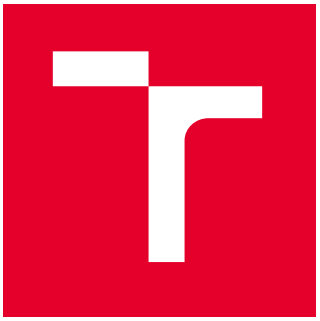


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY

INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

OPTIMALIZACE REDISTRIBUCE SDÍLENÝCH KOL MEZI STANICEMI

OPTIMIZATION OF BIKE REDISTRIBUTION BETWEEN STATIONS

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — **Bc. Ladislav Řezáč**

AUTHOR

VEDOUCÍ PRÁCE — **Ing. Vlastimír Nevrlý, Ph.D.**

SUPERVISOR

BRNO 2025

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav automatizace a informatiky
Student:	**Bc. Ladislav Řezáč**
Studijní program:	Aplikovaná informatika a řízení
Studijní obor:	bez specializace
Vedoucí práce:	**Ing. Vlastimír Nevrlý, Ph.D.**
Akademický rok:	2024/25

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Optimalizace redistribuce sdílených kol mezi stanicemi

Stručná charakteristika problematiky úkolu:

Tato diplomová práce se zaměří na vývoj a implementaci optimalizačního modelu pro rozvoz kol mezi stanicemi systému sdílených kol. Hlavním cílem práce bude efektivní plánování tras pro jednoho nebo více řidičů, přičemž budou zohledněna časová okna, omezení pracovní doby, kapacita vozidel a dva typy nákladu: rozbitá a funkční kola. Součástí práce bude také předpříprava dat, která zahrnuje kontrolu stavu stanic a kol, která se na nich nachází, a jejich vhodné zpracování před vstupem do modelu. Optimalizační model bude implementován v jazyce Python s využitím knihovny OR–Tools. Práce se zaměří na návrh a implementaci flexibilního modelu, který bude zahrnovat různé varianty reálných úloh a zohledňovat rozličné okrajové podmínky. Práce bude zahrnovat také testování funkčnosti na různých instancích. Práce navazuje na úspěšnou meziústavní spolupráci Ústavu automatizace a informatiky a Ústavu procesního inženýrství v rámci závěrečných prací na FSI VUT v Brně.

Cíle diplomové práce:

1. Seznámit se s matematickými modely svozových úloh.
2. Zpracovat základní požadavky pro redistribuci sdílených kol: definice úlohy a její matematická formulace.
3. Implementace optimalizačního modelu pomocí jazyka Python a knihovny OR–Tools.
4. Návrh modulu pro automatickou přípravu dat a vyhodnocení stavu stanic a kol.
5. Testování a ladění modelu na různých instancích.

Seznam doporučené literatury:

[1] GHIANI, G.; LAPORTE, G.; MUSMANNO, R. Introduction to Logistics systems planning and control. Wiley-interscience series in systems and optimization, John Wiley & Sons, Chichester, 2004.

[2] WILLIAMS, H. P. Model Building in Mathematical Programming. London School of Economics, UK: Wiley, 5th ed., 432, 2013. ISBN 978-1-118-44333-0.

[3] VANDERBEI, R. J. Linear Programming: Foundations and Extensions. International Series in Operations Research & Management Science, Springer, 2007.

[4] OR-Tools CP-SAT v9.11. Laurent Perron and Frédéric Didier. Dostupné z: https://developers.google.com/optimization/cp/cp_solver

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2024/25

V Brně, dne

L. S.

doc. Ing. Zdeněk Hadaš, Ph.D. ředitel ústavu	doc. Ing. Jiří Hlinka, Ph.D. děkan fakulty

ABSTRAKT

Tato diplomová práce se zabývá návrhem a implementací optimalizačního modelu pro redistribuci sdílených kol mezi stanicemi s využitím reálných dat společnosti Nextbike. Cílem je zefektivnit plánování tras servisních techniků tak, aby byly zohledněny kapacity vozidel, časová okna, role zaměstnanců, typy nákladu (funkční vs. poškozená kola) a specifické okrajové podmínky. Práce popisuje teoretický rámec kombinatorické optimalizace, zejména varianty problému rozvozu vozidel (VRP), a podrobně analyzuje jednotlivé přístupy včetně Rich VRP. Pro řešení je využita knihovna OR-Tools a programovací jazyk Python. Výsledný model je integrován do webové aplikace rozšířené o prediktivní modul, grafické rozhraní a nástroje pro přípravu a validaci dat. Výsledky testování na datech z města Brna potvrzují schopnost modelu generovat smysluplné trasy s vysokou úrovní realismu a flexibilitou vůči požadavkům provozu. Práce přispívá k rozvoji inteligentního plánování v oblasti městské mobility a logistiky.

ABSTRACT

This thesis focuses on the design and implementation of an optimization model for the redistribution of shared bicycles between stations, using real data from the company Nextbike. The goal is to streamline the route planning of service technicians while considering vehicle capacities, time windows, employee roles, types of cargo (functional vs. damaged bicycles), and specific boundary conditions. The work outlines the theoretical framework of combinatorial optimization, particularly variants of the Vehicle Routing Problem (VRP), and provides a detailed analysis of various approaches, including Rich VRP. The solution is implemented using the OR-Tools library and the Python programming language. The resulting model is integrated into a web application extended with a predictive module, graphical interface, and tools for data preparation and validation. Testing results on data from the city of Brno confirm the model's ability to generate meaningful routes with a high level of realism and flexibility with respect to operational demands. The thesis contributes to the advancement of intelligent planning in the field of urban mobility and logistics.

KLÍČOVÁ SLOVA

Optimalizace rozvozu, sdílená kola, Vehicle Routing Problem, OR-Tools

KEYWORDS

Routing optimization, bike-sharing, Vehicle Routing Problem, OR-Tools

ÚSTAV AUTOMATIZACE A INFORMATIKY

2025

BIBLIOGRAFICKÁ CITACE

ŘEZÁČ, Ladislav. *Optimalizace redistribuce sdílených kol mezi stanicemi*. Brno, 2025. Dostupné také z: `https://www.vut.cz/studenti/zav-prace/detail/165746`. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav automatizace a informatiky, Vedoucí práce: Ing. Vlastimír Nevrlý, Ph.D.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že tato práce je mým původním dílem, vypracoval jsem ji samostatně pod vedením vedoucího práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury.

Jako autor uvedené práce dále prohlašuji, že v souvislosti s vytvořením této práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků.

V Brně dne 23. 5. 2025 .....

Ladislav Řezáč

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu práce Ing. Vlastimíru Nevrlému, Ph.D. za odborné vedení, trpělivost a pomoc při řešení všech částí tohoto zadání.

Mé poděkování dále patří doc. Ing. Radovanu Šomplákovi, Ph.D. za cenné rady při směřování práce a za možnost se na projektu aktivně podílet.

Děkuji také celému vývojovému týmu, konkrétně tedy Bc. Adéle Ondrouchové, Bc. Andriimu Semeniukovi a Bc. Jakubu Hadašovi, za spolupráci a společné úsilí. Za technickou podporu při práci s knihovnou OR-Tools děkuji Ing. Lucii Němcové.

Velké díky patří také mé rodině a přátelům, kteří mi byli oporou během celého studia i při tvorbě této práce.

Závěrem bych rád poděkoval také celému projektu TAČR Doprava 2030 CL01000161 Optimalizace infrastruktury a provozu mikromobility ve městech ČR, díky němuž mohla tato práce vzniknout.

OBSAH

1 Úvod

V posledních letech lze v městských aglomeracích po celém světě pozorovat rostoucí důraz na rozvoj udržitelných forem dopravy. Jedním z klíčových nástrojů v této oblasti se stala služba sdílených kol, která nabízí uživatelům flexibilní, ekologickou a zpravidla cenově dostupnou alternativu k individuální automobilové dopravě. Její úspěch je však podmíněn nejen zájmem veřejnosti, ale především spolehlivým provozem a efektivní logistikou zajišťující dostupnost kol ve správný čas a na správném místě.

Zajištění rovnoměrného rozložení kol mezi jednotlivými stanicemi představuje složitou logistickou úlohu. V reálném provozu totiž dochází k přirozeným disproporcím v poptávce – některé lokality zaznamenávají trvale vyšší výpůjčky, zatímco jiné naopak fungují spíše jako cílové destinace. Bez odpovídající redistribuce by se tak mohly některé stanice zcela vyprázdnit, zatímco jiné by byly přeplněné. Tento problém je o to náročnější, že vedle samotného přesunu kol je třeba řešit i další operace, jako jsou opravy, technické kontroly, přelepování kampaní nebo svoz poškozených kol do dep.

Důležitou roli v celém procesu sehrávají servisní technici, jejichž úkolem je zajišťovat veškeré výše uvedené činnosti v terénu. Ti se v průběhu dne pohybují mezi desítkami stanic, přičemž musejí respektovat svá časová omezení, specifické role a kapacity přidělených vozidel. Plánování jejich směn, tras a jednotlivých úkonů tak tvoří komplexní úlohu, kterou nelze efektivně řešit ručně – zejména pokud má být zohledněna dynamika provozu a reálná data ze systému.

Ačkoliv již existuje řada teoretických přístupů k rozvozovým problémům, aplikace těchto modelů do prostředí sdílených kol vyžaduje úpravy a přizpůsobení konkrétním podmínkám provozovatele. Mezi ty patří například práce s více typy nákladu (funkční vs. rozbitá kola), specifikace úkonů na stanicích, časová okna, více směn, role techniků, otevřené i uzavřené trasy nebo možnost predikce poptávky.

Tato diplomová práce si klade za cíl navrhnout a implementovat flexibilní optimalizační modul, který bude schopen zpracovat výše zmíněné okrajové podmínky a poskytnout kvalitní návrh směn a tras pro jednoho i více techniků zároveň. Důraz je přitom kladen nejen na kvalitu výsledného plánu, ale i na jeho praktickou využitelnost v rámci širšího softwarového řešení pro společnost nextbike Czech Republic s.r.o., které bylo v průběhu projektu postupně rozšiřováno o další komponenty – frontendové rozhraní, databázové služby, prediktivní modul či vizualizační nástroje.

Součástí práce je podrobná analýza teoretických základů plánování tras, popis jednotlivých variant úloh VRP, jejich matematická formulace a praktická implementace v jazyce Python za využití knihovny Google OR-Tools. V další části jsou popsány jednotlivé komponenty technického řešení, přístup k přípravě vstupních dat, návrh datového modelu a implementace výpočetního jádra (Enginu). Závěr práce pak tvoří vyhodnocení výsledků na reálných datech z města Brna a úvahy o možnostech dalšího rozšíření systému.

2 Motivace a zadání projektu

Sdílená kola jsou v dnešní době stále častěji užívaným typem městské dopravy, která se po světě v posledních letech značně zpopularizovala. Klíčovým faktorem úspěchu je kromě individuální přepravy uživatelů, ekologického přínosu a svobody pohybu také její dostupnost. Právě tato dostupnost se od počátků této služby stala důležitým prvkem její oblíbenosti.

Zajištění dostupnosti kol však vyžaduje rozsáhlou logistickou a servisní podporu. Kromě pravidelné údržby je potřeba zajišťovat také jejich přesun mezi stanicemi dle aktuální poptávky, provádět opravy, nebo další operace nezbytné pro hladký chod služby. Tyto činnosti jsou většinou časově i finančně náročné, kladou vysoké nároky na plánování a koordinaci pracovníků v terénu.

Přestože existuje široká škála modelů a přístupů pro řešení těchto úloh, jejichž vývoj se v posledních letech značně zintenzivnil, odlišnosti mezi konkrétními implementacemi mohou být značné – zejména kvůli rozdílným obchodním strategiím jednotlivých poskytovatelů služeb. Z tohoto důvodu si většina firem vytváří vlastní modely, které lépe odpovídají jejich specifickým potřebám a způsobu fungování.

Tato diplomová práce vznikla jako součást širšího projektu realizovaného pro společnost nextbike, jehož cílem bylo rozšíření stávajícího softwarového řešení o modul pro plánování úkonů v rámci směn zaměstnanců, který by umožnil efektivnější organizaci provozu v reálném čase.

V rámci projektu vzniklo několik samostatných komponent, které se navzájem doplňují a společně tvoří výslednou aplikaci. Mezi ně patří například rozšíření vizuálního rozhraní, vstupní a výstupní formuláře pro zadávání okrajových podmínek směn jednotlivých zaměstnanců, predikční modul pro odhad poptávky po výpůjčkách, nebo plánovací výpočetní modul optimalizující rozvrh a trasy servisních techniků.

