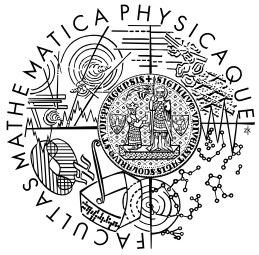


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Vojtěch Raja

Location Routing Problem

Katedra pravděpodobnosti a matematické statistiky

Vedoucí bakalářské práce: Ing. Vít Procházka Ph.D.
Studijní program: Finanční matematika

Praha 2025

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ....

Podpis autora

Rád bych poděkoval Ing. Vítu Procházkovi, Ph.D., za odborné vedení mé bakalářské práce. Jeho vstřícný přístup a ochota konzultovat v případě jakýchkoliv nesnází — vždy s úsměvem — pro mě byly při psaní velkou oporou.

Název práce: Location Routing Problem

Autor: Vojtěch Raja

Katedra: Katedra pravděpodobnosti a matematické statistiky

Vedoucí bakalářské práce: Ing. Vít Procházka Ph.D., Katedra pravděpodobnosti a matematické statistiky

Abstrakt: Location Routing Problem je výpočetně náročná kombinatorická optimalizační úloha, která nalézá uplatnění v logistice. Vzhledem k její výpočetní složitosti je však nalezení optimálního řešení pro rozsáhlejší instance prakticky neproveditelné. Proto se v praxi využívají aproximační metody, které sice negarantují nalezení optima, ale umožňují získat kvalitní řešení v přijatelném čase. Cílem práce je formální definice modelu, porovnání možných metod řešení s důrazem na vybraný aproximační algoritmus, jeho implementace v jazyce Python a následná analýza dosažených výsledků.

Klíčová slova: Location Routing Problem, Kombinatorická optimalizace, Metaheuristiky, Tabu prohledávání, Logistické plánování

Title: Location Routing Problem

Author: Vojtěch Raja

Department: Department of Probability and Mathematical Statistics

Supervisor: Ing. Vít Procházka Ph.D., Department of Probability and Mathematical Statistics

Abstract: The Location Routing Problem is a computationally demanding combinatorial optimization problem with applications in logistics. Due to its computational complexity, finding an optimal solution for larger instances is practically impossible. Therefore, approximation methods are commonly used in practice. Although they do not guarantee an optimal solution, they can provide high-quality results within a reasonable time. The aim of this thesis is to formally define the model, compare available solution methods with a focus on a selected approximation algorithm, implement it in Python, and analyze the obtained results.

Keywords: Location Routing Problem, Combinatorial optimalization, Metaheuristics, Tabu search, Logistics planning

Obsah

Úvod

Tato práce pojednává o *Location Routing Problem* (LRP), výpočetně složitém kombinatorickém optimalizačním problému. Úloha je formulována pomocí továren, které mohou být otevřeny, a zákazníků, jejichž poptávka musí být naplněna. Ti jsou přiřazeni k právě jedné rozvozové trase a následně obslouženi vybraným vozidlem, které vyjíždí z některé továrny a zase se do ní vrací. Úkolem je rozhodnout, které továrny otevřít a jak přiřadit zákazníky do tras tak, aby byly celkové náklady minimální.

Model LRP nachází široké uplatnění v logistice, která tvoří nezanedbatelnou část celkových nákladů mnoha podniků a jejíž efektivní nastavení může vést ke značným finančním úsporám. Nalezení optimálního řešení je však pro velké vstupní instance výpočetně neproveditelné, což v praxi vede k použití jiných metod - tzv. metaheuristik. Ty sice nalezení globálního optima nezaručují, ale dokáží se k němu přiblížit v dostatečné kvalitě a v rozumném čase.

Hlavním cílem této práce je formálně definovat model LRP, podrobně popsat vybraný metaheuristický algoritmus, implementovat ho v programovacím jazyce Python a následně diskutovat dosažené výsledky.

1 Prerekvizity

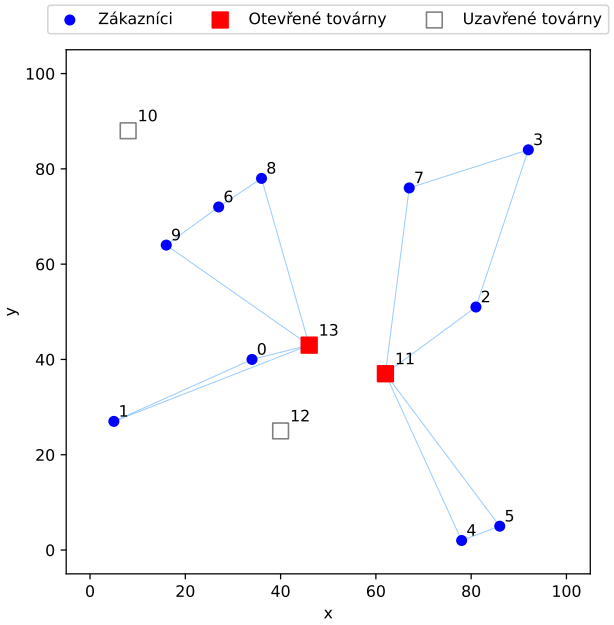
1.1 Popis modelu a jeho aplikace

Jak již bylo uvedeno v úvodu, LRP nalézá své využití nejčastěji v odvětví logistiky, typicky v distribuci zboží od dodavatele ke koncovým odběratelům. Pro jeho formulaci používáme následující základní komponenty, které jsou zároveň ukázány na obrázku 1.1.

- **Továrny** - výrobní místa, která je možné otevřít, vždy s předem danými fixními náklady. Prostřednictvím otevřené továrny je následně uspokojována poptávka zákazníků, kteří jsou k ní přiřazeni.
- **Vozidla** - zajišťují dopravu zboží z otevřených továren k zákazníkům, přičemž každé vozidlo má dané fixní náklady na pořízení. Do celkových nákladů se navíc započítává i vzdálenost, kterou vozidlo ujede, například formou nákladů na pohonné hmoty. Souhrnná poptávka zákazníků obsluhovaných jedním vozidlem nesmí překročit jeho kapacitu.
- **Zákazníci** - každý z nich poptává předem určené množství jednotek homogenního zboží.

Nalezení řešení problému spočívá v minimalizaci celkových nákladů tohoto logistického modelu, které sestávají z nákladů na otevření továren a pořízení a provoz vozidel. Je důležité zmínit, že proces určování, které továrny otevřít, alokace zákazníků k továrnám a následné určení tras vozidel probíhá v tomto modelu současně, separátní řešení může vést k neoptimálním řešením [1].

K tomu, abychom mohli formálně model LRP zavést, je třeba využít základní pojmy z teorie grafů a teorie optimalizace, těm jsou věnovány následující dvě podkapitoly.



Obrázek 1.1 Grafová reprezentace možného řešení modelu LRP o 10 zákaznících a 4 továrnách.

1.2 Využité pojmy z teorie grafů

Díky struktuře LRP se nabízí popsat model grafem. Vrcholy tohoto grafu představují zákazníky a továrny, zatímco hrany odpovídají cestám, po nichž vozidla zákazníky obsluhují. Následují vybrané definice z teorie grafů, díky nimž bude další práce s modelem jednodušší.

Definice 1. *Nechť V je neprázdná množina vrcholů a E množina hran, přičemž každá hrana je dvojice (u, v), kde $u, v \in V$, $u \neq v$. Potom graf definujeme jako uspořádanou dvojici $G = (V, E)$.*

Definice 2. *Graf $G = (V, E)$ nazveme úplným, pokud pro všechny dvojice vrcholů $u, v \in V$, $u \neq v$ existuje hrana (u, v).*

Definice 3. *Nechť $G = (V, E)$ je graf. Hranu (u, v) nazveme orientovanou, pokud je dvojice (u, v) uspořádaná. V takovém případě říkáme, že hrana vede z vrcholu u do vrcholu v. G nazveme orientovaným, pokud jsou všechny jeho hrany orientované.*

Definice 4. *Orientovaný graf $G = (V, E)$ nazveme ohodnoceným, pokud má k sobě každá hrana (u, v) přiřazena číselnou hodnotu c_{uv}. Tuto hodnotu nazýváme cena hrany. Pokud platí, že $c_{uv} = c_{vu}$ $\forall u,v \in V$, pak graf G nazveme symetrickým.*

Definice 5. *Řekneme, že úplný, ohodnocený graf $G = (V, E)$ splňuje trojúhelníkovou nerovnost, pokud pro každou trojici vrcholů $u, v, w \in V$ platí, že $c_{uw} \leq c_{uv} + c_{vw}$.*

Dále je potřeba formalizovat termín minimalizace nákladů za platnosti podmínek, které chceme, aby byly dodrženy. K tomu nám poslouží nadcházející podkapitola, která se týká teorie optimalizace.

