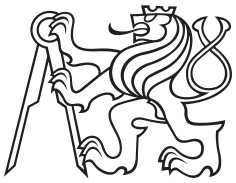
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Plánování zásobovacích tras

Makhambet Ismukhambetov

Vedoucí: Ing. Pavel Šedek
Obor: Softwarové inženýrství a technologie
Květen 2021

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Ismukhambetov** Jméno: **Makhambet** Osobní číslo: **474483**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Softwarové inženýrství a technologie**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Plánování zásobovacích tras

Název bakalářské práce anglicky:

Supply route planning

Pokyny pro vypracování:

Cílem práce je vytvořit software, který najde vhodné pořadí, ve kterém má být provedeno zásobování vybraných lokalit v existujícím podniku. Algoritmus nemusí nutně najít optimální trasu, nalezená trasa by se však od té optimální neměla výrazně lišit. Pro naplnění tohoto cíle by se mělo postupovat dle následujících bodů:
1. Analyzovat požadavky na plánování.
2. Provést rešerši existujících postupů k řešení podobných problémů.
3. Vybrat vhodný algoritmus, který by bylo možné následně implementovat.
4. Implementovat software.
5. Otestovat vytvořený software.

Seznam doporučené literatury:

1. G. B. Dantzig, J. H. Ramser, The Truck Dispatching Problem (1959)
2. Vidal, T. Crainic, M. Gendreau, Ch. Prins, A unified solution Framework for multi-attribute vehicle routing problems (2013)
3. J. Kolář, Teoretická informatika (2009)
4. El-Sherbeny, Nasser A. Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods. Journal of King Saud University – Science. Elsevier, 2010

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Pavel Šedek, katedra ekonomiky, manažerství a humanitních věd FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **21.02.2021** Termín odevzdání bakalářské práce: **21.05.2021**

Platnost zadání bakalářské práce: **19.02.2023**

Ing. Pavel Šedek podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. Datum převzetí zadání

Podpis studenta

Poděkování

Chtěl bych poděkovat vedoucímu své práce Ing. Pavlu Šedekovi za cenné rady, čas a zkušenosti, které jsem získal v rámci práce na tomto projektu. Také bych chtěl poděkovat své rodině, přítelkyni a kamarádům za jejich neustálou podporu při studiu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze, 21 května 2021

Abstrakt

Tato bakalářská práce se zabývá návrhem a implementací algoritmu pro optimalizace plánování každodenních tras existujícího podniku.

V teoretické části práce byly provedeny analýzy požadavků a podobných popsaných problémů. Na základě těchto analýz byly sestaveny: vlastní problém, matematický model problému a návrh algoritmu, který odpovídá všem požadavkům. Praktická část práce představuje algoritmus, který má najít optimální řešení problému v rozumném čase.

Klíčová slova: Optimalizace plánování tras, Problém obchodního cestujícího, Problém okružních jízd, Problém okružních jízd s časovými okny, heuristické metody, algoritmus, Metoda nejbližších sousedů

Vedoucí: Ing. Pavel Šedek

Abstract

This bachelor thesis deals with the design and implementation of an algorithm for optimizing the planning of daily routes in an existing company.

Requirements analysis and research of similar described problems were conducted in the theoretical part of the thesis. Based on these analyzes, the own problem, the mathematical model of the problem, and the design of an algorithm were formulated. The practical part of the thesis presents an algorithm that meets the requirements to find a good solution to the problem in a reasonable time.

Keywords: Optimization of routes planning, Travelling salesman problem, Vehicle routing problem, Vehicle routing problem with time windows, heuristic methods, algorithm, Nearest neighbour method

Title translation: Supply route planning

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

Tato bakalářská práce se zaměřuje na návrh a implementaci algoritmu, který bude plánovat trasy pro společnost Blahobyty.

1.1 Motivace

V současné době každá společnost hledá prostředky k vyššímu zhodnocení kapitálu, a proto neustále vzrůstá zájem o systémech, které optimalizují a automatizují pracovní procesy. Nedílnou součástí celého pracovního systému společnosti Blahobyty je doprava a její plánování, a proto společnost hledá řešení, které pomůže usnadnit každodenní plánování tras pro zaměstnance.

1.2 Cíl práce

Cílem práce jsou zanalyzovat stávající provoz zadavatele, specifikovat požadavky, prozkoumat popsané problémy, sestavit vlastní problém a jeho matematický model. Dalším cílem je prozkoumat existující řešení podobných problémů a na tomto základě navrhnout vlastní řešení, které splní všechny požadavky zadavatele. Posledním a hlávním cílem práce je naimplementovat a otestovat navržené řešení.

1.3 Struktura práce

Práce je rozdělena do tří části: analytická, teoretická a praktická. Analytická část práce popisuje aktuální problém a požadavky zadavatele. Teoretická část práce popisuje podobné a existující problémy, řešení těchto problémů, formulací vlastního problému a jeho matematického modelu, který odpovídá všem požadavkům z analytické části. Praktická část práce obsahuje návrh algoritmu, popis jeho implementace a výsledky testování.

Kapitola 2
Analýza aktuálního stavu

V této kapitole je stručně popsána společnost Blahobyty, která je zadavatelem této práce. Následně tato kapitola obsahuje popis aktuálního problému a požadavky zadavatele na algoritmus řešící tento problém.

2.1 Popis zadavatele

Společnost Blahobyty se sídlem v Praze byla založena v roce 2016 a svým klientům spravuje byty, které jsou pronajímaný na kratkodobé pronájmy. Společnost používá a neustále rozvíjí svůj vlastní informační systém, který umožňuje zjefektivnit velké množství pracovních procesů. Momentálně společnost Blahobyty spravuje 69 bytů. Ve flotile vozidel společnost má 3 auta a 4 skútry.

2.2 Seznámení s problémem

Každodenně zaměstnanci společnosti Blahobyty plánují trasy pro řidiče, které mají na sobě úkoly v různých částech města. Jelikož se trasy plánují ručně, jejich plánování je časově náročné a kvalita je závislá na vlastní zkušenosti a intuici zaměstnance, který tyto trasy plánuje. Kvůli tomu, že trh krátkodobých nájmu neustále roste, roste i počet nájmů, které spravuje společnost Blahobyty. Spolu s tím stoupá i počet úkolů, které zaměstnanci společnosti mají řešit včas, aby zákazníci byli vždy spokojení s poskytnutými službami. Tudíž společnost Blahobyty potřebuje optimalizovat a automatizovat tuto část své práci, aby mohli zvýšit její efektivitu a snížit náklady na dopravu a její plánování.

2.3 Analýza požadavků

Po jednání se zástupci společnosti Blahobyty byly sebrány a definovány požadavky na algoritmus. Podle požadavků algoritmus má plánovat trasy a při plánování dodržovat následující podmínky:

1. Denní plán se sestává z jedné nebo mnoha tras.

2. Pro každý den je k dispozici variabilní počet řidičů, kde každý řidič má svůj mzdový náklad.
3. Každý den je potřeba naplánovat přibližně 100-150 úkolů.
4. Každá trasa se sestává z úkolů, kde každý úkol má přiřazenou lokaci.
5. Každá trasa začíná a končí v kanceláři.
6. Ke každé trase může být přiřazeno pouze jedno vozidlo.
7. Trasa může být vyplněna autem, skútrem nebo městskou dopravou a pěšky. Každý typ dopravy má svůj provozní náklad a matici časových vzdálenosti mezi každou dvojicí lokací.
8. Ke každé trase může být přiřazen pouze jeden řidič.
9. Každý řidič má stanovenou pracovní dobu, ve které by měla být naplánována jeho trasa.
10. Každý úkol má nastavenou lokaci, ve které se vyplňuje.
11. Každý úkol má definovanou množinu vozidel, kterým může být obsloužen.
12. Každý úkol má definovanou množinu řidičů, kteří mohou úkol vykonat.
13. Každý úkol má definovanou časovou alokaci v minutách.
14. Každý úkol má definovanou prioritu.
15. Každý úkol může mít časové okno, ve kterém je potřeba začít a dokončit jeho vyplnění.

Kapitola 3

Teoretická východiska

Za posledních několik desítek let problematika logistiky, distribuce a plánování tras stala samotnou oblastí výzkumu, kterou rozvíjí velká řada vědců, díky čemuž bylo publikováno velké množství článků popisujících řešení různých problémů v této oblasti. Problém plánování tras společnosti Blahobyty je podobný již známým a popsaným problémům jako "Problém obchodního cestujícího" (TSP) a "Problém okružních jízd" (VRP). V následujících podkapitolách budou popsány výše uvedené problémy.

