**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., MBA

České Budějovice 2015

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 je 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). Finally, the application of these tools to the collection of separated waste for part 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

Cílem této práce je přiblížit optimalizaci pomocí mravenčí kolonie (ACO), jak vznikala a postupně se vyvíjela do formy použitelnou pro efektivní řešení NP problémů, jako je *problém obchodního cestujícího* (traveling salesman problem - TSP) nebo *vehicle routing problem* (VRP).

Na začátku práce si přiblížíme optimalizaci mravenčí kolonie. Od prvních pokusů převedení pravidel, kterými se mravenci rozhodují při vytvoření tras mezi hnízdem a potravou, po sofistikované metaheuristické algoritmy řešící složité problémy.

Následně budou popsány dva problémy spadající do třídy NP . Problém obchodního cestujícího, který patří mezi nejstudovanější NP problémy a z něho vycházející vehicle routing problem. Oba problémy budou blíže definovány a uvedeny některé jejich modifikace a jiné způsoby řešení než pomocí ACO.

Závěrem této práce bude aplikace ACO na svoz tříděného odpadu pro část Českých Budějovic.

## 1.1 Formulace problému

V dnešním světě se klade čím dál tím větší důraz na to, aby se vše provádělo rychleji, efektivněji a úsporněji. Jednou možností jak toho dosáhnout je využití takzvané rojové inteligence. Jedná se o odvětví umělé inteligence, které je založeno na algoritmech odvozených z chování různých společenství v přírodě.

Již v minulosti si lidé pouhým pozorováním všimli zdánlivé organizace rojů hmyzu, stád zvířat nebo hejn ptáků či ryb. První studie zaměřená na chování organizovaného společenství proběhla v šedesátých letech minulého století biologem Brianem Partridgem na univerzitě v Miami. [5] Jeho výzkum byl zaměřen na chování hejna tresek v nádrži. Při následném rozboru nafilmovaných záběrů ryb, které pořídil s kolegy, zjistil, že se tyto ryby řídí dvěma jednoduchými pravidly: následovat tresku vpředu a udržovat rychlost s rybou vedle.[5] Podobnými pravidly se řídí i lidé v davech, které by se daly také považovat za superorganizmy.

Superorganizmus se nazývá společenstvo organizmů, které se chová jako jeden celek. [6]

Každé společenství má svůj řád, podle kterého se chovají jedinci v různých situacích. Jako je například lov ve smečce nebo hledání úkrytu. Dalším příkladem je vytváření mravenčích stezek mezi hnízdem a potravou. Jednou z charakteristik těchto cest je jejich vysoká efektivita. Aplikací odvozených pravidel jsme schopni dosáhnout podobných výsledků. Toho se dá velmi dobře využít nejen v dopravě, kde je délka trasy důležitým parametrem, ale dosazením jiných veličin při výpočtu dokážeme optimalizovat například i časové úkony. Proto není divu, že algoritmus založen právě na mravencích je jeden z nejvýznamnějších součástí rojové inteligence.

## 1.2 Výzkumné otázky

Je možné efektivně aplikovat mravenčí algoritmy na optimalizaci svozu tříděného odpadu?

Dílčí otázky:

1) Jak probíhal vývoj optimalizace mravenčí kolonie?
2) Jaké jsou výhody použití metaheuristiky na porovnání s přesnými algoritmy?
3) Jaké jsou další možnosti využití optimalizace mravenčí kolonie?

## 1.3 Cíl práce

Cílem práce je najít nejoptimálnější trasu pro svoz tříděného odpadu v Suchém Vrbném v Českých Budějovicích.

Dílčí cíle a úkoly:

1) Získat lokality kontejnerů tříděného odpadu v Suchém Vrbném.
2) Zanést lokality do mapy a vytvořit z ní graf pro ACO
3) Aplikovat ACO na nashromážděná data

## 1.4 Použité nástroje a metody

Pro dosáhnutí stanovených cílů je nejdříve využita dostupná odborná literatura a seznámení s problematikou optimalizace a VRP.

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.

# 2. Mravenci v přírodě

Tento algoritmus je založen na pravidlech a postupech, kterými se řídí mravenčí kolonie, přesněji na postupu od nalezení potravy až k následnému předání informace o její pozici.

Přestože jednotlivý mravenci jsou poměrně jednoduší a sami o sobě toho moc nesvedou, mravenčí kolonie je proti tomu velmi výkonný a efektivní celek.

Mravenci řeší různá důležitá rozhodnutí, například, naleznutí co nejkratší cesty k potravě a zároveň ušetření energie při návratu s potravou zpět do hnízda.

Většina druhů má slabý zrak nebo 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, který 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 přispívají k samo-

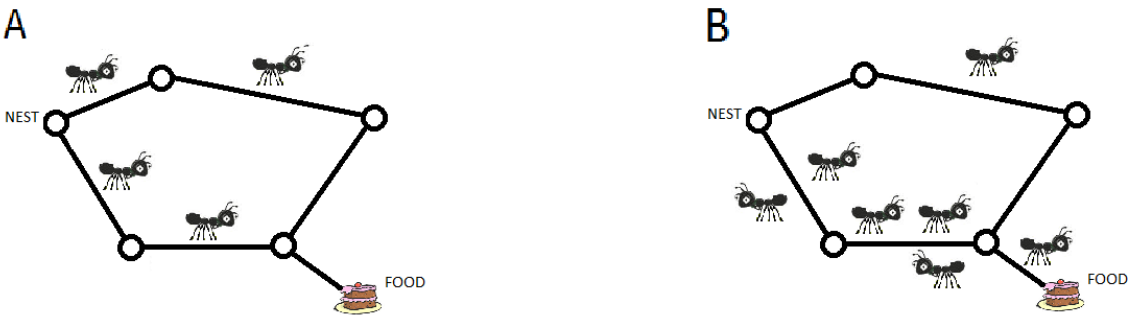
organizaci roje. To znamená, že není třeba, aby někdo dohlížel nebo kontroloval činnost kolonie. Veškerá komunikace probíhá pouze lokálně pomocí feromonů. [3]

## 2.1 Postupy mravenců v přírodě při hledání potravy

Mravenci nejprve náhodně prohledávají okolí, dokud nenaleznou potravu. Když je některý z nich úspěšný, 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. Opět provádí značení trasy pomocí feromonů. Z tohoto důvodu je tato již několikrát označená trasa přitažlivější, než cesta mravenců, kteří se ještě nevrátili do hnízda. Nový 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 pozitivního feedbacku je ukázkou samo-organizace celého roje. [3]

Někteří mravenci se přesto nevydají po označené cestě, a i tak můžou při svém průzkumu nalézt ten samý zdroj potravy. Tím se vytvoří více feromonových cest. Feromon se postupem času vypařuje a díky tomu ho bude na kratší cestě více. Protože mravenci chtějí šetřit energii, využívají jen tu trasu, po které se rychleji vracejí do kolonie a zpět k potravě. Kvůli větší koncentraci látky na kratší cestě se po ni začnou pohybovat i ti mravenci, kteří doposud používali delší trasu, což v důsledku bude znamenat, že časem všichni budou používat tu nejkratší objevenou cestou.

Ovšem jak tomu v přírodě bývá, je více variant značení si cest, například některé druhy vypouští feromon pouze při návratu. Jiný druh zas není schopen objevit nové nebo jiné dostupné trasy k potravě, když je první cesta silně nasycená feromonem. [2]




A) Mravenci začnou náhodně prohledávat okolí.

B) Průzkumníci, kteří se vydali kratší cestou, se také dříve vracejí do hnízda a díky této informaci se následující mravenci spíše vydají takto feromonem označenou cestou.