1.3 Stručný souhrn teorie optimalizace

Tato část vychází z knihy Dupačové a Lachouta [2]. K matematické formulaci modelu pomůže následující definice, díky které lze celý problém, tedy minimalizaci celkových nákladů za platnosti omezení, jako například uspokojení poptávky, přesně formulovat.

Definice 6. *Úlohou matematického programování rozumíme úlohu minimalizovat (maximalizovat) funkci $f(x_1, \ldots, x_n)$ na množině řešení soustavy rovnic a nerovnic*

$$\begin{aligned} g_j(x_1, \ldots, x_n) &\leq 0, \quad j = 1, \ldots, m, \\ h_k(x_1, \ldots, x_n) &\geq 0, \quad k = 1, \ldots, p, \\ i_l(x_1, \ldots, x_n) &= 0, \quad l = 1, \ldots, r. \end{aligned}$$

Funkci f nazveme účelovou funkcí a funkce g_j, $j = 1, \ldots, m$, h_k, $k = 1, \ldots, p$, a i_l, $l = 1, \ldots, r$ jejími omezeními. Proměnné $x_1, \ldots, x_n$ nazýváme rozhodovací nebo také stavové proměnné.

Hodnoty rozhodovacích proměnných $x_1, \ldots, x_n$, které splňují všechna omezení nazveme přípustným řešením. Přípustné řešení, pro které je hodnota účelové funkce zároveň minimální (maximální) nazveme optimálním řešením dané úlohy.

Termín přípustné řešení je v této práci ekvivalentní pojmu stav a termín množina přípustných řešení pojmu stavový prostor.

Dle typu účelové funkce a omezení rozlišujeme více typů úloh matematického programování.

- Pokud je účelová funkce a všechna omezení lineární, tak mluvíme o úlohách lineárního programování, ty jsou již dobře prozkoumané a k jejich řešení jsou využívány efektivní algoritmy.
- Pokud jsou některé stavové proměnné, ale ne všechny, celočíselné, mluvíme o úloze smíšeného celočíselného programování. Pokud jsou všechny rozhodovací proměnné celočíselné, jedná se o úlohu celočíselného programování. Obě zmíněné metody jsou již výpočetně náročnější a jejich řešení často spočívá v postupném řešení relaxovaných úloh, tedy takových, ve kterých neuvažujeme omezení na celočíselnost.
- Pokud je množina přípustných řešení diskrétní, často generovaná kombinatorickými strukturami (např. permutacemi, kombinacemi či podmnožinami), mluvíme o úloze kombinatorické optimalizace. Tyto úlohy bývají modelovány pomocí celočíselných nebo binárních proměnných a jejich řešení je proto obvykle výpočetně velmi náročné.

Do posledně zmíněné kategorie spadá i LRP, ve kterém je otevření továrny, případně využití hrany modelováno právě binární rozhodovací proměnnou. V následující kapitole je již formálně zaveden celý model jako úloha matematického programování na grafu.

2 Matematická formulace

2.1 Značení

V následující části předpokládejme, že $G = (V = I \cup J,\ E)$ je úplný, orientovaný, ohodnocený a symetrický graf, který splňuje trojúhelníkovou nerovnost. Dále budeme termín hrana $(u,\ v)$ považovat za ekvivalentní pojmu cesta $(u,\ v)$. Trasou pak rozumíme posloupnost navazujících cest, která začíná a končí v libovolné továrně, a která neobsahuje vnitřní cyklus, tedy žádný zákazník na ní není navštíven vícekrát.

Na tomto grafu zaveďme následující značení:

I $\{i \mid i = 1, \ldots, R\}$ je množina továren, které je možné otevřít

J $\{j \mid j = 1, \ldots, N\}$ je množina zákazníků, které je nutné obsloužit

V $I \cup J$ je množina všech vrcholů

S $\{l \mid l = 1, \ldots, L\}$ je množina vozidel použitelných k dopravě zboží

c_{uv} jsou náklady na využití cesty (cena hrany) z u do v, $u,\ v \in V$

C_l jsou fixní náklady na pořízení vozidla $l,\ l \in S$

F_i jsou fixní náklady na otevření továrny $i,\ i \in I$

q_j je poptávka zákazníka j v jednotkách zboží, $j \in J$

Q_l je kapacita vozidla $l,\ l \in S$

Dále zaveďme tyto binární rozhodovací proměnné:

$$x_{uvl} = \begin{cases} 1, & \text{využije-li vozidlo } l \text{ hranu z vrcholu } u \text{ do vrcholu } v, \\ 0, & \text{jinak,} \end{cases}$$

$$z_i = \begin{cases} 1, & \text{je-li otevřena továrna } i, \\ 0, & \text{jinak.} \end{cases}$$

A nechť R_j je rozhodovací proměnná, která modeluje pořadí zákazníka j na jeho trase. I přesto, že R_j může nabývat pouze hodnot na množině přirozených čísel, tak je vhodné ji uvažovat jako spojitou a nepřidávat do modelu dodatečná omezení.

2.2 Model

Účelová funkce LRP může mít v různých variantách modelu lehce odlišnou podobu, která spočívá v tom, jaké veškeré náklady se rozhodneme do modelu zahrnout. Kromě nákladů na trasy a pořízení vozidel a na otevření továren, které jsou v modelech přítomné v zásadě vždy, mohou být také uvažovány náklady na mzdy zaměstnanců či jiné provozní výdaje.

Kvůli názornosti a vysoké srozumitelnosti uvádím matematickou formulaci tak, jak ji popsali autoři Tuzun a Burke [3]. Účelovou funkcí modelu *Location Routing Problem* rozumíme

$$\sum_{u\in V}\sum_{v\in V}\sum_{l\in S} c_{uv}x_{uvl} \; + \sum_{l\in S}(C_l \sum_{i\in I}\sum_{j\in J} x_{ij}) \; + \sum_{i\in I} F_i z_i, \tag{2.1}$$

kde první výraz odpovídá nákladům na dopravu zboží z továren k zákazníkům, druhý fixním nákladům na pořízení vozidel a poslední fixním nákladům na otevření továren. Omezeními na účelovou funkci modelu *Location Routing Problem* rozumíme následující výrazy:

$$\sum_{l\in S}\sum_{u\in V} x_{ujl} = 1 \quad \forall j \in J, \tag{2.2}$$

$$\sum_{j\in J}\sum_{u\in V} q_j x_{ujl} \leq Q_l \quad \forall l \in S, \tag{2.3}$$

$$\sum_{u\in V} x_{uwl} = \sum_{v\in V} x_{wvl} \quad \forall l \in S,\ w \in V, \tag{2.4}$$

$$\sum_{i\in I}\sum_{j\in J} x_{ijl} \leq 1 \quad \forall l \in S, \tag{2.5}$$

$$\sum_{l\in S} x_{hil} + z_h + z_i \leq 2 \quad \forall h,\, i \in I \tag{2.6}$$

$$\sum_{l\in S}\sum_{j\in J} x_{ijl} \geq z_i \quad \forall i \in I, \tag{2.7}$$

$$\sum_{j\in J} x_{ijl} \leq z_i \quad \forall l \in S,\ i \in I. \tag{2.8}$$

$$R_j - R_k + (R+N)\sum_{l\in S} x_{jkl} \leq R + N - 1 \quad \forall j,\, k \in J,\, j \neq k \tag{2.9}$$

$$x_{uvl} \in \{0,1\} \quad \forall u, v \in V,\ \forall l \in S \tag{2.10}$$

$$z_i \in \{0,1\} \quad \forall i \in I \tag{2.11}$$

Interpretace omezení:

- (2.2) - Počet využitých cest ke každému zákazníkovi je roven jedné.
- (2.3) - Celková poptávka zákazníků na trase každého vozidla nepřevyšuje jeho kapacitu.
- (2.4) - Pro každé vozidlo a každého zákazníka je počet využitých hran do vrcholu roven počtu využitých hran z vrcholu.
- (2.5) - Pro každé vozidlo je počet tras roven maximálně jedné.
- (2.6) - Není využita žádná hrana, která vede z továrny do továrny.
- (2.7) - Z každé otevřené továrny vyjede alespoň jedno vozidlo.
- (2.8) - Z uzavřené továrny žádné vozidlo nevyjede.
- (2.9) - Žádná trasa neobsahuje vnitřní cyklus složený pouze ze zákazníků. Pokud by některá trasa takový cyklus obsahovala, tak by nutně existovala využitá hrana, která vede od zákazníka j k zákazníkovi k a zároveň $R_j > R_k$, čímž bychom se dostali do sporu s touto podmínkou ($R_j - R_k > -1$).
- (2.10) a (2.11) - Omezení oboru hodnot stavových proměnných.