Právě návrh a implementace plánovacího modulu je hlavním cílem práce. Rozsah vlastního řešení se pak vymezuje na zpracování vstupních dat získaných od uživatele a z databáze, jejich úpravu pro vstup do výpočetního jádra, samotný výpočet optimálního rozvrhu a tras, a následné formátování výstupu pro jeho další využití v databázi spolu s front-end rozhraním aplikace.

2.1 O společnosti nextbike

Nextbike je mezinárodní společnost, která od roku 2004 poskytuje služby sdílených kol. Původně vznikla v Německu, ale během svého vývoje expandovala do více než 20 zemí a aktuálně působí ve více než 300 městech po celém světě. [1]

V České republice působí od roku 2018 prostřednictvím společnosti nextbike Czech Republic, jejíž licenci získali dva dlouholetí kamarádi z hokeje, Lukáš Luňák a Tomáš

Karpov. V současnosti nextbike provozuje v ČR přes 7 500 kol ve více než 40 městech, z toho v Brně, na jehož datech je tato práce testována, je v provozu více než 1 000 kol rozprostřených přibližně na 300 odběrových místech [2].

Vzhledem k rozsahu této sítě a neustálé dynamice poptávek po kolech je efektivní plánování servisních akcí a směn nad rámec schopností ruční organizace. Proto bylo nutné využít automatizovaných nástrojů a algoritmů, které by byly schopny poskytnout v reálném čase řešení těchto úloh.

Obr. 1: Mapa měst, kde aktuálně nextbike Czech Republic působí (převzato z [2])

2.2 Zadání projektu a specifikace požadavků

Původní zadání projektu bylo formulováno velmi volně. Jediným pevným požadavkem bylo vytvořit rozšíření stávající aplikace, které by umožnilo uživatelům plánovat směny jak pro sebe, tak pro své podřízené. Pro interakci s tímto rozšířením bylo plánováno využití čtyř uživatelských rozhraní – mapy a vstupního formuláře pro nastavení směn, a dále navigace s výstupním formulářem pro zobrazení navržené trasy.

Výpočetní modul, který je jádrem této práce, musí být schopen zohlednit různé kombinace vstupních parametrů a omezení. Mezi nejvýznamnější patří:

- **Počet zaměstnanců** – plánování může probíhat pro jednoho nebo více techniků současně.

- **Povinné stanice** – uživatel může určit stanice, které musí být během směny navštíveny. Přesněji povinně volitelné, vázané přímo pro daného technika, a povinně volné, možné navštívit libovolným z řidičů v případech, kdy je zadán jejich výpočet současně.
- **Volné stanice** - všechna místa, která nejsou zahrnuta v povinných stanicích, ale mohou být uvažována v řešení.
- **Zakázané stanice** – taktéž lze určit místa, která se nesmí během směny v seznamu úkonů objevit.
- **Časová okna** – každému zaměstnanci mohou být nastavena různá pracovní období včetně začátku, konce a případných pauz.
- **Role techniků** – jednotliví technici mohou mít odlišné role (opravy, kontrola, redistribuce, kampaně, svoz poškozených kol) s různými kombinacemi vykonávaných činností.
- **Typ vozidla a jeho kapacita** – k jednotlivým technikům musí existovat možnost přiřadit vozidlo,se kterým budou absolvovat směnu. Mezi jednotlivými typy vozidel jsou rozdíly v rychlostech, místech možných průjezdů, kapacitách, jejichž hodnota se nastavuje na počátku trasy dle požadavků (např. předem naložená kola z depa).
- **Počáteční a koncová pozice** – lze nastavit, kde a kdy má technik směnu začít či skončit.
- **Požadovaný stav nákladu na konci směny** – standardně je požadováno, aby vozidlo ukončilo směnu bez nákladu, ale toto omezení lze zrušit podle aktuálních potřeb.

Výše uvedené specifikace představují základ množiny okrajových podmínek, se kterými se výpočetní modul snaží pracovat. Seznam všech těchto nastavení je pak doplněn a řádně popsán v kapitole 6.

3 Optimalizace plánování tras

Plánování tras a jejich optimalizace patří mezi klíčové oblasti kombinatorické optimalizace. Tyto metody nacházejí široké uplatnění v různých praktických oblastech – od logistiky a dopravy, přes plánování výrobních procesů, až po služby a městskou mobilitu. V jádru této problematiky často stojí tzv. Problém obchodního cestujícího (TSP, *Travelling salesman problem*, 3.1), jehož kořeny sahají až do počátku 19. století [3]. Ačkoli jeho původ není plně znám, vědecký svět si získal převážně ve 30. letech 20. století, kdy byl oficiálně definován a rozběhly se první pokusy o jeho řešení. Tento vývoj představoval významný impuls pro vznik a rozvoj celé oblasti výpočetní optimalizace.

TSP je příkladem úplného NP-těžkého problému, u kterého se stává nalezení výsledku výpočetně neúnosným při rostoucím počtu míst, které je třeba navštívit. Tento fakt vedl k vývoji široké škály metod a přístupů pro jeho řešení; od exaktních algoritmů, jako je metoda *Branch and Bound*, nacházející optimální řešení systematickým procházením možných cest a efektivním prořezáváním nevyhovujících větví problému [4], až po různé heuristiky a metaheuristiky, mezi které lze zařadit simulované žíhání (*simulated anneahealing*, SA) nebo algoritmy inspirované přírodou, například genetický algoritmus (*genetic algorithm*, GA). Přes svou relativní jednoduchost si TSP stále drží pozici jednoho z nejvíce zkoumaných problémů v oblasti operačního výzkumu. [5]

Reálné aplikace však často vyžadují mnohem komplexnější přístup než základní scénář TSP. Proto vznikla řada jeho rozšíření a zobecnění, která zohledňují kupříkladu více vozidel, omezenou kapacitu, specifické požadavky zákazníků, více výchozích bodů, časová okna, různé varianty požadovaných služeb. Tyto obecnější úlohy jsou známé jako *Vehicle Routing Problem* (VRP, viz 3.2), tedy problém plánování tras vozidel. Stejně jako TSP i VRP patří mezi NP-těžké problémy, a proto je jejich kvalitní a efektivní řešení dnes nezbytné pro moderní logistické systémy, městskou dopravu a řadu dalších odvětví.

Cílem této kapitoly a jejích podkapitol je představit vývoj a jednotlivé varianty úloh z oblasti TSP a VRP. Nejprve je podrobně popsán klasický problém obchodního cestujícího, včetně jeho formálního zápisu a základních vlastností. Následně jsou představeny jeho rozšířené verze, jako například mTSP, a dále problematika VRP, která je v práci popsána nejpodrobněji. VRP totiž představuje klíčovou úlohu, na kterou navazují další prvky a rozšíření.

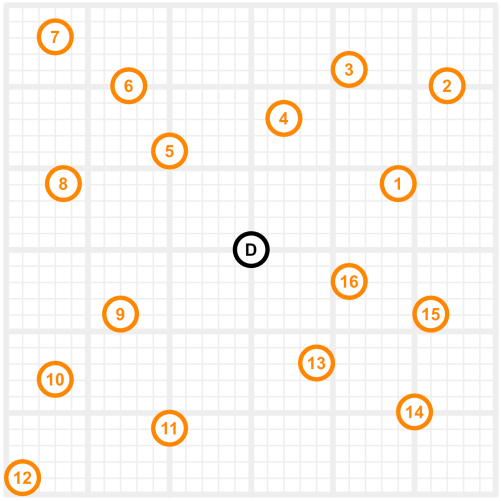
3.1 Travelling Salesman Problem (TSP)

Problém obchodního cestujícího (TSP) je jednou z nejznámějších úloh kombinatorické optimalizace, jejímž cílem je nalézt optimální pořadí návštěvy všech zadaných uzlů tak, aby byl každý navštíven právě jednou, a po dokončení cesty se vozidlo vrátilo do výchozího

bodu. Optimalizací se zde rozumí minimalizace celkové „ceny“ trasy, která může být vyjádřena například vzdáleností, časem nebo jinými náklady.

Problém lze formálně definovat na neorientovaném grafu $G = (V, E)$ jehož příkladem může být právě (Obr. 2), kde:

- $V = \{0, 1, .., n\}$ je množina uzlů, ve které nultý index obsahuje počáteční stanici, tedy depo D,
- $E = \{(i, j) : i, j \in V, i < j)\}$ je množina všech hran spojující uzly (v Obr. 2 jsou vzdálenosti reprezentovány sítí s rozměry 30×30 jednotek).



Obr. 2: Neorientovaný graf - pro čitelnost bez hran

Matematicky může být úloha popsána:

$$\min \quad \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{1}$$

$$\text{subject to} \quad \sum_{i \in V, i \neq k} x_{ik} + \sum_{j \in V, j \neq k} x_{kj} = 2 \qquad (k \in N) \tag{2}$$

$$\sum_{i,j \in S} x_{ij} \leq |S| - 1 \qquad (S \subset N, 2 \leq |S| \leq n - 1) \tag{3}$$

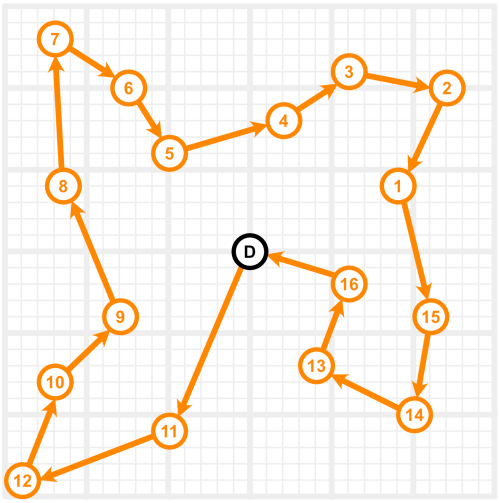
$$x_{ij} \in \{0, 1\} \qquad ((i, j) \in E) \tag{4}$$

Jednotlivé rovnice a omezení lze popsat následovně:

- (1) **Účelová funkce (Objective function)** je definována tak, že c_{ij} představuje cenu (například vzdálenost nebo čas) pro přechod z vrcholu i do vrcholu j.
- (2) **Omezení stupňů (Degree constraints)** zajišťuje, že každý vrchol má v řešení právě dvě hrany — jednu vstupní a jednu výstupní, což odpovídá podmínce existence Hamiltonovského okruhu [6].

- (3) **Omezení eliminace podcyklů (Subtour elimination)** zabraňuje vzniku podcyklů, které by mohly vzniknout rozdělením množiny vrcholů na podmnožiny S, čímž by se znemožnilo vytvoření jediné uzavřené trasy pokrývající všechny vrcholy. Množina S představuje libovolnou vlastní podmnožinu vrcholů grafu, u které je existence podcyklu nežádoucí.
- (4) **Omezení integrity proměnných (Integrality constraints)** definuje proměnnou x_{ij} jako binární, kde hodnota 1 znamená zařazení hrany do výsledného řešení a hodnota 0 její vyloučení.

Výsledek problému TSP z grafu (Obr. 2) je znázorněn na následujícím obrázku (Obr. 3). V tomto případě je cesta považována za optimální v obou směrech, jelikož byl graf definován jako neorientovaný. Směr šipek je proto pouze ilustrativní a ukazuje jednu z možných variant průchodu vrcholy.



Obr. 3: Výsledek úlohy TSP

TSP se může vztahovat jak na neorientované, tak i na orientované grafy, přičemž v orientovaných grafech je směrování hran významné kvůli rozdílnému ohodnocení. Pro snazší pochopení a klasifikaci těchto variant pak bylo přijato následující značení [5]:

- **Symetrické TSP (sTSP)**, kde jsou ceny mezi dvěma vrcholy shodné v obou směrech. Graf je považován za neorientovaný a platí, že $c_{ij} = c_{ji}$ pro všechny hrany.
- **Asymetrické TSP (aTSP)**, kde se ceny hran liší v závislosti na směru průchodu. Graf je označen jako orientovaný a existuje alespoň jedna hrana, pro kterou platí $c_{ij} \neq c_{ji}$.