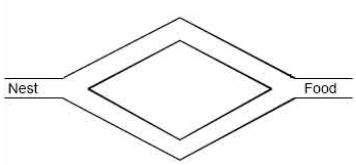
## 2.2 Double bridge experiment

Komunikace roje pomocí feromonu byla testována různými experimenty. Mezi hlavní patří tzv. Double bridge experiment pod vedením Jean Louis Deneubourgem. Ten spočíval ve spojení mravenčího hnízda a potravy dvojitým mostem a pozorování jakou cestu si průzkumníci vyberou.

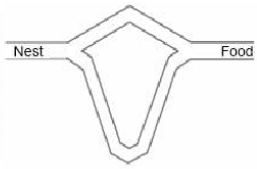
Zprvu byli obě větve stejně dlouhé (1.1). Když se mravenci začali vydávat za potravou, vybírali cestu náhodně, protože nebyl přítomen žádný feromon. Přestože obě cesty byli stejně dlouhé, postupem času se všichni mravenci pohybovali po jedné cestě. To se přisuzuje náhodné složce při výběru cesty, přesněji jednu z možností si vybere více mravenců a díky tomu na této možnosti bude silnější feromon a tím bude lákavější pro následující průzkumníky. Jak je vidět z výsledků experimentu (2.1) výsledná trasa je náhodná a výběr je přibližně stejný pro obě trasy. [3]

Při druhém experimentu byla jedna cesta mostu dvakrát delší jak druhá (1.2). Při této variantě se po krátké chvíli téměř všichni pohybovali po kratším mostě. Stejně jako v předchozí variantě si zezačátku mravenci vybrali cestu náhodně, avšak ti kteří si vybrali tu kratší, se i rychleji dostali k potravě a tím při zpáteční cestě se vraceli po silněji značené feromonové stopě kterou po sobě zanechali. [5] Z grafu (2.2) je jasně vidět naprostá převaha mravenců, kteří se vydali kratší trasou. Ovšem i poté co se na kratší cestě vytvořila silná feromonová stopa, byli jedinci, co se vydali delší cestou v pokuse o průzkum okolí mraveniště. [3]

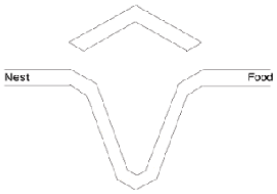
Při jiném experimentu byla zprvu mravencům poskytnuta jen dlouhá trasa a po půl hodině byla přidána druhá, kratší (1.3). Tím se ukázalo, že i mezi jednotlivými druhy mravenců jsou rozdíly. Konkrétně průzkumníci druhu *Limepithema humile* nebyli schopni upravit trasu po přidání kratší cesty. Přestože se někteří mravenci vydali nově přidanou trasou, nebylo jich dostatek, aby přemohli silnou feromonovou stopu na delší cestě a jeho pomalé odpařování a tím uvázli v lokálním extrémním řešení. Kdežto zástupci druhu *Lasius Niger* jsou schopni využít i později dostupné, výhodnější trasy. [4]



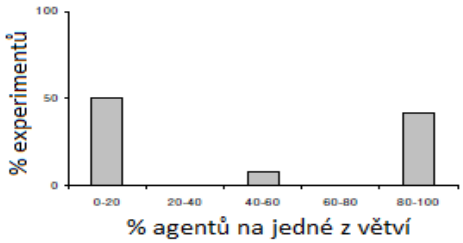

1.1 Obě cesty jsou stejně dlouhé



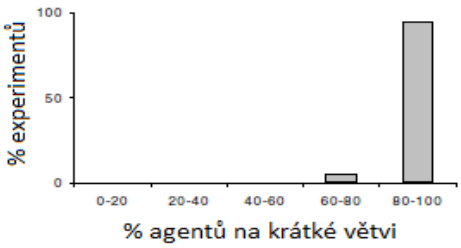

1.2 Spodní větev je dvakrát delší jak horní




1.3 Kratší cesta je oddělena a průzkumníci se mohou dostat k potrave pouze delší trasou




2.1 Zde vidíme, že náhodné vybrání trasy při stejně dlouhých větvích je přibližně rovnaké.




2.2 Kdežto u rozdílných velikostí je ve značné většině případů vybrána kratší trasa.

[1]

# 3. Ant Colony optimization – ACO

## 3.1 Základní vlastnosti.

Založen na chování průzkumníků některých druhů mravenců, patří optimalizace mravenčí kolonie mezi nejvýznamnější algoritmy převzaté z mravenčího světa.

Jedná se o meta-heuristickou techniku, neboli algoritmus, který 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.

Je velmi flexibilní a jeho použití jsou rozsáhlá. Ať už se jedná o *the job shop scheduling problem, the quadratic assignment problem, the sequential ordering problem, the graph coloring problem* a *the shortest common supersequence problem* (Dorigo a Stützle 2004), *open shop problem* (Blum 2005) nebo plánování studní pro kontrolu podzemní vody (Li a Chan Hilton 2007). [7]

[1] Zdroj obrázků: [3] strany 3 a 5

## 3.2 Začátky

První pokusy o umělého mravence provedl tým Deneubourga s kolegy v roce 1990, kde z jejich pozorování a výsledků z double bridge experimentu vytvořili poměrně jednoduchý matematický model popisující chování průzkumníků.

V tomto stochastickém modelu překročí ψ agentů most. Mravenci se pohybují v obou směrech konstantní rychlostí *v* (cm/s) a zároveň 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 $\varphi_{ia}(t)$.

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

Z toho dostaneme pravděpodobnost v čase t, že si agent vybere kratší větev. Pravděpodobnost pro delší cestu dopočítáme z následujícího vztahu.

$$p_{is}(t) + p_{il}(t) = 1 \qquad (4.2)$$

Tyto diferenciální rovnice popisují postupnou změnu feromonu na jednotlivých větvích.

$$\mathrm{d}\varphi_{is}/\mathrm{d}t = \psi p_{js}(t - t_s) + \psi p_{is}(t), \qquad (i=1, j=2;\ i=2, j=1), \qquad (4.3)$$

$$\mathrm{d}\varphi_{il}/\mathrm{d}t = \psi p_{jl}(t - t_s \cdot r) + \psi p_{il}(t), \qquad (i=1, j=2;\ i=2, j=1). \qquad (4.4)$$

Dají se přeložit jako změna feromonu na větvi *s* za čas *t* na uzlu *i* se rovná tok mravenců (ψ) znásobený pravděpodobností, že si agent vybere kratší cestu z uzlu *j* v čase t-$t_s$ 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.

Když následně zkoušeli počítačovou simulaci double bridge experimentu pro jeden tisíc agentů, výsledky byli velmi podobné jako ty, které zaznamenali při samotném experimentu s reálnými mravenci. [3]

## 3.3 Diskrétní model

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

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

Rovnice pro pravděpodobnosti tento model, když by se použil pro double bridge experiment by byli následující:

$$p_{is}(t) = \frac{[\varphi_{is}(t)]^{\alpha}}{[\varphi_{is}(t)]^{\alpha}[\varphi_{il}(t)]^{\alpha}} \quad (5.1)$$

$$p_{il}(t) = \frac{[\varphi_{il}(t)]^{\alpha}}{[\varphi_{is}(t)]^{\alpha}[\varphi_{il}(t)]^{\alpha}} \quad (5.2)$$

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:

$$\varphi_{is}(t) = \varphi_{is}(t-1) + p_{is}(t-1)m_i(t-1) + p_{js}(t-1)m_j(t-1), \text{ (i=1, j=2; i=2, j=1)} \quad (5.3)$$

$$\varphi_{il}(t) = \varphi_{il}(t-1) + p_{il}(t-1)m_i(t-1) + p_{jl}(t-r)m_j(t-r), \text{ (i=1, j=2; i=2, j=1)} \quad (5.4)$$

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

$$m_i(t) = p_{js}(t-1)m_j(t-1) + p_{jl}(t-r)m_j(t-r), \text{ (i=1, j=2; i=2, j=1)} \quad (5.5)$$

Tyto rovnice, ale platí, pouze pokud 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 tak tím kolonie ztrácí 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.