Řešení takto definovaného modelu spočívá v nalezení globálního minima funkce 2.1 na množině řešení rovností a nerovností 2.2 - 2.11.

Algoritmy pro řešení tohoto problému se dělí na dvě hlavní skupiny, a to exaktní a heuristické. První skupina garantuje nalezení optimálního řešení úlohy, ale pouze pro velmi omezenou velikost vstupu, a proto se v praxi častěji využívá spíše druhá skupina algoritmů, které umožňují pracovat s větší velikostí vstupních dat. Ty na druhou stranu nezaručují nalezení přesného řešení, pouze se k němu dokážou přiblížit na rozumnou, většinou poměrně dostačující úroveň.

Exaktním procedurám využívaných pro nalezení optimálního řešení je věnována následující kapitola.

3 Exaktní algoritmy

LRP spadá do třídy NP-hard problémů, což znamená, že jeho řešení je alespoň tak obtížné, jako řešení kteréhokoli problému třídy NP - množiny rozhodovacích problémů, jejichž řešení lze v polynomiálním čase ověřit, pokud je k dispozici [4]. Důsledkem je značné omezení velikosti vstupních instancí, které lze exaktními metodami vyřešit v přijatelném čase. V praxi se tyto algoritmy využívají jen zřídka a proto se jim v této práci podrobně nevěnuji, pouze stručně zmiňuji principy, na kterých jsou založeny. Následující podkapitoly vycházejí z knihy autorů Chen a kol. [5].

3.1 Branch and bound

Tato metoda je založena na řešení relaxovaných lineárních podproblémů původní úlohy a následném osekávání těch částí množiny přípustných řešeních, ve kterých se nemůže vyskytovat optimum. Stavový prostor je v tomto případě vhodné reprezentovat stromem rozšiřujícím se směrem dolů. Procedura prohledává stavový prostor iterací následující sekvence kroků:

1. Nalezneme řešení aktuální relaxované úlohy.
2. Podle jejího řešení rozsekneme stavový prostor do větví (branching) přidáním nerovnosti podle minima aktuální relaxované úlohy. Rozhodneme, zda je nutné obě tyto větve dál prohledávat, či jestli neposkytují lepší přípustné řešení (bounding).

Jakmile jsou všechny stavy prohledány nebo zamítnuty, vrátíme přípustné řešení s nejnižší hodnotou účelové funkce, to je hledaným optimálním řešením.

3.2 Cutting Planes

Tato metoda se zaměřuje na zpřesňování relaxovaných lineárních úloh přidáváním řezných nadrovin (cutting planes), které odstraňují nepřípustná řešení. Tento proces se provádí iterativně:

1. Nalezneme řešení aktuální relaxované úlohy.
2. Pokud řešení obsahuje neceločíselné hodnoty, přidáme nové omezující podmínky, které je odstraní.

Jakmile jsou všechny stavy prohledány, vrátíme přípustné řešení s nejnižší hodnotou účelové funkce. Na rozdíl od metody 3.1 zde nedochází k větvení stavového prostoru, ale k postupnému zpřesňování modelu přidáváním dodatečných podmínek.

3.3 Branch and Cut

Tato metoda vzniká kombinací 3.1 a 3.2 a používá tedy jak větvení, tak přidávání řezných nadrovin pro zlepšení relaxace. Tento přístup probíhá následovně:

1. Nalezneme řešení aktuální relaxované úlohy.
2. Přidáme řezné nadroviny pro zlepšení relaxace a odstranění neceločíselných řešení (cutting planes).
3. Pokud není nalezeno celočíselné řešení, rozdělí se problém do menších podproblémů (branching).
4. V každé větvi se opět aplikují řezné nadroviny k dalšímu zlepšení relaxace.

Jakmile jsou všechny stavy prohledány, vrátíme přípustné řešení s nejnižší hodnotou účelové funkce. Na tomto principu je založeno exaktní řešení modelu LRP softwarem Gurobi, jehož analýza je umístěna v poslední kapitole.

3.4 Branch and Price

Jedná se o metodu, která kombinuje branching a výběr vhodných rozhodovacích proměnných, a to tzv. oceňováním (pricing). Proces je následující:

1. Nalezneme řešení aktuální relaxované úlohy s omezenou množinou proměnných.
2. Dynamicky generujeme nové proměnné, které mohou vést k lepšímu řešení.
3. Jakmile je nalezeno lepší řešení, aplikujeme větvení podobně jako v metodě 3.1.

Jakmile jsou všechny stavy prohledány, vrátíme řešení s nejnižší hodnotou účelové funkce.

Všechny výše uvedené metody prohledávají celý stavový prostor s cílem nalézt optimální řešení. V případě LRP je však tento prostor velmi rozsáhlý a již pro instance s nižšími desítkami zákazníků a jednotkami továren je nalezení optimálního řešení výpočetně nereálné. Z tohoto důvodu se v praxi častěji využívají tzv. heuristické nebo metaheuristické metody, které umožňují řešit problémy většího rozsahu v přijatelném čase. Právě těm se věnuje následující kapitola.

4 Heuristiky

Heuristické metody, na rozdíl od těch exaktních, neprohledávají celý stavový prostor, ale pouze jeho část, ve které se spíše vyskytuje globální optimum. Tím se sice zbavují schopnosti jej vždy a přesně nalézt, ale výměnou za to umožňují efektivní řešení úloh s výrazně většími vstupními daty. Výsledkem je tedy řešení, které se sice nemusí shodovat s optimem, ale jeho hodnota účelové funkce je mu dostatečně blízko. Tyto aproximační procedury začínají nalezením dostatečně dobrého řešení, které je následně modifikováno posloupností drobných změn.

4.1 Nalezení počátečního přípustného řešení

K určení počátečního stavu lze využít širokou škálu heuristik – od jednoduchých metod, jako je náhodné propojování sousedních zákazníků do tras, až po složitější přístupy. Podkapitola 4.1.1 vychází z článku autorů Clarke a Wright [6], kteří v něm zavedli přímočarou a efektivní metodu nalezení tras pro jednu továrnu. V modelu LRP je třeba tento algoritmus provést opakovaně, pro každou otevřenou továrnu zvlášť.

4.1.1 Algoritmus úspor

Nejdříve určíme počet továren, se kterými chceme začít, a následně náhodně otevřeme odpovídající počet z nich. Každému zákazníkovi přiřadíme nejbližší otevřenou továrnu. Pokud některá z továren nemá žádného přiřazeného zákazníka, bude uzavřena. Pro každou otevřenou továrnu poté algoritmus nalezne odpovídající trasy ve třech fázích:

1. **Inicializace**

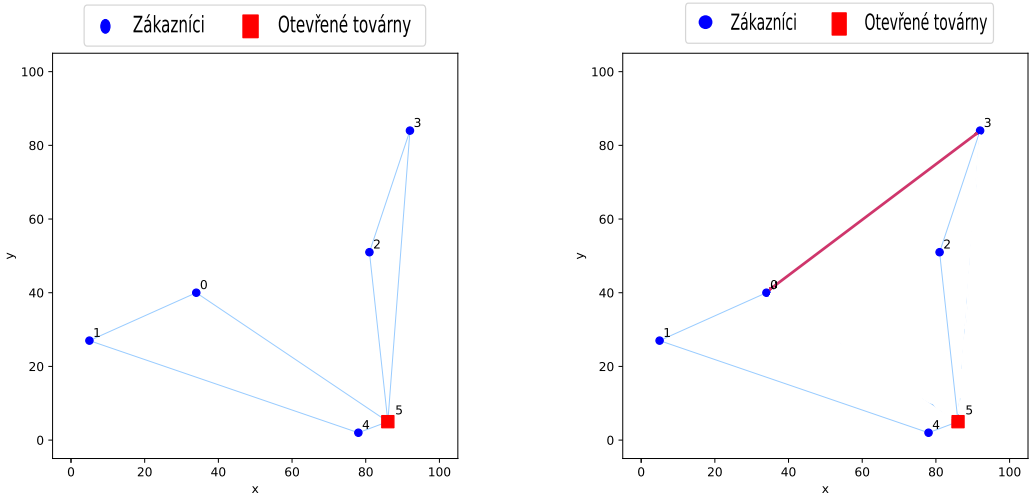
   Každému zákazníkovi přiřadíme vozidlo, které obslouží pouze jeho, tedy vyjede z továrny k zákazníkovi a poté se do ní vrátí.