3.1.1 Multi Travelling Salesman Problem (mTSP)

Jednou ze základních modulací je mTSP, kde se úloha nově rozšiřuje o více cestujících, kteří mezi městy putují. Tím se však drobně mění i matematický zápis. Pro ten existuje hned několik forem, v závislosti na složitosti a dalších využití úlohy. Pro návaznost však stačí jednoduchá formulace celočíselného programování založeného na přiřazení. Účelová funkce (1), integrita proměnné (4) a eliminace podcyklů (3) zůstávají stejné, změnou však prochází zbytek omezení:

$$\sum_{j=2}^{n} x_{1j} = m, \tag{5}$$

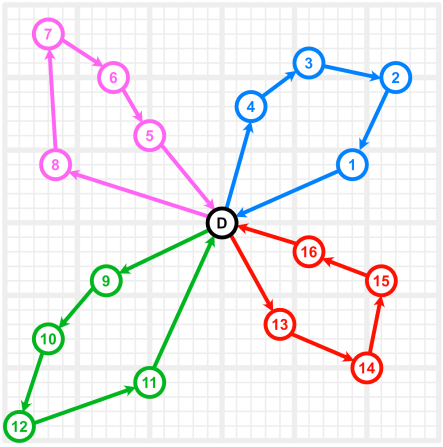
$$\sum_{i=2}^{n} x_{i1} = m, \tag{6}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \qquad (j = 2, \ldots, n) \tag{7}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad (i = 2, \ldots, n) \tag{8}$$

Podmínky (5) a (6) definují nutnost vycestování a následného navrácení shodného počtu cestujících m do počáteční stanice (depa). Kritéria (7) a (8) pak zajišťují, že každý jiný uzel je navštíven právě jednou [7].

Pro vizualizaci rozdělení tras je možné využít stejný graf jako u TSP (Obr. 2). Na rozdíl od situace, kdy byla města navštívena jedním cestujícím, jak je znázorněno na obrázku 3, je při rozdělení měst mezi více cestujících (v tomto případě čtyři) výsledkem obrázek 4. Města jsou zde podle barev přiřazena jednotlivým cestujícím. Žádné z nich není navštíveno vícekrát a každá z tras byla minimalizována tak, aby byl minimalizován i celkový součet ohodnocení.



Obr. 4: Ukázka rozdělení měst mezi více řidičů

3.2 Vehicle Routing Problem (VRP)

V případě, kdy uzly není třeba pouze navštívit, ale například do nich také přivážet zboží, je nutné zohlednit nové omezení v podobě kapacity. Takto rozšířená úloha již není označována jako *Travelling Salesman Problem* (TSP), nýbrž jako *Vehicle Routing Problem* (VRP), a může být řešena jak pro jedno, tak i více vozidel.

Typickým příkladem tohoto typu úlohy je rozvoz zboží mezi zákazníky realizovaný za využití homogenního vozového parku. V tomto případě již cílem není pouze návštěva všech zákazníků, ale zároveň i minimalizace počtu nasazených vozidel při současném splnění všech požadavků na rozvoz zboží. To znamená, že musí být respektována také kapacita, která je k jednotlivým vozidlům přiřazena. [8]
Formulace základní úlohy VRP se matematicky definuje:

$$\min \quad \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} c_{ij} x_{ij}^{k} + \sum_{k=1}^{m} f_k y_k \tag{9}$$

$$\text{subject to} \quad \sum_{j=2}^{n} x_{1j}^{k} = m, \tag{10}$$

$$\sum_{i=2}^{n} x_{i1}^{k} = m, \tag{11}$$

$$\sum_{j=1, j\neq i}^{n} x_{ij}^{k} = 1, \qquad i = 2, \ldots, n, \tag{12}$$

$$\sum_{i=1, i\neq j}^{n} x_{ij}^{k} = 1, \qquad j = 2, \ldots, n, \tag{13}$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij}^{k} \leq |S| - 1, \qquad S \subset V, 3 \leq |S| \leq n - 3, \tag{14}$$

$$m_L \leq m \leq m_U, \tag{15}$$

$$x_{ij}^{k} \in \{0, 1\}, \qquad \forall (i, j) \in E,\ \forall k \in K \tag{16}$$

$$y_k \in \{0, 1\}, \qquad \forall k \in K. \tag{17}$$

$$\sum_{j=1}^{n} x_j^k \leq M y_k, \qquad \forall k \in K \tag{18}$$

Oproti TSP (3.1) se v modelu nachází několik rozšíření. Účelová funkce (9) nově obsahuje člen $\sum_{k=1}^{m} f_k y_k$, kde f_k značí fixní náklad za aktivaci vozidla (pokud je uvažován). Binární proměnná y_k (17) nabývá hodnoty 1, pokud je vozidlo k, z množiny všech vozidel $K = \{1, \ldots, m\}$, skutečně využito, jinak 0. Tato proměnná je svázána s pohybem vozidla vazbou (18), která zajišťuje, že pokud není vozidlo aktivní ($y_k = 0$), nemůže být použito na žádné trase. Zároveň umožňuje optimalizaci rozhodnout, kolik vozidel bude ve výsledném řešení skutečně využito. Odpovídající omezení (15) umožňuje pracovat s variabilním počtem vozidel v rámci intervalu $[m_L, m_U]$, kde m_L značí minimální a m_U maximální počet dostupných vozidel.

Mezi mezními hodnotami počtu vozidel mohou panovat odlišné vztahy. V případě, že se dolní a horní mez nerovnají, tedy $m_L \neq m_U$, hovoří se o flexibilním počtu vozidel. V takto definované úloze je cílem nalézt takovou hodnotu m, při níž jsou požadavky všech zákazníků splněny co nejefektivněji. Pokud však platí rovnost $m_L = m_U$, přechází se k tzv. fixní variantě problému. Hodnota m je pak předem daná a její volba bývá ovlivněna různými faktory, které závisí na konkrétních požadavcích řešené úlohy.

- **Kapacita vozidel (Capacitated VRP – CVRP)**, kde má každé vozidlo přidělenou maximální kapacitu Q_k a jednotliví zákazníci poptávku d_i. V takovém případě musí být počet vozidel m zvolen tak, aby splňoval následující podmínku:

$$m \geq \left\lceil \frac{\sum_{i=1}^{n} d_i}{Q_k} \right\rceil \tag{19}$$

- **Maximální pracovní doba nebo vzdálenost**, pokud je nutné nastavit horní hranici na dobu jízdy nebo celkovou ujetou vzdálenost jednoho vozidla. Do výpočtu vstupuje maximální povolená hodnota $T_{\max}$, která omezuje celkový čas, jež může jedno vozidlo strávit jízdou a obsluhou zákazníků. Zahrnujeme jak časy potřebné na obsluhu jednotlivých zákazníků t_i, tak odhad minimálního času přejezdů mezi nimi, označený jako $\mathrm{MTS}(G)$ (např. součet hran v minimální kostře grafu zákazníků). Odhad minimálního počtu potřebných vozidel pak určíme vztahem:

$$m \geq \left\lceil \frac{\sum_{i=1}^{n} t_i + \mathrm{MTS}(G)}{T_{\max}} \right\rceil \tag{20}$$

- **Počet zákazníků na trasu**, který omezuje maximální počet zákazníků, jež může jedno vozidlo obsloužit během své trasy (např. kvůli počtu zastávek nebo organizačním omezením). Pokud n značí celkový počet zákazníků a n_{max} maximální počet zákazníků na jedno vozidlo, pak dolní mez počtu vozidel určuje následující vztah:

$$m \geq \left\lceil \frac{n}{n_{max}} \right\rceil \tag{21}$$

Uvedená omezení slouží především k odhadnutí dolní hranice potřebného počtu vozidel. Reálný počet použitých vozidel se následně určuje až během řešení úlohy na základě konkrétních podmínek. Zbytek omezení i proměnných takových úloh zůstává beze změny, není tedy třeba je procházet. [9]

Jak bylo popsáno v kapitole TSP (3.1), existuje několik způsobů zobecnění úloh plánování cest mezi stanicemi. Jejich definice se ve formulaci pro VRP a TSP zásadně nemění. Z tohoto důvodu jsou následující zobecnění popsána pouze pro případ VRP.

3.2.1 Multi-Depot VRP (MDVRP)

V praxi však nemusí být podmínkou, že by všechna vozidla začínala a končila své trasy ve stejném místě. Zatímco základní varianta VRP předpokládá jedno společné depo pro všechna vozidla, úloha MDVRP (*Multi-Depot Vehicle Routing Problem*) se zabývá situací, kdy existuje více výchozích a cílových stanic. Tato definice lépe odráží realitu distribučních sítí, kde je pro pokrytí větší oblasti zboží rozváženo z několika samostatných skladovacích míst.

Stejně jako v předchozích kapitolách TSP a VRP, i tato úloha je definována na neorientovaném grafu $G = (V, E)$, s jediným rozdílem, že množina V se skládá ze dvou podmnožin. Množina $D = \{d_1, \ldots, d_m\}$ reprezentuje všechna depa a množina $C = \{1, \ldots, n\}$ pak lokality zákazníků. Pro každé depo $d \in D$ je k dispozici množina vozidel K_d. Matematický model oproti VRP prochází celkem rozsáhlými změnami v indexování, omezení podcyklů za využití MTZ (14) zůstává beze změny, a proto není zbytečně opakováno:

$$\min \quad \sum_{d \in D} \sum_{k \in K_d} \sum_{(i,j) \in E} c_{ij} \cdot x_{ij}^k \tag{22}$$

$$\sum_{d \in D} y_{id} = 1 \qquad \forall i \in C \tag{23}$$

$$\sum_{k \in K_d} \sum_{j \in V} x_{ij}^k = y_{id} \qquad \forall i \in C,\ \forall d \in D \tag{24}$$

$$\sum_{j \in V} x_{dj}^k = 1 \qquad \forall d \in D,\ \forall k \in K_d \tag{25}$$

$$\sum_{i \in V} x_{id}^k = 1 \qquad \forall d \in D,\ \forall k \in K_d \tag{26}$$

$$\sum_{j \in V} x_{ij}^k = \sum_{j \in V} x_{ji}^k \qquad \forall i \in C,\ \forall d \in D,\ \forall k \in K_d \tag{27}$$

$$x_{ij}^k \in \{0, 1\} \qquad \forall (i,j) \in E,\ \forall d \in D, \forall k \in K_d, \tag{28}$$

$$y_{id} \in \{0, 1\} \qquad \forall i \in C,\ \forall d \in D \tag{29}$$

Formulace úlohy MDVRP vychází z minimalizace celkových nákladů na přepravu mezi zákazníky a depy. Účelová funkce (22) minimalizuje součet nákladů c_{ij} po hranách $(i, j) \in E$ využitých vozidly k přiřazenými k jednotlivým depům $d \in D$.

Každý zákazník $i \in C$ musí být přiřazen právě k jednomu depu, jak popisuje rovnice (23). Toto přiřazení je reprezentováno binární proměnnou y_{id}, která nabývá hodnoty 1, pokud je zákazník obsluhován z depa d, a 0 jinak. Následně, pokud je zákazník přiřazen k nějakému depu, musí být zároveň obsloužen právě jedním vozidlem z tohoto depa, což zajišťuje rovnice (24).

Každé vozidlo $k \in K_d$ musí začít trasu výjezdem z depa d (25) a ukončit ji návratem zpět do téhož depa (26). Dále je zajištěna rovnováha toku pro všechny zákazníky pomocí rovnice (27), která říká, že počet hran vstupujících k zákazníkovi i se musí rovnat počtu hran od něj odcházejících.

Proměnné x_{ij}^k určují, zda vozidlo k projíždí hranou (i, j), a jsou definovány jako binární (28). Stejně tak přiřazení zákazníků k depům je reprezentováno binárními proměnnými y_{id} (29).

V této variantě problému přibývá rozhodovací aspekt spočívající v tom, ke kterému depu a vozidlu budou jednotliví zákazníci přiřazeni. Při řešení je tak nezbytné důsledně respektovat indexaci proměnných v rámci množin zákazníků C a dep D. [10] [11]

3.2.2 Time windows VRP (VRPTW)

Časová vytíženost nebo otevírací doby některých podniků mohou být jedním z faktorů omezujících možné časy návštěv jednotlivých uzlů. Tato skutečnost přidává k řešenému problému další dimenzi složitosti. Pro zachycení této složky reálného světa bylo zavedeno zobecnění s časovými okny. Ta jsou zpravidla reprezentována intervaly přiřazenými ke každé stanici, které vymezují časové rozmezí, během něhož musí být daný uzel obsloužen (například v minutách či sekundách od začátku cesty, kdy se počáteční okamžik označuje jako $t = 0$).

