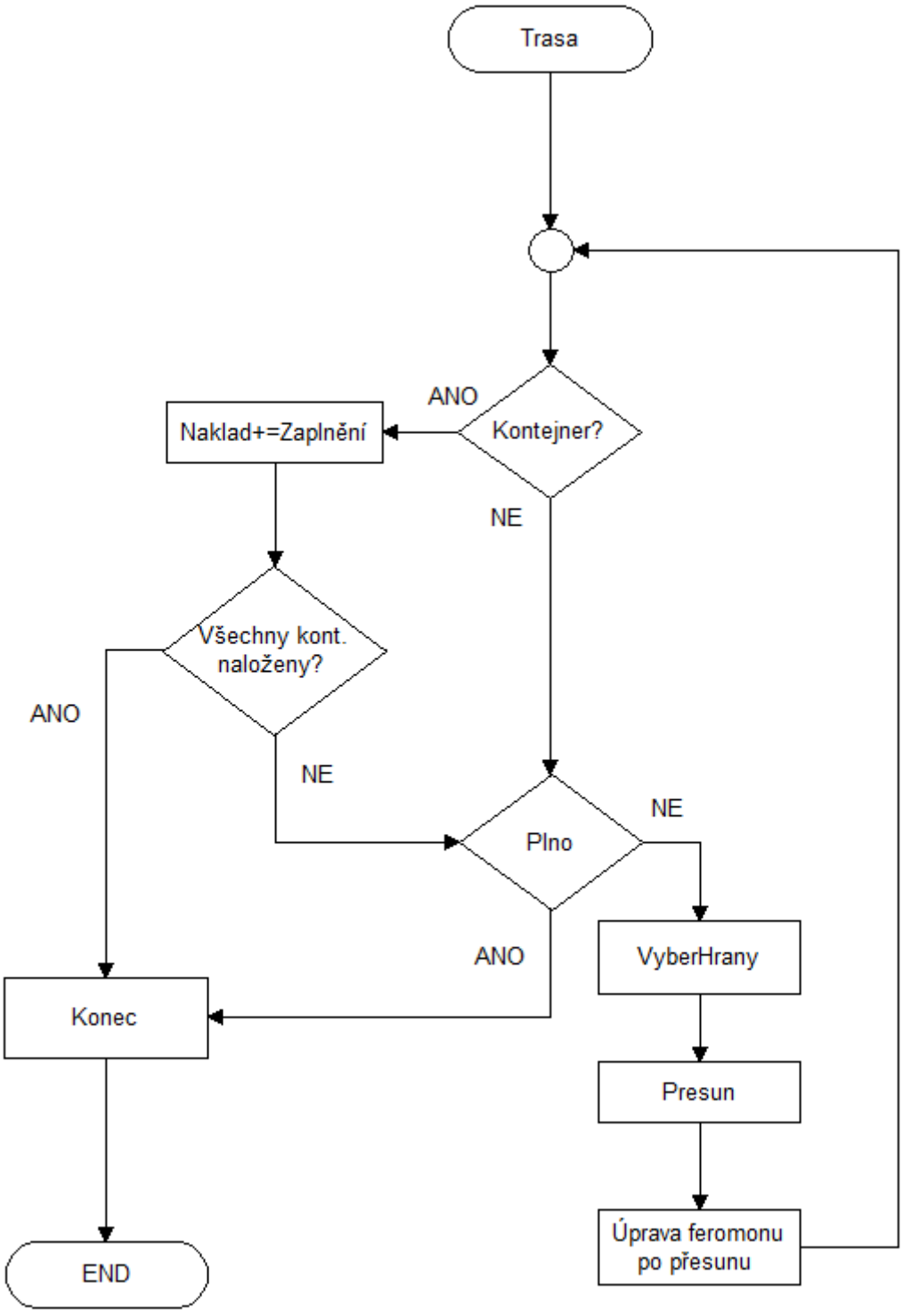
Velikost hran je dána hodnotami získanými z obrázku 8 pomocí http://maps.google.com. Délky mezi jednotlivými vrcholy jsou udány v metrech. Modře označené vrcholy reprezentují sběrné kontejnery a zelený vrchol značí výchozí bod pro všechna svozová vozidla.

6.3 Návrh řešení algoritmu

6.3.1 Diagram programu

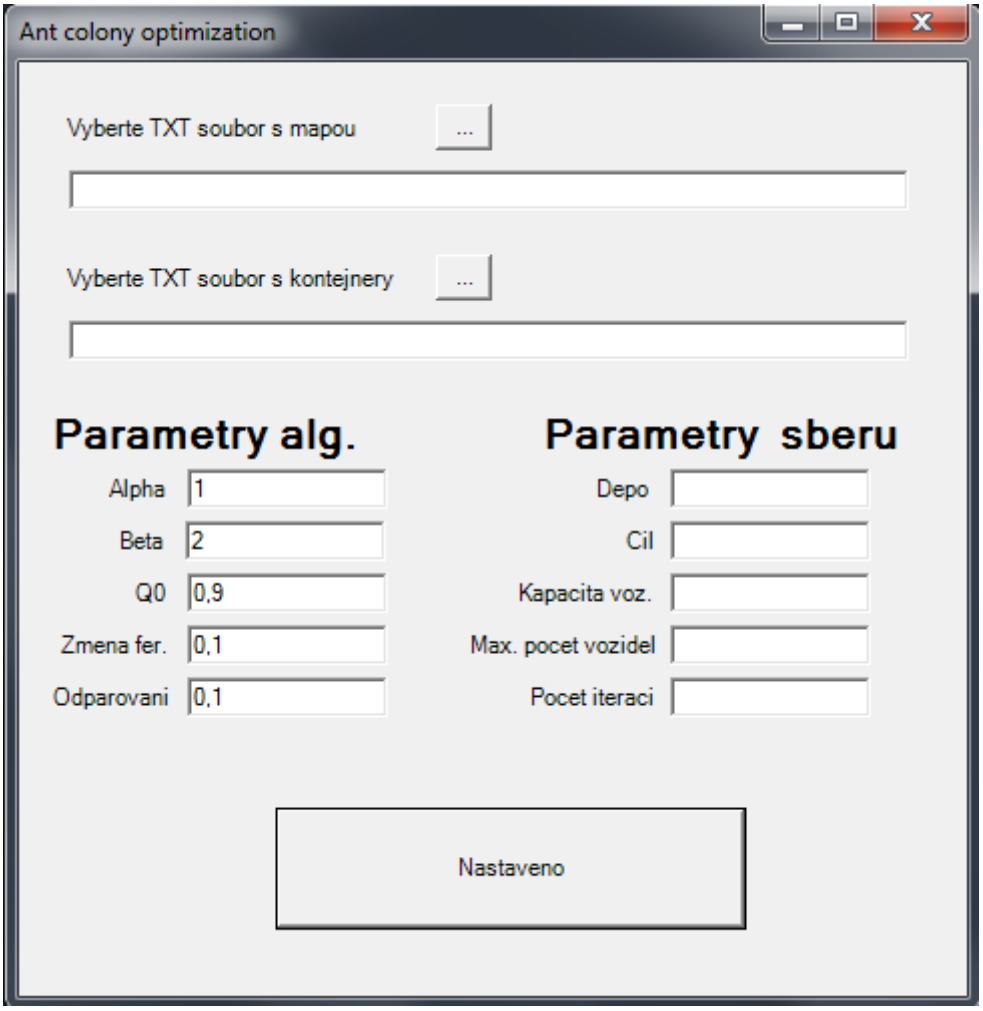


6.3.2 Diagram pro metodu Trasa



6.3.3 Formuláře programu

Program obsahuje dva formuláře, první pro nastavení parametrů a načtení vstupních textových souborů a druhý k zobrazení výsledků.