3.1 Problém obchodního cestujícího

Problém obchodního cestujícího nebo Travelling Salesman Problem (TSP) je kombinatorický optimalizační problém, jeden z nejznámějších a jednoduchých úloh v oblasti dopravy [2].

Cílem problému je najít nejkratší hamiltonovskou kružnice v úplném, ohodnoceném a neorientovaném grafu $G = (V, A)$, kde V je množina vrcholů a A je množina ohodnocených hran [2].

Nejkratší hamiltonovská kružnice je kružnice, která prochází všechny vrcholy grafu pouze jednou [3].

Úplný, ohodnocený a neorientovaný graf je graf, ve kterém každé dva vrcholy spojené mezi sebou jednou neorientovanou a ohodnocenou hranou [3].

Obecně TSP spočívá v tom, že je dáno n měst a jedno depo, kde mezi každou dvojicí uzlů existuje právě jedna cesta. Úkolem je najít nejkratší možnou trasu, která začíná a končí v depu, přičemž prochází každý uzel pouze jednou (Obrázek 3.1).

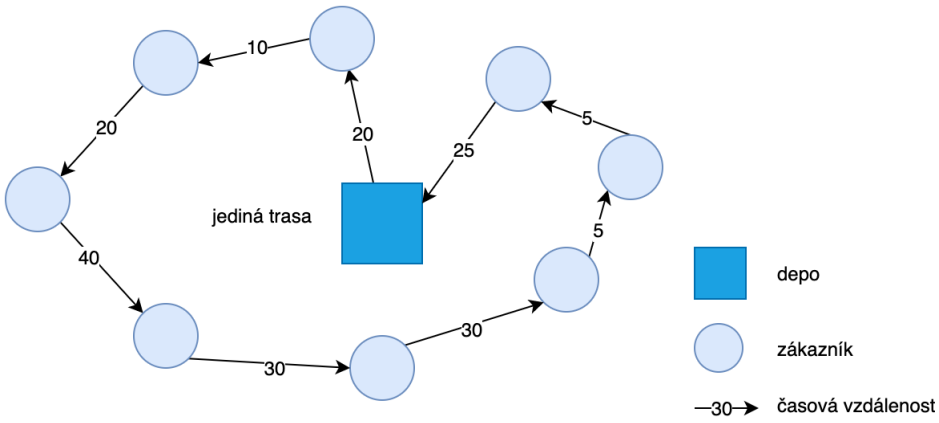
Kvůli jednoduchosti TSP není často používán pro řešení problémů, které se vyskytují v reálných situacích, protože kromě nalezení minimální cesty v rámci jedné trasy nezahrnuje žádné další požadavky a podmínky.

3.2 Problém okružních jízd

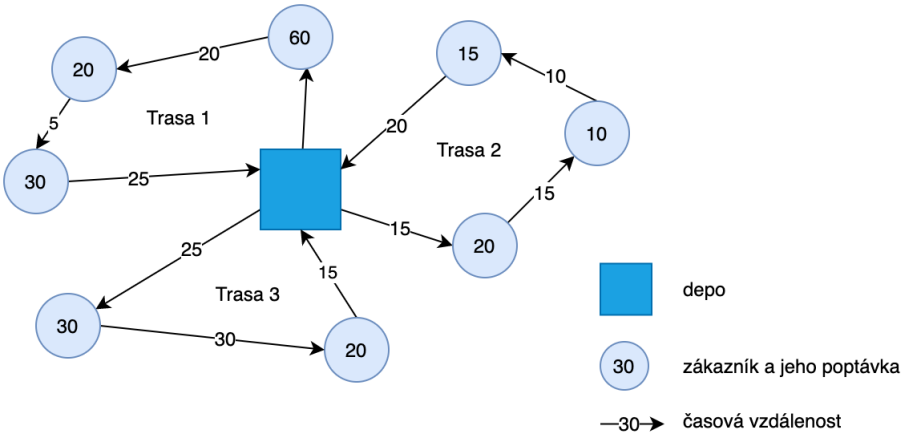
Problém okružních jízd nebo Vehicle Routing Problem (VRP) je kombinatorický, optimalizační problém, jeden z často vyskytujících problémů v oblasti

dopravy, logistiky a distribuce. Poprvé tento problém byl popsán Dantzigem a Ramserem v roce 1959 jako "The truck dispatching problem" [1].

Na rozdíl od TSP v depu jsou k vozidel, pro které je potřeba nalézt k tras, které stejně jako i v TSP začínají a končí v depu (Obrázek 3.2). Dalším rozdílem jsou vrcholy, které jsou představené jako zákazníky a mají hodnoty, které ukazují množství zboží, které má být dodáno zákazníkovi z depa. [4]



Obrázek 3.1: Problém obchodního cestujícího



Obrázek 3.2: Problém okružních jízd

3.2.1 Definice problému okružních jízd

Problém okružních jízd je definován na úplném neorientovaném grafu $G = (V, E)$. Množina V reprezentuje vrcholy grafu. Množina vrcholů $Z = V \setminus \{0\}$ představuje zákazníky a vrchol $v_0 \in V$ představuje depo. Každý vrchol $i \in Z$ má nezápornou poptávku q_i. Každá hrana $e \in E = \{(i, j)|i, j \in V, i \neq j\}$ je ohodnocená dopravním nákladem c_{ij}. V depu je k dispozici flotila homogenních vozidel, která je označována množinou K. Každé vozidlo má stejnou maximální kapacitu Q. [5]

Cílem VRP je najít řešení, které se skládá z m tras, jejichž celkové dopravní

náklady jsou minimalizovány. Každý zákazník musí být přiřazen přesně na jednu trasu pouze jednou. Každá trasa musí začínat a končit v depu, přičemž celková poptávka všech zákazníků trasy nepřesahuje maximální kapacitu vozidla. [5]

3.2.2 Matematický model problému okružních jízd

Matematický model pro problém okružních jízd popsaný Z. Borcinovou [6] obsahuje binární proměnné x_{ijk}, kde $i, j \in V, i \neq j, k \in K$, a které mají hodnotu:

- 1, pokud hrana mezi vrcholy i a j je přiřazená vozidlu k
- 0, pokud hrana mezi vrcholy i a j není přiřazená vozidlu k

Účelová funkce, kterou je potřeba minimalizovat, reprezentuje celkové náklady na dopravu a má následující tvar:

$$min \sum_{k=1}^{|K|} \sum_{i=0}^{|V|} \sum_{j=0, j\neq i}^{|V|} c_{ij} x_{ijk} \tag{3.1}$$

Účelová funkce musí podléhat podmínkám:

$$\sum_{k=1}^{|K|} \sum_{i=0, i\neq j}^{|V|} x_{ijk} = 1 \text{ , pro } \forall j \in Z \tag{3.2}$$

(3.2) podmínka, která zajišťuje, že každý zákazník j bude navštíven přesně jednou.

$$\sum_{j=1}^{|V|} x_{0jk} = 1 \text{ , pro } \forall k \in K \tag{3.3}$$

(3.3) podmínka, která zajišťuje, že každé vozidlo k může opustit depo pouze jednou.

$$\sum_{i=0, i\neq j}^{|V|} x_{ijk} = \sum_{i=0}^{|V|} x_{jik} \text{ , pro } \forall j \in Z, \forall k \in K \tag{3.4}$$

(3.4) podmínka, která zajišťuje, že každé vozidlo k, které navštíví zákazníka j, opustí zákazníka j.

$$\sum_{i=0}^{|V|} \sum_{j=0, j\neq i}^{|V|} d_j x_{ijk} \leq Q \text{ , pro } \forall k \in K \tag{3.5}$$

(3.5) podmínka, která zajišťuje, že celková poptávka všech zákazníků, které jsou přiřazení na trasu vozidla k, nepřevýší maximální kapacitu tohoto vozidla.