Pro definici modelu VRPTW však nedochází k zásadním změnám oproti modelu VRP (viz 3.2, rovnice 9–18). Rozšiřuje se pouze množina omezení o následující vztahy:

$$t_j \geq t_i + s_i + c_{ij} - M(1 - x_{ij}) \qquad \forall i, j \in V,\ i \neq j \tag{30}$$

$$e_i \leq t_i \leq l_i \qquad \forall i \in V \tag{31}$$

$$t_i \geq 0 \qquad \forall i \in V \tag{32}$$

kde t_i značí čas obsluhy uzlu i a řídí se podmínkou (31), v níž e_i představuje nejranější přípustný čas obsluhy a l_i nejpozdější možný čas. Konstanta M, často označovaná jako *Big-M*, ve vztahu (30) reprezentuje dostatečně velkou hodnotu, která v případě, že hrana (i, j) není aktivní ($x_{ij} = 0$), činí danou podmínku triviálně splněnou a tedy neaktivní. Rozšíření o index vozidel k, respektive množinu vozidel K, zde není provedeno, ale jedná se o přímočaré zobecnění. [11] [12]

3.2.3 VRP with Pick up and Delivery (VRPPD)

Rozvozy nemusí probíhat pouze mezi zákazníky a depy, ale také přímo mezi zákazníky samotnými. Typickým příkladem takového plánování jsou donášky jídla, kde je třeba plánovat cesty mezi restauracemi a konečnými spotřebiteli s ohledem na správné pořadí doručení. Každý přepravní požadavek v tomto případě obsahuje informace o výchozí i konečné stanici. Řešením tedy není pouze nalezení trasy s minimálními náklady, ale také zajištění obsluhy všech objednávek v požadovaném pořadí.

Každý z dříve zmíněných úkolů představuje pár uzlů ve tvaru (p_i, d_i), kde vozidlo musí nejprve navštívit **pickup** uzel p_i a následně **delivery** uzel d_i. Pokud není uvažována kapacita vozidla, lze tuto úlohu formulovat jako rozšíření klasického VRP s následujícími rozdíly:

Množina vrcholů je nyní definována jako $V = \{0\} \cup P \cup D$, kde $\{0\}$ označuje depo, množina $P = \{p_1, \ldots, p_n\}$ obsahuje všechny uzly pro vyzvednutí (pickup) a množina $D = \{d_1, \ldots, d_n\}$ uzly pro doručení (delivery). [13] [11]

Pro zachování návštěv příslušných uzlů v rámci jednoho vozidla, obdobně jako u modelu (3.2.1), je zavedena množina vozidel $K = \{1, \ldots, m\}$, kde m je počet dostupných vozidel. Dále je definována množina zakázek $R = \{(p_i, d_i) \mid i = 1, \ldots, n\}$, která obsahuje všechny požadované páry vyzvednutí a doručení. V rámci zajištění správného pořadí návštěv uzlů jsou přidána následující omezení:

- **Precedenční podmínka**, která garantuje, že doručení může nastat pouze po předchozím vyzvednutí. Formálně platí pro každé vozidlo $k \in K$:

$$u_{p_i}^k + 1 \leq u_{d_i}^k \quad \forall (p_i, d_i) \in R, \tag{33}$$

  kde u_i^k představuje pořadí návštěvy uzlu i v rámci trasy vozidla k.
- **Rozsah pořadí uzlů** definuje platné hodnoty pořadí návštěv tak, že každý uzel je buď nakládací nebo vykládací, a pořadí návštěvy je v rozmezí od 1 do $2n$, přičemž n je počet zakázek:

$$1 \leq u_i^k \leq 2n \quad \forall i \in P \cup D, \forall k \in K. \tag{34}$$

- **Integrita proměnných** vyžaduje, aby pořadí návštěv bylo celočíselné a proměnné reprezentující výběr hran byly binární:

$$u_i^k \in \mathbb{Z}, \quad x_{ij}^k \in \{0, 1\} \quad \forall i, j \in V, \forall k \in K. \tag{35}$$

3.2.4 Time-dependent VRP (TDVRP)

Toto rozšíření klasické úlohy se snaží zachytit situace, kdy se časy přesunů mezi stanicemi mohou měnit a rychlosti přejezdů během dne nejsou konstantní. Výpočet nákladů na hranu je proto závislý na čase odjezdu z předchozí navštívené stanice. Tím *Time-dependant VRP* (TDVRP) lépe reflektuje reálné podmínky rozvozu, jako jsou dopravní špičky, noční klid či dynamické změny v infrastruktuře, které významně ovlivňují volbu nejkratších či nejrychlejších tras.

Pro takové modely je třeba získávat časová okna, která predikují očekávané penalizace v průběhu celého časového horizontu, během něhož má být cesta vykonána. Výpočet v těchto úlohách nabývá na celkové obtížnosti, jelikož cena ohodnocení hran se musí přepočítávat po každé návštěvě stanice.

Na nastavení penalizací, které budou k hranám přičteny, se volí z několika metod, přičemž výběr závisí na dostupných datech a jejich kvalitě. Mezi používané přístupy patří například:

- diskretizace časového horizontu,
- prediktivní nebo historické informace o dopravních podmínkách,

- pokročilé metody pro výpočty cestovních časů, jako jsou *piecewise* funkce nebo tabulkové metody, které umožňují modelovat proměnlivé rychlosti a časy přejezdů podle aktuálního časového intervalu či jiných parametrů. *Piecewise* funkce rozdělují časový horizont na několik segmentů s různými rychlostmi, zatímco tabulkové metody využívají předem vypočtené nebo měřené hodnoty cestovních časů pro konkrétní časové okamžiky a směry.

K běžné úloze VRP (rovnice 9–18) je přidáno omezení, které zaručuje správné načasování cesty mezi uzly s ohledem na časově závislé cestovní časy. Formálně je toto omezení vyjádřeno:

$$a_j \geq a_i + c_{ij}(a_i) - M(1 - x_{ij}) \quad \forall i, j \in V, \tag{36}$$

kde proměnné mají stejný význam jako v případě (30), s tím rozdílem, že cena hrany je nyní časově závislá a označena jako $c_{ij}(a_i)$.

V praxi se však často stává, že řidiči disponují omezenou pracovní dobou, což znamená, že není možné obsloužit všechny zákazníky. V takových případech musí model zahrnout i rozhodnutí o tom, které zákazníky obsloužit, a které vynechat, čímž se úloha stává složitější variantou VRP s omezenou kapacitou a výběrem zákazníků (tzv. VRP s výběrem neboli *Selective VRP*). Taková formulace pak zahrnuje nejen časově závislé cestovní časy, ale i rozhodovací proměnné určující výběr zákazníků, což umožňuje efektivnější plánování a reálnější využití dostupných zdrojů. [14]

3.2.5 Site-dependent VRP (SDVRP)

Site-Dependent Vehicle Routing Problem (SDVRP) představuje rozšíření klasického problému rozvozu (VRP), ve kterém je umožněna pouze omezená obsluha zákazníků jednotlivými vozidly. Každý zákazník může být přiřazen pouze k určité podmnožině vozidel na základě předem definovaných kritérií, jako jsou například technické parametry vozidla, přístupové možnosti, specifické požadavky na přepravu zboží či provozní omezení. Tento typ omezení významně zužuje rozhodovací prostor a zvyšuje složitost optimalizačního problému. Zároveň však může být tato úloha chápána jako speciální případ klasického VRP s přidanými omezeními na přiřazení zákazníků k vozidlům, což umožňuje její redukci na VRP s omezeným výběrem vozidel pro jednotlivé zákazníky.

Model SDVRP se přirozeně vyskytuje v celé řadě reálných aplikací, především v městské logistice, specializované distribuci nebo v oblastech se specifickými regulačními podmínkami. Například některé lokality mohou být nepřístupné pro větší vozidla, zatímco jiné zákazníky je možné obsloužit pouze prostřednictvím vozidel vybavených pro převoz určitého typu zboží. Zohledněním těchto faktorů dochází k přesnějšímu modelování logistického procesu, čímž se zvyšuje kvalita plánování a realizovatelnost navržených tras.

Formálně je SDVRP modelován pomocí dodatečných restrikcí, které určují, zda dané vozidlo smí obsloužit konkrétního zákazníka. Tato omezení jsou vyjádřena prostřed-

nictvím binárních rozhodovacích proměnných, jejichž hodnota je vyloučena v případě nekompatibility mezi zákazníkem a vozidlem. Začlenění těchto podmínek do matematického modelu zajišťuje dodržení provozních omezení již na úrovni optimalizační úlohy a současně umožňuje nalezení efektivnějších a realističtějších tras v souladu s konkrétními požadavky daného distribučního systému.

Oproti zjednodušenému modelu VRP (rovnice 9–18) se úloha rozšiřuje pouze o podmínku zamezující návštěvu hran, pokud vozidlo nesplňuje podmínky vrcholu na něj navázanou:

$$x_{ij}^k = 0 \quad \text{pokud} \quad k \notin K_i \text{ nebo } k \notin K_j, \tag{37}$$

kde x_{ij}^k indikuje, zda vozidlo k projíždí hranou mezi uzly i a j, a množiny K_i, K_j představují vozidla oprávněná obsloužit uzly i, resp. j.

Alternativou k přímému vyloučení návštěvy zákazníka nekompatibilním vozidlem je úprava cenové matice tak, že se cena (nebo čas) spojený s návštěvou takového zákazníka tímto vozidlem výrazně zvýší. Tato metoda spočívá v nastavení velmi vysoké hodnoty cestovních nákladů nebo doby jízdy na hranách vedoucích k nekompatibilním uzlům, čímž se jejich výběr v optimalizaci stává extrémně nevýhodným. V případě, že je v úloze přítomno omezení na maximální délku pracovní doby vozidel, může toto zdražení způsobit, že cesty zahrnující nekompatibilní zákazníky budou pro dané vozidlo neproveditelné (*infeasible*). Takto se zajišťuje nepřímé respektování omezení kompatibility, přičemž model zůstává formálně jednodušší a nevyžaduje explicitní binární omezení na návštěvu zákazníků jednotlivými vozidly. Nevýhodou této metody však může být nižší přehlednost a menší kontrola nad konkrétními přiřazeními, což může vést k suboptimálním řešením v případech, kdy jsou parametry zdražení nevhodně nastaveny. [15]

3.2.6 VRP with multiple trips (MTVRP)

Multi-Trip Vehicle Routing Problem (MTVRP) je zobecněním klasického problému VRP, které umožňuje každému vozidlu během dne vykonat více cest, tzv. tripů. Tripy jsou definovány jako samostatné cesty, kdy vozidlo může opakovaně vycestovat z depa (výchozího místa) a po absolvování trasy se k němu vrátit, a to více než jednou za den. Tento přístup lépe odpovídá reálným provozním podmínkám svozu odpadu či dalších oblastech, kde je třeba provádět opakované výjezdy v rámci jedné lokality.

Tento model je pak převážně užitečný v situacích, kdy přepravní kapacita vozidla není dostatečně velká vůči celkové poptávce, ale jeho provoz není omezen pouze na jedinou cestu. Umožněním vícenásobných tripů lze navíc výrazně snížit velikost vozového parku, a to i přes zvýšené náklady spojené s častějšími návraty do výchozích stanic. [16]

Model a jeho notace pak vypadá takto:

$$\min \quad \sum_{k\in K}\sum_{t\in T}\sum_{(i,j)\in E} c_{ij}\cdot x_{ij}^{kt} \tag{38}$$

$$\text{s.t.} \quad \sum_{k\in K}\sum_{t\in T}\sum_{j\in V, j\neq i} x_{ij}^{kt} = 1 \qquad \forall i \in C \tag{39}$$

$$\sum_{j\in V, j\neq i} x_{ij}^{kt} = \sum_{j\in V, j\neq i} x_{ji}^{kt} \qquad \forall i \in V,\ \forall k \in K,\ \forall t \in T \tag{40}$$

$$\sum_{j\in V, j\neq 0} x_{0j}^{kt} \leq 1 \qquad \forall k \in K,\ \forall t \in T \tag{41}$$

$$\sum_{i\in V, i\neq 0} x_{i0}^{kt} \leq 1 \qquad \forall k \in K,\ \forall t \in T \tag{42}$$

$$u_i - u_j + n\cdot \sum_{k\in K}\sum_{t\in T} x_{ij}^{kt} \leq n-1 \qquad \forall i,j \in C,\ i \neq j \tag{43}$$

$$1 \leq u_i \leq n \qquad \forall i \in C \tag{44}$$

$$x_{ij}^{kt} \in \{0,1\} \qquad \forall (i,j) \in E,\ \forall k \in K,\ \forall t \in T \tag{45}$$

$$u_i \in \mathbb{Z} \qquad \forall i \in C \tag{46}$$

kde oproti klasickému VRP (kap. 3.2) nově platí:

- $T = \{1, \ldots, t\}$ reprezentuje maximální počet možných cest (tripů) pro jedno vozidlo; jde o horní odhad počtu opakovaných výjezdů z depa během plánovaného období.
- $x_{ij}^{kt} \in \{0,1\}$ je binární rozhodovací proměnná, která je rovna 1, pokud vozidlo k v rámci svého t-tého tripu projíždí hranu mezi uzly i a j, jinak 0.
- u_i je pomocná celočíselná proměnná, která slouží k eliminaci podcyklů pomocí tzv. MTZ formulace. Její hodnota vyjadřuje pořadí, v jakém je uzel i navštíven.
- n označuje počet zákaznických uzlů, tedy velikost množiny C, a je využívána v podmínce pro eliminaci podcyklů.