Obrázek 10: Vstupní formulář (settings)

Vstupní formulář se vždy načte s doporučenými parametry algoritmu, přičemž všechny zapsané výsledky v této práci byly získány právě s tímto nastavením.

Textový soubor s mapou musí být ve formátu matice sousednosti, kde jsou hodnoty od sebe oddělené dvojtečkou. Pokud nejsou vrcholy přímo propojené, musí na souřadnici tohoto spojení být hodnota -2. Navíc musí být v každé řádce dva sloupce, jedná se o poslední dva, kde předposlední sloupec obsahuje adresu vrcholu a v posledním sloupci se nachází GPS souřadnice vrcholu.

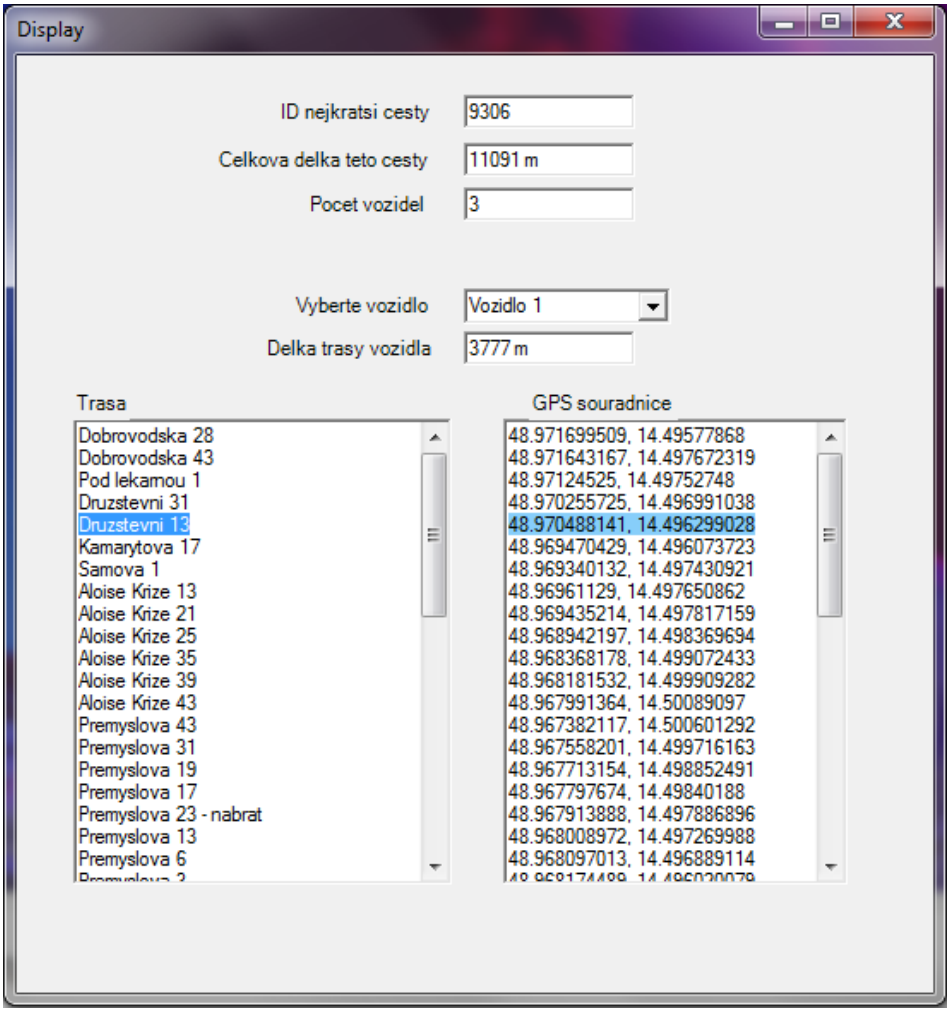
Vstup pro seznam kontejnerů je realizován dvěma řádky. V prvním řádku jsou ID vrcholů, které se mají navštívit, oddělené dvojtečkou. Druhá řádka je určena pro zaplnění

kontejnerů. Jak bylo zmíněno v kapitole 6.1.2, jsou zde dvě možnosti jak toto zaplnění zadat. Pokud chceme pro všechny kontejnery maximální zaplnění, musí první člen obsahovat frázi *max*. Jakmile je načtena tato fráze, jsou všechna ostatní data v této řádce ignorována. Pokud chceme zadat každému kontejneru specifické zaplnění, zadáváme ho v desetinném čísle mezi dvojtečky ve stejném pořadí, jako jsou zadány kontejnery.

Depo a cíl označují ID vrcholu, kde bude algoritmus začínat a kam vozidlo pojede, jakmile bude naplněné nebo budou všechny kontejnery vysypané.

Kapacita vozidla značí kolik plných kontejnerů je vozidlo schopno naložit, udává se v desetinném čísle. Maximální počet vozidel udává velikost dostupného parku. Důležité je, aby konečná kapacita vozového parku byla schopna pojmout požadované množství odpadu.

Počet iterací odpovídá počtu, kolik řešení se vygeneruje a následně porovná. Kvůli vysokému počtu vrcholů je vhodné toho číslo nastavit dostatečně vysoko.



Obrázek 11: Ukázka výstupního formuláře (display)

Výstupní formulář obsahuje nejlepší získané řešení za běh programu. Je zde zobrazen ID iterace, celková délka trasy všech vozů, počet potřebných vozidel, celá trasa jednotlivých vozů, spolu s GPS souřadnicemi jednotlivých křižovatek a délka trasy vybraného vozidla. Kontejnery, které má vozidlo naložit, jsou označené vedle adresy heslem nabrat.

6.3.4 Třídy

Součástí tohoto programu jsou tři třídy. Třída Base je třídou, přes kterou voláme jak formuláře, tak metody, které vytvářejí trasy a načítají vrcholy. Mimo tuto funkci zároveň je tato třída zodpovědná za použití metody nejbližšího souseda pro výpočet prvotní vrstvy feromonu, úpravy feromonu po každé iteraci a porovnávání vypočítaných řešení.

Třída Vrchol získává hodnoty ze vstupních souborů a upravuje je pro další zpracování.

Poslední třídou je třída Mravenec. Ta zodpovídá za vytvoření trasy pro každého mravence (vozidlo), od jednotlivých kroků při průzkumu grafu, přes obsluhu nakládání kontejnerů, po hledání trasy od posledního nabraného kontejneru do cílového vrcholu a tím ukončení trasy pro tohoto agenta.

6.4 Implementace algoritmu

V předchozí části byly naznačeny jednotlivé třídy programu, zde v několika větách popíšu funkce významnějších metod těchto tříd.

6.4.1 NactiVrcholy

Jedná se o první metodu, která je volaná po vyplnění vstupního formuláře (settings). Její funkcí je získat data z textových souborů a připravit tyto data do polí a listů.

6.4.2 NNvypocet

Tato metoda získává pomocí metody nejbližšího souseda potřebnou vzdálenost pro navštívení všech vrcholů. Tato hodnota je klíčová pro prvotní nastavení hladiny feromonu.

6.4.3 GetFeromon

Pomocí výstupu z NNvypocet a maximálního počtu vozidel dosadíme do vzorce 3.27 a tím dostaneme hladinu feromonu pro první iteraci algoritmu. Poté je nanesen na všechny

hrany grafu.

Tato hodnota feromonu je o to významnější, protože je potřebná jako proměnná FeromonIn, která se používá při lokální úpravě feromonu během procházení grafu.