3.3 Problém okružních jízd s časovými okny

Problém okružních jízd s časovými okny nebo Vehicle Routing Problem With Time Windows (VRPTW) je jednou z nejznámějších generalizací Problému okružních jízd a popisuje situaci, ve které zákazník požaduje, aby jeho obsloužení se začalo pouze ve stanoveném časovém intervalu (Obrázek 3.3). [9]

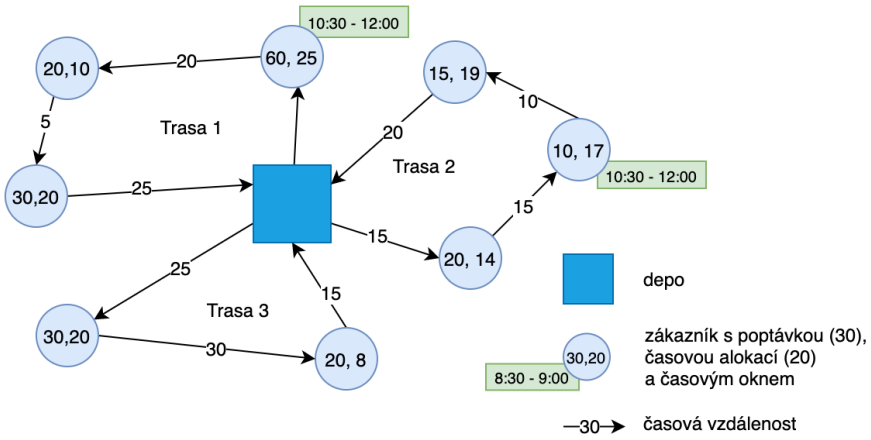
3.3.1 Definice problému okružních jízd s časovými okny

Problém popsaný S.N Kumarem a R. Panneerselvamem [9] je definován jako graf $G = (V, E)$, kde V je množina vrcholů a E je množina hran. Množina vrcholů $Z = \{1, ...n\}$ reprezentuje zákazníky a vrchol 0 reprezentuje depo. Prvky množiny E reprezentují hrany mezi všemi dvojicemi vrcholů množiny V. Pro každou hranu $(i, j) \in E$ jsou stanoveny náklady na dopravu c_{ij} a časová vzdálenost t_{ij}.

V depu je umístěná množina identických vozidel K, kde každé vozidlo k má maximální povolenou dobu trasy r_k a maximální kapacitu Q.

Každý zákazník i má poptávku m_i. Pro každého zákazníka i je definována doba f_i potřebná k obsloužení daného zákazníka a časový interval $[e_i, l_i]$, ve kterém požaduje, aby byla zahájena obsluha. Vozidlo může dorazit k zákazníkovi i i před časem e_i, ale bude muset počkat.

Každá trasa řešení musí začínat a končit v depu. Každý zákazník v rámci jednoho řešení musí být navštíven přesně jednou. V rámci jednoho řešení k jednomu vozidlu může být přiřazená pouze jedna trasa.



Obrázek 3.3: Problém okružních jízd s časovými okny

3.3.2 Matematický model problému okružních jízd s časovými okny

Pro matematický model VRPTW musíme definovat proměnné [9]:

- T_i - čas příjezdu do vrcholu i.

- w_i - doba čekání u vrcholu i, pokud vozidlo dorazí dřív.
- $x_{ijk} \in \{0, 1\}$, kde $i, j \in V, i \neq j, k \in K$ - binární proměnná, která má hodnotu:
  - 1, pokud cesta mezi vrcholy i a j je přiřazená vozidlu k
  - 0, jinak

Matematický model VRTPW má následující tvar [9]:

$$min \sum_{k=1}^{|K|} \sum_{i=0}^{|V|} \sum_{j=0, j\neq i}^{|V|} c_{ij} x_{ijk} \tag{3.6}$$

(3.6) účelová funkce, která minimalizuje celkové náklady všech tras a podléhá podmínkám 3.6 - 3.16.

$$\sum_{k=1}^{|K|} \sum_{j=1}^{|V|} x_{0jk} \leq |K| \tag{3.7}$$

(3.7) podmínka, která zajišťuje, že počet hran, které vedou z depa, není větší než počet vozidel.

$$\sum_{j=1}^{|V|} x_{0jk} = 1 \text{ , pro } \forall k \in K \tag{3.8}$$

(3.8) podmínka, která zajišťuje, že pro každou trasu, která se vyplňuje vozidlem k, existuje přesně jedna hrana, která vede z depa.

$$\sum_{i=1}^{|V|} x_{i0k} = 1 \text{ , pro } \forall k \in K \tag{3.9}$$

(3.9) podmínka, která zajišťuje, že pro každou trasu, která se vyplňuje vozidlem k, existuje přesně jedna hrana, která vede do depa.

$$\sum_{k=1}^{|K|} \sum_{j=0, i\neq j}^{|V|} x_{ijk} = 1 \text{ , pro } \forall i \in Z \tag{3.10}$$

(3.10) podmínka, která zajišťuje, že v celém řešení existuje přesně jedna hrana, která vede z vrcholu i.

$$\sum_{k=1}^{|K|} \sum_{i=0, i\neq j}^{|V|} x_{ijk} = 1 \text{ , pro } \forall j \in Z \tag{3.11}$$

(3.11) podmínka, která zajišťuje, že v celém řešení existuje přesně jedna hrana, která vede do vrcholu j.

$$\sum_{i=1}^{|V|} m_i \sum_{j=0, i\neq j}^{|V|} x_{ijk} \leq Q \text{ , pro } \forall k \in K \tag{3.12}$$

(3.12) podmínka, která zajišťuje, že celková poptávka všech vrcholů trasy, která se vyplňuje vozidlem k, nepřevýší maximální kapacitu tohoto vozidla.

$$\sum_{i=1}^{|V|} \sum_{j=0, i \neq j}^{|V|} x_{ijk}(t_{ij} + f_i + w_i) \leq r_k \text{ , pro } \forall k \in K \tag{3.13}$$

(3.13) podmínka, která zajišťuje, že celá doba trasy vozidla k nepřekročí jeho maximální povolenou dobu jedné trasy.

$$T_0 = w_0 = f_0 = 0 \tag{3.14}$$

(3.14) podmínka, která nastavuje čas příjezdu, dobu čekání a dobu potřebnou k obsloužení v depu na nulu.

$$\sum_{k=1}^{|K|} \sum_{i=0, i \neq j}^{|V|} x_{ijk}(T_i + t_{ij} + f_i + w_i) \leq T_j \text{ , pro } \forall j \in Z \tag{3.15}$$

(3.15) podmínka, která zajišťuje, že čas příjezdu do vrcholu j není větší než zadaný čas příjezdu pro tento vrchol.

$$e_i \leq (T_i + w_i) \leq l_i \text{ , pro } \forall i \in Z \tag{3.16}$$

(3.16) podmínka, která zajišťuje, že obsluha zákazníka i bude zahájena v časovém intervalu tohoto zákazníka.

Kapitola 4

Formulace vlastního problému

Vlastní problém odpovídající požadavkům z podkapitoly 2.3 bude popsán jako generalizace VRPTW. Pro formulace vlastního problému potřebujeme přidat do VRPTW následující změny:

- Množinu homogenních vozidel nahradíme množinou heterogenních vozidel, kde každé vozidlo má vlastní náklad a matici časových vzdáleností.
- Problém musí brát v úvahu i množinu řidičů, kde každý řidič má svou pracovní dobu a mzdový náklad. Každému řidiči může být přiřazená maximálně jedna trasa.
- Pojmy "depo" a "zákazník" budou nahrazeny pojmy "kancelář" a "úkol".
- Spojení mezi úkoly se bude počítat pomocí lokací, ve kterých se úkoly vyplňují.
- VRPTW popisuje situaci, ve které obsluha zákazníka musí být zahájena v jeho časovém okně, ale vlastní problém musí popisovat situaci, ve které vyplnění úkolu musí být zahájeno a dokončeno v časovém intervalu.
- Přidáme možnost plánovat několik tras pro jedno vozidlo v rámci jednoho řešení.
- Zrušíme poptávku zákazníků, maximální kapacitu a maximální povolenou dobu trasy u vozidel.
- Čas příjezdu do vrcholu a dobu čekání u vrcholu nahradíme časem, ve kterém bylo zahájeno vyplnění úkolu.

4.1 Popis problému

Je definována dopravní síť v podobě úplného grafu $G = (V, E)$, kde V je množina vrcholů a E je množina hran. Množina vrcholů $Z = V \setminus \{0\} = 1, ...n$

představuje úkoly. Vrchol $v_0 \in V$ představuje kancelář. Množina E obsahuje hrany mezi každou dvojicí vrcholů z množiny V.

Množina heterogenních vozidel je definována jako K, kde každé vozidlo $k \in K$ má provozní náklad c_k.

Pro každé vozidlo $k \in K$ definujeme matici T_k typu $|V|x|V|$. Prvek matice t_{ijk} reprezentuje časovou vzdálenost mezi vrcholy $i \in V$ a $j \in V$ pro vozidlo k.