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 tím 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.

## 3.4 Simple Ant Colony Optimization - S-ACO

Vznikají kvůli nanášení feromu už při průzkumu a ne jen když 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 tak je možné smyčkám předejít. Tento algoritmus právě obsahuje tyto změny.

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 proměná *umělá feromonová stopa* $\tau = \tau_{ij}(t)$, 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:

$$p_{ij}^k = \begin{cases} \frac{\tau_{ij}^\alpha}{\sum_{j \in N_i^k} \tau_{ij}^\alpha} & \text{pro } j \in N_i^k \\ 0, & \text{pro } j \notin N_i^k \end{cases} \quad (6.1)$$

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 tuďíš 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], kde se postupně prohledává jeho cesta od hnízka k potravě. Tímto způsobem sice odstraníme smyčky, ale odstraňují se ty, na které se příjde v prohledávání jako na první. To znamená, že se může stát že se sice jedna smyčka odstraní, ale může být menší než jiná, která se sice tímto také odstraní, ale už to nemusí být nejktratší řešení z této cesty.

Po odstranění smyček se průzkumník vydá po upravené cestě nazpět do hnízda a na každé hraně, kterou teď bude procházet upravý hladinu feromonu podle 6.2.

$$\tau_{ij} = \tau_{ij} + \Delta\tau^k \tag{6.2}$$

Hodnotu $\Delta\tau^k$ v nejjednoduším 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. To 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 6.3.

$$\tau = (1 - \rho)\tau \tag{6.3}$$

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

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

## 3.5 Ant systém – AS

Ant systém je základní verzí mravenčích algoritmů a byl použit na řešení TSP jako příklad použití (Dorigo, 1992). [3] Přestože jeho základní verze nebyla tak účinná jako jiné tehdy dostupné algoritmy, řešící ten samí 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 $p_{ij}^{k}$.

$$p_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in N_i^k}[\tau_{il}]^{\alpha}[\eta_{il}]^{\beta}} \tag{7.1}$$

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]
Přibila tu veličina $\eta_{ij}$, která reprezentuje heuristickou a-priori informaci, jenž značí vhodnost přechodu z vrcholu $i$ do $j$. [1]

$$\eta_{ij} = \frac{1}{d_{ij}} \tag{7.2}$$

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

Parametrem α upravujeme výraznost feromonu. 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 a to 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. Paralelní implementaci, kde se všichni agenti pohybují do následujícího města. 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 an 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.

$$\tau_{ij} = (1-\rho)\tau_{ij}\,, \quad \forall(i,j) \in L \tag{7.3}$$

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.

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{1}{C^k}, & a_{ij} \in T^k \\ 0\,, & a_{ij} \notin T^k \end{cases} \tag{7.4}$$

$C^k$ je suma délky tras cesty vygenerované mravencem *k*.
$T^k$ je množina hran trasy agenta *k*.

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

$$\tau_{ij} = \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{7.5}$$

Veličina *m* označuje celkový počet mravenců.

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

Ovšem AS, na porovnání s ostatními meta-heuristickými algoritmy, má tendenci značně snižovat svůj výkon. Proto byly navrhnuty některé vylepšení, aby se tento problém podařilo minimalizovat. [3]

## 3.6 Varianty

### 3.6.1 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. Tím se algoritmus zrychlí, ale zároveň se vystavujeme riziku uváznutí v lokálním optimu.

Dosáhneme toho úpravou nanášení feromonu, kde se na hrany, po kterých vede zatím nejlepší řešení ($T^{bs}$), nanese větší množství, jehož velikost je funkce kvality trasy.

$$\Delta\tau_{ij}^{bs} = \begin{cases} \frac{1}{C^{bs}}, & a_{ij} \in T^{bs} \\ 0\ , & a_{ij} \notin T^{bs} \end{cases} \tag{7.6}$$

$\mathrm{C}^{\mathrm{bs}}$ je velikost celé trasy $T^{bs}$.

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

$$\tau_{\mathrm{ij}} \leftarrow \tau_{\mathrm{ij}} + \sum_{\mathrm{k=1}}^{\mathrm{m}} \Delta\tau_{\mathrm{ij}}^{\mathrm{k}} + \mathrm{e}\Delta\tau_{\mathrm{ij}}^{\mathrm{bs}} \tag{7.7}$$

### 3.6.2 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 nejoptimálnější 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 $\tau^{bs}$. Samotná úprava hran je pak následovná:

$$\tau_{ij} = \tau_{ij} + \sum_{r=1}^{w-1}(w - r)\,\Delta\tau_{ij}^{r} + w\Delta\tau_{ij}^{bs} \quad (7.8)$$

Pořadí mravence je označeno *r*.
$\Delta\tau_{ij}^{r}$ dostaneme použitím vzorce 7.4 a úprava feromonu pro nejlepší řešení je stejná jako u EAS.

Podle Bullnheimera (1999) má tento algoritmus mírně lepší výsledky než EAS, ale je znatelně výhodnější než pouze AS. [3]

### 3.6.3 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 pomalím odpařováním, podporuje rychlé prozkoumání grafu. [1]

Feromon pokládá pouze jeden agent. Tento mravenec je buďto zatím nejlepší nalezené řešení ($T^{bs}$) nebo nejlepší řešení nalezené v této iteraci ($T^{ib}$). [3]

$$\tau_{ij} = \tau_{ij} + \Delta\tau_{ij}^{best} \quad (7.9)$$

$$\Delta\tau_{ij}^{best} = \frac{1}{C^{bs}} \quad (7.10)$$

$$\Delta\tau_{ij}^{best} = \frac{1}{C^{ib}} \quad (7.11)$$

Pokud se pokládá podle $T^{bs}$, feromonová stopa na této trase se stane rychle velmi silnou a pro následující agenty neodolatelnou. Zatímco když se vyhledává podle iterace, je graf spíše prohledáván, než aby se vytvořila jedna 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]

### 3.6.4 Ant colony system - ACS

Vychází z AS, ale na rozdíl od něj a jeho úprav, které byli víceméně orientované okolo pokládání feromonu a jeho množství, ACS podniklo razantnější kroky. Hlavní rozdíly se dají shrnout do těchto tří částí. Zaprvé používá pseudonáhodné proporční pravidlo. 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 odebrán feromon. [1] [3]

#### 3.6.4.1 Pseudonáhodné proporční pravidlo

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

Zneužití (exploitation) je dán 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 $\eta_{ij}$ a $\tau_{ij}$.

$$j = \begin{cases} algmax_{l \in N_i^k} \left\{ [\tau_{ij}]^{\alpha} [\eta_{ij}]^{\beta} \right\}, & q \leq q_0 \ (exploitation) \\ J, & q > q_0 \ (exploration) \end{cases} \tag{8.1}$$

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]

#### 3.6.4.2 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]

$$\tau_{ij} = (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij}^{bs}, \qquad \forall(i,j) \in T_{ij}^{bs} \tag{8.2}$$

Nanášené množství je upravováno parametrem ρ.
Hodnotu $\tau_{ij}^{bs}$ dostaneme stejně jako u EAS ze vztahu 7.6.

#### 3.6.4.3 Úprava feromonové stopy po použití

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

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

$$\tau_{ij} = (1-\xi)\tau_{ij} + \xi\tau_0 \tag{8.3}$$

Přibili tu dva parametry $\tau_0$ a ξ.