6.4.4 Trasa

Jedná se o hlavní metodu třídy Mravenec. Zodpovídá za vytvoření trasy, volání ostatních metod a zároveň kontroluje, zda je vozidlo plné a má ukončit svou trasu pro sběr odpadu nebo jestli jsou všechny kontejnery naložené.

6.4.5 Nakladani

První metoda volaná při tvorbě trasy. Kontroluje, zda se vozidlo nachází na vrcholu označeném jako kontejner. Pokud je vozidlo schopné naložit celý náklad, zadá se tento vrchol do listu Nabrano, který slouží pro kontrolu ukončení prohledávání v metodě Trasa. Poté zkontroluje, zda seje vozidlo schopno naložit některý z dalších kontejner. V případě, že vozidlo nemá dostatek volného místa pro žádný další kontejner, změní se logická proměnná plno na hodnotu true a začne se vytvářet trasa do cílového vrcholu. Tímto se snažíme zaplnit každé vozidlo co nejvíce.

6.4.6 Presun

Po naložení, pokud se neukončuje trasa, se algoritmus rozhoduje, do jakého následujícího vrcholu se přesune. Samotné vyhodnocení dalšího vrcholu provádí funkce VyberHrany. Presun obsluhuje ostatní potřebné akce. Patří mezi ně úprava feromonu po přesunutí, zavedení lokace, ze které se přesouvá, do zásobníku, přičtení velikosti vybrané větve do délky řešení momentálního vozidla a samotnou změněnu lokace.

Změna feromonu se provádí podle vzorce 3.25, kde se využívá proměnná FeromonIn získaný z metody GetFeromon.

6.4.7 VyberHrany

Tato metoda rozhoduje o následujícím vrcholu. Ze všeho nejdřív se vypočítá pravděpodobnost pro každou větev spojenou s vrcholem, ve kterém se algoritmus nachází. Tato pravděpodobnost je získána vzorcem 3.13. Následně se vygeneruje náhodné číslo Q z intervalu 0 až 1, které se porovná s parametrem Q0. Pokud je náhodné číslo větší než Q0 bude se algoritmus rozhodovat pomocí pravděpodobností získaných na začátku metody. V opačném případě se algoritmus přesune po hraně, která má nejvyšší pravděpodobnost.

Pokud se stane, že není možné se přesunout na nenavštívený vrchol, použije se zásobník k prohledávání předchozích vrcholů.

6.4.8 Konec

Jakmile je vozidlo plné nebo jsou všechny kontejnery v grafu vysypány, je zavolána tato metoda, která pomocí Dijkstrova algoritmu hledá nejkratší cestu z posledního kontejneru do cílového vrcholu. Jelikož hledáme nejkratší trasu mezi dvěma vrcholy, je tento algoritmus vhodnější pro tuto část aplikace. Výhoda tohoto algoritmu je zejména naleznutí zaručeně nejkratší trasy od posledního nabraného kontejneru do cílového vrcholu, navíc při tomto přesunu se nesnižuje feromon na použitých hranách, což by mohlo způsobit zkreslení při průzkumu grafu.

6.4.9 FeromonGlobal

Poslední metoda, která se použije před ukončením iterace, je změna feromonu na hranách, jež jsou využity v dosud nejlepším řešení. Samotná úprava je dána vzorcem 3.25.

7. Aplikace programu na svoz odpadu

7.1 Data

Prvním krokem pro optimalizaci problému svozu odpadu je získání a předzpracování dat. Ať už se jedná o vytvoření souborů, které budou reprezentovat mapu a seznam kontejnerů nebo informace o lokaci depa a místa skládky.

7.2 Nastavení vstupních hodnot

Pro parametry algoritmu jsem použil doporučené hodnoty z literatury. Jedná se o stejné hodnoty, které jsou zobrazené v obrázku číslo 10.

Podle lokality depa a svozového místa jsem pro vstupní a výstupní vrchol určil vrchol číslo 87. Budeme předpokládat, že každé vozidlo je schopno naložit čtyři plné kontejnery a k dispozici pro danou část města je maximálně 5 vozidel.

7.3 Výstup programu

Nejprve vygenerujeme řešení pro případ, ve kterém jsou všechny kontejnery maximálně zaplněné. Jedná se o extrémní případ, což znamená, že popelářské vozy budou vždy schopné tímto způsobem naložit všechny kontejnery. Nevýhodou je, že vozidla bývají

tímto způsobem málokdy plně naložená.

Číslo iterace	Kapacita vozidla	Počet vozidel	Délka trasy v metrech
12 778	4	3	11 739
12 066	4	3	11 573
12 840	4	3	10 754
21 668	4	3	10 707
4 674	4	3	10 453
1 961	4	3	11 132
13 852	4	3	12 679
2 420	4	3	11 557
38 337	4	3	11 085
9 306	4	3	11 091

Tabulka 1: výsledky pro případ, když jsou všechny kontejnery zaplněné.

V druhém případě jsem každému kontejneru přiřadil určité procento zaplnění. Textový soubor pro kontejnery tedy vypadal následovně:

7:23:46:51:53:68:77:92:90:84:39

0,2:0,741:0,99:0:0,8:1:0,45:0,77:0,33:0,2546:0,123

Průměrné zaplnění je 0,514. Toto nastavení obsahuje i extrémní hodnoty jak 1 tak 0.

V konstrukci řešení bude kontejner s ID 51 ignorován, protože je jeho kapacita naplněna z 0%. Kvůli snížení potřebné kapacity nutné pro naložení všech kontejnerů se tímto redukuje počet potřebných vozidel na dvě.

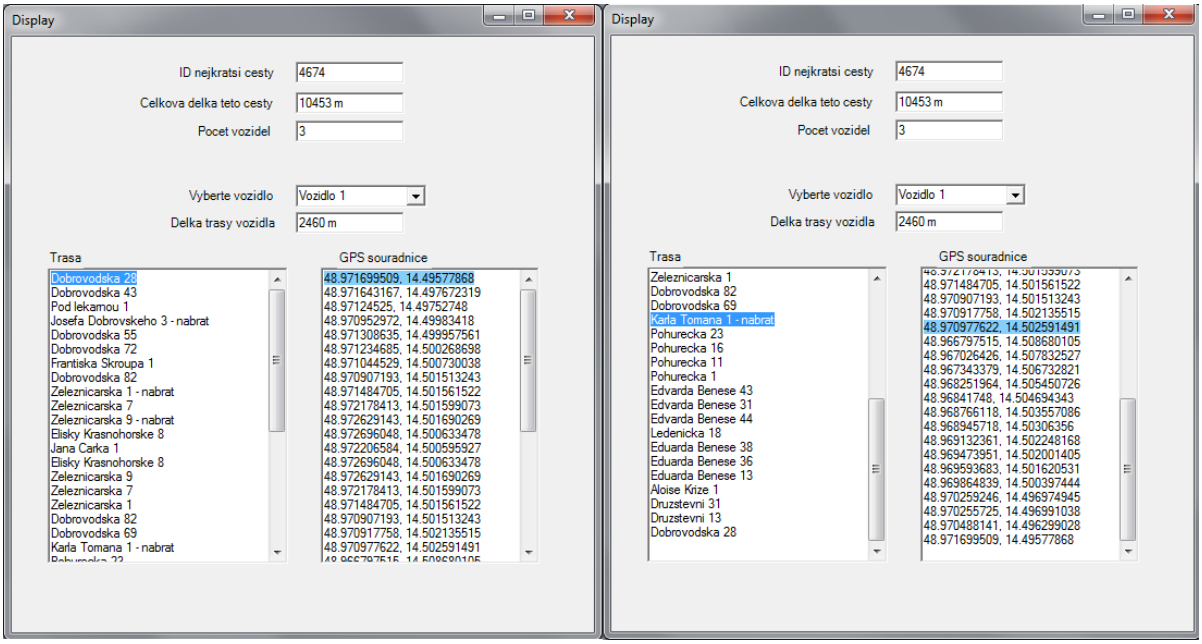
Číslo iterace	Kapacita vozidla	Počet vozidel	Délka trasy v metrech
3613	4	2	8162
27391	4	2	10088
485	4	2	6474
30772	4	2	7124
7121	4	2	7857
10782	4	2	7494
17200	4	2	6722
12390	4	2	6766
21492	4	2	8469
12637	4	2	6844

Tabulka 2: hodnoty pro různě zaplněné kontejnery.