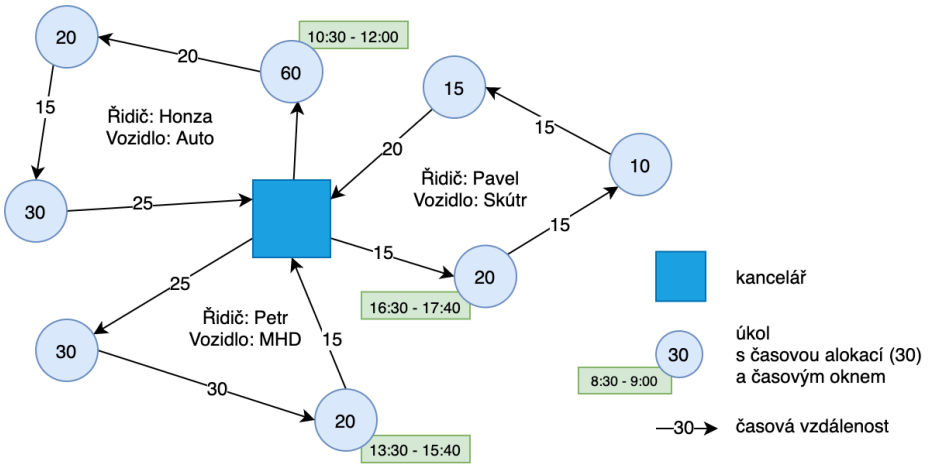
Množina řidičů je definována jako N. Každý řidič $n \in N$ má stanovený mzdový náklad w_n a pracovní dobu $[a_n, b_n]$. Naplánovaná trasa řidiče n musí být zahájena po čase a_n a musí být dokončena před časem b_n.

Hodnocení každého spojení mezi vrcholem $i \in V$ a vrcholem $j \in V$ je dopravním nákladem, který se skládá z mzdových nákladů na řidiče a provozních nákladů na vozidlo.

Každý vrchol $i \in Z$ má časový interval $[e_i, l_i]$, ve kterém požaduje, aby bylo zahájeno a dokončeno vyplnění úkolu. Doba vyplnění úkolu i je definována jako f_i.

Pro každý vrchol $i \in Z$ definujeme čas p_i, ve kterém bylo zahájeno vyplnění úkolu.

Cílem problému je navrhnout množinu tras R, kde každá trasa začíná a končí v kanceláři, a jejichž celkové dopravní náklady jsou minimalizovány. V rámci celého řešení každý úkol musí být přiražen pouze na jednu trasu přesně jednou.



Obrázek 4.1: Vlastní problém

4.2 Matematický model problému

Pro formulace matematického modelu problému je nutné definovat proměnnou x_{ijkn}, kde $(i, j) \in E$, $i \neq j$, $k \in K$, $n \in N$. Tato proměnná je binární a má hodnotu:

- 1, pokud cesta mezi vrcholy i a j je přiřazená vozidlu k a řidičovi n.
- 0, jinak.

Matematický model problému bude mít následující tvar:

$$min \sum_{k=1}^{|K|} \sum_{n=1}^{|N|} \sum_{i=0}^{|V|} \sum_{j=0, j \neq i}^{|V|} x_{ijk}(t_{ijk}c_k + (t_{ijk} + f_i)m_n) \tag{4.1}$$

(4.1) účelová funkce, která minimalizuje celkové náklady všech tras a podléhá podmínkám 3.18 - 3.26.

$$\sum_{k=1}^{|K|} \sum_{n=1}^{|N|} \sum_{j=1}^{|V|} x_{0jkn} \leq |N| \tag{4.2}$$

(4.2) podmínka, která zajišťuje, že počet hran, které vedou z kanceláře, není větší než počet řidičů.

$$\sum_{k=1}^{|K|} \sum_{j=1}^{|V|} x_{0jkn} \leq 1 \text{ , pro } \forall n \in N \tag{4.3}$$

(4.3) podmínka, která zajišťuje, že v celém řešení pro řidiče n existuje maximálně jedna hrana, která vede z kanceláře.

$$\sum_{k=1}^{|K|} \sum_{i=1}^{|V|} x_{i0kn} \leq 1 \text{ , pro } \forall n \in N \tag{4.4}$$

(4.4) podmínka, která zajišťuje, že v celém řešení pro řidiče n existuje maximálně jedna hrana, která vede do kanceláře.

$$\sum_{k=1}^{|K|} \sum_{n=1}^{|N|} \sum_{j=0, i \neq j}^{|V|} x_{ijkn} = 1 \text{ , pro } \forall i \in Z \tag{4.5}$$

(4.5) podmínka, která zajišťuje, že v celém řešení existuje přesně jedna hrana, která vede z vrcholu i.

$$\sum_{k=1}^{|K|} \sum_{n=1}^{|N|} \sum_{i=0, i \neq j}^{|V|} x_{ijk} = 1 \text{ , pro } \forall j \in Z \tag{4.6}$$

(4.6) podmínka, která zajišťuje, že v celém řešení existuje přesně jedna hrana, která vede do vrcholu j.

$$p_0 = f_0 = 0 \tag{4.7}$$

(4.7) podmínka, která nastavuje čas zahájení úkolu a dobu vyplnění úkolu v kanceláři na nulu.

$$x_{ijkn}(p_i + f_i + t_{ij}) \leq p_j \text{ , pro } \forall i, j \in V,\ i \neq j,\ \forall k \in K,\ \forall n \in N \tag{4.8}$$

(4.8) nerovnost, která zajišťuje, že čas příjezdu do vrcholu j není větší než naplánovaný čas zahájení úkolu.

$$e_i \leq (p_i + f_i) \leq l_i \text{ , pro } \forall i \in Z \tag{4.9}$$

(4.9) nerovnost, která zajišťuje, že vyplnění úkolu ve vrcholu i bude zahájeno a dokončeno v jeho časovém okně.

$$a_n \leq \sum_{k=1}^{|K|} \sum_{i=0}^{|V|} \sum_{j=0, j \neq i}^{|V|} x_{ijk}(f_i + t_{ijk}) \leq b_n \text{ , pro } \forall n \in N \tag{4.10}$$

(4.10) nerovnost, která zajišťuje, že trasa pro řidiče n bude naplánována v jeho pracovní době.

Kapitola 5

Existující metody řešení VRPTW

N. El-Sherbeny ve své práci [8] uvádí tři základní druhy metod, které slouží k řešení problému okružních jízd s časovými okny: exaktní metody, heuristické metody a metaheuristické metody. Jelikož exaktní metody zvládají v rozumném čase pracovat jen s malými úlohami a metaheuristické metody slouží k výpočtu složitých a rozsáhlých úloh, v následující podkapitole budou představeny pouze heuristické metody. Tyto metody musí zvládnout nalézt optimální řešení v rozumném čase pro počet úkolů, který je uveden v požadavcích zadavatele v podkapitole 2.3.

5.1 Heuristické metody

Heuristické metody jsou speciální metody pro řešení poměrně rozsáhlých problémů rozumném čase. Řešení, která dávají heuristické metody nejsou vždycky optimální. M.M. Solomon [13] popsal čtyři druhy heuristických metod: Metoda výhodnostních koeficientů, Metoda nejbližších sousedů, Vkládací metoda a Stírací metoda.

5.1.1 Metoda výhodnostních koeficientů

Metoda výhodnostních koeficientů pro VRPTW je rozšíření Clark-Wrightovy metody [14], která řeší VRP. Princip této metody spočívá ve sdružování dvou tras $(0, i, 0)$ a $(0, j, 0)$ do jedné podle tzv. výhodnostního koeficientu $v_{ij} = c_{i0} + c_{j0} - c_{ij}$, kde c_{i0} vzdálenost mezi depem a zákazníkem i, c_{j0} vzdálenost mezi depem a zákazníkem j a c_{ij} vzdálenost mezi zákazníky i a j. Vždy sdružujeme ty dvě trasy, které mají nejvyšší výhodnostní koeficient a dodržují omezující podmínky jako maximální kapacita vozidla a časová okna zákazníků.