2. **Spojení dvou tras**

   Pro každou dvojici tras vypočítáme možné úspory celkových nákladů, které vzniknou jejich propojením. Neuvažujeme propojení, v jejichž důsledku překračuje souhrn poptávek zákazníků na trase kapacitu přiřazeného vozidla, ani propojení podle zákazníků, kteří jsou na trase mezi dvěma dalšími zákazníky. Pro úsporu propojení zákazníka i se zákazníkem j, S_{ij} tedy platí, že $S_{ij} = c_{0i} + c_{0j} - c_{ij}$, kde c_{ij} značí vzdálenost mezi zákazníkem i a j a c_{0i} a c_{0j} jsou vzdálenosti z mateřské továrny k zákazníkovi i resp. j. Příklad propojení je uveden na obrázku 4.1. Vybereme tu úsporu s maximální kladnou hodnotou a propojení provedeme.

3. **Iterace**

   Opakujeme krok 1, dokud existuje proveditelné propojení.



Obrázek 4.1 Přípustné řešení instance s 5 zákazníky, 1 továrnou a 2 trasami a jedno z možných propojení krajních zákazníků.

4.2 Modifikace aktuálního stavu

Jakmile máme k dispozici počáteční přípustné řešení, tak je z něj možné začít stavový prostor prohledávat. K tomu je třeba zavést kroky, které aktuální stav převedou na stav budoucí. Při řešení problému LRP se nejčastěji využívají kroky typu *swap* a *add*.

4.2.1 *Swap*

Tento krok může být aplikován na dvojici továren nebo na dvojici zákazníků a dle toho buď:

- otevře momentálně uzavřenou továrnu a zároveň uzavře momentálně otevřenou továrnu nebo
- prohodí dva zákazníky na libovolných trasách.

4.2.2 *Add*

Tento krok může být aplikován na továrnu nebo na zákazníka a dle toho buď:

- otevře momentálně uzavřenou továrnu nebo
- přeřadí zákazníka na jinou libovolnou pozici kterékoliv trasy.

Uvedené kroky jsou přípustné pouze tehdy, když jejich provedení nepřekročí kapacitu příslušného vozidla, v opačném případě provedení kroku zamítneme. Konkrétní příklad heuristického algoritmu využívajícího tyto kroky je uveden v následující podkapitole.

4.3 Příklad heuristického algoritmu

Jednoduchým příkladem řešení LRP heuristickým způsobem je následující procedura, která na počáteční přípustné řešení aplikuje kroky vedoucí ke snížení hodnoty účelové funkce.

Popis algoritmu

1. Zvolíme úvodní množinu otevřených továren a nalezneme odpovídající trasy každé z nich algoritmem úspor.

2. Pro aktuálně otevřené továrny provádíme sekvenčně ten *add* zákazníka nebo *swap* zákazníků, který nejvíce snižuje hodnotu účelové funkce.

3. Jakmile všechny možné kroky vedou ke zhoršení hodnoty účelové funkce, provedeme *swap* na továrny. Pokud již byly prohledány všechny možné kombinace aktuálního počtu otevřených továren, přistoupíme k *add* na továrny. Trasy momentálně otevřených továren aktualizujeme znovu algoritmem úspor a pokračujeme předchozím bodem.

4. Jakmile jsme prozkoumali všechny možné kombinace otevření továren či dosáhli časového limitu, tak vrátíme nalezené přípustné řešení s nejnižší hodnotou účelové funkce.

Tento postup však díky averzi vůči akceptování kroků vedoucích ke zvýšení hodnoty účelové funkce může vést pouze k nalezení lokálního minima. Algoritmům, které se z něj snaží uniknout a efektivněji prohledávají stavový prostor, se věnuji v následující kapitole.

4.4 Metaheuristické algoritmy

Pro tak výpočetně náročnou úlohu, jako je LRP, je k dosažení lepších výsledků vhodné heuristiky modifikovat za účelem úniku ze stavů lokálního minima. Takto pozměněné heuristiky nazýváme metaheuristikami a na jejich pozadí vždy stojí sofistikovaná úvaha o tom, jakým způsobem vhodně prohledávat stavový prostor. K řešení LRP je často využíváno simulované žíhání, genetický algoritmus a tabu prohledávání. Následující část je věnována obecným principům, na nichž tyto metody stojí.

4.4.1 Simulované žíhání

Tato sekce vychází z článku od Yu et al. [7]. Algoritmus se snaží z lokálního minima uniknout tím způsobem, že s předem danou pravděpodobností akceptuje krok vedoucí ke zvýšení hodnoty účelové funkce. Své jméno získal po procesu z metalurgie, kdy je kov pomalu ochlazován za účelem získání pevnější struktury.

Popis algoritmu

1. Nalezneme počáteční přípustné řešení.

2. Náhodně vybereme krok, jehož provedení neporuší zadaná omezení. Pokud je v budoucím stavu hodnota účelové funkce nižší než ve stavu původním, krok provedeme. V opačném případě provedeme krok s pravděpodobností určenou Boltzmannovým rozdělením, $\exp(-\Delta/kT)$, kde Δ je rozdíl v hodnotě účelových funkcí stavů, k je Boltzmannova konstanta a T momentální teplota, námi volený parametr. Krok opakujeme.

3. Po počtu iterací daných parametrem I_{iter} je teplota snížena: $T = \alpha\, T$, kde $\alpha \in (0{,}1)$ je námi voleno.

4. Končíme v případě provedení daného počtu iterací bez snížení účelové funkce nebo při poklesu T pod úroveň zvoleného parametru T_{min}. Vrátíme přípustné řešení s nejnižší hodnotou účelové funkce.

Z počátku je teplota vysoká, tedy pravděpodobnost ($\exp -\Delta/kT$) také vysoká, a to odpovídá většímu prohledávání stavů s vyšší hodnotou účelové funkce s nadějí na budoucí zlepšení. Ke konci se tato pravděpodobnost snižuje, což nutí algoritmus rychleji tlačit dolů hodnotu účelové funkce.

4.4.2 Genetický algoritmus

Tato metoda spadá do širší skupiny evolučních algoritmů, které imitují proces přirozeného výběru v populaci. Jejich princip spočívá ve výběru rodičů - řešení – jejichž křížením vzniká potomek, řešení nové. Následně je tento potomek podroben mutaci, která do výsledku vnáší prvek náhodnosti a zvyšuje diverzitu populace. Cílem je nalezení nejsilnějšího jedince, tedy přípustného řešení s nejnižší hodnotou účelové funkce. Tato sekce vychází ze článku Rybičkové, Burketové a Mockové [8].

Popis algoritmu

1. Vygenerujeme počáteční populaci stavů.

2. Z počáteční populace náhodně vybereme rodiče, přičemž s větší pravděpodobností volíme stavy s nižší hodnotou účelové funkce.

3. Na vybrané stavy aplikujeme námi zvolené křížení, čímž získáme nové řešení - potomka. Může se například jednat o vzájemnou výměnu části řešení.

4. Potomka následně podrobíme zvolené mutaci, např. záměnu jeho části. Tímto způsobem vygenerujeme celou následující generaci potomků.

5. Sekvenčně aplikujeme body $2-4$ a proceduru ukončíme po provedení námi zvoleného počtu iterací, které nevedou ke snížení účelové funkce. Vrátíme přípustné řešení s nejnižší hodnotou účelové funkce.

Na populaci jsou tedy celkově aplikovány tři operátory - selekce, která vybírá potencionálně dobrá řešení, křížení, které se z nich snaží vytvořit ještě lepší řešení a mutace, která se snaží vyhnout lokálnímu optimu.

4.4.3 Tabu prohledávání

Základní ideou této metody je akceptování i těch kroků, které vedou ke zvýšení účelové funkce, čímž se snažíme vyhnout případům, ve kterých by algoritmus označil lokální optimum za globální. Za tímto účelem jsou již navštívené stavy přidány do datové struktury nazvané tabu seznam a po dočasnou dobu je jejich opětovné navštívení zakázáno. Tato podkapitola a následující kapitola vychází z článku, jehož autory jsou Tuzun a Burke [3].

Popis algoritmu

1. Nalezneme počáteční přípustné řešení a vytvoříme tabu seznam.
2. Ze všech možných kroků vybereme ten, který není na seznamu tabu a zároveň nejvíce snižuje účelovou funkci, provedeme ho. Pokud žádný takový není, akceptujeme i ten krok, který účelovou funkci zvyšuje. V tom případě je ale i s opačným krokem přidán do tabu seznamu, čímž je jejich provedení na několik iterací zakázáno. Pokud je tabu seznam zaplněn, odebereme z něj nejstarší prvek.
3. Končíme v případě provedení daného počtu iterací bez snížení hodnoty účelové funkce. Vrátíme přípustné řešení s nejnižší hodnotou účelové funkce.

Tím, že zakazujeme algoritmu ihned se vracet do míst s nižší hodnotou účelové funkce, ho nutíme více prohledávat stavový prostor s nadějí na nalezení lepšího řešení. Tato idea je v následující kapitole podrobně rozebrána a aplikována na model LRP.