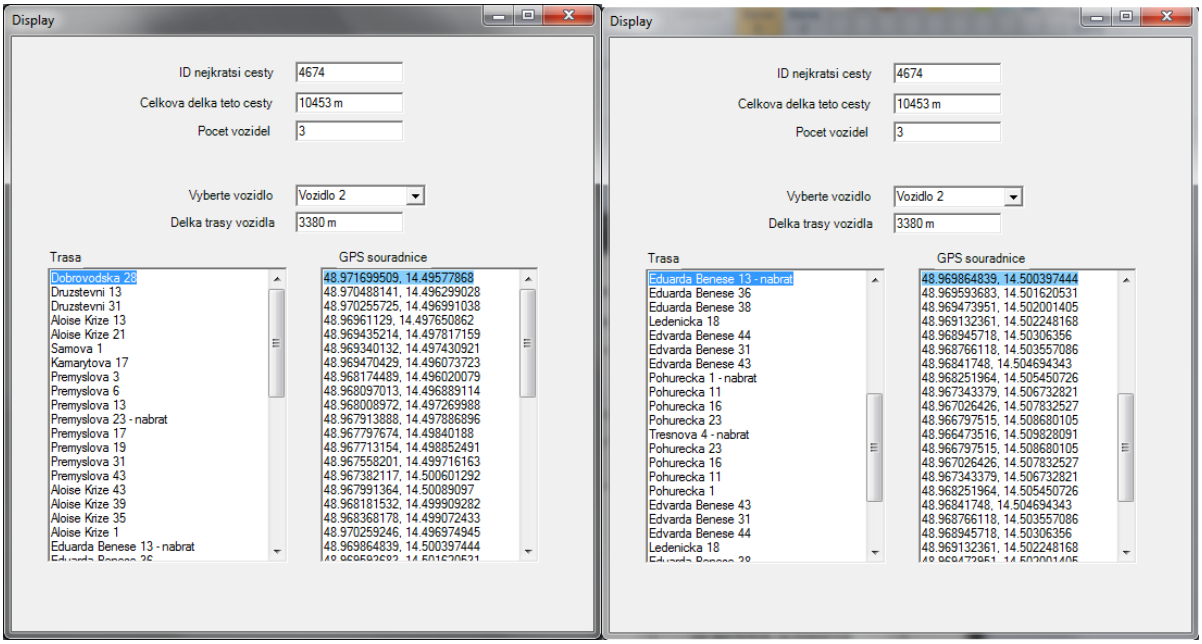
7.4 Nejlepší získaná řešení

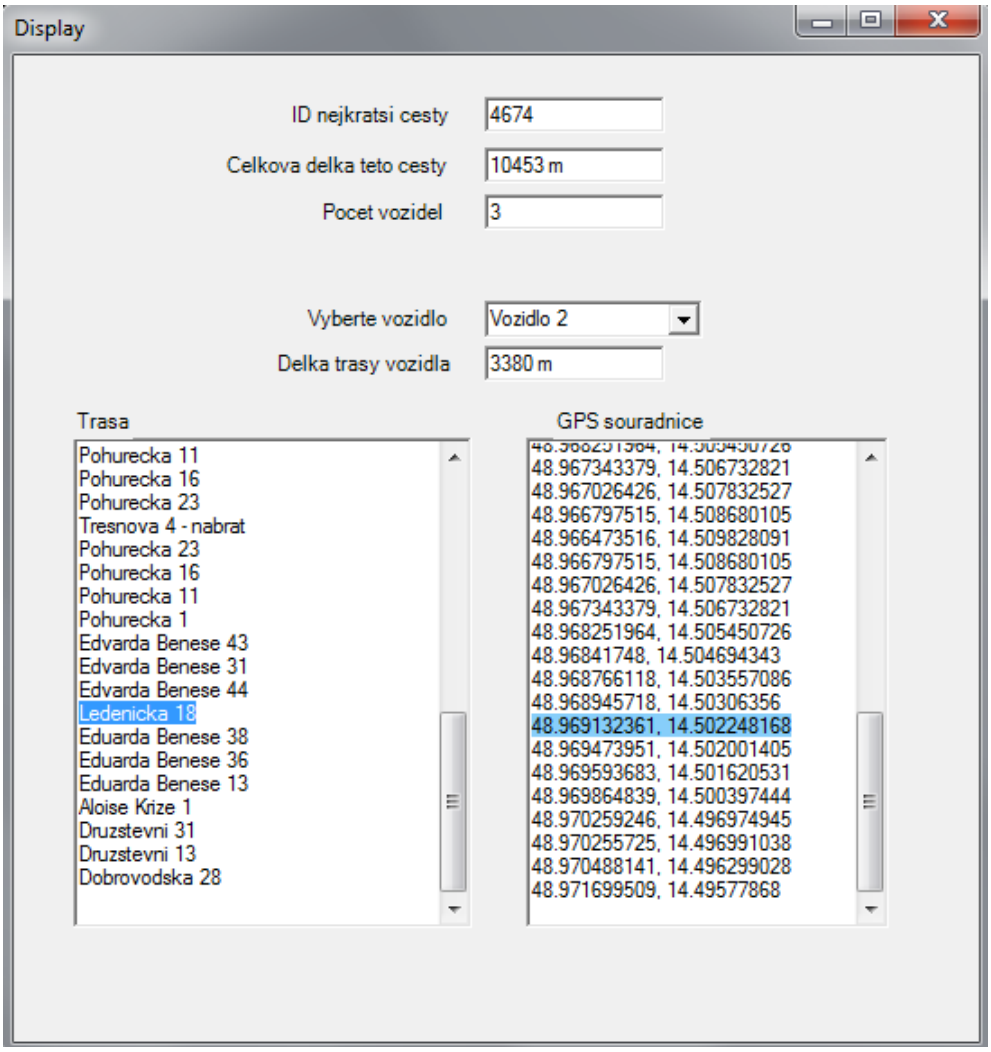
Pro všechny kontejnery zaplněné je nejkratší trasa dlouhá 10 453 metrů a potřebuje tři vozidla na obsluhu trasy. Samotný výstup pro toto řešení vypadá následovně:

Vozidlo 1

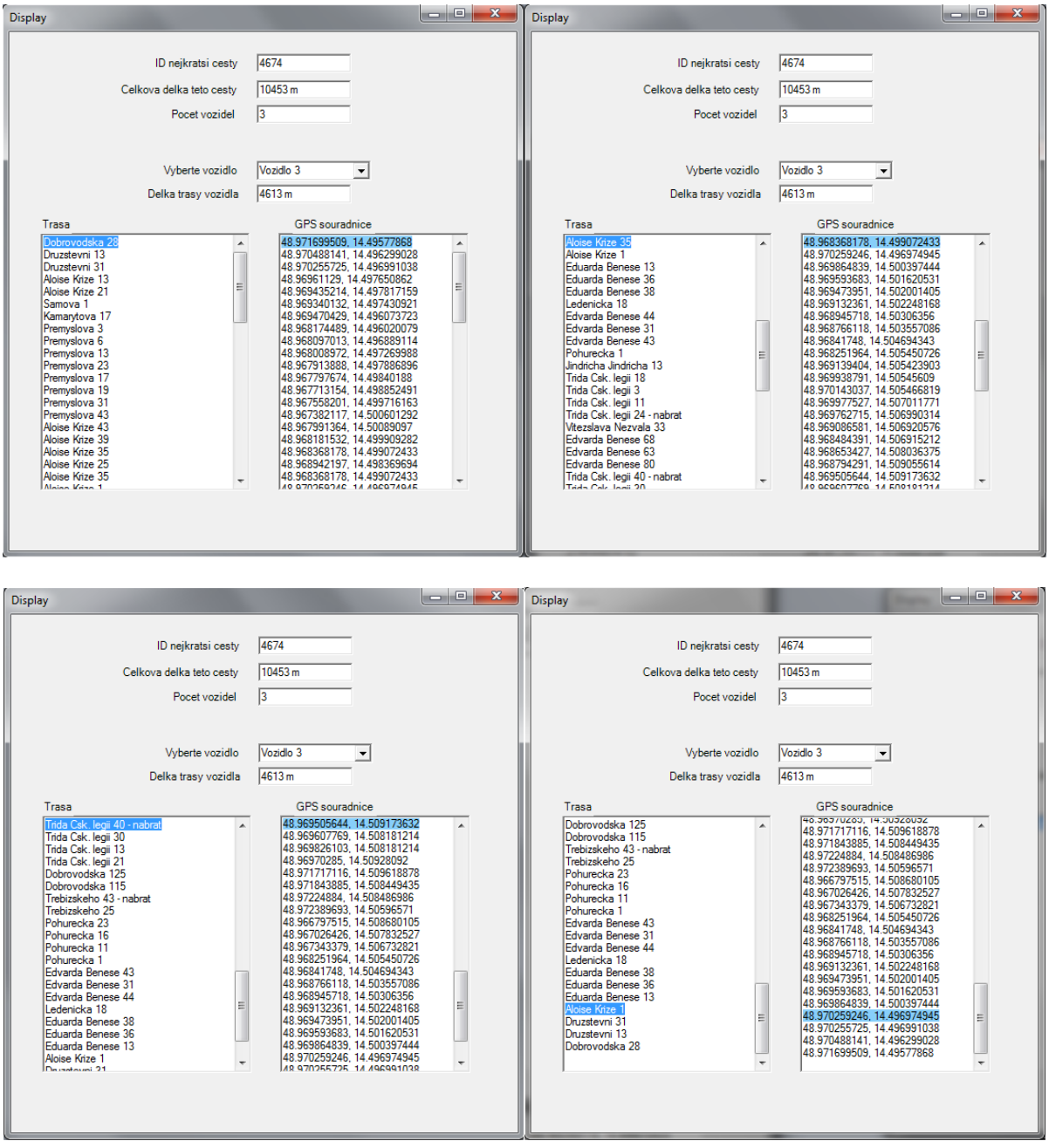


Vozidlo 2


Display
ID nejkratsi cesty
4674
Celkova delka teto cesty
10453 m
Pocet vozidel
3
Vyberte vozidlo
Vozidlo 2
Delka trasy vozidla
3380 m
Trasa
GPS souradnice
Dobrovodska 28
Druzstevni 13
Druzstevni 31
Aloise Krize 13
Aloise Krize 21
Samova 1
Kamarytova 17
Premyslova 3
Premyslova 6
Premyslova 13
Premyslova 23 - nabrat
Premyslova 17
Premyslova 19
Premyslova 31
Premyslova 43
Aloise Krize 43
Aloise Krize 39
Aloise Krize 35
Aloise Krize 1
Eduarda Benese 13 - nabrat
48.971699509, 14.49577868
48.970488141, 14.496299028
48.970255725, 14.496991038
48.96961129, 14.497650862
48.969435214, 14.497817159
48.969340132, 14.497430921
48.969470429, 14.496073723
48.968174489, 14.496020079
48.968097013, 14.496889114
48.968008972, 14.497269988
48.967913888, 14.497886896
48.967797674, 14.49840188
48.967713154, 14.498852491
48.967558201, 14.499716163
48.967382117, 14.500601292
48.967991364, 14.50089097
48.968181532, 14.499909282
48.968368178, 14.499072433
48.970259246, 14.496974945
48.969864839, 14.500397444
Eduarda Benese 13 - nabrat
Eduarda Benese 36
Eduarda Benese 38
Ledenicka 18
Edvarda Benese 44
Edvarda Benese 31
Edvarda Benese 43
Pohurecka 1 - nabrat
Pohurecka 11
Pohurecka 16
Pohurecka 23
Tresnova 4 - nabrat
Pohurecka 23
Pohurecka 16
Pohurecka 11
Pohurecka 1
Edvarda Benese 43
Edvarda Benese 31
Edvarda Benese 44
Ledenicka 18
48.969864839, 14.500397444
48.969593683, 14.501620531
48.969473951, 14.502001405
48.969132361, 14.502248168
48.968945718, 14.50306356
48.968766118, 14.503557086
48.96841748, 14.504694343
48.968251964, 14.505450726
48.967343379, 14.506732821
48.967026426, 14.507832527
48.966797515, 14.508680105
48.966473516, 14.509828091
48.966797515, 14.508680105
48.967026426, 14.507832527
48.967343379, 14.506732821
48.96841748, 14.504694343
48.968766118, 14.503557086
48.968945718, 14.50306356
48.969132361, 14.502248168