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

Druhý parametr je závislí na grafu, kde provádíme výpočty.

$$\tau_0 = \frac{1}{nC^{nn}} \tag{8.4}$$

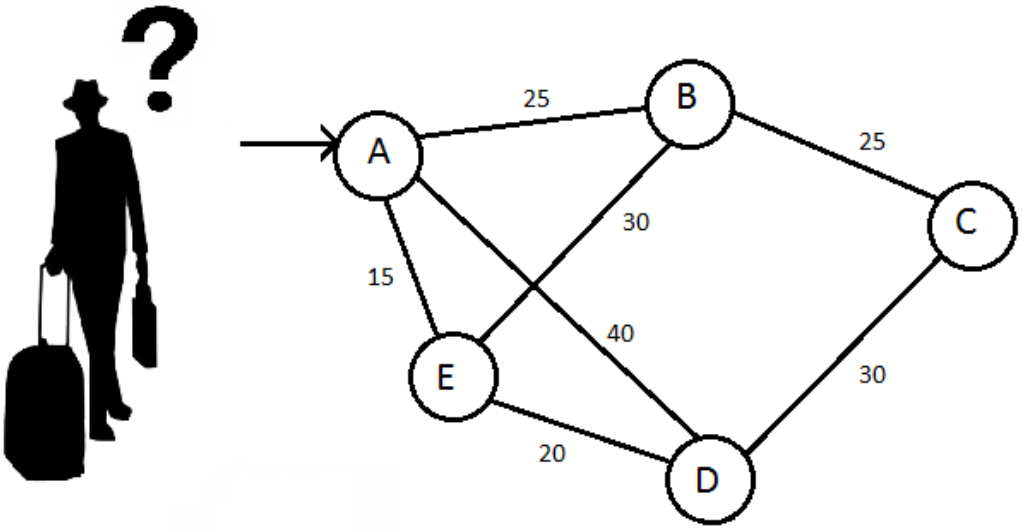
Proměnná $n$ je počet vrcholů v grafu a $C^{nn}$ je délka trasy vygenerovaná pomocí nejbližšího souseda (nearest-neighbor tour). [9] Generování této cesty probíhá tak, že se vždy vybere hranu, která vede k nejbližšímu městu nejkratší cestou. Nevrací se do vrcholu, kterým již prošel kromě původního města. To 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 tím paralelní a sériový průzkum probíhá jinak. Většinou se používá paralelní, ale zatím nebyly experimenty prokázány výhody jedné možnosti nad druhou. [3]

# 4. Traveling salesman problem (TSP)

## 4.1 Problém

Tento problém hledá nejkratší možnou cestu mezi městy s návratem do startovního bodu co má obchodník navštívit, tak, že každé město navštíví nanejvýš jednou. Samotný výsledek je pak pořadí měst.




Tento problém rozlišujeme na dva základní typy podle grafu. Buď je symetrický, nebo asymetrický.

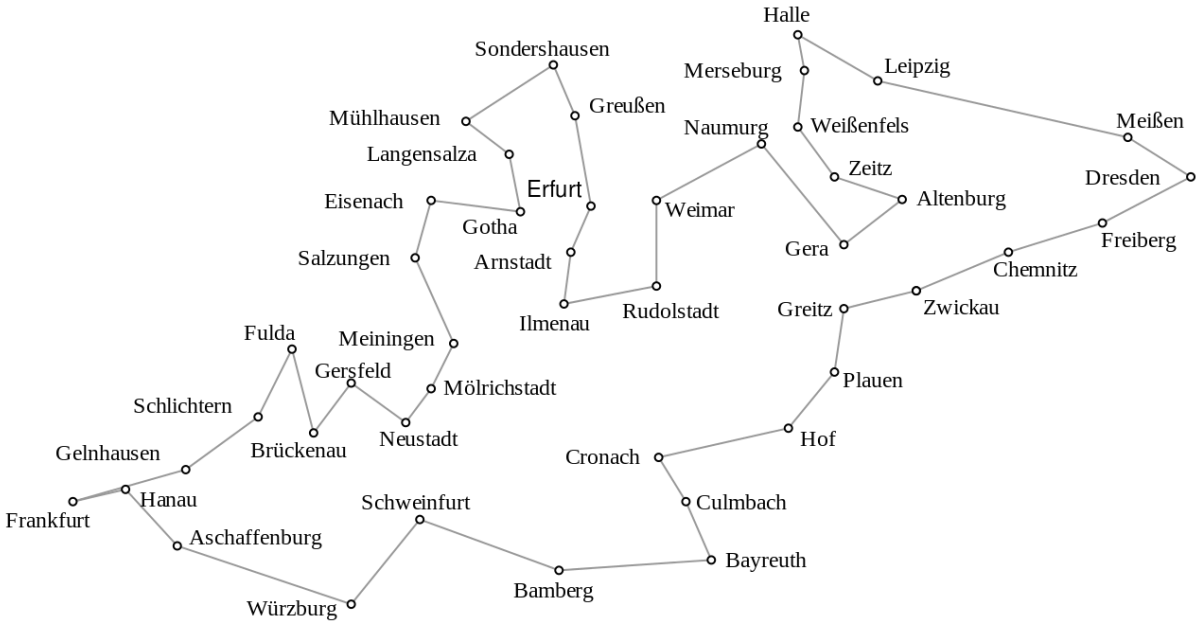
Asymetrický znamená, že vzdálenost z vrcholu A do B není stejná jako vzdálenost z B do A. Matematicky se to může zapsat jako $d_{AB} \neq d_{BA}$.

Graf znázorněný výše má na každé hraně pouze jednu hodnotu a tím pádem se jedná o graf symetrický.

## 4.2 Historie

První matematický výzkum tohoto problému začal roku 1934 na Princetonově univerzitě Hesslerem Whitneyem. Ovšem TSP byl naznačen v německé příručce pro cestující prodejce roku 1832 *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*. Bylo tam uvedeno pět tras, z toho čtyři byli organizované způsobem, kde některá města představovala základnu, do kterých se obchodník vracel po návštěvě okolních lokací.

Ovšem pátá cesta byla řešení TSP pro města v okolí Frankfurtu.




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

## 4.3 Aplikace

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 algoritmu analyzovat data z psychologie, rentgenové krystalografie a opravy vzduchových turbín. [27]

Jedna z nejvíce studovaných aplikací je pořadí prací pro sekvenční stroje. U tohoto příkladu je na hranách místo vzdálenosti cena provedení jedné práce hned po ukončení předchozí zakázky. To se využívá zejména u různých linek, kde při přecházení na výrobu jiného produktu se musí pozměnit některé části výrobního procesu. 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 řidičů 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. Nejen, že za stejný čas je řidič schopen vyřídit více práce, ale zároveň se při správně navržené cestě ušetří pohonné hmoty a tím se stává 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 *chinese postman problem* a *vehicle routing problem*.

Tento algoritmus se dá použít i pro osobní plány, například o dovolené. Můžeme si tak naplánovat trasu pro prohlížení památek a zajímavostí v dané destinaci. Vznikne tak plán, kde navštívíme požadované lokality v dostupném čase.

Za zmínku stojí také trasy, které prováděli obvodoví soudci v Británii v patnáctém století. Ti měli předem stanovené cesty, kde se budou provádět po určitou dobu soudy. Tento princip pak převzali v USA. [10] Kdyby tyto cesty nebyly optimalizované, mohlo by se stát, že by soudci trávili více času na cestách mezi městy, než samotným vykonáváním své práce.

## 4.4 Hledání řešení

Tento problém je NP-hard. 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 nemusí najít nejoptimálnější trasu, ale poskytují dostatečně dobré výsledky za značně kratší dobu.