4.5 Dvoufázové tabu prohledávání

Algoritmus se skládá ze dvou částí: location fáze, v níž je určeno, které továrny budou otevřeny, a routing fáze, ve které dochází k přiřazení zákazníků do rozvozových tras a jejich následné modifikaci. Myšlenka tabu prohledávání je v rámci tohoto přístupu aplikována opakovaně, pro každý bod (a) a (b) v popisu níže zvlášť. Výsledkem jsou čtyři oddělené tabu seznamy. V popisu algoritmu je často odkazováno na kroky uvedené v sekci 4.2. Jedná se o obsáhlý algoritmus, jehož struktura je na obrázku 4.2.

Popis algoritmu

1. **Inicializace**

   Zvolíme počáteční počet otevřených továren a náhodně vybereme ty, které otevřeme. Pro každou otevřenou továrnu nalezneme trasy algoritmem úspor, po jehož provedení aplikujeme sérii dodatečných *swap*ů na zákazníky.

2. **Location fáze**

   Cílem této fáze je najít kombinaci otevřených továren vedoucí k nejnižší hodnotě účelové funkce.

(a) **Přidání továrny**

Provedeme *add* továrny, která má nejvyšší odhadovaný přínos v podobě snížení celkových nákladů. Zmíněný odhad získáme následovně:

- Každého zákazníka přiřadíme k nejbližší otevřené továrně, přičemž mu přiřadíme vozidlo, které ho obslouží a vrátí se zpět. Náklady na vzniklé přímé trasy sečteme a výsledek označíme jako $costsOld$.
- Stejným způsobem spočteme náklady pro novou kombinaci otevřených továren, která by vznikla provedením daného kroku. Tyto náklady označíme jako $costsNew$.
- Odhad snížení nákladů určíme jako rozdíl $costsOld$ a $costsNew$ (ten je vždy kladný, protože přidáním továrny se náklady přímých tras zhoršit nemůžou) a od něj odečtená fixní cena otevírané továrny.

Nově otevřenou továrnu přidáme na příslušný tabu seznam a pokračujeme bodem (b) této fáze.

(b) **Prohození továren**

Provedeme *swap* továren s nejvyšším odhadem snížení celkových nákladů. Zmíněný odhad získáme stejně jako v minulém bodě, pouze s tím rozdílem, že k němu dodatečně přičteme fixní cenu továrny, která je v důsledku provedení kroku uzavřena.

Dvojici továren společně s dvojicí opačnou přidáme na příslušný seznam tabu a aktualizujeme trasy otevřených továren algoritmem úspor. Metoda pokračuje bodem (a) v routing fázi.

3. **Routing fáze**

Cílem je nalézt rozvozové trasy pro aktuálně otevřené továrny s minimálními náklady.

(a) **Přeřazení zákazníka**

Provedeme *add* zákazníka s nejvyšším zlepšením celkových nákladů, které je spočteno jako rozdíl celkových nákladů před a po aktuálním kroku. Uvažujeme jen kroky, které nejsou tabu a které přiřazují zákazníka k jedné z jeho $fMax$ nejbližších továren. Tím je snížena výpočetní náročnost.

Pokud je v průběhu evaluace nalezen takový krok, který vede ke snížení nejmenší nalezené hodnoty účelové funkce a není tabu, tak je ihned proveden.

Zákazníka, kterého přeřadíme, přidáme na příslušný tabu seznam.

(b) **Prohození zákazníků**

Provedeme *swap* zákazníků s nejvyšším zlepšením celkových nákladů, které je spočteno stejně jako v bodě (a) této fáze. Uvažujeme pouze kroky prohazující zákazníky, kteří jsou navzájem mezi $cMax$ nejbližšími.

Pokud je v průběhu evaluace nalezen takový krok, který vede ke snížení nejmenší nalezené hodnoty účelové funkce a není tabu, tak je ihned proveden.

Dvojici prohazovaných zákazníků přidáme včetně dvojice opačné na příslušný tabu seznam.

4. **Ukončení algoritmu a výstup**

   U částí (a) a (b) v location a routing fázi si ukládáme počet kroků, které v dané části vedly ke zhoršení nejmenší aktuální hodnoty účelové funkce. Pokud by tento počet překonal námi volenou hranici, tak tuto část ukončíme a pokračujeme k další. Také je možné omezit se pouze na předem stanovený počet iterací, případně pro menší instance nechat metodu prohledat všechny možné kombinace otevřených továren.

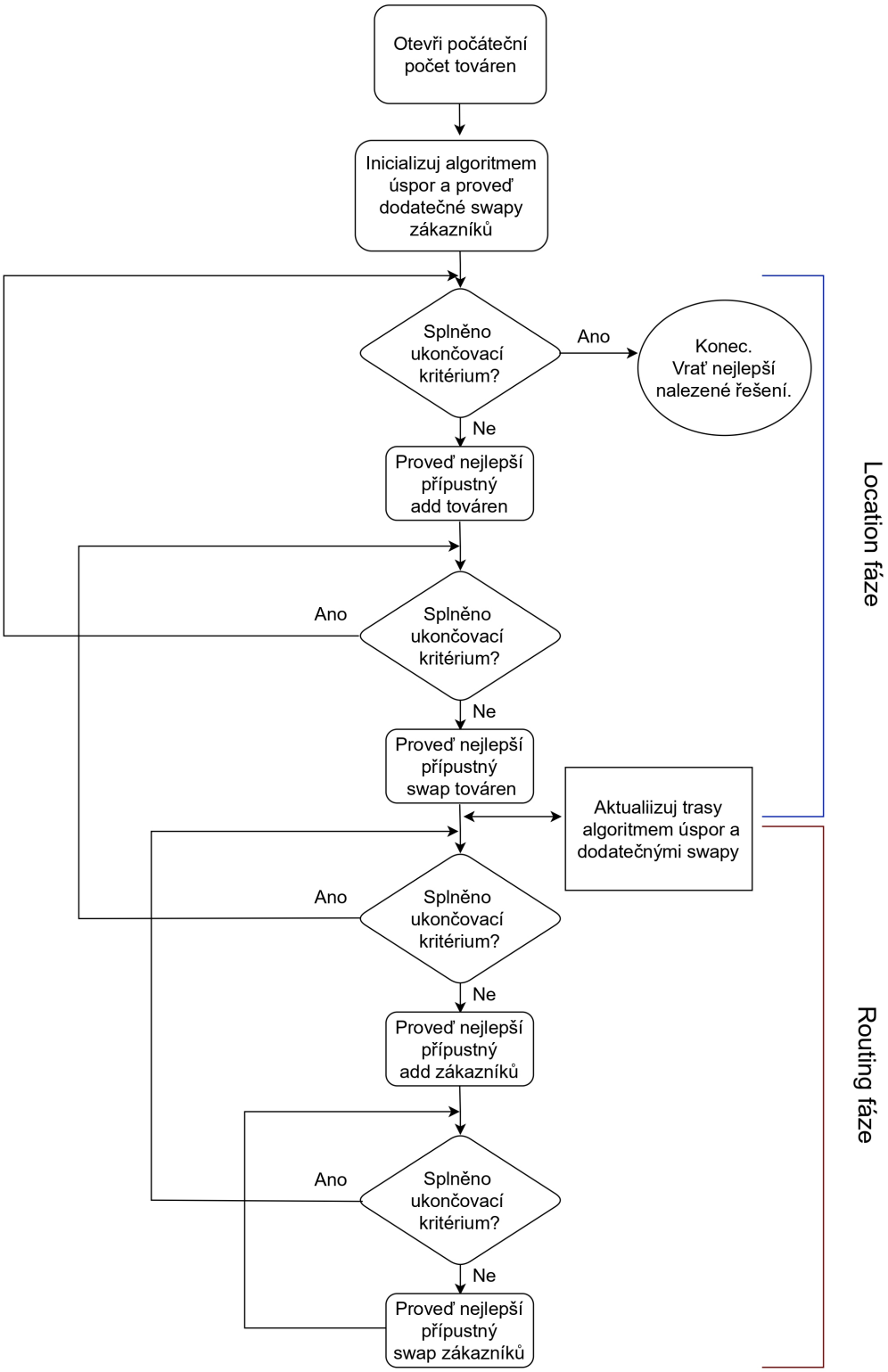
   Po celou dobu evidujeme nejlepší nalezené rozdělení továren a jejich trasy a tento objekt na konci vrátíme.

Parametry algoritmu

V postupu se vyskytuje velké množství parametrů, které jsou voleny dle úvahy uživatele.

- Velikost tabu seznamů
- Počet kroků, po který tolerujeme zvýšení hodnoty účelové funkce v každé části.
- Počet nejbližších továren $fMax$.
- Počet nejbližších zákazníků $cMax$.

Poslední kapitola je věnována numerické studii, která vychází z naší implementace dvoufázového tabu prohledávání v programovacím jazyce Python. Zdrojový kód je k nalezení v příloze bakalářské práce.