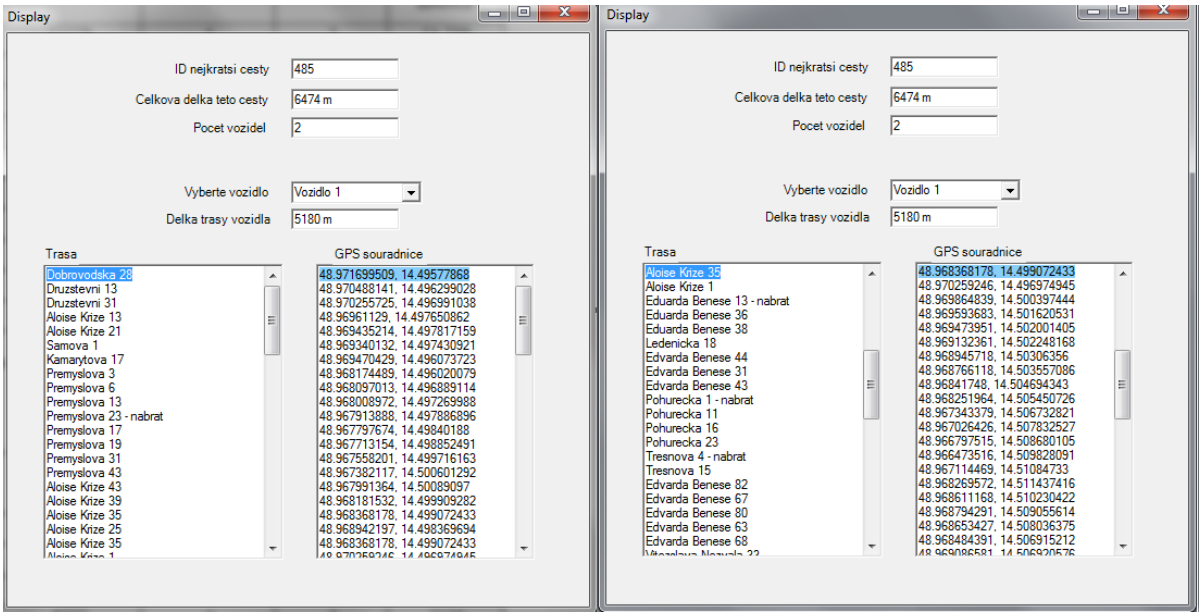

Display
ID nejkratsi cesty
4674
Celkova delka teto cesty
10453 m
Pocet vozidel
3
Vyberte vozidlo
Vozidlo 2
Delka trasy vozidla
3380 m
Trasa
GPS souradnice
Pohurecka 11
Pohurecka 16
Pohurecka 23
Tresnova 4 - nabrat
Pohurecka 23
Pohurecka 16
Pohurecka 11
Pohurecka 1
Edvarda Benese 43
Edvarda Benese 31
Edvarda Benese 44
Ledenicka 18
Eduarda Benese 38
Eduarda Benese 36
Eduarda Benese 13
Aloise Krize 1
Druzstevni 31
Druzstevni 13
Dobrovodska 28
48.967343379, 14.506732821
48.967026426, 14.507832527
48.966797515, 14.508680105
48.966473516, 14.509828091
48.966797515, 14.508680105
48.967026426, 14.507832527
48.967343379, 14.506732821
48.968251964, 14.505450726
48.96841748, 14.504694343
48.968766118, 14.503557086
48.968945718, 14.50306356
48.969132361, 14.502248168
48.969473951, 14.502001405
48.969593683, 14.501620531
48.969864839, 14.500397444
48.970259246, 14.496974945
48.970255725, 14.496991038
48.970488141, 14.496299028
48.971699509, 14.49577868

Vozidlo 3

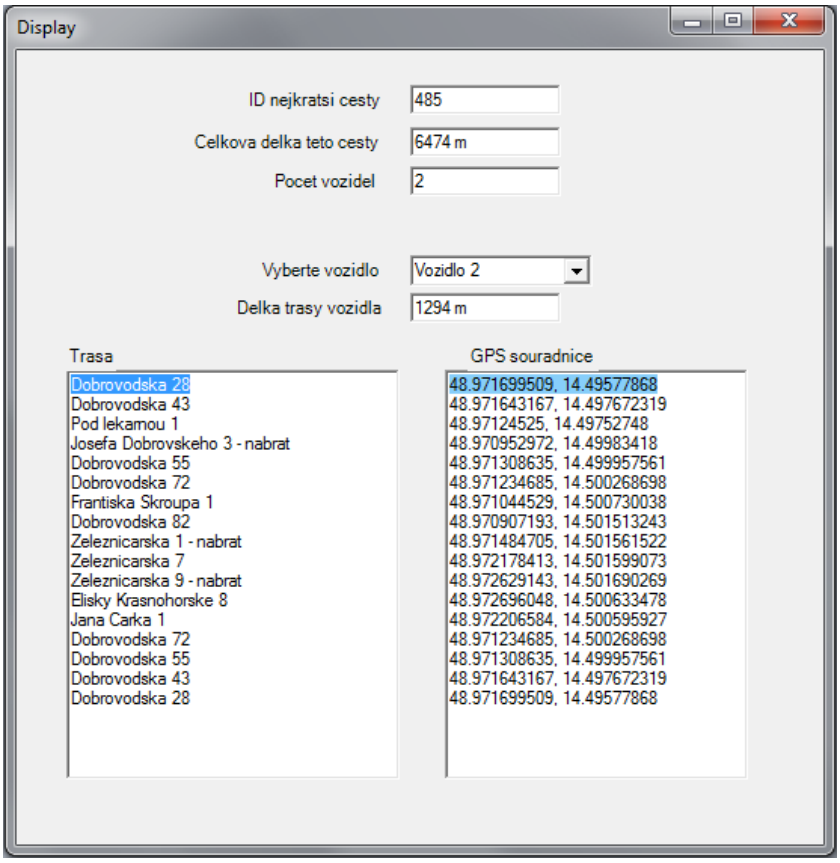

Display
ID nejkratsi cesty
4674
Celkova delka teto cesty
10453 m
Pocet vozidel
3
Vyberte vozidlo
Vozidlo 3
Delka trasy vozidla
4613 m
Trasa
GPS souradnice
Dobrovodska 28
48.971699509, 14.49577868
Druzstevni 13
48.970488141, 14.496299028
Druzstevni 31
48.970255725, 14.496991038
Aloise Krize 13
48.96961129, 14.497650862
Aloise Krize 21
48.969435214, 14.497817159
Samova 1
48.969340132, 14.497430921
Kamarytova 17
48.969470429, 14.496073723
Premyslova 3
48.968174489, 14.496020079
Premyslova 6
48.968097013, 14.496889114
Premyslova 13
48.968008972, 14.497269988
Premyslova 23
48.967913888, 14.497886896
Premyslova 17
48.967797674, 14.49840188
Premyslova 19
48.967713154, 14.498852491
Premyslova 31
48.967558201, 14.499716163
Premyslova 43
48.967382117, 14.500601292
Aloise Krize 43
48.967991364, 14.50089097
Aloise Krize 39
48.968181532, 14.499909282
Aloise Krize 35
48.968368178, 14.499072433
Aloise Krize 25
48.968942197, 14.498369694
Aloise Krize 35
48.968368178, 14.499072433
Display
ID nejkratsi cesty
4674
Celkova delka teto cesty
10453 m
Pocet vozidel
3
Vyberte vozidlo
Vozidlo 3
Delka trasy vozidla
4613 m
Trasa
GPS souradnice
Aloise Krize 35
48.968368178, 14.499072433
Aloise Krize 1
48.970259246, 14.496974945
Eduarda Benese 13
48.969864839, 14.500397444
Eduarda Benese 36
48.969593683, 14.501620531
Eduarda Benese 38
48.969473951, 14.502001405
Ledenicka 18
48.969132361, 14.502248168
Edvarda Benese 44
48.968945718, 14.50306356
Edvarda Benese 31
48.968766118, 14.503557086
Edvarda Benese 43
48.96841748, 14.504694343
Pohurecka 1
48.968251964, 14.505450726
Jindricha Jindricha 13
48.969139404, 14.505423903
Trida Csk. legii 18
48.969938791, 14.50545609
Trida Csk. legii 3
48.970143037, 14.505466819
Trida Csk. legii 11
48.969977527, 14.507011771
Trida Csk. legii 24 - nabrat
48.969762715, 14.506990314
Vitezslava Nezvala 33
48.969086581, 14.506920576
Edvarda Benese 68
48.968484391, 14.506915212
Edvarda Benese 63
48.968653427, 14.508036375
Edvarda Benese 80
48.968794291, 14.509055614
Trida Csk. legii 40 - nabrat
48.969505644, 14.509173632
Display
ID nejkratsi cesty
4674
Celkova delka teto cesty
10453 m
Pocet vozidel
3
Vyberte vozidlo
Vozidlo 3
Delka trasy vozidla
4613 m
Trasa
GPS souradnice
Trida Csk. legii 40 - nabrat
48.969505644, 14.509173632
Trida Csk. legii 30
48.969607769, 14.508181214
Trida Csk. legii 13
48.969826103, 14.508181214
Trida Csk. legii 21
48.96970285, 14.50928092
Dobrovodska 125
48.971717116, 14.509618878
Dobrovodska 115
48.971843885, 14.508449435
Trebizskeho 43 - nabrat
48.97224884, 14.508486986
Trebizskeho 25
48.972389693, 14.50596571
Pohurecka 23
48.966797515, 14.508680105
Pohurecka 16
48.967026426, 14.507832527
Pohurecka 11
48.967343379, 14.506732821
Pohurecka 1
48.968251964, 14.505450726
Edvarda Benese 43
48.96841748, 14.504694343
Edvarda Benese 31
48.968766118, 14.503557086
Edvarda Benese 44
48.968945718, 14.50306356
Ledenicka 18
48.969132361, 14.502248168
Eduarda Benese 38
48.969473951, 14.502001405
Eduarda Benese 36
48.969593683, 14.501620531
Eduarda Benese 13
48.969864839, 14.500397444
Aloise Krize 1
48.970259246, 14.496974945
Display
ID nejkratsi cesty
4674
Celkova delka teto cesty
10453 m
Pocet vozidel
3
Vyberte vozidlo
Vozidlo 3
Delka trasy vozidla
4613 m
Trasa
GPS souradnice
Dobrovodska 125
Dobrovodska 115
Trebizskeho 43 - nabrat
Trebizskeho 25
Pohurecka 23
Pohurecka 16
Pohurecka 11
Pohurecka 1
Edvarda Benese 43
Edvarda Benese 31
Edvarda Benese 44
Ledenicka 18
Eduarda Benese 38
Eduarda Benese 36
Eduarda Benese 13
Aloise Krize 1
Druzstevni 31
Druzstevni 13
Dobrovodska 28
48.971717116, 14.509618878
48.971843885, 14.508449435
48.97224884, 14.508486986
48.972389693, 14.50596571
48.966797515, 14.508680105
48.967026426, 14.507832527
48.967343379, 14.506732821
48.968251964, 14.505450726
48.96841748, 14.504694343
48.968766118, 14.503557086
48.968945718, 14.50306356
48.969132361, 14.502248168
48.969473951, 14.502001405
48.969593683, 14.501620531
48.969864839, 14.500397444
48.970259246, 14.496974945
48.970255725, 14.496991038
48.970488141, 14.496299028
48.971699509, 14.49577868