### 4.4.1 ACO a Nearest Neighbor

Jeden z takových algoritmů je i optimalizace mravenčí kolonie. Mezi další často používané metody patří například nejbližší soused, který byl zmíněn u ACS. Jedná se o rychlý algoritmus (O ($n^2$)), ale jeho výsledky nejsou příliš dobré. [12]

### 4.4.2 Greedy

Další možností je algoritmus *Greedy*. U tohoto postupu se nejprve seřadí všechny hrany vzestupně a postupně se přidávají do výsledného řešení. Hrany, které jsou přidávané, ale nesmějí porušit následující pravidla: nesmí se vytvořit smyčka před tím, než se projdou všechny vrcholy a k žádnému vrcholu nesmí vést více než dvě hrany. Tento algoritmus má lepší výsledky než nejbližší soused, ale už je náročnější na výpočet (O ($n^2 log_2(n)$)). [12]

### 4.4.3 Genetický algoritmus

Tím se dostáváme k časově náročnějším algoritmům jako třeba genetické algoritmy. Jedná se o meta-heuristiku. Stejně jako optimalizace mravenčí kolonie 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. Při 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 dalšího rodiče a tím se vytvoří dva potomci. Toto rozdělení může proběhnout na jednom (single-point crossover) nebo více místech. Experimentovalo se hlavně s dvěma body křížení (two-point crossover), protože rozdělení pouze v jednom bodě nemůže vytvořit všechny možné variace a zároveň schéma s dlouhou délkou mají tendenci být zničena tímto způsobem křížení. Další možností je *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 neměnili. 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. U tohoto algoritmu to znamenalo přidávat rodiče s nejvyšší fitness s jeho potomky do vytváření nových potomků. Tím se zabezpečilo, že algoritmus nemohl postupem času generovat horší výsledky. [13]

## 4.5 Varianty

Tyto upravené vznikly se základem situací z reálného života a potencionálních aplikací. Většinou se liší od TSP pouze malými úpravami.

### 4.5.1 The Max TSP

Na rozdíl od TSP, kde se hledá nejkratší cesta mezi všemi vrcholy, Max TSP hledá nejdelší možnou trasu, kterou se může obchodník vydat. Toho lze dosáhnout negací velikosti hran.

### 4.5.2 The bottleneck TSP

V tomto případě hledáme trasu grafem G s co nejmenší nejdelší hranou. Toho dosáhneme umocněním velikosti hran před zahájení algoritmu.

### 4.5.3 Messenger problem

Známí také jako *wandering salesman problem*. Tento problém hledá nejkratší hamiltonovu 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 tak k tomu také můžeme 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]

#### 4.5.4 TSP s možností víckrát navštívit města (TSPM)

Zde musí výsledná trasa alespoň jednou navštívit každé město a skončit stejně jako u TSP v prvním vrcholu. Úprava grafu pro takovýto způsob spočívá v přidání nejkratších tras vypočítaných podle jiných algoritmů.

# 5. Vehicle routing problem (VRP)

## 5.1 Problém

VRP řeší rozvoz nebo svoz předmětů pomocí většího počtu vozidel, tak aby nebyly porušeny předem stanovené konstanty, s co nejkratší nebo nejrychlejší trasou, přičemž vždy vyjíždí z jednoho nebo více dep. Mezi tyto konstanty patří zejména velikost, váha, typ a kapacita vozidel. Jsou důležité kvůli, různým možným omezení na silnicích. Ať už se jedná uzavření v centrech měst, nemožnost překročení limitů mostů, podjezdů, tunelů a podobně. Kapacita nám udává, kolik toho může vozidlo 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 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 do té samé destinace mimo dopravní špičku.

K tomuto se vztahují takzvaná časová okna, která určují časový úsek, kdy má být předmět vyzvednut / dopraven. Může být jedno, ale i více pro jednu lokaci. Rozdělujeme je na dva typy. První představuje takové časové okno, které není možné nedodržet. Ty nazýváme tvrdé (hard). Pokud je možné okno ignorovat, jedná se o měkká (soft). Pokud se měkké časové okno nedodrží, bude zaplacena pokuta. To si můžeme z reálného světa představit, jako jsou různé akce u rozvozu pizzy, pokud nebude doručena do 30 minut, bude jídlo zdarma a jiné.

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

Graf pro tento problém má $n$+1 vrcholů, vrchol 1 je, kde každé vozidlo začíná a končí svou trasu v depu, $n$ je počet zákazníků. Zákazníci jsou pak zbylé vrcholy $S_c = \{2,3,4, \dots, n+1\}$. Každý zákazník $i$ má předem známý objem zboží $q_i$. Pokud je vozový park homogenní tak počet takových vozidel je $m$ $S_v = \{1,2,3, \dots, m\}$. 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. Dantzig a J. H. Ramser roku 1959. [22] Kde řešili nejoptimálnější trasu pro auta rozvážející benzín.

## 5.2 Aplikace

Protože se jedná o více specifický problém, než byl předešlý TSP, nachází uplatnění zejména u firem zaměřených na transport.

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

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

Z těchto čísel vidíme, že nesprávná volba tras může mít nemalé důsledky, nejen pro ekonomiku dodavatele, ale má to vliv i na jeho ceny a tím pádem pro všechny ostatní zákazníky ať už přímo (cena jízdenek) nebo nepřímo (cena zboží v obchodech).

O jaké příklady se tedy přesněji jedná? Mezi hlavní problémy co spadají pod VRP patří rozvozy jídel, jako pizza zmíněná v předešlé části, 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][18][17]

## 5.3 Hledání řešení

Stejně jako TSP tak i VRP je NP-hard problém. Přestože máme k dispozici postupy pro přesné řešení, nejsme schopni dostat výsledek v rozumném čase. Mezi algoritmy pro

přesné řešení patří například Column Generation, branch-and-cut, set-covering approuches a Lagrangian relaxation. V závislosti na metodě můžeme tyto postupy použít na 50 – 100 zákazníků. [19] Kvůli tomuto omezení se hojně používají metaheuristické algoritmy jako Tabu Search, Ant Colony Optimization, evoluční a genetické algoritmy a heuristicky, zejména local search. [7][19][21] Pokud máme malý objem zákazníků nebo míst, které je třeba navštívit, je možné použít některý z přesných řešení. [19]

Kvůli omezením, kterým podléhají přesné algoritmy (zejména dlouhé výpočetní časy pro větší obsah dat), nejsou příliš vhodné pro praktické aplikace. Proto se obracíme na metaheuristické řešení. Patří mezi ně ACO, evoluční a genetické algoritmy a Tabu Search. Tabu Search vychází z Local Search, ale je upraven o možnosti vymanění z lokálního optima.

Termín metaheuristika byl definován ve stejné práci jako Tabu Search (Glover 1986). Jedná se o metodu, která upravuje a navádí jinou heuristickou metodu a tím může dosáhnout lepších výsledků. [25]

### 5.3.1 Local Search

Tento typ algoritmu nevytváří řešení, ale prohledává množinu možných řešení s cílem nalézt to nejoptimálnější. Vždy se prohledává sousedy momentálního řešení.

Definice souseda závisí na řešeném problému, u TSP a VRP 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 se dá nalézt jak nejkratší tak nejdelší trasa.

Pokud změníme pouze dvě hrany, jedná se o takzvaného souseda dvou změn (*2-change heighborhood).* [24] Řešení A, B jsou sousedi, pokud můžeme z řešení A předem definovanou změnou dostat řešení B. U 2-change heighborhood 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 a tím to stávající nahradí. Tento způsob se nazývá *iterative improvement algorithm* známý také jako *hill climbing algorithm*. Velkým rizikem při používání takového postupu je uvíznutí v nedostatečném lokálním optimu. [23] Tomu se dá do jisté míry omezit povolením přesouvání na horší řešení nebo vícery běhy algoritmu s jinou počáteční hodnotou. [23]

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

### 5.3.2 Tabu Search

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

V paměti nejsou zaznamenávány 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 upravené množiny $N^*_{(x)}$, ve které se nevyskytují sousedi, kterých by se dosáhlo úpravami uloženými v Tabu listu. V této nové množině se můžou vyskytovat i taková řešení, která nebyla v původní $N_{(x)}$, mezi nově zařazené možnosti patří dříve použité řešení, popřípadě dobře hodnocené sousedy těchto starších řešení. To znamená, že sousedi řešení $x$ nejsou statický, ale dynamický. [26]