Obrázek 4.2 Schéma algoritmu 4.5. Každá část location a routing fáze si uchovává počet kroků, které vedly ke zvýšení hodnoty účelové funkce. Příslušné ukončovací kritérium je splněno, pokud tento počet překoná námi zvolený parametr odpovídající části.

5 Dosažené výsledky

5.1 Analýza parametrů

V následující části je popsán jeden z možných přístupů k volbě hodnot jednotlivých parametrů k dosažení co nejlepších výsledků.

Počet nejbližších továren a zákazníků

Nalezení vhodných hodnot $fMax$ a $cMax$ je důležité proto, abychom se vyhnuli evaluaci zbytečných kroků a zároveň se nezbavili řešení s nižší hodnotou účelové funkce. Dobrým indikátorem je počet kroků, kdy dochází k přeřazení zákazníka k první nejbližší továrně, druhé nejbližší továrně... Jakmile se tento počet začne blížit nule, je vhodné zamítnou evaluaci kroků, které by přeřazovaly dál. U prohození zákazníků je postup analogický.

Počet akceptovatelných zvýšení hodnoty účelové funkce

Každý bod algoritmu, tedy přidání a prohození továrny a přeřazení a prohození zákazníků, má svůj parametr a jeho vhodné nastavení dokáže snížit čas nutný k výpočtu. Možným přístupem k nalezení dobrých hodnot těchto parametrů je experimentálně zvyšovat hodnoty těchto parametrů až do chvíle, kdy opakovaně dochází ke zvýšení výpočetního času bez snížení hodnoty účelové funkce.

Velikost tabu seznamů

Kvůli zmíněnému předpokladu je vhodné odlišit velikost seznamů v location a routing fázi. Autoři článku o dvoufázovém tabu prohledávání volí velikosti příslušných seznamů zaokrouhlením náhodně vygenerovaného čísla z rovnoměrného rozdělení na celé číslo. Krajní body rozdělení jsou voleny mezi 0.25× počet továren (počet zákazníků) a 0.75× počet továren (počet zákazníků) pro location tabu seznam (routing tabu seznam), pro tyto hodnoty bylo opakovaně dosahováno nejlepších výsledků. Pokud je velikost seznamů příliš malá, tak nedochází k úniku z některých lokálních minim. Pokud je naopak velikost seznamů příliš velká, pak je stavový prostor prohledán příliš povrchně a algoritmus se nedokáže dostat k menším hodnotám účelové funkce.

5.2 Numerická studie

K účelu numerické studie jsou souřadnice zákazníků a továren náhodně generovány z dvojrozměrného rovnoměrného rozdělení na množině $(0,100)^2$. Dále předpokládáme následující.

- Fixní cena otevření továrny je stejná pro všechny továrny a je rovna 50.
- Každá továrna má k dispozici neomezenou flotilu vozidel s kapacitou 25 jednotek zboží a fixní cenou použití 10.
- Poptávka každého zákazníka je celočíselná a je volena náhodně mezi 5 a 10 včetně.
- Algoritmus inicializujeme s jednou otevřenou továrnou.

V dalších podkapitolách jsou diskutovány dosažené výsledky z dvou pohledů, a to velikosti vstupní instance a správnosti nalezeného řešení pro ověřitelné řešení.

5.2.1 Velikost vstupní instance

Tato část se věnuje chování implementované metody pro zvětšující se velikost vstupu. V tabulce 5.1 jsou shrnuty dosažené výsledky. Každý pokus byl omezen 10 minutami výpočetního času na stroji s procesorem Intel Core i7 − 10510U CPU @ 1.80GHz a 16GB RAM.

Č. pokusu	**# zákazníků**	**# továren**	**Optimum**	**2 - f. tabu proh.**
1	5	5	354	354
2	10	5	521	521
3	15	5	593	593
4	20	5	-	823
5	30	5	-	1101
6	10	10	492	492
7	15	10	-	605
8	20	10	-	716
9	30	10	-	965
10	70	10	-	1801

Optimum = exaktní řešení nalezené softwarem Gurobi, 2 - f. tabu proh. = aproximace nalezená algoritmem 4.5.

Tabulka 5.1 Výsledky algoritmu pro různé instance

Pro menší vstupy o 20 vrcholech bylo zjištěno, že exaktní řešení se shoduje s řešením, které označila naše implementace dvoufázového tabu prohledávání jako optimální řešení. Příklad nalezeného optimálního řešení je uveden na obrázku 5.1.

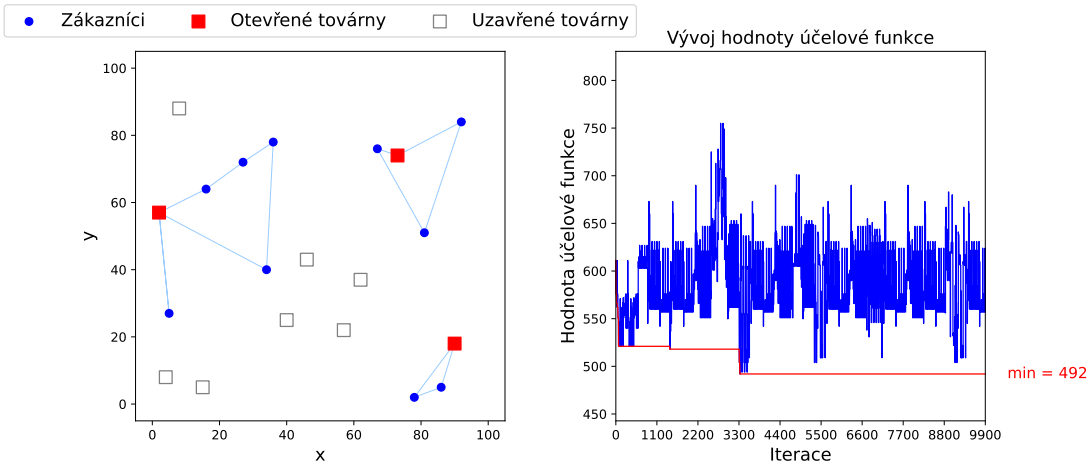
Optimalizační software Gurobi nicméně již pro instance od 25 vrcholů dál nedokázal v 10 minutách najít optimum a přesnost námi nalezených řešení tedy není možné ověřit. Obrázek 5.2 však ukazuje, že kvalita tohoto řešení pro 30 zákazníků a 10 továren je velmi dobrá. Zároveň můžeme zhruba u 7500 iterace pozorovat zhoršení účelové funkce, po kterém dochází k nalezení řešení s nejnižší

hodnotou účelové funkce. Algoritmu se úspěšně podařilo překonat lokální minimum, přiblížený výřez je k dispozici na obrázku 5.3.

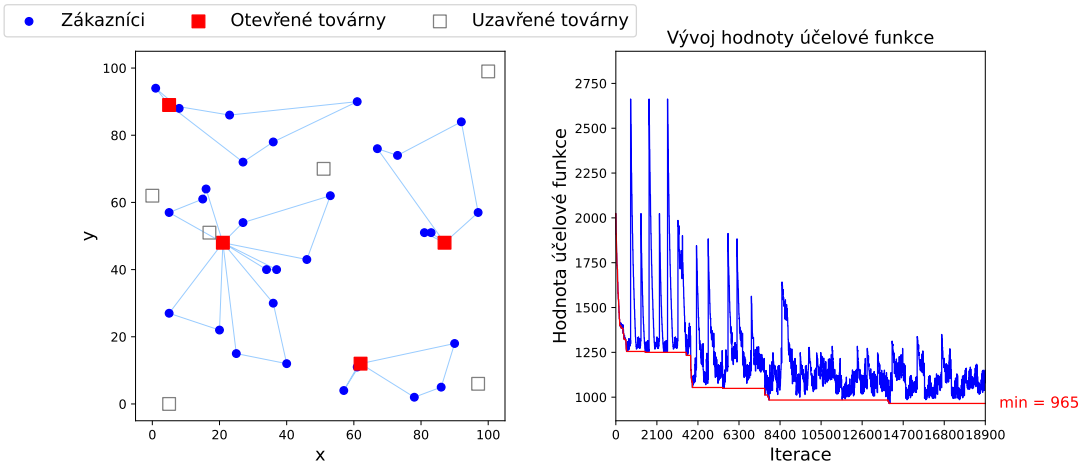
Největší vstupní instance, jejíž přípustné řešení bylo v tomto časovém omezení nalezeno, byla tvořena 70 zákazníky 10 továrnami, jak je ukázáno na obrázku 5.4. Zde ale již poměrně často dochází k překryvu sousedních tras a pravděpodobně se o přesné optimální řešení nejedná, jeho kvalita nicméně vypadá uspokojivě.