V druhém případě je nejkratší trasa 6 474 metrů za použití pouze dvou vozidel.
Trasy jednotlivých vozidel:

Vozidlo 1



Vozidlo 2



Jak bylo zmíněno v kapitole 6.1.2 je možné tento způsob použít po získání většího množství dat pro přesné předpovídání pravděpodobného zaplnění kontejneru. Avšak může nastat, taková situace, že budou kontejnery více zaplněné, než bývá zvykem, což zapříčiní nedostatek nákladového prostoru a musel by se tento problém dále řešit. Tento způsob bych doporučil pouze v případě, že máme způsob jak zjistit zaplnění kontejnerů těsně před tvorbou trasy.

8. Zhodnocení algoritmu

Protože algoritmus nevrací vždy stejné hodnoty, rozhodl jsem se vygenerovat více řešení pro různá nastavení a porovnat výsledky. Zároveň zjistit efektivitu optimalizaci mravenčí kolonií oproti metodě nejbližšího souseda.

Za použití grafu získaného z mapy pro Suché Vrbné, jsem vygeneroval řešení pro počet iterací 20 000 a 50 000, pro zjištění jaký vliv má na průměrnou kvalitu řešení. Vstupní hodnoty, kromě počtu iterací a kapacity vozidel, jsou stejné, jako v kapitole 7 a všechny kontejnery jsou zaplněné.

Číslo iterace	Kapacita vozidla	Počet vozidel	Délka trasy v metrech
7771	11	1	6 428
9635	11	1	5 641
5682	11	1	5 437
6461	11	1	6 403
2095	11	1	6 450
4654	11	1	6 806
18156	11	1	5 639
12156	11	1	6 284
10295	11	1	5 919
4083	11	1	6 327
8090	11	1	5 437
12673	11	1	6 325
16817	11	1	5 810
1126	11	1	6 012
5036	11	1	5 654
9608	11	1	6 259
9861	11	1	6 492
7085	11	1	5 919
14652	11	1	5 919
12617	11	1	6 151

2796	6	2	8 205
8848	6	2	8 311
2577	6	2	7 459
6886	6	2	6 907
902	6	2	7 656
10004	6	2	8 335
17815	6	2	8 206
8220	6	2	8 275
8888	6	2	7 632
5223	6	2	8 717
18459	6	2	7 864
4104	6	2	7 632
18641	6	2	8 224
2958	6	2	7 846
11469	6	2	8 428
18578	6	2	7 778
18260	6	2	8 387
6738	6	2	7 913
1010	6	2	8 772
4617	6	2	8 668
64	4	3	13 938
5869	4	3	13 299
2918	4	3	12 683
3843	4	3	11 229
16023	4	3	11 390
8018	4	3	11 029
8928	4	3	12 772
6676	4	3	11 604
4894	4	3	10 780
8345	4	3	12 683
321	4	3	12 209
6366	4	3	12 683
8068	4	3	10 659

17245	4	3	12 545
112	4	3	12 107
1784	4	3	11 604
5351	4	3	10 190
15738	4	3	12 507
16257	4	3	10 598
19636	4	3	11 367

Tabulka 3: Řešení získaná pro 20 000 iterací.

Číslo iterace	Kapacita vozidla	Počet vozidel	Délka trasy v metrech
2908	11	1	6 517
33825	11	1	5 919
997	11	1	5 565
16532	11	1	5 437
25156	11	1	5 764
25644	11	1	6 472
1935	11	1	6 893
19540	11	1	5 759
973	11	1	5 937
36396	11	1	5 500
6880	6	2	8 867
33174	6	2	7 991
5905	6	2	8 690
9534	6	2	7 632
7953	6	2	7 946
24513	6	2	7 669
6499	6	2	8 808
21345	6	2	7 529
45514	6	2	7 632
27407	6	2	7 682

12778	4	3	11 739
12066	4	3	11 573
12840	4	3	10 754
21668	4	3	10 707
4674	4	3	10 453
1961	4	3	11 132
13852	4	3	12 679
2420	4	3	11 557
38337	4	3	11 085
9306	4	3	11 091

Tabulka 4: Získané hodnoty pro 50 000 iterací

Počet iterací	Počet vozidel	Průměrná délka trasy v m	Maximální délka trasy	Rozdíl max. délky od průměrné v %	Minimální délka trasy	Rozdíl min. délky od průměrné v %
20 000	1	6 065.6	6 806	12,2	5 437	10,4
20 000	2	8 060.75	8 772	8,8	6 907	14,3
20 000	3	11 893.8	13 938	17,2	10 190	14,3
50 000	1	5 976,3	6 893	15,3	5 437	9
50 000	2	8 044,6	8 867	10,2	7529	6,4
50 000	3	11 282.9	12 679	12,4	10 453	7,4

Tabulka 5: Porovnání výsledků pro různý počet iterací

Z výsledků je vidět, že i přes značně větší počet iterací nemusí algoritmus najít kratší trasu grafem. Přestože pouze v jednom případě má algoritmus s více iteracemi lepší krajní hodnotu, než menší počet iterací, vždy poskytuje lepší průměrnou trasu a menší rozptyl generovaných hodnot.

Počet vozidel	Délka trasy NN	Rozdíl NN oproti nejdelší trase ACO v %	Rozdíl NN oproti nejkratší trase ACO v %
1	9 302	25,9	41,5
2	13 209	32,8	47,7
3	17 091	18,4	40,5

Tabulka 6: Porovnání metody nejbližšího souseda (NN) s opt. mravenčí kolonií (ACO)

Při vypočítání trasy metodou nejbližšího souseda jsem pro cestu od posledního kontejneru do koncového vrcholu použil stejný Dijkstrův algoritmus, jaký je použit i u optimalizace mravenčí kolonií, aby byly porovnány pouze trasy vytvořené při průzkumu.

Z tabulky 4 vidíme naprostou dominanci optimalizace mravenčí kolonií. Jednou z příčin je zejména to, že při průzkumu metody NN se vozidla pohybují k poslednímu nabranému kontejneru, které obsloužilo předchozí vozidlo, a teprve poté prohledá graf. Jelikož u optimalizace mravenčí kolonií není průzkum závislý pouze na vzdálenosti a feromonu, ale využívá i pravděpodobností, je tento algoritmus schopen efektivněji prohledávat graf a tím nalézt optimálnější cestu.