Klíčovým aspektem je správné využívání paměti mezi vyhledáváním nových řešení (intensification) využíváním dobrých řešeních naleznutých 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 neboli bohaté VRP, většinou se jedná od zařazení více dep, vozidla musí provést více tras, rozdílné kapacity vozidel a jiné. [21]

### 5.4.1 Kapacitní VRP (CVRP)

Základní verze VRP je CVRP. 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 50 zákazníků. [7]

### 5.4.2 VRP s časovými okny (VRPTW)

Zde se navíc ke kapacitě z CVRP přibudou časová okna a service time. Pro zákazníka i je interval dán $[a_i, b_i]$ a service time $s_i$. Kallehauge a kolegové (2006) vyřešili tento problém pomocí *Lagrangian relaxation* pro grafy o 400 a 1000 vrcholů, ale časová okna musí být tvrdá. [7]

### 5.4.3 VRP s vyzvednutí a doručení (VRPPD)

Pokud se musí balíky před dodání 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, ty se můžou vztahovat jak k vyzvednutí, tak k donášce, případně lze nastavit časy pro obě aktivity. [17]

### 5.4.4 Vehicle scheduling problems (VSPs)

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

### 5.4.5 Otevřené VRP (OVRP)

Od CVRP se liší pouze v tom, že se po obsloužení posledního zákazníka vozidlo nevrací do depa. [16] Použití 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 with Backhauls (VRPB)

Jedná se o rozšíření CVRP, kde jsou vrcholy rozděleni 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 obsloužit všechny dodávky. [17]

### 5.4.7 VRP s heterogenním vozovým parkem

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

### 5.4.8 VRP s rozdělenou dodávkou (Split Delivery VRP)

Pokud je zásilka větší, než kapacita dostupných vozidel je třeba ji rozdělit. V některých případech může být rozdělení i malých zásilek, může výrazně ušetřit na konečné ceně. Podle Dror a Trudeau je možné jakoukoliv zásilku rozdělit na více menších zásilek. [17]

### 5.4.9 Periodický VRP (PVRP)

Pokud zákazních 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 rozváží materiály pro výrobní závody a jiná zařízení. [20]

#### 5.4.9.1 Periodic VRP with Service Choice (PVRP-SC)

Jedná se o rozšíření PVRP, kde se frekvence návštěv zákazníků získává pomocí modelu. Každý zákazník může mít jiné nároky, ať už jak často má být či pokud jsou vůbec ochotní za častější dodávky platit. Správným modelem tedy můžeme zabránit zbytečnému zásobování lokací, které to nevyžadují a zároveň, aby ve více vytížených obchodech nenastal nedostatek zboží. [20] Dá se považovat za mezi stupeň PVRP a IRP, které bude zmíněno níže. [17]

#### 5.4.9.2 Dodatek k PVRP a PRVP-SC

Tyto problémy spadají pod kategorii opakovaného zásobování, mezi podobné problémy patří *Inventory routing problem* (IRP), který se od PVRP liší zejména v tom, že nemá pořadí zákazníků, místo toho funguje na principu doplňování zásob, tak aby cílová lokace nikdy nebyla bez zásoby. Sama společnost rozhoduje jak velkou dodávku a kdy ji pošlou. Kvůli tomu musí být trasa navrhnuta pro každý den zvlášť, přičemž závisí na zákaznících, kteří se budou ten den obsluhovat a množství, které se každému bude dodávat.[17]

Od toho se dostáváme k *Vendor managed inventory* neboli inventář řízený dodavatelem. Dodavatel je tedy zodpovědný za dostatek zboží na destinaci 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, že pokryje co nejvíce zákazníkům z daného okolí nebo vyslání plného vozidla 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. Aby tento systém fungoval, 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 i Shell Chemical, Fruit of the Loom a jinými. [18] Hlavní uplatnění VMI nalézá u doplňování supermarketů, udržování dostatek materiálu pro výrobní linky, zásobování výdejních automatů a paliva do benzínových pump. [17]

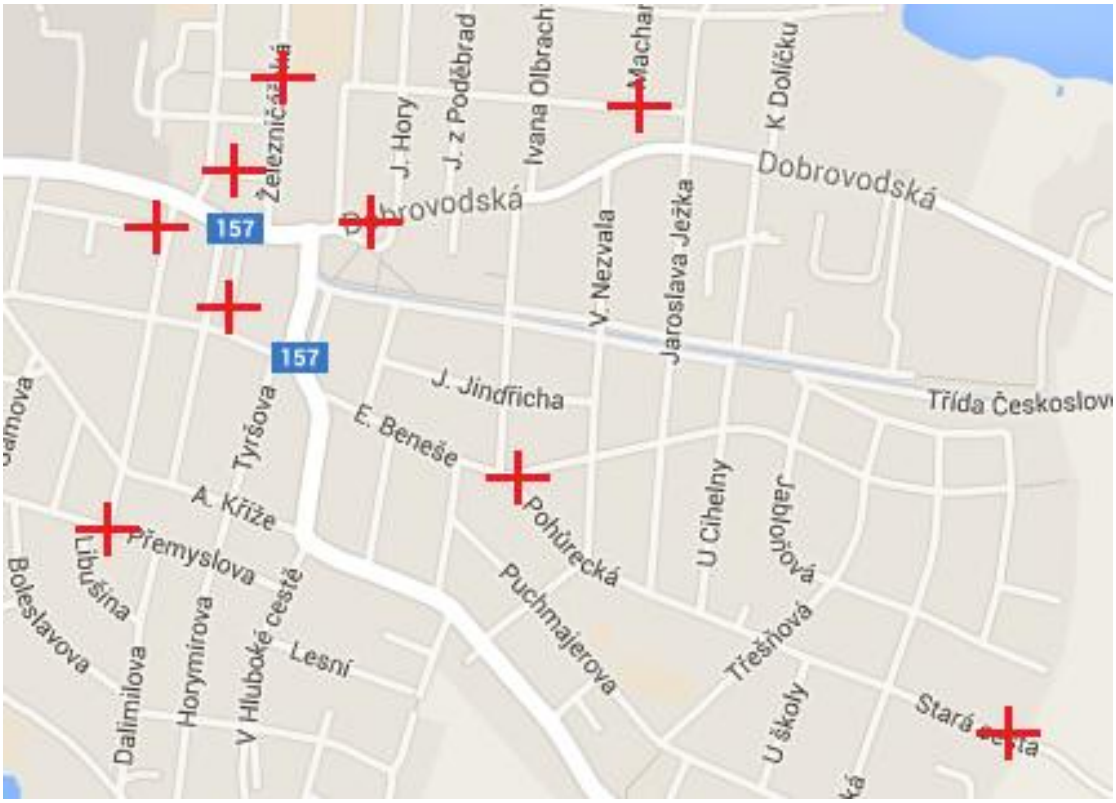
# 6. Aplikace Optimalizace mravenčí kolonií na svoz odpadu v Českých Budějovicích

Optimalizace mravenčí kolonií bude aplikována na část Českých Budějovic, přesněji na Suché Vrbné. Bude použita optimalizace mravenčím systémem (kapitola 3.6.4).