5.1.2 Metoda nejbližších sousedů

Metoda nejbližšího souseda začíná plánování každé trasy tím, že nalezne nejbližšího zákazníka od depa, který dosud není naplánován do žádné trasy. V každém následném kroku metoda hledá nejbližšího dostupného zákazníka k poslednímu přiřazenému zákazníku aktuální trasy. Přiřazení zákazníka na

trasu musí dodržovat jeho časová okna a nepřevyšovat maximální kapacitu vozidla. Plánování nové trasy se začne, pokud už nebude možné přiřadit dalšího zákazníka na aktuální trasu. Plánování celého řešení se ukončí, pokud všechny vozidla budou obsazena nebo všichni zákazníci budou naplánovaní do vytvořených tras.[13]

5.1.3 Vkládací metoda

Vkládací metoda začíná plánování každé trasy sadou uzlů (i_o, i_m), která reprezentuje aktuální formující trasu, která začíná a končí v depu $(i_0 = i_m = 0)$. V každém následném kroku metoda spočítá pro každého nezařazeného zákazníka nejlepší vkládací místo ve formující trase a pak do ní zařazí zákazníka, který má nejlepší hodnotu vkládacího místa, přičemž zařazení dodržuje maximální kapacitu vozidla a časová okna všech zákazníků ve formující trase. Pokud nebude možné zařadit dalšího zákazníka k formující trase, začne se plánování nové trasy. Pokud všechna vozidla budou obsazena nebo všichni zákazníci budou zařazeni do tras, plánování celého řešení se ukončí. [13]

5.1.4 Stírací metoda

Stírací metoda dělí proces řešení na fázi shlukování a fázi plánováni. V první fázi se každému vozidlu vytváří shluk zákazníků, kterým ještě nebyla naplánována trasa. Tyto shluky musí dodržovat maximální kapacitu vozidla a jednotliví zákazníci od sebe nesmí být příliš daleko vzdálení. Ve druhé fázi se vytváří ze shluku samotná trasa vozidla, kde se dodržují časová okna všech přiřazených zákazníků. Kvůli omezením časového okna někteří zákazníci v shluku mohou zůstat neplánovaní, a proto budeme opakovat proces plánování stejným způsobem, dosud všichni zákazníci nebudou naplánováni do tras. [13]

Kapitola 6

Algoritmus pro řešení vlastního problému

V této kapitole bude popsán návrh a proces implementace algoritmu, který řeší problém popsaný v kapitole 4. Následně bude popsán proces testování algoritmu.

6.1 Základní metoda algoritmu

Jelikož budeme rozšiřovat původní existující metodu tak, aby dodržovala všechny omezující podmínky vlastního problému, musíme vytvořit algoritmus na základě metody, která je jednoduchá a zároveň i efektivní. Pro tyto účely je zvolena heuristická metoda nejbližších sousedů, která by měla zvládnout nalézt optimální a odpovídající všem požadavkům řešení v rozumném čase, jelikož v každém hledání dalšího zákazníka formující trasy prochází množinu dostupných zákazníků pouze jednou.

6.2 Vstupní data

Algoritmus požaduje na výpočet řešení sadu dat, do které patří:

1. Množina vozidel, kde každé vozidlo má unikátní ID, název, typ a náklad na hodinu. Typ vozidla může být auto, skútr nebo městská doprava.

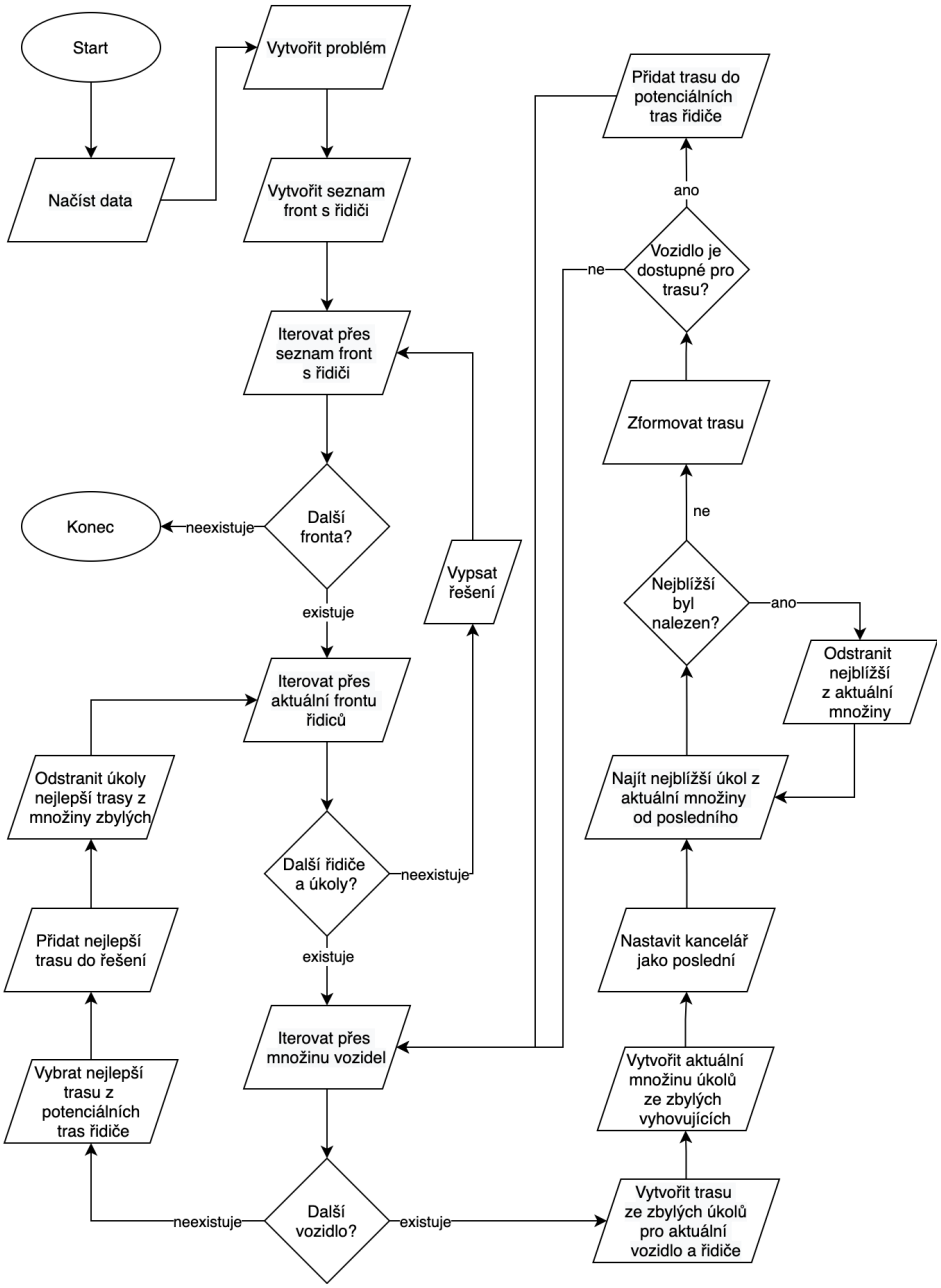
2. Množina řidičů, kde každý řidič má unikátní ID, jméno, mzdu na hodinu a časový interval, který reprezentuje pracovní dobu.

3. Množina lokací, ve kterých se řidiče vyplňují úkoly. Každá lokace má ID a název. První lokace množiny vždy reprezentuje kancelář.

4. Množina distančních matic pro každý typ vozidla. Distanční matice reprezentuje časové vzdálenosti v minutách mezi všemi dvojicemi z množiny lokací.

5. Množina úkolů, kde každý úkol má unikátní ID, název, lokace, kde se vyplňuje, prioritu vyplnění, časovou alokaci v minutách, časový interval, ve kterém tento úkol musí být vyplněn, množina řidičů,

které mohou jej vyplnit, a množina vozidel, kterými úkol může být obsloužen.



Obrázek 6.1: Vývojový diagram algoritmu

6.3 Návrh algoritmu

V této podkapitole bude popsána činnost algoritmu pro nalezení optimálního řešení podle vytvořeného vývojového diagramu (Obrázek 6.1).

6.3.1 Vytváření řešení

Pokud algoritmus má na vstupu správná data, vytvoří se na jejich základě problém, do kterého se budou uloženy všechna data. Po vytvoření problému algoritmus vytvoří z množiny řidičů dvě seřazené množiny, kde každá obsahuje všechny řidiče z původní množiny. První množina je seřazena podle mzdových nákladů řidiče. Druhá množina je seřazena podle množství úkolů, ve kterých řidič je jeden z požadovaných řidičů. Pro každou množinu se vytvoří řešení.

Každé řešení obsahuje množinu tras, kde každá trasa má množinu úkolů, přiřazeného řidiče a vozidlo, čas zahájení a ukončení. Jednotlivé úkoly trasy mají naplánovaný čas zahájení a dokončení vyplnění.

Jednotlivá řešení se vytváří následujícím způsobem:

1. Zkontrolovat, jestli množina nepřiřazených úkolů ještě obsahuje prvky. Pokud množina je prazdná, ukončit vytváření řešení.