3.2.7 Heterogeneous Fleet VRP (HFVRP)

V klasickém VRP je vozový park homogenní, tedy všechna vozidla jsou identická, se stejnou rychlostí, kapacitou, spotřebou, dostupností míst a cest nebo dalším z řady parametrů. Ve skutečnosti se však tyto vlastnosti mezi vozidly ve flotile často liší. Heterogenní flotilové VRP (HFVRP) se tak více přibližuje k realitě v logistických sítích, kde je k dispozici více různých typů vozidel a jejich přiřazení k trasám a zákazníkům může výrazně ovlivnit efektivitu a celkové náklady rozvozu. Jeho řešení je však často náročnější než klasické VRP. [11]

Při tomto zobecnění prochází změnou v rámci VRP jen pár omezení a také účelová funkce, zbytek úlohy (rovnice 10–18) zůstává beze změny. Jejich zápis může vypadat následovně:

$$\min \quad \sum_{k \in K} \sum_{(i,j) \in E} c_{ij}^k \cdot x_{ij}^k + \sum_{k \in K} f_k \cdot z_k \tag{47}$$

$$\sum_{j \in V} q_j \cdot x_{ij}^k \leq Q_k \qquad \forall k \in K,\ \forall i \in V \tag{48}$$

Parametry a proměnné použité v modelu:

- q_j označuje poptávku (například množství zboží) zákazníka v uzlu j.
- Q_k je kapacita vozidla k, tedy maximální množství zboží, které může vozidlo během své cesty najednou přepravovat.

3.2.8 Open VRP (OVRP)

Otevřené VRP (OVRP) slouží k popsání situací, kdy počátek a konec trasy nemusí být totožný. Vozidla tak nemusí začínat ani končit trasu v rámci depa, ale mohou začít i skončit v kterékoliv z dostupných lokalit. Tento model popisuje situace, kdy je návrat do depa zbytečný nebo neefektivní. Příkladem může být jednosměrná distribuce, kdy vozidlo zůstane v cílové destinaci a řidič může pokračovat jiným způsobem.

OVRP se od klasického VRP liší především strukturou tras — místo uzavřených okruhů se jedná o otevřené cesty. Tento rozdíl ovlivňuje nejen logistickou stránku, ale i využívané algoritmy a optimalizační techniky pro řešení úlohy.

Model OVRP tak lépe odpovídá situacím, kdy není nutné či možné, aby se vozidla vracela zpět do výchozí pozice, což může vést k významným úsporám v čase a nákladech. V matematickém modelu pak dochází pouze k drobným změnám v omezeních:

$$\sum_{j \in V \setminus 0} x_{0j}^k = 1 \qquad \forall k \in K \tag{49}$$

$$\sum_{i \in V \setminus 0} x_{ij}^k - \sum_{i \in V \setminus 0} x_{ji}^k = 0 \qquad \forall k \in K,\ \forall j \in C \tag{50}$$

kde v omezení určující návrat a konec trasy (49) již nemusí splňovat podmínku návratu do depa (nahrazení za 10 a 11). Podmínka (50) pak nastavuje rovnováhu toku, tedy vozidlo, které do uzlu přijede, z něj také musí odjet (záměna za rovnice 12 a 13). Ve VRP se běžně aplikuje pro všechny uzly, avšak u OVRP se soustředí pouze na zákazníky, jelikož podmínka návratu zpět do depa už neplatí. [17]

3.2.9 VRP with Backhauls (VRPB)

VRP with Backhauls (VRPB) představuje rozšíření běžné VRP úlohy, v níž kromě doručení nákladu zákazníkům (tzv. *linehaul*) vozidla musí při návratu posbírat zboží zpět (tzv. *backhaul*). Klíčovým prvkem jsou pak v tomto modelu pořadí návštěv jednotlivých stanic. Je totiž potřeba prvně navštívit všechny linehaul stanice, a až následně na to backhaul. Toto omezení odráží praktické požadavky na logistiku a efektivitu nákladů do vozidel.

Tyto modely jsou využívány například distribučními společnostmi, které kromě dodávek zboží také sváží použité obaly nebo přepravní prostředky.

VRPB se od původní definice úlohy pro rozvoz liší především v dělení dostupných stanic z množiny V na dvě podmnožiny B (backhaul) a L (linehaul). Jedinou změnou, kterou pak prochází celý model, je přidání podmínky pro jejich postupné navštívení. To lze popsat pomocí pořadových proměnných u_i a podmínky:

$$u_i < u_j \quad \forall i \in L, j \in B \quad \text{takové, kde} \quad x_{ij}^k = 1. \tag{51}$$

Kapacitní omezení zajišťuje, že celkové množství nákladu q_i, které vozidlo k doručuje zákazníkům z podmnožiny linehaul L, nepřekročí kapacitu vozidla Q_k. V podstatě tedy hlídá, aby během cesty vozidlo nepřijalo více nákladu, než kolik unese. [18]

$$\sum_{i \in L} q_i \sum_{j \in V} x_{ij}^k \leq Q_k, \quad \forall k \in K \tag{52}$$

4 Technologie a software

V rámci celého projektu bylo využito několik klíčových technologií. Pro pochopení vývoje je vhodné stručně představit alespoň ty, které se přímo týkaly výpočetního modulu.

Za zmínku stojí především verzovací systém Git a na něm postavená platforma GitLab, které poskytly spolehlivé zázemí pro týmovou spolupráci, správu kódu, revize i průběžné zálohování. Dále pak programovací jazyk Python spolu s knihovnou OR-Tools, která tvoří základní stavební kámen výpočetního jádra celé práce.

Každá z těchto technologií je blíže popsána v následujících podkapitolách.

4.1 Git a GitLab

Git představuje moderní a široce využívaný systém pro správu verzí (VCS - *Version Control System*), který umožňuje efektivní a snadné sledování změn u souborů v průběhu času. Na rozdíl od centralizovaných systémů, jako je Subversion nebo CVS, Git uchovává celou historii projektu lokálně. Díky tomu umožňuje většinu operací provádět i bez nutnosti připojení k internetové síti.

Nejdůležitějším rozdílem oproti dalším VCS je způsob, jakým Git data ukládá. Git totiž vždy pracuje se snímky všech souborů (tzv. *snapshoty*), nikoli pouze s rozdíly mezi jednotlivými verzemi. Při každém potvrzení změn pak provádí tzv. *commit*, který vytvoří nový snímek a uloží pouze odkaz na něj.

Pro zvýšení efektivity při práci s úložištěm však Git neukládá duplikáty nezměněných souborů, ale odkazuje se na jejich předchozí verze. To z Gitu činí, za pomoci dalších velmi užitečných nástrojů, výkonný a flexibilní lokální verzovací systém.

Díky ukládání všech změn přímo na zařízení nelze při verzování s Gitem narazit na stavy, kdy by nebylo možné provádět či ukládat změny. Jelikož Git nevyužívá výhradně ukládání na servery, není nutné být připojen k síti, což zajišťuje schopnost pracovat odkudkoli bez obav o uložení posledních úprav.

Další klíčovou vlastností je i zajištění integrity pomocí kryptografických součtů (SHA-1 hashů). Do těchto hashů jsou zahrnuty informace jako čas uložení, struktura složky a další parametry, čímž je zajištěno, že nemohou být provedeny žádné nenápadné modifikace, které by zůstaly bez povšimnutí.

Práce se soubory je v systému Git zajišťována prostřednictvím tří stavů – *modified* (změněný), *staged* (připravený k uložení) a *committed* (uložený v repozitáři, který představuje lokální složku, do níž jsou změny ukládány). K těmto stavům se vztahují i tři odpovídající části projektu – *working tree* (pracovní kopie), *staging area* (připravená oblast) a *Git directory* (vlastní databáze Gitu). Tento model umožňuje velmi flexibilní, rychlý a zároveň bezpečný systém pro vývoj softwaru.

Díky všem výše uvedeným vlastnostem je Git široce využíván jak při tvorbě malých individuálních projektů, tak i v open-source komunitě či ve vývojových týmech, jako byl ten, v jehož rámci vznikal i tento projekt. [19]

GitLab představuje webovou platformu určenou pro správu a koordinaci vývoje softwaru, která je postavena na systému Git. Na rozdíl od jiných služeb, jako je třeba GitHub, nabízí GitLab prostředí pro celý vývojový cyklus softwaru. Jedná se tak o platformu, která poskytuje služby od plánování, verzování a testování, až po monitoring a zajištění bezpečnosti – bezplatně.

Tím je vytvořeno efektivní prostředí pro spolupráci více vývojářů na jediném projektu. Platforma nabízí nástroje pro správu repozitářů, větví, požadavků na jejich sloučení (tzv. *merge requests*), sledování úprav nebo tvorbu dokumentace. [20]

4.2 Python

Python je vysokoúrovňový, interpretovaný programovací jazyk, který je známý svou srozumitelnou a přehlednou syntaxí, dynamickým typováním a širokou podporou různých programovacích paradigmat – od imperativního a objektově orientovaného přístupu až po funkcionální styl. Díky své čitelnosti, jednoduchosti a rozsáhlé nabídce knihoven je Python považován za ideální volbu nejen pro rychlý vývoj aplikací, ale také pro skriptování, automatizaci úloh, analýzu dat, vědecké výpočty či optimalizační úlohy.

Pythonu je věnována aktivní komunita a rozsáhlý ekosystém rozšiřujících balíčků, díky kterým je možné tento jazyk efektivně využívat i v náročnějších aplikačních doménách. Mezi tyto oblasti patří také operační výzkum a optimalizace, kde je Python hojně používán pro modelování a řešení úloh typu rozvrhování, trasování, alokace zdrojů nebo plánování.

Následující kapitola se proto zaměřuje na knihovnu OR-Tools, která je v Pythonu snadno dostupná a umožňuje efektivní řešení kombinatorických a optimalizačních problémů pomocí moderních algoritmů, jako jsou algoritmy založené na heuristikách, lineární programování a další. Python zde slouží jako jednoduché, přehledné a flexibilní rozhraní, které umožňuje uživatelům soustředit se na samotný model problému, aniž by se museli zabývat složitostmi implementace algoritmů.

4.3 OR-Tools

OR-Tools (*Operations Research Tools*) je open-source optimalizační knihovna vyvinutá společností Google, která je určena pro řešení širokého spektra úloh z oblasti operačního výzkumu. Primárně je zaměřena na optimalizaci úloh lineárního programování, celočíselného programování (MIP), různých variant rozvozu a plánování tras (TSP a VRP),

plánování směn a dalších kombinatorických optimalizačních problémů. Pro tato zadání jsou poskytovány výkonné nástroje pro modelování a řešení.

Knihovna je implementována v programovacím jazyce C++, avšak je dostupná a pravidelně podporována i v dalších programovacích jazycích, mezi které patří C#, Java nebo Python. V případě Pythonu je OR-Tools zpřístupněna prostřednictvím balíčku *ortools*, který slouží jako wrapper obalující původní C++ implementaci, což umožňuje efektivní a pohodlné využití knihovny přímo z Pythonu.

Díky dobře zpracované dokumentaci je OR-Tools hojně využívána jak v průmyslové praxi, tak v akademické sféře. Velkou zásluhu na tom má i přehledné a snadno použitelné programátorské rozhraní (API), které vývojářům umožňuje jednoduše definovat a řešit i složité optimalizační problémy, aniž by museli znát všechny detaily vnitřního fungování knihovny, nebo psát komplexní kód od začátku.

V případě využití OR-Tools pro plánování tras, což je hlavní zaměření této práce, nabízí knihovna pokročilý modul *Routing Solver*, speciálně navržený pro řešení problémů typu TSP a VRP. Uživatelé si v něm mohou definovat vlastní parametry a funkce, například:

- **Nákladové funkce** (*Cost Functions*) – představují hodnoty vzdáleností nebo časů potřebných pro přejezdy mezi jednotlivými uzly (*nodes*). Tyto funkce určují náklady tras a slouží k optimalizaci celkové délky nebo doby jízdy.
- **Požadavky a kapacita** (*Demands, Capacity*) – definují váhové nebo objemové limity jednotlivých vozidel, včetně podpory více typů nákladu. Tím se zajišťuje, že kapacita vozidla není přiřazeným nákladem překročena.
- **Časová okna** (*Time Windows*) – vymezují intervaly, ve kterých je možné obsloužit zákazníka nebo dokončit určitou část trasy, včetně omezení délky pracovních směn vozidel.
- **Počet a startovní pozice vozidel** (*Number of vehicles, Starts*) – umožňují definovat, kolik vozidel je k dispozici, jejich výchozí (startovní) pozice a případně různé depa nebo směny, ve kterých vozidla pracují.
- **Přestávky a povinné zastávky** (*Breaks*) – jsou implementovány pomocí tzv. intervalových proměnných (*interval variables*), které umožňují plánovat povinné pauzy a další zastávky v rámci trasy vozidla.
- **Další funkce** – zahrnují například omezení na maximální délku trasy, preference návštěv, penalizace za neobsloužení některých zákazníků a další specifická pravidla potřebná pro daný problém.