## 6.1 Data

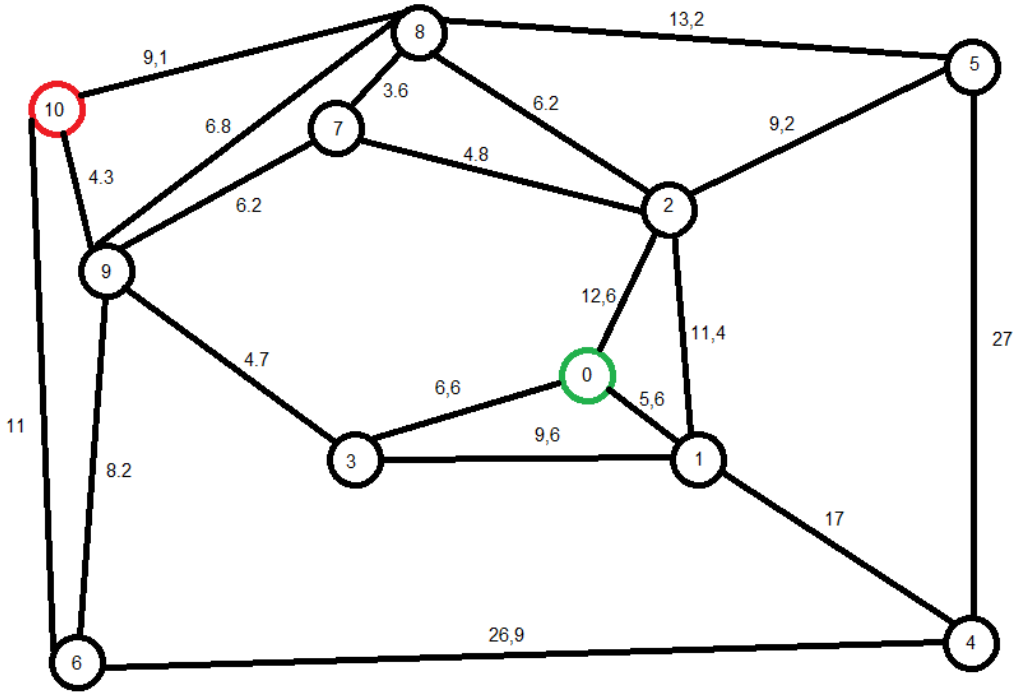
Veškerá použitá data byla obstarána z internetu. Mapa ze stánky https://www.google.cz/maps/@48.9712627,14.5044861,15.75z. Lokality kontejnerů na tříděný odpad má na svých stránkách dostupný magistrát města České Budějovice. http://www.c-budejovice.cz/cz/magistrat/odbory/osvs/stranky/nadoby-na-trideny-odpad.aspx

Po dosazení umístění kontejnerů na mapu dostaneme následující obrázek.




Z toho získáme vzdálenosti mezi jednotlivými lokalitami. Úpravou této mapy dostaneme graf, kterým bude následně algoritmus procházet.

## 6.2 Graf




Velikost hrany je dána součtem trasy nutné k dosáhnutí jiného kontejneru. Zároveň zde přibili další dva vrcholy. Zelený značí start, odkud budou všechna auta vyjíždět. Červený vrchol je konečná destinace všech aut, které ze zeleného vyjedou. Vzdálenosti nejsou dány v žádné jednotce, slouží pouze k udání poměru.

## 6.3 Implementace

Hodnoty a vztahy (propojení) mezi vrcholy jsou součástí programu.

Nejprve zjistíme základní vrstvu feromonu, která bude nanesena na všechny hrany. Tu získáme podle vzorce 8.4. Jsou zde dvě proměnné, počet mravenců (n) a $C^{NN}$, kterou získáme pomocí algoritmu *nearest neighbor search.* Kvůli možnému uvíznutí využijeme třídu Stack, pomocí které, se v takovém případě vrátíme na předchozí vrcholy, ze kterých může vést hrana na zatím nenavštívené vrcholy. Samotný kód pak vypadá následovně.

### 6.3.1 Inicializace

```
public static double Inicializace()
    {
      for (int k = 0; k < PocetVrcholu; k++)
      {
        Vrchol vrchol = new Vrchol(k);

        Vrcholy[k] = vrchol;
      }
      Stack<int> s = new Stack<int>();
      Visited.Add(0);
      a = new int[100];
      s.Push(0);
      double NNDelkaTrasy = 0;
      int Curr_loc = 0;
      int Next_loc = 0;
      double[,] Dostupny = new Double[PocetVrcholu, PocetVrcholu];

      while(a.Count()!=PocetVrcholu)
      {
        for (int j = 0; j < PocetVrcholu; j++)
        {
          if (j == Curr_loc || Visited.Contains(j) || !Vrcholy[Curr_loc].Spojeni.Contains(j))
          {
            Dostupny[Curr_loc, j] = double.MaxValue;
          }
          else Dostupny[Curr_loc, j] = Vzdalenost[Curr_loc, j];
          if (Dostupny[Curr_loc, j] != double.MaxValue)
          {
            Next_loc = j;
          }
        }
        for (int i = 0; i < PocetVrcholu; i++)
        {
```

```
            if (Dostupny[Curr_loc, i] < Dostupny[Curr_loc, Next_loc])
            {
                Next_loc = i;
            }
        }
        if (Dostupny[Curr_loc, Next_loc] == double.MaxValue)
        {
            s.Pop();
            Next_loc = s.Pop();
        }
        NNDelkaTrasy += Vzdalenost[Curr_loc, Next_loc];
        Curr_loc = Next_loc;

        if (!Visited.Contains(Curr_loc))
        {
            s.Push(Curr_loc);
        }
        Visited.Add(Curr_loc);
        a = Visited.Distinct().ToArray();
    }

    return NNDelkaTrasy;
}
```

Třída Vrchol obsahuje konstruktor Vrchol (int Curr_loc), který podle momentálně navštíveného vrcholu (Curr_loc) vrací list, který obsahuje vrcholy, na které se můžeme dostat. Pole dostupné slouží k uložení vzdáleností mezi vrcholy, které můžeme bez postihu upravit. Pokud se vzdálenost upravuje, znamená to že se z Curr_loc nemůžeme dostat na vrchol Next_loc (vrchol, který se zvažuje jako následný). Pokud se z vrcholu můžeme pohnout pouze po upravené vzdálenosti vrátí se algoritmus na předchozí vrchol pomocí zásobníku. To probíhá, dokud list Visited neobsahuje všech 11 vrcholů. Návratová hodnota této metody je vzdálenost, kterou algoritmus prošel, než nalezl všechny vrcholy.

### 6.3.2 Feromon

Základní feromon se nanese na hrany pomocí metody

```
public static double[,] GetFeromon(double NNDelkaTrasy)
    {
        Feromon = new Double[PocetVrcholu, PocetVrcholu
      for (int i = 0; i < PocetVrcholu; i++)
      {
        for (int j = 0; j < PocetVrcholu; j++)
        {
          if (Vrcholy[i].Spojeni.Contains(j))
          {
            Feromon[i, j] = 1 / (NNDelkaTrasy * PocetVrcholu);
          }

          else Feromon[i, j] = 0;
        }

      }
      FeromonIn = 1 / (NNDelkaTrasy * PocetVrcholu);
      return Feromon;
    }
```

Znovu se využívá list z třídy Vrchol, aby se feromon nanášel pouze na existující hrany. Za zmínku zde stojí také proměnná FeromonIn, ta je potřebná při lokální úpravě feromonu během procházení grafu mravenci. Po každé iteraci se zároveň provede globální úprava feromonu podle vzorce 8.2. Tato úprava, přestože se nazývá globální, se provádí pouze na nejlepší nalezené trase.

### 6.3.3 Presun

Po získání základních proměnných se vytvoří instance třídy Mravenec. V této třídě se aplikuje samotný algoritmus. Voláním metody Trasa () započne vytváření cesty pro první vozidlo této iterace. Tato metoda pouze volá jiné metody a kontroluje podmínky ukončení iterace. Ty jsou následující: vozidlo nesmí být plné, nesmí být vybrány všechny kontejnery, zároveň pokud jsou všechny kontejnery navštíveny, algoritmus se ukončí. Jednou z volaných metod je Presun(int curr_loc).

```
public void Presun(int curr_loc)
    {
        int next=VyberHrany(curr_loc);
        if(curr_loc!=next){Graf.Feromon[curr_loc, next] = (1 - FeromonUpdate) *
Graf.Feromon[curr_loc, next] + Graf.FeromonIn;}
        if (curr_loc == next)
        {
            this.DelkaTrasy += 0;
        }
        else this.DelkaTrasy+=Graf.Vzdalenost[curr_loc,next];

        if (this.Curr_loc != next)
        {
            this.Visited.Add(next);
            this.zasobnik.Push(next);
            this.Curr_loc = next;
        }
        Nakladani();
    }
```

Její hlavní účely jsou zavolat VyberHranz(curr_loc), vkládat do zásobníku navštívené vrcholy a zajistit lokální úpravu feromonu.