2. Vybrat prvního řidiče ze seřazené množiny volných řidičů. Pokud množina již neobsahuje další řidiče, ukončit vytváření řešení.

3. Z množiny nepřiřazených úkolů získat množinu úkolů, které požadují vyplnění aktuálním řidičem. Z úkolů získané množiny získat typy vozidla, kterými musí být obslouženy. Pokud množství takových typů vozidel větší než 0, z množiny všech vozidel vynechat vozidla jiného typu.

4. Vytvořit trasy pro aktuálního řidiče se všemi vozidly z předchozího kroku. Trasy budou vytvořeny podle návodu z podpodkapitoly 6.3.2. Pokud vozidlo bude dostupné po celou dobu aktuální trasy, přidat tuto trasu do množiny potenciálních tras aktuálního řidiče.

5. Pro každou trasu z množiny potenciálních tras spočítat průměrný náklad, který se počítá pomocí vzorce $\overline{C} = C/T$, kde C je celkový náklad trasy a P je počet úkolů trasy.

6. Pokud množina potenciálních tras aktuálního řidiče obsahuje alespoň jednu trasu, vybrat z nich nejlepší a přidat ji do aktuálního řešení. Nejlepší trasa je taková, která má nejmenší průměrný náklad. Všechny úkoly z nejlepší trasy odstranit z množiny nepřiřazených úkolů. Řidiče trasy odstranit z množiny volných řidičů a vrátit se zpět k 1. kroku.

6.3.2 Vytváření jednotlivých tras

Všechny trasy musí odpovídat omezujícím podmínkám problému z kapitoly 4 a vytváří se následujícím způsobem [13]:

1. Z množiny nepřiřazených úkolů odstranit takové úkoly, které mají být obslouženy jinými vozidly a řidiči než aktuální. Následně odstranit úkoly, u kterých se časová okna nepřekrývají s pracovní dobou

aktuálního řidiče. Získanou množinu označit jako aktuální množina úkolů.

2. Označit kancelář jako poslední přiřazený úkol.

3. Zjistit čas, ve kterém řidič dokončí poslední úkol trasy.

4. Zjistit pro každý úkol z aktuální množiny časovou vzdálenost od posledního úkolu aktuální trasy.

5. Určit pro každý úkol z aktuální množiny největší čas mezi:

   - časem, ve kterém začíná časové okno lokace,
   - časem, ve kterém se řidič dostaví do aktuální lokace.

   Největší čas nastavit jako možný čas zahájení vyplnění úkolu.

6. Z aktuální množiny vynechat úkoly, které není možné stihnout v rámci časového okna nebo po dokončení kterých se nestíhá vrátit zpět do kanceláře před ukončením pracovní doby řidiče.

7. Spočítat pro každý úkol z aktuální množiny časový rozdíl mezi časem opuštění lokace, ve kterém se byl vyplněn poslední úkol, a časem zahájení vyplnění aktuálního úkolu.

8. Pro každý úkol z aktuální množiny spočítat "blízkost" od posledního úkolu, která se rovná součtu časové vzdálenosti a časového rozdílu z bodu 7. Po získání součtu je potřeba přepočítat "blízkost" pomocí priority, časového okna úkolu, počtu řidičů a vozidel, kterými se může být obsloužen úkol.

   Vysoká priorita úkolu vynásobí hodnotu "blízkosti" číslem 0.5, střední priorita úkolu vynásobí hodnotu číslem 0.7 a nízká priorita úkolu vynásobí hodnotu číslem 1.

   Zbytek časového okna aktuálního úkolu vynásobit číslem 0.05 a výsledek připočítat ke "blízkosti".

9. Vybrat úkol s nejmenší "blízkosti" a označit jej jako nejbližší.

10. Pokud nejbližší úkol bude nalezen, přiřadit jej na trasu, označit jako poslední v trase, odstranit z aktuální množiny úkolů. Pokud nalezený nejbližší úkol je přiřazen do trasy po kanceláři, nastavit pro aktuální trasu čas její zahájení. Čas zahájení trasy se rovná času, ve kterém řidič potřebuje opustit kancelář, aby se dostavil do lokace úkolu v naplánovaný čas zahájení vyplnění úkolu. Poté se vrátit do 3. bodu a pokračovat ve formování trasy.

    Pokud nejbližší úkol nebude nalezen, tak ukončit plánování trasy a nastavit čas její ukončení. Čas ukončení trasy se rovná času, ve kterém se řidič dostane zpátky do kanceláře po vyplnění posledního úkolu trasy.

6.4 Výstupy algoritmu

Výstupem algoritmu jsou řešení pro každou seřazenou množinu řidičů. Každá řešení obsahuje trasy, kde každá trasa má:

- přiřazené vozidlo,
- přiřazeného řidiče,
- naplánované časy začátku a konce,
- seřazenou množinu úkolů, kde každý úkol bude mít naplánovaný čas zahájení a dokončení vyplnění,
- cenu, která se skládá z průměrných náklad na řidiče a dopravu.

6.5 Implementace algoritmu

Jelikož logická část informačního systému společnosti Blahobyty je naimplementována v jazyce Java, navržený algoritmus byl naimplementován ve stejném programovacím jazyce pro snadnou integraci. Zdrojový kód algoritmu je přílohou práce (viz Příloha E).

6.5.1 Třídy

Tato podpodkapitola popisuje třídy, které byly vytvořeny v průběhu implementaci algoritmu.

Výčtové typy

1. VehicleType - výčtový typ, který reprezentuje typy vozidla. Má tři hodnoty: CAR, SCOOTER, PUBLIC_TRANSPORT.
2. TaskPriority - výčtový typ, který reprezentuje priority vyplnění úkolu. Má tři hodnoty: HIGH, MEDIUM, LOW.

Datové modely

1. Driver - třída, která reprezentuje řidiče a má následující atributy:
   - ID (Long) - unikátní identifikační číslo.
   - name (String) - jméno řidiče.
   - hourlyRate (int) - mzdový náklad na hodinu řidiče.
   - shiftStart (int) - začátek pracovní doby řidiče.
   - shiftEnd (int) - konec pracovní doby řidiče.
2. Vehicle - třída, která reprezentuje vozidlo a má následující atributy:

- ID (Long) - unikátní identifikační číslo.
- name (String) - název vozidla.
- type (VehicleType) - typ vozidla.
- hourlyRate (double) - dopravní náklad na hodinu vozidla.

3. Location - třída, která reprezentuje lokaci, ve které se vyplňují úkoly. Má následující atributy:
   - ID (Long) - unikátní identifikační číslo.
   - name (String) - název lokace.
   - office (boolean) - true, pokud lokace je kancelář, nebo false, pokud lokace není kancelář.
4. DistanceMatrix - třída, která reprezentuje distanční matici a má následující atributy:
   - vehicleType (VehicleType) - typ vozidla, pro který tato matice určena.
   - distanceMatrix (int[ ][ ]) - matice, která obsahuje časové vzdálenosti v minutách mezi všemi dvojicemi z množiny lokací.
5. Task - třída, která reprezentuje úkol a má následující atributy:
   - ID (Long) - unikátní identifikační číslo.
   - name (String) - název úkolu.
   - location (Location) - lokace, ve kterých se úkol vyplnuje.
   - priority (TaskPriority) - priorita vyplnění úkolu.
   - requiredTime (int) - časová alokace úkolu.
   - timeWindowStart (int) - začátek časového okna.
   - timeWindowEnd (int) - konec časového okna.
   - requiredDrivers (List<Driver>) - seznam řidičů, kteří mohou vyplnit úkol.
   - requiredVehicleTypes (List<VehicleType>) - seznam typů vozidel, kteří mohou obsloužit úkol.
   - plannedStart (int) - naplánovaný čas, ve kterém se začíná vyplnění úkolu.
   - plannedEnd (int) - naplánovaný čas, ve kterém se dokončí vyplnění úkolu.
6. Problem - třída, která reprezentuje aktuální problém a obsahuje všechna vstupní data. Má následující atributy:
   - drivers (List<Driver>) - seznam všech řidičů.
   - vehicles (List<Vehicle>) - seznam všech vozidel.

- locations (List<Location>) - seznam všech lokací, kde první lokace je kancelář.
- tasks (List<Task>) - seznam všech úkolů.
- distanceMatrices (List<DistanceMatrix>) - seznam distančních matic.