Velkou výhodou je možnost nastavit vlastní omezení pomocí tzv. dimenzí (*dimensions*), které umožňují sledovat různé veličiny, jako je čas, kapacita nebo vzdálenost napříč trasami. Díky tomu je OR-Tools velmi flexibilní a dokáže řešit i složité úlohy, například redistribuci sdílených prostředků, plánování směn, svoz odpadu nebo logistiku zásobování.

4.3.1 Počáteční řešení

Pro řešení a následnou optimalizaci problému pomocí modulu Routing Solver je vždy nezbytné nejprve nalézt počáteční řešení. Tyto inicializační trasy mají výrazný vliv na kvalitu a rychlost konvergence k lepším výsledkům. Proto je klíčové pečlivě zvolit vhodnou metodu generování počátečního řešení. OR-Tools nabízí širokou škálu takových strategií, včetně možnosti automatického výběru nejvhodnějšího přístupu podle charakteru dané úlohy. Přehled dostupných technik je uveden v tabulce 1.

Strategie	**Popis**
AUTOMATIC	Automatický výběr strategie na základě detekovaných vlastností modelu.
PATH_CHEAPEST_ARC	Z počátečního uzlu se iterativně přidává hrana s nejnižším nákladem.
PATH_MOST_ CONSTRAINED_ARC	Podobné jako PATH_CHEAPEST_ARC, ale vybírá hrany podle největších omezení.
EVALUATOR_STRATEGY	Využívá uživatelsky definovanou hodnotící funkci. Nedoporučuje se pro obecné použití.
SAVINGS	Clarke-Wrightova heuristika – začíná samostatnými trasami pro každého zákazníka a následně je slučuje podle úspor.
SWEEP	Rozdělí uzly podle úhlu z centrálního bodu (depa) a každou skupinu řeší zvlášť jako TSP.
CHRISTOFIDES	Aproximační heuristika s garancí řešení do 1,5násobku optima pro metrický TSP. Nelze použít s časovými okny. [21]
ALL_UNPERFORMED	Všechny uzly jsou označeny jako neobsloužené – vhodné jen pokud lze vše vynechat.
BEST_INSERTION	Postupně přidává nejvýhodnější uzel na nejlepší pozici v trase. Vyžaduje možnost vynechání uzlů.
PARALLEL_CHEAPEST_ INSERTION	Paralelně buduje více tras (např. podle vozidel) a vkládá nejlevnější uzly.
LOCAL_CHEAPEST_ INSERTION	Buduje jednu trasu a přidává nejlevnější uzel s ohledem na přírůstek nákladů.

Strategie	Popis
GLOBAL_CHEAPEST_ARC	Globálně vybírá nejlevnější dostupnou hranu v celé instanci.
LOCAL_CHEAPEST_ARC	Podobné jako předchozí, ale vybírá z pohledu aktuální pozice.
FIRST_UNBOUND_ MIN_VALUE	Přiřazuje první volný uzel nejnižší možnou hodnotou (např. index nebo náklad) – vhodné např. pro testování.

Tab. 1: Přehled strategií počátečního řešení v OR-Tools

4.3.2 Metaheuristiky – algoritmy pro hledání optima

Metaheuristiky jsou optimalizační techniky používané pro řešení úloh, které jsou příliš rozsáhlé nebo složité na to, aby je bylo možné efektivně vyřešit exaktními metodami. V takových případech by vyhledání optimálního řešení klasickými přístupy bylo výpočetně nebo časově příliš náročné. Metaheuristiky lze obecně rozdělit do dvou základních kategorií: **populační metody** (*population-based methods*) a **metody s jedním řešením** (*single-solution methods*).

Populační algoritmy pracují s celou množinou řešení, které se v průběhu optimalizace iterativně aktualizují s cílem nalézt co nejlepší výsledek. Mezi typické představitele této skupiny patří například **genetický algoritmus** (*Genetic algorithm*, GA) nebo **optimalizace hejnem částic** (*Particle swarm optimization*, PSO).

Naopak algoritmy založené na jediném řešení operují vždy s jedním aktuálním kandidátem, který se postupně vylepšuje pomocí **lokálního prohledávání** (*local search*). Místo práce s celou populací se soustředí na inteligentní prozkoumávání okolí aktuálního řešení.

Při hledání optimálních tras v rámci OR-Tools se využívá právě tato druhá třída přístupů, tedy metaheuristiky s jedním řešením, jak je uvedeno v tabulce 2.

Metaheuristika	Popis
AUTOMATIC	Solver automaticky zvolí vhodnou heuristiku na základě charakteru úlohy.
GREEDY_DESCENT	Také známý jako horolezecký (*hill climbing*) nebo gradientní algoritmus. Vždy volí pouze zlepšující sousední řešení a pokračuje, dokud není dosaženo lokálního optima. Výhodou je rychlost, nevýhodou riziko uvíznutí v lokálním minimu. [22]

Metaheuristika	Popis
GUIDED_LOCAL_SEARCH	Rozšiřuje lokální prohledávání o penalizaci často používaných komponent, čímž zlepšuje schopnost uniknout z lokálních minim. [23]
SIMULATED_ANNEALING	Simulované žíhání napodobuje proces ochlazování kovů. Algoritmus občas přijímá i horší řešení, aby se vyhnul lokálním minimům, a postupně tuto pravděpodobnost snižuje. [24]
TABU_SEARCH	Vylepšený horolezecký algoritmus využívající paměť (tzv. tabu seznam) k zabránění návratu ke stejným řešením a úniku z lokálních minim. [25]
GENERIC_TABU_SEARCH	Obecnější a pokročilejší varianta algoritmu Tabu Search rozšířená o další možnosti konfigurace. [26]

Tab. 2: Přehled metaheuristických algoritmů v OR-Tools

4.4 Formát JSON

JSON (*JavaScript Object Notation*) – formát pro výměnu dat, který je navržený tak, aby byl snadno čitelný pro uživatele i počítač. Jedná se o textový formát založený na zápisu objektů a polí podobně jako v jazyce JavaScript, přičemž jeho využití není nijak omezeno jen pro něj. Díky své jednoduchosti a univerzálnosti se JSON stal standardem pro přenos strukturovaných dat mezi aplikacemi, zejména ve webovém prostředí.

Strukturu JSON dokumentu tvoří páry klíč–hodnota, které mohou být dále vnořeny do polí či dalších objektů. Tento způsob zápisu umožňuje snadno reprezentovat složité hierarchické datové struktury v přehledném zápisu. Výhodou je i to, že většina moderních programovacích jazyků (včetně Pythonu) poskytuje přímou podporu pro práci s JSON daty, což z něj činí velmi praktický nástroj pro ukládání a přenos informací. [27]

V rámci této práce je formát JSON využíván především jako prostředník pro ukládání a předávání informací o řešených optimalizačních úlohách, jako jsou vstupní parametry nebo výstupy instrukcí pro jednotlivé uživatele. Díky snadné serializaci a deserializaci v Pythonu je JSON ideální volbou pro efektivní propojení mezi datovým modelem, výpočetní částí založenou na OR-Tools a vizualizačním uživatelským rozhraním.

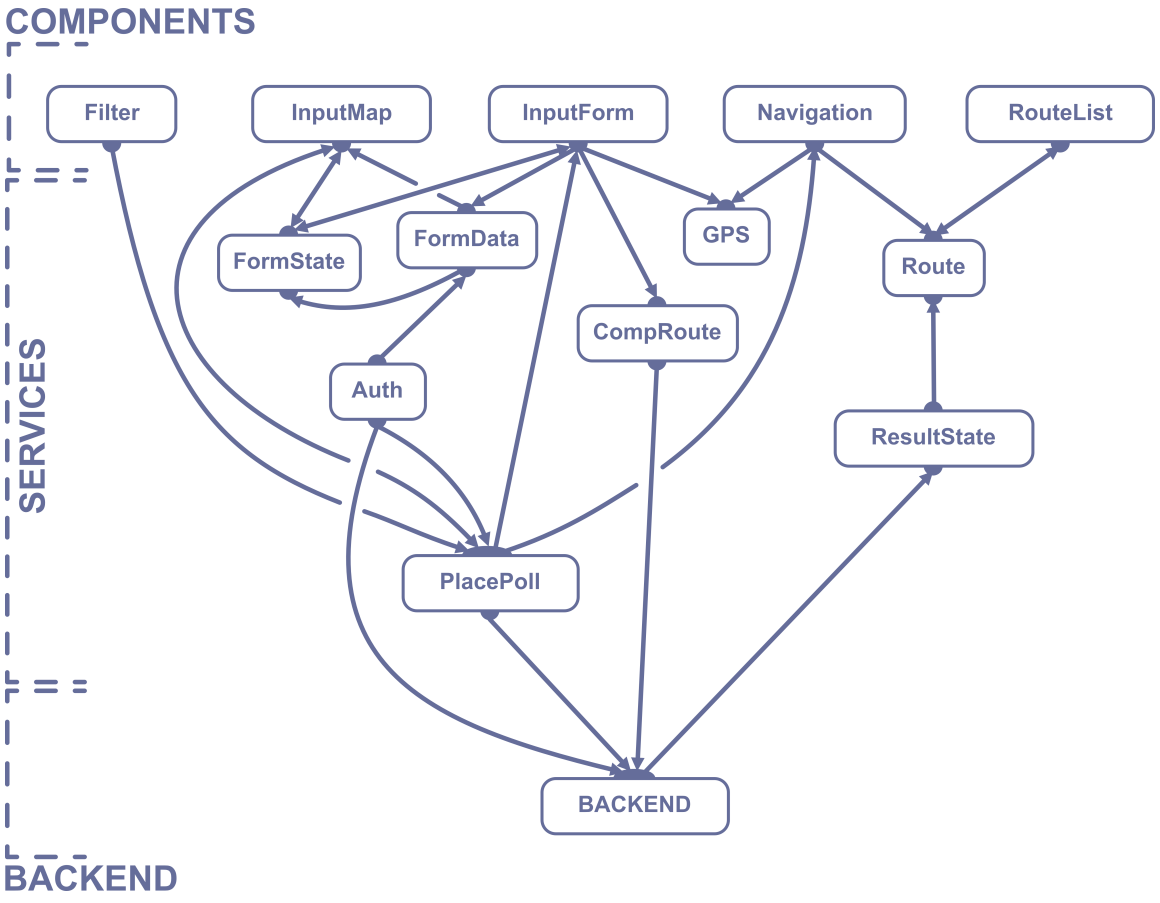
5 Technická realizace aplikace

Pro rozšíření stávající aplikace o novou službu, která by zajišťovala výše popsanou funkcionalitu, bylo nezbytné vytvořit také systém pro vizualizaci a zpracování dat na pozadí.

Aby bylo možné porozumět komunikaci mezi jednotlivými částmi systému, procesu zpracování dat při tvorbě úloh, i samotnému provádění výpočtů, je vhodné tyto postupy znázornit a popsat pomocí schémat – konkrétně pro frontendovou a backendovou část aplikace.

5.1 Frontend

Frontend představuje uživatelské rozhraní, ve kterém uživatel zadává vstupní data pro výpočet a následně získává výsledky. Aplikace je navržena jako webové rozhraní, které je optimalizované pro použití na desktopových počítačích, tak i na mobilních zařízeních.



Obr. 5: Schéma návazností na frontendu aplikace

V rámci rozdělení podle schématu (Obr. 5) lze celý frontend rozdělit na komponenty (*components*) a služby (*services*), přičemž každá z těchto kategorií představuje odlišnou úroveň zpracování dat.

Komponenty lze popsat jako konečné uživatelské rozhraní, které zprostředkovává jednotlivé akce uživateli. Patří sem prvky pro zadávání výpočtu – `InputMap` a `InputForm`, zobrazení výsledků – `Navigation` a `RouteList`, nebo speciální komponenty jako `Filter`.