Výběr hrany se provádí pomocí pseudonáhodného pravidla (vzorec 8.1), podle náhodného čísla se algoritmus rozhodne, zda bude prohledávat graf pro nová řešení, nebo jestli vybere nejpravděpodobnější hranu.

### 6.3.4 VyberHrany

```
    public int VyberHrany(int Curr_loc)
    {
        double Q = rnd.NextDouble();
        double MaxPrav;
        double[,] Pravdepodobnost;
        Pravdepodobnost = new double[PocetVrcholu, PocetVrcholu];
        double[] sance;
```

```
            double sanceComlp = 0;
            double total;
            double OkoliVysledek = Okoli(Curr_loc);
            Vrcholy = new Vrchol[this.PocetVrcholu];
            Vrchol vrchol = new Vrchol(Curr_loc);
            Vrcholy[Curr_loc] = vrchol;

            {
               for (int i = 0; i < PocetVrcholu; i++)
               {
                  if (this.Visited.Contains(i) || !Vrcholy[Curr_loc].Spojeni.Contains(i))
                  {
                     Pravdepodobnost[Curr_loc, i] = 0;
                  }
                  else Pravdepodobnost[Curr_loc, i] =
(System.Math.Pow(Graf.Feromon[Curr_loc, i], Alfa) * System.Math.Pow(1 /
Graf.Vzdalenost[Curr_loc, i], Beta)) / OkoliVysledek;
               }
               if (Q0 < Q) //explore
               {
                  sance = new double[PocetVrcholu];
                  for (int i = 0; i < PocetVrcholu; i++)
                  {
                     sance[i] = Pravdepodobnost[Curr_loc, i];
                     sanceComlp += Pravdepodobnost[Curr_loc, i];
                  }
                  if (sanceComlp == 0)
                  {
                     return this.zasobnik.Pop();
                  }
                  total = 0;
                  double random;
                  do
                  {
```

```
                random = rnd.NextDouble();
            } while (random == 0);
            for (int i = 0; i < PocetVrcholu; i++)
            {
                total += sance[i];
                if (total >= random)
                {
                    Next_loc = i;
                    break;
                }
            }
            return Next_loc;
        }
        else //exploit
        {
            MaxPrav = double.MinValue;
            Next_loc = int.MinValue;
            for (int i = 0; i < PocetVrcholu; i++)
            {
                if (MaxPrav < Pravdepodobnost[Curr_loc, i])
                {
                    MaxPrav = Pravdepodobnost[Curr_loc, i];
                    Next_loc = i;
                }
            }
            if(Pravdepodobnost[Curr_loc,Next_loc]==0)
            {
                return this.zasobnik.Pop();
            }
            return Next_loc;
        }
    }
}
```

Tato velmi obsáhlá metoda rozhoduje o následujícím vrcholu, jak bylo zmíněno, výše rozhoduje mezi explore a exploit grafu. Toho je dosáhnutu náhodným číslem Q.
Pokud se stane, že není možné přesunout se na nenavštívený vrchol, použije se zásobník k prohledávání předchozích vrcholů.

### 6.3.5 Nakladani

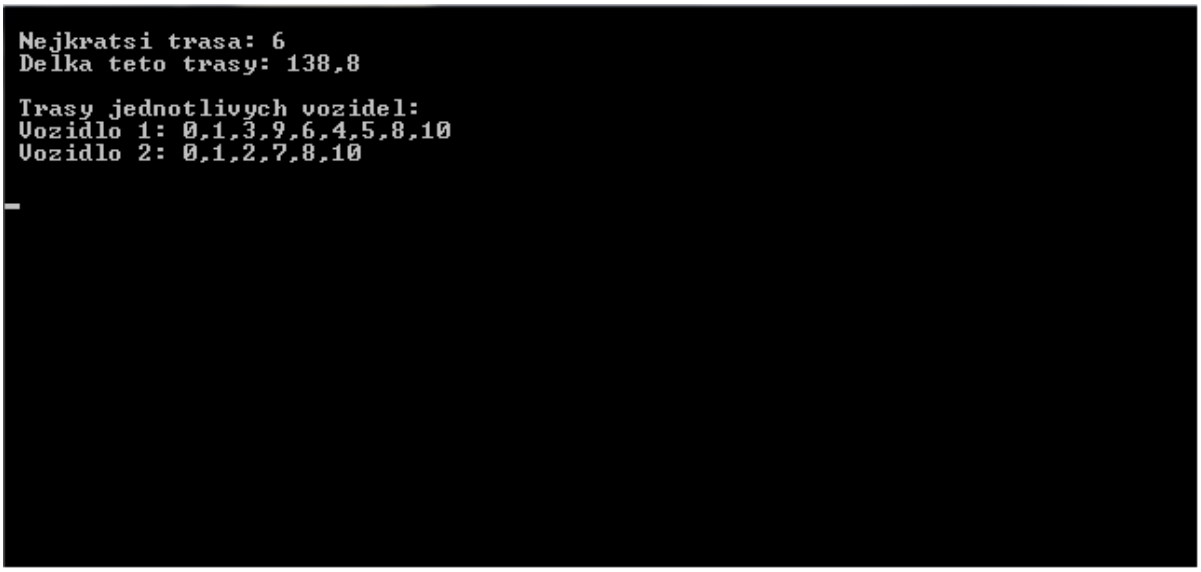
Při prvním navštívení vrcholu během každé iterace se tento vrchol zavede do listu Nabrano, ale pouze pokud je vozidlo na tomto vrcholu schopné odvést všechen náklad na stanovišti. Nabrano se po každé iteraci nuluje.

```
public void Nakladani()
    {
      double pomocNaklad = 0;
      if(Curr_loc!=0){
          pomocNaklad = this.Naklad + pravVse[this.Curr_loc-1];
      }
       if (!Nabrano.Contains(this.Curr_loc))
       {
          if (this.Curr_loc != 0 && this.Curr_loc != 10 && pomocNaklad<KapacitaMax)
          {
             this.Naklad = pomocNaklad;
             Nabrano.Add(this.Curr_loc);
          }
          else this.plno = true;
       }
    }
```

Množství na každém vrcholu je náhodné, ale pro jedno spuštění programu jsou tyto hodnoty pro všechny iterace stejné. Přestože druhé vozidlo ví, kolik kontejnerů bylo vybráno, neví které, takže prohledává graf, dokud nenajde zbývající lokace nebo dokud není naplněno.

Při splnění některé z podmínek ukončení dané v Presun (curr_loc) provede se ukončení. To spočívá v poslání vozidla nejkratší cestou k poslednímu vrcholu. Tato cesta je předem dána. Po ukončení všech tras vozidel v iteraci, spojí se jejich trasy do jednoho řetězce znaků, kde si každé vozidlo zachová svou trasu.

Výsledek je pak ID nejlepší iterace, její délka pořadí navštívených vrcholů pro jednotlivá vozidla.



# 7. Závěr

Tato práce popsala historii optimalizace mravenčí kolonie, od matematického popsání chování mravence při vytváření feromonových stezek přes řešení prvních problémů spojených s implementací tohoto chování po algoritmy schopné řešit obsáhlé objemy dat, které nejsou přesné algoritmy schopné vyřešit během přípustného času nebo nejsou schopné je řešit.

Optimalizace by měla být vždy jedním z cílů jakéhokoliv projektu.

# 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