Na všech zmíněných obrázcích dochází v průběhu evaluace ke značným pozitivním skokům. Zde dochází k inicializaci algoritmem úspor a až následné swapy a postupné iterace hodnotu účelové funkce více snižují.

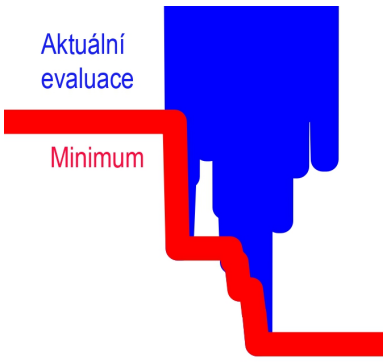
V poslední řadě je nutno okomentovat počáteční zacyklení implementované metody, které je k vidění na obrázcích 5.2 a 5.4. Opakované navštívení stavů je způsobeno tím, že velikost některého tabu seznamu je nedostatečná a po několika krocích je metodě dovoleno vrátit se do lokálního minima. Nicméně ladění tohoto parametru je obtížné a úplná prevence tohoto jevu výpočetně nákladná. Možným řešením je občas provést krok, který náhodně aktuální stav změní.



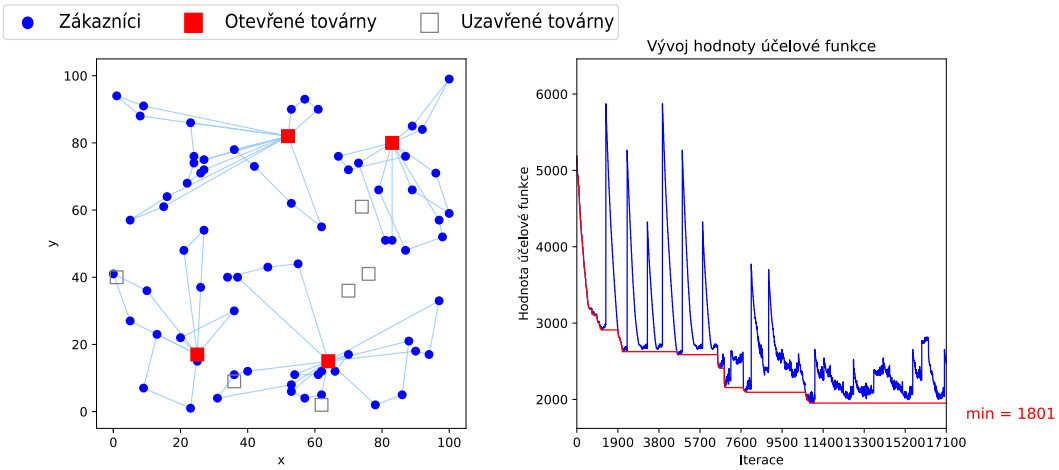
Obrázek 5.1 Přípustné řešení LRP s 10 zákazníky a 10 továrnami a graf hodnoty účelové funkce. Dvoufázové tabu prohledávání nalezlo optimální řešení.



Obrázek 5.2 Přípustné řešení LRP s 30 zákazníky a 10 továrnami a graf hodnoty účelové funkce.



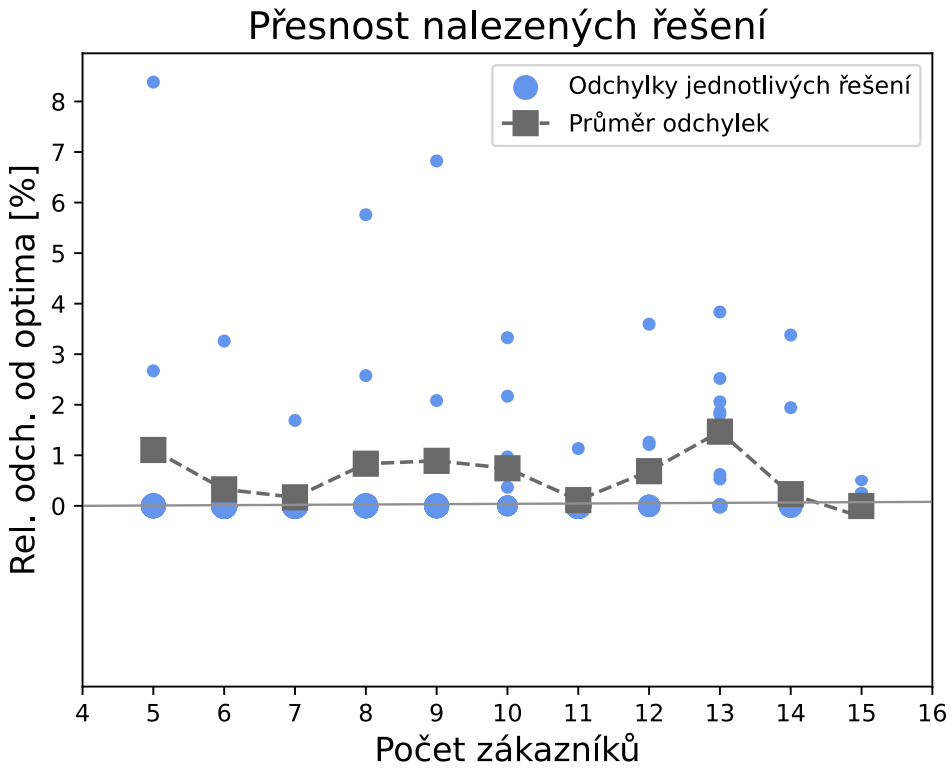
Obrázek 5.3 Úspěšné překonání lokálního minima algoritmem dvoufázového tabu prohledávání.



Obrázek 5.4 Přípustné řešení LRP s 70 zákazníky a 10 továrnami a graf hodnoty účelové funkce.

5.2.2 Relativní odchylka

Tato podkapitola se zabývá přesností metody dvoufázového tabu prohledávání pro vstupní instance, jejichž exaktní řešení bylo nalezeno softwarem Gurobi do 10 minut. Na obrázku 5.5 je k vidění závislost přesnosti aproximace na počtu zákazníků pro fixní počet 5 továren. Pro menší instance, pro které máme k dispozici exaktní řešení, se průměrná odchylka od skutečné optimální hodnoty účelové funkce pohybuje kolem 1%. V praxi se optimalizační algoritmy často pouštějí opakovaně s jinými parametry či z jiného počátečního řešení, skutečná přesnost procedury je tedy vyšší.



Obrázek 5.5 Závislost relativní odchylky aproximačního řešení od exaktního na počtu zákazníků pro fixní počet 5 továren. Pro každý počet zákazníků bylo provedeno 10 měření a následně spočítán aritmetický průměr. Velikost bodů s 0% odchylkou je určena počtem pozorování.

5.3 Ilustrace na reálných datech

Za účelem ukázky možné aplikace modelu LRP byl aplikován algoritmus dvoufázového tabu prohledávání na stanice metra v Praze. Skutečné vzdálenosti potřebné k jízdě vozidlem mezi jednotlivými stanicemi byly získány pomocí API Openrouteservice.

Mějme firmu, která v každé stanici metra vlastní prodejní automaty, ve kterých jsou prodávány plechovky slazeného nápoje stejného druhu. Vstupní parametry jsou dále následující:

- Na výběr je 10 malých skladů k pronájmu, každý za 10000,-/měsíc.
- Ke každému skladu je k dispozici jedno vozidlo s kapacitou 3000 plechovek a spotřebou 10l/100km.
- Brigádník si za jeden rozvoz bere 1000,-.
- Náklady na pohonné hmoty činí 40,-/l.