Služby pak představují další prvky, které slouží k oddělení a zpracování logiky aplikace od samotného uživatelského rozhraní. Mohou sdílet či uchovávat různé vlastnosti, jako jsou data, funkce, komunikace s backendem, validace, formátování či další možnosti. Jejich úkolem může být také zlepšení čitelnosti a údržby kódu nebo usnadnění testování.

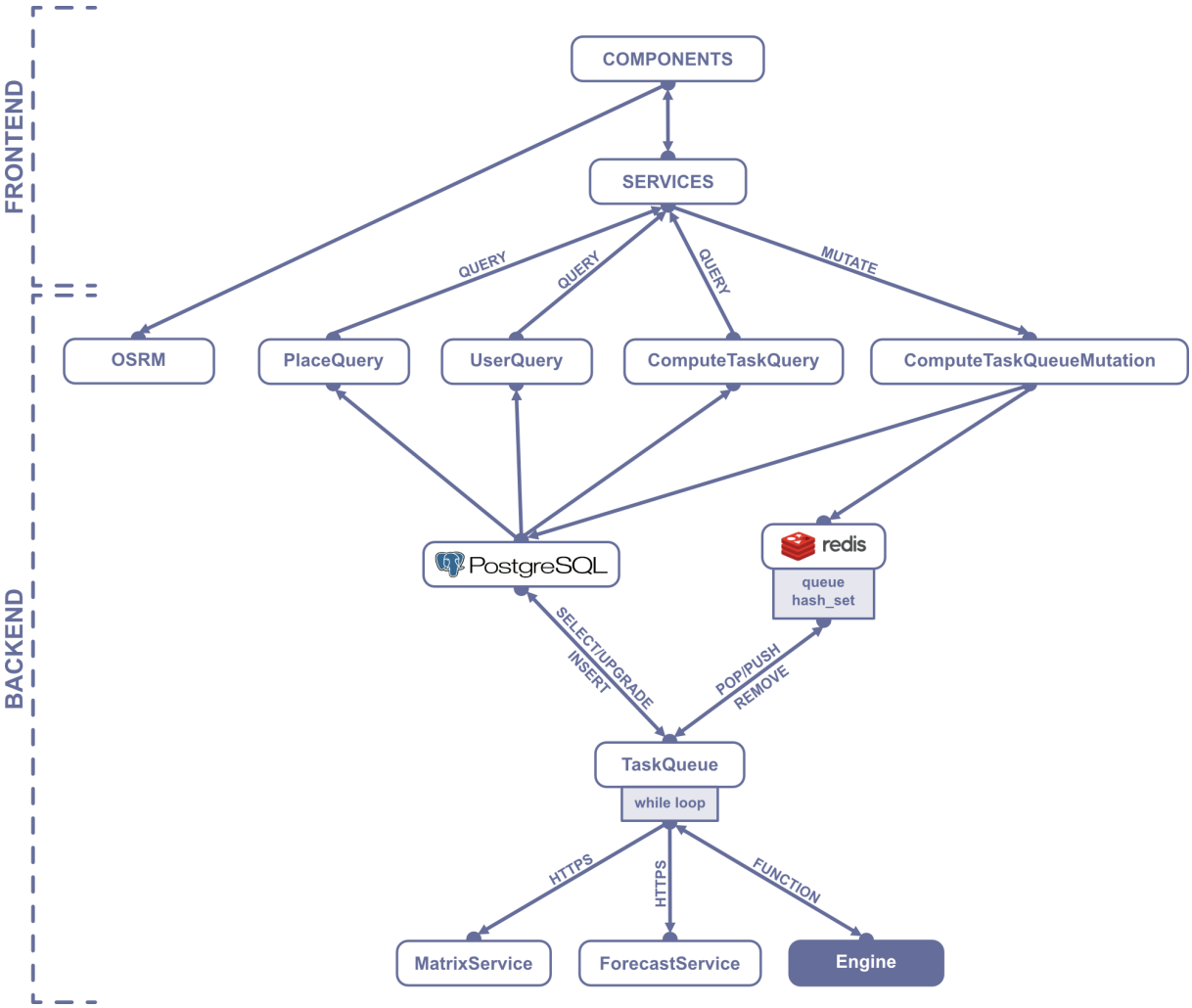
Ačkoli tyto moduly nejsou součástí této práce, jejich popis umožňuje lépe porozumět zadávání, principům zpracování a hodnotám, které vstupují do výpočtů. Z tohoto důvodu je vhodné nastínit jejich funkcionalitu, alespoň pokud jde o části grafického uživatelského rozhraní, se kterými se uživatel může setkat:

- **Filter** – jedinečná komponenta, jelikož pracuje nad mapami vstupní i výstupní částí (`InputMap` a `Navigation`). Umožňuje zobrazování a výběr stanic na základě vybraných kritérií. Slouží částečně jako první validace uskutečnitelného zadání, jelikož nepovoluje zobrazení, a tedy přidání, stanic, které nesplňují některé z požadavků.
- **InputMap** (vstupní mapa) – komponenta poskytující náhled do interaktivní mapy, s možností výběru stanic a úkonů, které se dále propisují do vstupního formuláře, kde se s nimi následně pracuje.
- **InputForm** (vstupní formulář) – představuje rozhraní pro úplné nastavení dat výpočtu. Staví se tak na úplný konec procesu výběru. Po potvrzení zadání se data odesílají na službu, kde probíhá validace. Při splnění všech kritérií přesouvá informace dále na backend, kde se rozšiřují, doplňují a pokračují až do výpočetního modulu (`Enginu`). Poskytuje výběr odeslání jak pro jednoho, tak i více řidičů.
- **RouteList** (výstupní formulář) – zpracovává a zobrazuje úkony na jednotlivých stanicích získané z výpočtu. Slouží pro potvrzování a aktualizaci zadaných činností v rámci průběhu cesty.
- **Navigation** (navigace) – podle názvů poskytuje navigaci pro průjezd stanic ze seznamu `RouteList`. Celé rozhraní probíhá na mapě, kde kromě jednotlivých míst lze sledovat i úkony včetně pořadí průjezdů.

5.2 Backend

Při odeslání zadaných dat ke zpracování a výpočtu probíhá jejich kontrola jak na frontendu, tak i backendu. Cílem je zajistit, aby stanice a úkony na nich nebyly pro daný den již zahrnuty v plánech jiných servisních pracovníků. K tomu se využívají údaje z databáze. Pokud alespoň jedna stanice tuto unikátnost nesplňuje, úkon je spolu s chybovou zprávou vrácen zpět na frontend, kde je uživatel vyzván k úpravě okrajových podmínek tak, aby se těmto stanicím vyhnul.

Pokud však toto omezení porušeno není, validovaná data jsou přeposlána do fronty, kde vzniká takzvaný *task*. K tomuto novému objektu se následně získává řada informací potřebných pro celý výpočet. Pro zpracování se využívá několik backendových služeb, které jsou znázorněny ve schématu (Obr. 6).



Obr. 6: Schéma návazností backendu aplikace

Jelikož tyto služby poskytují podstatnou část dalších rozšiřujících dat nezbytných pro výpočet, je vhodné si je stručně představit:

- **Databáze PostgreSQL** – obsahuje všechny informace o stavech jednotlivých stanic, dep a kol na nich umístěných (například počty pro dané úkony, technické stavy či typy kol). Význam těchto informací, jejich účel a podrobnější popis je uveden v kapitole 6.
- **MatrixService** – počítá a vrací matici časové náročnosti a vzdáleností vzájemných přejezdů mezi stanicemi. Při výpočtu zohledňuje typ vozidla (včetně omezení na cestách), jednosměrné ulice, zpomalení při průjezdu zatáčkami, drobnou penalizaci za průjezd světelně řízenými křižovatkami a další faktory.

- **ForecastService** – prediktivní služba sloužící k předpovědi výpůjček jednotlivých stanic. Byla vyvinuta rovněž v rámci projektu a zaměřuje se na odhad poptávky na základě historických dat z nedávné doby. Předpověď je poskytována ve formě procentuálních odhadů příjezdů a odjezdů kol z každé stanice v 15minutových intervalech během dne. Ke každé stanici tak doplňuje odhad, kolik kol do ní v daném časovém okně pravděpodobně přijede a kolik z ní odjede. Součástí výstupu je rovněž informace o počtu kol, která se v danou chvíli s největší pravděpodobností nacházejí mimo stanice, tedy jsou aktuálně vypůjčena. Více o tomto modelu v kapitole 6.6.

Po převzetí všech informací ze zmíněných služeb a zkompletování celého souboru je výsledný objekt předán výpočetnímu modulu. Pokud je úloha řešitelná a výpočet proběhne úspěšně, `Engine` vrací výsledek zpět na backend. V této fázi se z vyřešené úlohy načtou data, která jsou následně zapsána do databáze. Tato operace slouží jak k aktualizaci stavu jednotlivých stanic, tak k zamezení výběru obsažených míst dalšími pracovníky. Posledním krokem procesu je předání výsledných informací na frontend, kde je uživateli zobrazena naplánovaná trasa v obou dříve zmíněných komponentách – výstupním formuláři (`RouteList`) a navigaci (`Navigation`). V rámci těchto rozhraní lze stav jednotlivých úkonů dále aktualizovat (v případě neočekávaných změn během dne) a zároveň potvrzovat jejich splnění.

6 Vstupní parametry a okrajové podmínky výpočtu

Celý proces vývoje výpočetního modulu je poměrně komplexní, a pro jeho správné pochopení je nejprve nezbytné definovat základní pojmy a stavy, s nimiž tato práce dále pracuje. Ačkoli některé z nich již byly částečně nastíněny v kapitole 2.2, pro úplné porozumění je vhodné je dále rozšířit a upřesnit. Cílem této části je pojmy systematicky objasnit a podrobněji popsat všechna nastavení a stavy, se kterými výpočetní modul může při své činnosti pracovat.

6.1 Kola, úkony na stanicích a role zaměstnanců

Jednotlivá kola a stanice mohou vyžadovat různé typy úkonů, jejichž provedení zajišťují servisní technici. Ti se mezi stanicemi přesouvají s vozidly, kde vykonávají předepsané činnosti. Pro pochopení požadavků, které je nutné při výpočtech zohlednit, se musí jejich významy jednoznačně definovat:

Stavy kol:

- **Kontrola kol** – každé kolo by mělo ideálně projít kontrolou alespoň jednou za dva až tři týdny. Cílem je udržovat přehled o technickém stavu a předcházet situacím, kdy by kolo bylo chybně evidováno jako provozuschopné, přestože by neodpovídalo požadovaným standardům. Tento úkon nevyžaduje žádné speciální vybavení – jedná se převážně o vizuální kontrolu, při níž technik naskenuje QR kód kola v aplikaci a potvrdí jeho aktuální stav.
- **Opravy** – zahrnují drobné závady, které lze odstranit přímo na stanici bez potřeby specializovaného nářadí. Typickými příklady jsou shozené řetězy, rozbitá světla nebo jiná menší poškození vzniklá běžným provozem. Doba nutná k provedení opravy je zpravidla krátká – v řádu několika minut.
- **Vážné opravy** – jedná se o závažnější technické poruchy, které nelze řešit přímo na stanici. Kola ve vážném stavu je nutné stáhnout zpět do depa, což je speciální servisní stanice vybavená potřebným zázemím a nářadím pro náročnější servisní zásahy. Typickými příklady jsou poškození baterií u elektrokol, poruchy GPS modulů a jiné technické závady vyžadující zásah odborného personálu.
- **Odemčená kola** – nacházejí se ve stanici, avšak zůstala z nějakého důvodu odemčená. Příčinou může být nepozornost uživatele při ukončení jízdy, případně technická závada, například selhání elektronického zámku. Taková kola představují riziko odcizení nebo dalšího poškození. Z tohoto důvodu je nezbytné je co nejdříve zkontrolovat, uzamknout, případně opravit.
- **Kola mimo stanice** – tento stav označuje kola, která se nacházejí mimo oficiální stanice a zároveň nejsou součástí aktivní výpůjčky. Pokud takové kolo zůstane dlouhodobě bez pohybu nebo se nachází mimo vymezené území, je automaticky

evidováno v systému jako samostatná „stanice“. Správa těchto kol má vysokou prioritu, zejména v případě, že se jedná o elektrokola, u nichž vzniká nejvyšší potenciální finanční ztráta.