Přestože nemáme jistotu, že algoritmus vygeneruje nejkratší trasu, při porovnání s metodou nejbližšího souseda ukázala optimalizace mravenčí kolonií výrazně lepší výsledky.

Je vhodné algoritmus opakovat pro stejné hodnoty ze základního stavu, protože v některých případech se můžou nejkratší a nejdelší výsledky od sebe lišit v porovnání s průměrnou délkou trasy v intervalu od -14,3% po +17,2%.

9. Závěr

V první části práce byl shrnut postupný vývoj optimalizace mravenčí kolonií, od prvního matematického modelu popisující chování reálných mravenčích průzkumníků při experimentu dvojitého mostu, přes definování optimalizace mravenčí kolonií a jednotlivých úprav tohohle algoritmu po systém mravenčí kolonie.

Druhý úsek teoretické části je zaměřena na dva hlavní problémy, které se dají tímto algoritmem řešit. Jedná se o problém obchodního cestujícího a problém okružních jízd. U obou těchto problémů byly uvedeny definice, různé jejich varianty a jiný způsob řešení, než optimalizace mravenčí kolonií.

Praktická část začíná analýzou problému svozu odpadu a základním návrhem na řešení tohoto problému. Zároveň sem v té to části naznačil dva možné způsoby jak vytvořit trasy v závislosti na datech o zaplnění kontejnerů, které máme k dispozici.

Následuje převedení mapy části města na graf, který bude použit pro optimalizaci. V tomto grafu jsou vyznačeny všechny lokace kontejnerů, které jsem získal z dat poskytnutých z magistrátu města České Budějovice.

Další částí je popis struktury programu. Nejprve jsou zobrazeny vývojové diagramy, následně jsou popsány formuláře programu a lehce naznačen účel jednotlivých tříd. Poté jsou popsány metody, které jsou v programu použity.

Poté následuje samotná aplikace připravených dat na vstup programu a zaznamenání výsledků. Na stejný graf byly aplikovány dva různé soubory obsahující informace o kontejnerech. V prvním případě byly všechny kontejnery maximálně zaplněné, zatímco při druhé situaci byla každému kontejneru přiřazena hodnota odpovídající množství odpadu, které obsahoval. Přestože v druhém případě je výsledná trasa značně kratší a je potřeba méně vozidel, není zaručené, že při následujícím sběru budou stejně zaplněné a podobný způsob by se měl použít pouze v případě, že máme k dispozici prostředky pro zjištění zaplnění kontejnerů.

Poslední část práce je věnována analýze programu pro různé případy, pro zjištění více informací o kvalitě generovaných řešení. Z výsledků této analýzy jsou zjevný nedostatek tohoto algoritmu, kvůli intervalu délek generovaných tras, avšak i přes tuto

nevýhodu, jsou i nejhorší vypočítaná řešení znatelně lepší v porovnání s metodou nejbližšího souseda a to o desítky procent.

Vhodným směrem pro další vývoj této aplikace by byl pokus o snížení časové náročnosti. Zejména pro takto vysoký počet vrcholů a vysoký počet iterací by bylo i malé zlepšení v této oblasti velmi znatelné. Dalším možným vylepšením by mohl být vývoj systému, který by měřil zaplnění kontejneru a vytvářel by vstupní soubor pro tuto aplikaci pro přesnější a efektivnější optimalizaci.

Seznam použité literatury

[1] Krömer, P. *Optimalizace pomocí mravenčích kolonií* Dostupné z: http://homel.vsb.cz/~kro080/mravenci.pdf

[2] Deborah M. Gordon *Ant Encounters: Interaction Networks and Colony Behavior*, 2010

[3] M. Dorigo and T. Stützle, *Ant Colony Optimization*. Cambridge, MA: MIT Press, 2004

[4] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*. New York, NY, USA: Oxford University Press, Inc., 1999

[5] Len Fisher, *The Perfect Swarm: The Science of Complexity in Everyday Life*

[6] Peter Miller, *The Smart Swarm: How to Work Efficiently, Communicate Effectively, and Make Better Decisions Using the Secrets of Flocks, Schools, and Colonies*

[7] A. E. Rizzoli, R. Montemanni , E. Lucibello, L. M. Gambardella *Ant colony optimization for real-world vehicle routing problems From theory to applications*

[8] Marco Dorigo, Thomas Stützle, *An Experimental Study of the simple Ant Colony Optimization Algorithm* (2001) Dostupné z: http://www.wseas.us/e-library/conferences/tenerife2001/papers/622.pdf

[9] Godfrey C. Onwubolu,B. V. Babu, *New Optimization Techniques in Engineering*

[10] David L. Applegate, Robert E. Bixby, Vasek, *The Traveling Salesman Problem: A Computational Study*, 2006

[11] K. Aardal, George L. Nemhauser, R. Weismantel, *Handbooks in Operations Research and Management Science: Discrete Optimization Volume 12*

[12] Christian Nilsson, *Heuristics for the Traveling Salesman Problem* Dostupné z: https://web.tuke.sk/fei-cit/butka/hop/htsp.pdf

[13] Mitchell Melanie, *An Introduction to Genetic Algorithms*, 1999

[14] Upravili: G. Gutin, A. P. Punnen, *The Traveling Salesman Problem and Its Variations*, 2007

[15] Christian Prins, *Efficient Heuristics for the Heterogeneous Fleet Multitrip VRP with Application to a Large-Scale Real Case*, 2002

[16] Francisco Baptista Pereira, Jorge Tavares, *Bio-inspired Algorithms for the Vehicle Routing Problem*, 2009

[17] Upravili: Paolo Toth, Daniele Vigo, *Vehicle Routing: Problems, Methods, and Applications, Second Edition*, 2014

[18] Sıla Çetinkaya, Chung-Yee Lee, *Stock Replenishment and Shipment Scheduling for Vendor-Managed Inventory Systems*, 2000, dostupné z: http://www.ie.bilkent.edu.tr/~ie572/Papers/CetinkayaandLee.pdf

[19] Upravili: Geir Hasle, Knut-Andreas Lie, Ewald Quak, *Geometric Modelling, Numerical Simulation, and Optimization: Applied Mathematics at SINTEF*, 2007

[20] Peter Francis, Karen Smilowitz and Michal Tzury, *Flexibility and complexity in periodic distribution problems*, 2006, Dostupné z: http://www.transportation.northwestern.edu/docs/research/Smilowitz_PeriodicDistribution.pdf

[21] Roberto Baldacci, Maria Battarra and Daniele Vigo, *Routing a Heterogeneous Fleet of Vehicles*, 2007

[22] G. B. Dantzig and J. H. Ramser , *The Truck Dispatching Problem G. B. Dantzig and J. H. Ramser,* 1959

[23] Wil Michiels, Emile Aarts, Jan Korst, *Theoretical Aspects of Local Search,* 2007

[24] David S. Johnson, Christos H. Papadimitriou, Mihalis Yannakakis, *How Easy Is Local Search?,* 1988

[25] Fred W. Glover, Manuel Laguna, *Tabu Search, svazek 1,* 1997

[26] Enrique Alba, Rafael Martí, *Metaheuristic Procedures for Training Neural Networks,* 2006

[27] M. Jünger G. Reinelt G. Rinaldi, *The Traveling Salesman Problem,* 1996

[28] Keld Helsgaun, *Solving the Bottleneck Traveling Salesman Problem Using the Lin-Kernighan-Helsgaun Algorithm,* 2014