Snaha firmy o minimalizaci nákladů může být reprezentována modelem LRP s těmito parametry:

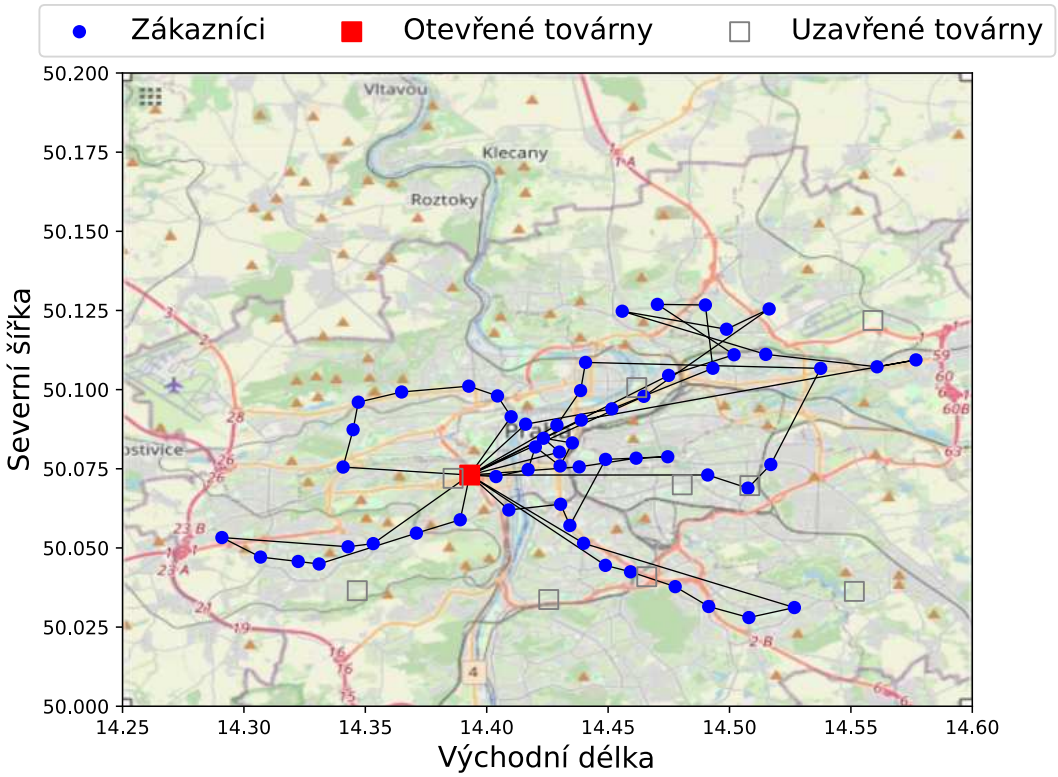
- $F_i = 10000,\ i \in \{1, \ldots, 10\}$
- $Q_l = 3000,\ l \in \{1, \ldots, 10\}$
- $C_l = 1000,\ l \in \{1, \ldots, 10\}$
- $c_{u,v} = distuvKm \times 40 \times 10/100$
- $q_j = randInt(26, 53) \times 10,\ j \in \{1, \ldots, 58\}$

Za předpokladu, že automaty jsou doplňovány vždy jednou za měsíc, dojdeme k výsledku, který je uveden na obrázku 5.6. Aproximace nejlepšího řešení byla nalezena implementací algoritmu 4.5 a měsíční náklady na provoz činí 18709,-.

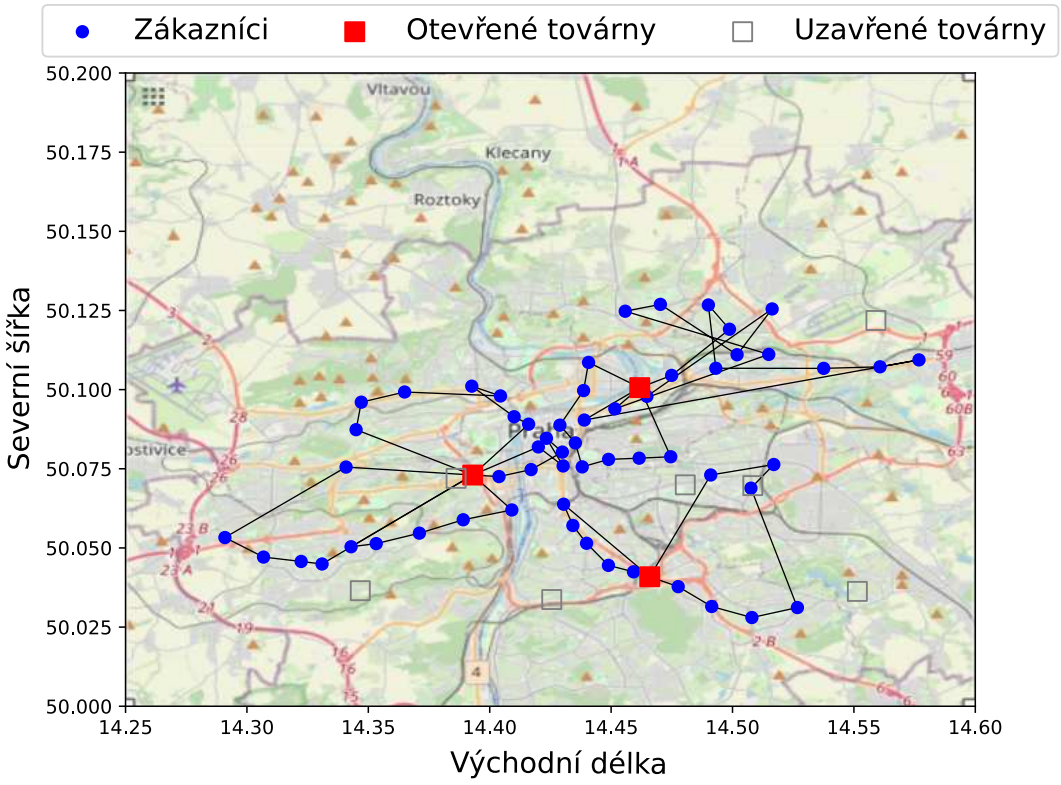
Pokud by firma věděla, že z kapacitních důvodů potřebuje pronajmout alespoň tři sklady, je možné přidat do modelu omezení

$$\sum_{i \in I} z_i \geq 3.$$

Nalezené řešení má hodnotu účelové funkce 39593,- a je k vidění na obrázku 5.7.



Obrázek 5.6 Aplikace implementované metody na rozvoz homogenního zboží po stanicích metra v Praze. Jako vzdálenost mezi dvěma body jsou brány průměrné náklady na pohonné hmoty.



Obrázek 5.7 Aplikace implementované metody na rozvoz homogenního zboží po stanicích metra v Praze s dodaným omezením tří a více skladů. Jako vzdálenost mezi dvěma body jsou brány průměrné náklady na pohonné hmoty.

Závěr

Tato práce se věnovala modelu Location Routing Problem a hledání optimální hodnoty příslušné účelové funkce. Pomocí simulací bylo ukázáno, že pro větší vstupní instance není možné nalézt optimální řešení v reálném čase. Z tohoto důvodu byla popsána a implementována aproximační metaheuristická metoda dvoufázového tabu prohledávání, která umožnila zvýšit zvládnutelnou velikost instance z několika desítek vrcholů až přibližně ke stu.

V závěrečné části práce bylo ilustrováno možné použití algoritmu na datech odpovídajících reálné geografii a poptávce.

Zvolený model LRP je deterministický, předpokládá, že zákazníci jsou jednoznačně daní a jejich poptávku lze uspokojit kdykoli. Vzhledem k tomu, že v praktických aplikacích se téměř vždy vyskytují náhodné vlivy (např. zpoždění, výpadky, proměnlivá poptávka), nabízí se jako přirozené rozšíření stochastická formulace úlohy. Ta by lépe odpovídala reálným podmínkám a mohla by vést k robustnějším řešením.

Literatura

1. Said Salhi, Graham K. Rand. The effect of ignoring routes when locating depots. *European Journal of Operational Research*. 1989, roč. 39, č. 1, s. 150–156. ISSN 0377-2217.
2. Dupačová, Lachout. *Úvod do optimalizace*. MatfyzPress, 2011. ISBN 978-80-7378-176-7.
3. Dilek Tuzun, Laura I. Burke. Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *European Journal of Operational Research*. 1999, roč. 116, č. 1, s. 87–99. ISSN 0377-2217. Dostupné z DOI: 10.1016/S0377-2217(98)00107-6.
4. Bovet, Daniel Pierre; Crescenzi, Pierluigi. *Introduction to the theory of complexity*. GBR: Prentice Hall International (UK) Ltd., 1994. ISBN 0139153802.
5. Der-San Chen Robert G.Batson, Yu Dang. *Applied Integer Programming*. John Wiley Sons, Ltd, 2009. ISBN 9780470373064. Dostupné z DOI: 10.1002/9781118166000.
6. Clarke, G.; Wright, J. W. A compact model and tight bounds for a combined location-routing problem. *Operations Research*. 1964, roč. 12, č. 4, s. 568–581. ISSN 0030364X, ISSN 15265463. Dostupné také z: http://www.jstor.org/stable/167703.
7. Yu, Vincent F.; Lin, Shih-Wei; Lee, Wenyih; Ting, Ching-Jung. A simulated annealing heuristic for the capacitated location routing problem. *Computers Industrial Engineering*. 2010, roč. 58, č. 2, s. 288–299. ISSN 0360-8352. Dostupné z DOI: https://doi.org/10.1016/j.cie.2009.10.007. Scheduling in Healthcare and Industrial Systems.
8. Rybičková, Alena; Burketová, Adéla; Mocková, Denisa. Solution to the location-routing problem using a genetic algorithm. 2016, s. 1–6. Dostupné z DOI: 10.1109/SCSP.2016.7501016.

Seznam obrázků