7. Route - třída, která reprezentuje trasu a má následující atributy:
   - start (Integer) - naplánovaný čas začátku trasy.
   - end (Integer) - naplánovaný čas konce trasy.
   - driver (Driver) - přiřazený řidič.
   - vehicle (Vehicle) - přiřazené vozidlo.
   - tasks (List<Task>) - seznam všech úkolů, které budou obslouženy v rámci této trasy.
   - lastTask (Task) - naposled přiřazený úkol trasy.
   - totalCost (double) - celkový náklad trasy.
   - averageTotalCost (double) - průměrný náklad trasy.
8. ProblemSolution - třída, která reprezentuje vytvořené řešení a má následující atributy:
   - problem (Problem) - aktuální problém, pro který bylo vytvořeno řešení.
   - routes (List<Route>) - seznam vytvořených tras.
   - solutionCost (double) - celkové náklady všech tras.
   - unroutedTasks (List<Task>) - všechny úkoly, které nebyly přiřazený do žádné trasy řešení.

Přehlednější informace o datových modelech obsahuje diagram tříd (viz Příloha D).

Obslužné třídy

1. ProblemReader - třída obsahující metody, které načítají data ze souborů. Všechny časy ze vstupních dat jsou převedeny z formátu "HH:mm" na minuty od půlnoci (například: 10:30 = 630).
2. ProblemSolver - třída, která řeší problém a má následující metody:
   - List<Queue<Driver>> getDriverQueues() - metoda, která vytvoří z původního seznamu řidičů dvě seřazené fronty řidičů.
   - routePlanningHeuristic() - metoda, která vytváří řešení pomocí metody *solutionCreator* pro každou frontu řidiče ze seznamu, který vrací metoda *getDriverQueues()*.

- ProblemSolution solutionCreator(Queue<Driver> drivers, List<Vehicle> vehicles, List<Task> tasks) - metoda, která vytváří trasy pomocí metody *routeCreator* a pak z vytvořených tras formuje řešení.
- Route routeCreator(List<Task> tasks, Driver driver, Vehicle vehicle) - metoda, která pro daného řidiče *driver* a vozidlo *vehicle* vytváří jednotlivou trasu z množiny úkolů *tasks*. Trasa se vytváří postupným vyhledáváním a přidáváním dalšího úkolu pomocí metody *findNearestNeighbour*.
- Task findNearestNeighbour(Route route, List<Task> tasks, int[ ] distancesToOffice, int[ ][ ] distancesTaskLocations) - metoda, která hledá nejbližší úkol ze seznamu *tasks* pro poslední úkol z dané trasy *route*.
- double getTaskNearness(Route route, Task task, int minVisitTime, int lastTaskEndTime, int distance) - metoda, která spočítá "blízkost" mezi aktuálním a posledním přiřazeným úkolem trasy.
- boolean checkVehicleAvailability(Route possibleRoute, List<Route> routes) - metoda, která ověří dostupnost vozidla během potenciální trasy *possibleRoute* mezi už naplánovanými trasami *routes*.

3. ProblemSolutionWriter - třídá, která vypíše jednotlivá řešení do konzole. Vyvolává se ze třídy *ProblemSolver* po každém nalezení řešení.

4. Main - třída s metodou *main*, která spustí metody načítání dat ze třídy *ProblemReader*, vytvoří z načtených dat problém pomocí třídy *Problem* a pomocí třídy *ProblemSolver* zkusí vyřešit aktuální problém.

6.6 Testování algoritmu

Pro testování algoritmu spolu se zaměstnancem společnosti Blahobyty byly vytvořeny 3 sady souborů s přibližně reálnými daty. Testovací data jsou uloženy v repozitáři zdrojového kódu (viz Příloha E). V průběhu testování taky byla udělána kalibraci algoritmu, kde byly zkoušené různé hodnoty koeficientů používané při počítání "blízkosti" úkolu a následně byly nastaveny ty nejlepší.

6.6.1 První sada dat

První sada dat obsahuje:

- soubor s 2 řidiči a jejími vlastnostmi;

- soubor s vozidly, kde jsou představeny 3 auty, 4 skútry, městská doprava a jejich náklady;
- soubor se 101 lokací, kde první je kancelář a zbylé jsou lokace, ve kterých se vyplňují úkoly;
- 3 soubory pro 3 typy vozidla, kde v každém souboru je distanční matice typu 101x101;
- soubor se 40 úkoly.

S touto sadou souborů algoritmus za 0.5 sekund vytvořil dvě řešení, kde nejlepší řešení bylo první, které mělo 2 trasy pro 2 řidiče. Jeden úkol s nízkou prioritou zůstal nenaplánovaný v rámci řešení.

Číslo trasy	Naplánovaná doba trasy	Pracovní doba řidiče	Vozidlo	Počet úkolů
1	07:00 - 20:51	07:00 - 21:00	Skůtr	26
2	08:54 - 14:58	08:00 - 19:00	Auto	13

Tabulka 6.1: Řešení pro 40 úkolů

6.6.2 Druhá sada dat

Druhá sada dat obsahuje:

- soubor s 5 řidiči a jejími vlastnostmi,
- soubor s vozidly, kde jsou představeny 3 auta, 4 skútry, městská doprava a jejich náklady,
- soubor se 101 lokací, kde první je kancelář a zbylé jsou lokace, kde se vyplňují úkoly,
- 3 soubory pro 3 typy vozidla, kde v každém souboru je distanční matice typu 101x101,
- soubor se 100 úkoly.

S touto sadou souborů algoritmus za 1 sekundu vytvořil dvě řešení, kde nejlepší bylo druhé řešení, které mělo 5 tras pro 5 řidičů. Všechny úkoly v rámci tohoto řešení byly naplánovány, zatímco jiné řešení mělo 2 nenaplánovaných úkolů se střední prioritou. Celkové náklady nejlepšího řešení je nižší o 10% než u jiného řešení.

Číslo trasy	Naplánovaná doba trasy	Pracovní doba řidiče	Vozidlo	Počet úkolů
1	06:00 - 17:56	06:00 - 18:00	MHD	23
2	05:00 - 19:00	05:00 - 19:00	Auto	25
3	09:00 - 22:54	09:00 - 23:00	Skůtr	29
4	07:00 - 16:40	07:00 - 17:00	Auto	19
5	05:00 - 9:26	05:00 - 22:00	MHD	4

Tabulka 6.2: Řešení pro 100 úkolů

6.6.3 Třetí sada dat

Třetí sada dat obsahuje:

- soubor se 7 řidiči a jejími vlastnostmi,
- soubor s vozidly, kde jsou představeny 3 auta, 4 skútry, městská doprava a jejich náklady,
- soubor se 101 lokací, kde první je kancelář a zbylé jsou lokace, kde se vyplňují úkoly,
- 3 soubory pro 3 typy vozidla, kde v každém souboru je distanční matice typu 101x101,
- soubor se 150 úkoly.

S touto sadou souborů algoritmus za 1.5 sekundy vytvořil dvě řešení, kde nejlepší bylo druhé řešení, které mělo 7 tras pro 7 řidičů. V rámci tohoto řešení zůstaly nenaplánovanými 2 úkoly, kde první úkol s nízkou prioritou a druhý úkol se střední prioritou, zatímco jiné řešení mělo 9 nenaplánovaných úkolů a skoro stejné náklady.

Číslo trasy	Naplánovaná doba trasy	Pracovní doba řidiče	Vozidlo	Počet úkolů
1	05:00 - 22:55	05:00 - 23:00	Auto	42
2	08:00 - 22:52	08:00 - 23:00	Skůtr	31
3	05:00 - 19:58	05:00 - 20:00	Auto	28
4	07:00 - 21:48	07:00 - 22:00	Auto	18
5	07:00 - 22:53	07:00 - 23:00	Skůtr	13
6	06:00 - 10:15	06:00 - 18:00	MHD	7
7	06:00 - 20:46	06:00 - 21:00	MHD	9

Tabulka 6.3: Řešení pro 150 úkolů

Kapitola 7

Závěr

Cílem této bakalářské práce bylo vytvořit algoritmus, který by automatizoval a optimalizoval proces plánování tras pro společnost Blahobyty.

Nejprve bylo provedeno seznámení s aktuálním problémem a definování požadavků zadavatele.

V další části práce byly prozkoumány podobné a již popsané problémy, do kterých patří Problém obchodního cestujícího, Problém okružních jízd a Problém okružních jízd s časovými okny.

Následně Problém okružních jízd s časovými okny byl vybrán jako nejvíc podobný, a proto na jeho základě byly sestaveny vlastní problém a matematický model vlastního problému.

V následující části práce bylo prozkoumání existujících heuristických metod, které řeší Problém okružních jízd s časovými okny. Ze všech prozkoumaných metod byla vybrána Metoda nejbližších sousedů jako základní metoda vlastního algoritmu.

V poslední části práce byl popsán a naimplementován vlastní algoritmus, který vytváří řešení v souladu s matematickým modelem problému z podkapitoly 4.2 a snaží se najít optimální řešení, která odpovídají požadavkům zadavatele z podkapitoly 2.3. Pak bylo provedeno testování algoritmu s různými vstupními daty, která byly vytvořeny spolu se zaměstnanci společnosti Blahobyty. Testování pomohlo odkalibrovat hodnoty koeficientů, které se používají při vytváření tras. Následně testování ověřilo, že algoritmus je schopen najít optimální řešení, ale je závislý na kvalitě vstupních dat.

Výsledkem celé práce je funkční algoritmus, který hledá optimální trasy a snaží se naplánovat co nejvíc úkolů v rozumném čase. V budoucnu se plánuje integrace algoritmu do informačního systému zadavatele, kde ho cílové uživatelé budou moci využívat. Po integraci dost možná, že na základě reálného používání budou provedeny další kalibrace, které pomohou určit lepší hodnoty a následně dostávat řešení, která jsou víc optimální. Doufám, že tato práce pomůže společnosti Blahobyty v budoucnu ušetřit čas a náklady spojené s plánováním tras a jejich vyplněním.

Příloha A

Literatura

[1] G.B. Dantzig and J.H. Ramser. *The Truck Dispatching Problem.* Management Science, vol. 6, 1959, no. 1, pp. 80-91.
Dostupné z: `https://www.jstor.org/stable/2627477`

[2] G. Laporte. *The traveling salesman problem: An overview of exact and approximate algorithms.* European Journal of Operational Research vol. 59, pp. 231-247, North-Holland, 1992.
Dostupné z: `https://doi.org/10.1016/0377-2217(92)90138-Y`

[3] J. Demel. *Grafy a jejich aplikace.* Academia, 2015, ISBN 978-80-260-7684-1.

[4] J.F. Cordeau, G. Laporte, M.W.P. Savelsbergh, D. Vigo. *Vehicle routing.* Handbooks in operations research and management science, vol. 14 (Transportation), 2007, pp. 367-428, Elsevier.
Dostupné z: `https://www.researchgate.net/publication/233843551`

[5] T. Carić, S. Pašagić, Z. Lanović. *Vehicle Routing Problem Models.* Promet-Traffic-Traffico, vol. 16, 2004, no. 1, pp. 59-62.
Dostupné z: `https://www.researchgate.net/publication/290029721`

[6] Z. Borcinova. *Two models of the capacitated vehicle routing problem.* Croatian Operational Research Review CRORR 8, 2017, pp. 463–469.
Dostupné z: `https://hrcak.srce.hr/ojs/index.php/crorr/article/view/4528`

[7] O. Bräysy, M. Gendreau. *Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms.* Transportation Science 39(1), pp. 104-118, 2005.
Dostupné z: `https://www.researchgate.net/publication/220413310`

[8] N.A. El-Sherbeny. *Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods.* Journal of King Saud University – Science, 2010, pp. 123-131, Elsevier.
Dostupné z: `https://doi.org/10.1016/j.jksus.2010.03.002`

[9] S.N. Kumar, R. Panneerselvam *A Survey on the Vehicle Routing Problem and Its Variants.* Intelligent Information Management, vol. 4, 2012, pp.

66-74.
Dostupné z: https://www.researchgate.net/publication/267202931

[10] P. Kozel. Úloha okružních jízd s časovými okny. Perner's Contacts, vol. 6(3), 2011, pp. 160-167.
Dostupné z: https://pernerscontacts.upce.cz/index.php/perner/article/view/861

[11] N. Kohl, O.B.G. Madsen. An Optimization Algorithm for the Vehicle Routing Problem with Time Windows Based on Lagrangian Relaxation. Operations Research, vol. 45, no. 3, 1997, pp. 395-406.
Dostupné z: https://www.jstor.org/stable/172017

[12] M. Desrochers, J. Desrosiers, M. Solomon A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows. Operations Research, vol. 40, no. 2, 1992, pp. 342-354.
Dostupné z: https://www.researchgate.net/publication/221704743

[13] M.M. Solomon. Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. Operations Research, vol. 35, no. 2, 1987, pp. 254-265.
Dostupné z: https://www.jstor.org/stable/170697

[14] G. Clarke, W. Wright. Scheduling of Vehicles from a Central Depot to A Number of Delivery Points. Operations Research, vol. 12, 1964, pp. 568-581.
Dostupné z: https://www.jstor.org/stable/167703

[15] Oracle. Overview of Java. Java developer's guide.
Dostupné z: https://docs.oracle.com/en/database/oracle/oracle-database/12.2/jjdev/Java-overview.html

[16] Oracle. Package java.util.stream. Java™ Platform, Standard Edition 8 API Specification.
Dostupné z: https://docs.oracle.com/javase/8/docs/api/java/util/stream/package-summary.html

Příloha B

Seznam zkratek

- TSP - Traveling salesman problem (Problém obchodního cestujícího)
- VRP - Vehicle routing problem (Problém okružních jízd)
- VRPTW - Vehicle routing problem with time windows (Problém okružních jízd s časovými okny)
- MHD - Městská hromadná doprava

Příloha C

Návod na spuštění

Pro spuštěni daného algoritmu je třeba mít nainstalované následující technologie:

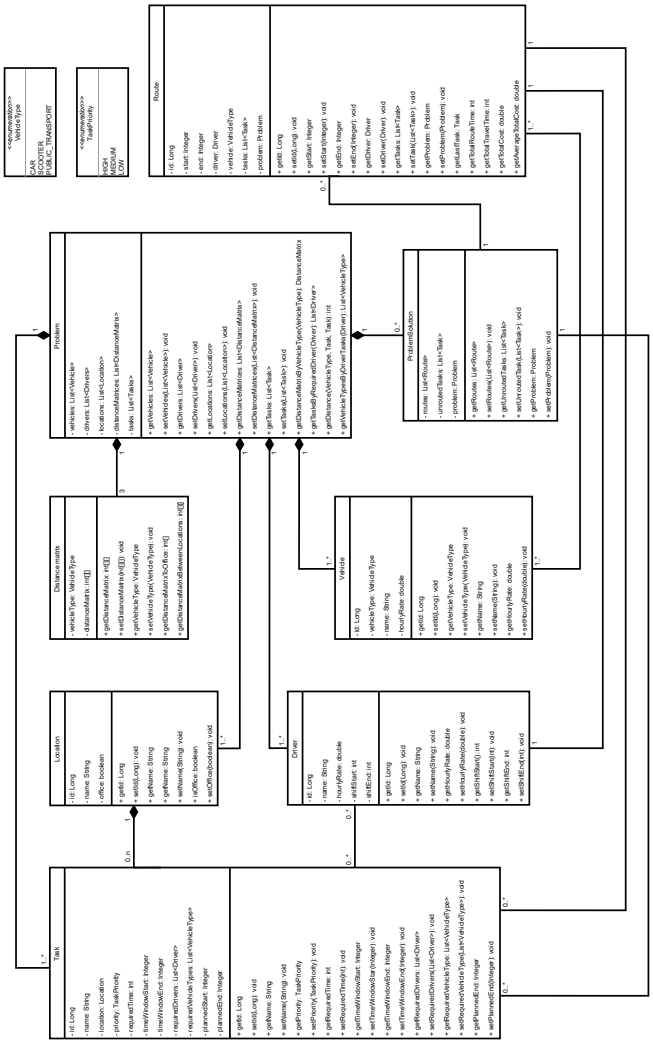
- Java (ve verzi 8 vyšší);
- Vývojové prostředí s Apache Maven (Intellij IDEA, NetBeans atd.).

Postup spuštění algoritmu:

1. Otevřete projekt ve vývojovém prostředí
2. Spusťte třídu **Main**
3. V konzoli vyberte sadu dat, kterou chcete otestovat.
4. Dostaňte možná řešení do konzole.

Příloha D

Diagram tříd



Obrázek D.1: Diagram tříd

Příloha E

Výstupy autora

1. Vývojový diagram algoritmu: *flowchart.png*
2. Diagram tříd: *class_diagram.png*
3. Zdrojový kód algoritmu: *routeplanner.zip*
4. Veřejný repozitář se zdrojovým kódem algoritmu a s testovacími daty je uložen na GitLab (online): `https://gitlab.com/imakhambet/supply-route-planning`