Možné úkony na stanicích:

- **Rebalanc** – tedy vyrovnávání počtu kol na stanici dle zadané hodnoty (*setpointu*) nebo dle výstupu prediktivního modelu (`ForecastService`). Cílem je zajistit, aby bylo v průběhu celého dne na stanicích k dispozici dostatečné množství kol odpovídající aktuální poptávce, a předešlo se tak situacím, kdy by zájemce o výpůjčku nebyl obsloužen. Oba přístupy pracují s historickými daty – rozdíl spočívá v tom, že setpointy jsou pevně dané hodnoty poskytované společností nextbike a platí univerzálně bez ohledu na den v týdnu, počasí či další proměnné. Naproti tomu prediktivní model sleduje dlouhodobé trendy ve výpůjčkách a umožňuje tak flexibilnější reakci na aktuální situaci.
- **Kampaně** – představují reklamní aktivity spojené s vizuálem kol. Jelikož je provoz služby částečně financován reklamou, je potřeba při změnách v obchodní spolupráci zajistit výměnu reklamních polepení. Ta probíhá přímo na stanici, formou výměny „kus za kus“, bez změny kapacity vozidla. V případě, že je nutné vyměnit větší množství kol, vozidlo nejprve sváží stávající kola s neaktuální reklamou na depo, kde dojde k přelepení, a následně jsou kola rozvezena zpět do provozu.
- **Opravy** – zahrnují drobné zásahy na kolech, které je možné provést přímo na stanici bez nutnosti specializovaného vybavení. Jak bylo uvedeno výše, jedná se například o opravu světel, řetězu či jiných menších závad.
- **Kontroly** – slouží jako inspekce stavu kol. Patří sem i odemčená kola na stanici, která může servisní technik v případě potřeby přehodnotit a zařadit do stavu rozbitých. Také vychází ze stavu popsaného výše.
- **Svoz rozbitých kol** – v rámci pozdější kategorizace v `Enginu` jsou sem zařazována vážně poškozená kola ze stanic spolu s koly mimo stanici, u nichž se předpokládá, že obsahují závady.

Role zaměstnanců/servisních techniků na stanicích

Role zaměstnanců definují oprávnění k vykonávání různých úkonů na stanicích, což umožňuje efektivní přerozdělení práce, omezení složitých úkolů pro nováčky a specializaci během směn (například prioritu při stahování kampaní). Konkrétně se jedná o:

- Vyrovnávání stavu kol (`REBALANC`),
- Opravy kol (`FIX`),
- Svoz vážně rozbitých kol na depo (`REMOVE`),
- Stahování kampaní (`CAMPAIGN`),
- Kontrola kol (`CHECK`).

6.2 Typy stanic a jejich fixace

V rámci nastavení jsou stanice rozlišeny podle priorit, které se řídí aktuálními potřebami redistribuce, a také v závislosti na technickém stavu kol. Tohle dělení primárně žádnou stanici výrazně nepreferuje, ani nevylučuje, nicméně bez dalších omezení by vozidlo pravděpodobně zůstávalo v centru města, kde je největší pohyb, a tudíž zpravidla i největší potřeba servisu.

Pro cílenou pozornost určitých míst byla přidána možnost jejich explicitního výběru nebo zákazu, čímž lze lépe kontrolovat sestavení směny.

Ve výpočtu se uvažuje celkem šest typů stanic, které se liší nejen důležitostí, ale i tím, pro kolik zaměstnanců jsou v rámci výpočtu dostupné. Jejich seznam a popis je následující:

- **Obecná stanice**:
  - **Běžná stanice** – stanice bez speciálních požadavků, zahrnující všechna odběrová místa.
  - **Depa/workshopy** – speciální lokace, minimálně jedno na město, sloužící k odkládání vážně rozbitých kol na opravu nebo přelepu kampaní. Alespoň jedna taková stanice musí být vždy zahrnuta ve výpočtu.
- **Společné stanice pro více řidičů**:
  - **Společné zakázané stanice** – tyto stanice jsou většinou vyřazeny již na frontendu nebo v části backendu a do výpočtu nevstupují. Ať už je důvodem manuální výběr uživatele nebo systémová logika, jsou tyto stanice záměrně ignorovány. Příkladem může být konání různých akcí, jako jsou závody dračích lodí nebo ohňostroje na přehradě, při nichž mohou být kola do okolních stanic svážena úmyslně mimo běžné setpointy. Tato místa jsou pak v systému evidována s výrazně vyššími hodnotami oproti běžným stavům, díky kterým by mohla vznikat snaha o snížení na běžnou úroveň. Jelikož však takový zásah nemusí být žádoucí, bývají tyto stanice ze zpracování zpravidla vyloučeny.
  - **Společné přikázané stanice** – množina stanic, které musí být navštíveny některým z řidičů v rámci směny, nezáleží však na tom, kterým konkrétně. Platí pouze v případech, kdy je výpočet zadán pro více pracovníků najednou.
- **Stanice pro jednotlivé řidiče**:
  - **Přikázané stanice** – stanice, které jsou pevně přiřazeny konkrétnímu řidiči a musí být navštíveny. Pro ostatní řidiče jsou tyto stanice v rámci výpočtů považovány za zakázané.
  - **Zakázané stanice** – stanice, na kterých danému pracovníkovi není dovoleno vykonávat zvolenou práci. Do této skupiny spadají také stanice přikázané jiným zaměstnancům a stanice, které si uživatel výslovně zablokoval. Filtrace těchto

stanic probíhá na více úrovních – stanice manuálně zablokované nebo již zahrnuté ve výpočtech jiných zaměstnanců jsou obvykle vyřazeny už na úrovni frontendu, zatímco zbylé případy jsou zohledněny až ve výpočetním jádře.

6.3 Časová okna směn, počáteční a koncové polohy

Plánovaná trasa nemusí být omezena pouze na jeden souvislý časový úsek, ale může být rozdělena do dvou samostatných částí, například na dopolední a odpolední směnu. Takové rozdělení poskytuje větší flexibilitu při sestavování denního plánu. Servisní technik může po absolvování první části trasy dorazit na depo, kde vykoná další činnosti, například údržbu nebo opravy kol, a teprve poté pokračovat ve druhé části rozvozu.

V rámci jednoho pracovního dne tak mohou být plánovány až dvě směny. Pokud by bylo požadováno větší členění, je nutné výpočet rozdělit do více samostatných běhů, přičemž mezi nimi musí být zajištěna návaznost a kontrola navštívenosti jednotlivých stanic. Takový postup předpokládá úplné dokončení první části trasy před zahájením druhé. Přesto zůstává i tato varianta v systému podporována a může být využita dle potřeby konkrétní situace.

Plánovaná trasa nemusí být omezena pouze na jeden souvislý pracovní cyklus, ale může být rozdělena do dvou samostatných částí, například na dopolední a odpolední směnu. Toto rozdělení poskytuje větší flexibilitu při sestavování směn a umožňuje lépe přizpůsobit pracovní režim potřebám jednotlivých techniků. Typickým příkladem může být situace, kdy pracovník absolvuje dopolední trasu, následně se vrátí na stanovené místo (např. depo), kde má možnost realizovat delší přestávku nebo vykonat dílčí opravy, a až poté pokračovat v rozvozu kol.

Jeden pracovní den tak může být v tomto rámci rozdělen až na dvě směny. V případě potřeby plánování více částí pro jeden den je nutné přistoupit k oddělenému výpočtu každé z nich, přičemž mezi jednotlivými částmi musí být zajištěna kontrola stavů navštívenosti a aktuální dostupnosti kol i kapacit na jednotlivých stanicích. Tento přístup je založen na podmínce úplného dokončení první trasy, než může být spuštěna výpočetní procedura pro trasu následující. Takový způsob plánování je podporován a lze jej v systému plnohodnotně realizovat.

Ke směnám v rámci jednoho požadavku se následně přiřazují výchozí a koncové polohy. Definice těchto poloh může být následující:

- **Případ jedné směny:**
  - **Pevně stanovený začátek směny, konec definovaný nebo volný** – trasu nelze vytvořit bez určení výchozí polohy. Konec směny však není nutné specifikovat; v takovém případě vozidlo ukončí trasu podle dostupného času bez

ohledu na konkrétní místo. Pro výchozí i koncové body je aktuálně dostupné následující nastavení:

  * Workshop (nebo také depo),
  * Libovolná poloha na mapě,
  * Aktuální pozice vozidla,
  * „Je mi to jedno“ – označení možnosti pro volně definovanou koncovou polohu; nelze využít pro výchozí bod trasy.

- **Případ dvou směn:**
  - **Definované začátky, pauzy a volitelný konec** – každý časový interval má přesně určenou výchozí a koncovou polohu. Toto nastavení umožňuje například naplánovat pauzu na oběd v konkrétním místě nebo ukončit první část směny na depu a využít tak delší přestávku na opravy. Úplný konec směny může být opět buď specifikován, nebo ponechán volný.
  - **Definován pouze začátek směny, pauza s volnou polohou** – v tomto případě je trasa plánována pouze mezi stanoveným počátečním bodem a koncem dne. Pauza je řešena výhradně časově, bez přesné lokalizace. Předpokládá se, že technik během přestávky zůstává poblíž místa, kde vykonával svou poslední činnost. Po jejím skončení pak plynule navazuje na druhou část směny ze stejné lokality.

6.4 Vozidla a stavy na počátku a konci směny

Existuje několik různých způsobů přepravy mezi stanicemi. Například v případě kontrol v blízkých lokalitách mohou servisní technici využívat kola. Mezi další způsoby dopravy patří automobily, dodávky či pěší chůze. Tyto varianty je třeba brát v úvahu při návrhu systému. Stejně tak je nutné zohlednit možnosti nastavení kapacity vozidla na začátku a konci směny. Podle manuálu servisního technika je doporučeno ukončit pracovní den s prázdným vozidlem, avšak tato podmínka není striktně závazná.

V rámci tohoto projektu však zatím nebyly uvažovány rozdíly v rychlostech přemístění mezi jednotlivými typy dopravy. Existuje možnost využití konstantních faktorů pro modifikaci matic vzdáleností a rychlostí přesunů, avšak tento přístup by z hlediska logiky řešení nepřinesl zásadní změnu. Z tohoto důvodu nebyl doposud implementován.

Zajímavým aspektem je analýza nákladů z hlediska počátečních a konečných stavů vozidla v průběhu pracovního dne. Tyto stavy jsou totiž jasně definované a jejich správné nastavení má často významný dopad na plánování tras a efektivitu rozvozu. Například na začátku dne může být stanovena počáteční skladba nákladu, včetně množství a typu jednotlivých položek. To zásadně ovlivňuje rozhodování o pořadí obsluhy jednotlivých stanic. Vozidlo může být například již předem naloženo funkčními koly, což mu umožní

přímo doplňovat zásoby na stanicích s nedostatkem, aniž by muselo nejprve navštěvovat místa s přebytky. Naopak, pokud je kapacita vozidla již zcela zaplněná, nebude možné během trasy žádný náklad naložit a bude nutné nejdříve navštívit depo, kde lze rozbité kusy vyložit. Těchto logistických pravidel a procesů je celá řada, což výrazně ovlivňuje plánování a optimalizaci tras.
Přehled všech možných stavů:

- **Na začátku dne**:
  - **Nulové náklady ve vozidle** – auto je při začátku směny prázdné.
  - **Obsazení již funkčními koly** – například z předchozího dne, nebo pokud začínají na depu s naloženými koly po opravě, případně bez kampaně při výměnách atd.
  - **Obsazení rozbitými koly nebo koly se starou kampaní** - pravděpodobný zůstatek z předchozího dne. Oba typy kol jsou interně ve výpočtu uvažována jako rozbitá - je tedy nutné pro jejich vyprázdnění prvně zajet na workshop.
  - Nebo kombinace obou typů kol dohromady.
- **Na konci dne**:
  - **Nulové náklady ve vozidle** – základní stav, jakým by měla být ukončena směna.
  - **Vozidlo s náklady** – bez ohledu na typ, jde o výjimky v rámci běžného provozu.

6.5 Otevřený/uzavřený výpočet

V rámci zadání úloh se výpočty dělí na dva typy: **otevřený** a **uzavřený**. Ty pak označují velikost úlohy, respektive kolik stanic bude celkově do výpočtu přidáno. To má velký vliv na výpočetní náročnost a také čas, který bude `Routing Solver` potřebovat pro nalezení řešení. Je tak na rozhodnutí uživatele, jak moc rozsáhlou úlohu s výpočetním časem je ochoten pro získání výsledku poskytnout.

Otevřený výpočet je tím náročnějším ze dvojice nastavení. Jedná se o stav, kdy se uvažují všechny dostupné stanice pro dané město. Dostupnými stanicemi se pak myslí ty, které se dostaly přes filtraci při výběru úkonů pro zaměstnance a zároveň nejsou již obsazené v jiných dřívějších výpočtech daného dne.

Oproti tomu uzavřený výpočet v základní formě představuje velmi často úlohu menšího typu. Zde jsou totiž v rámci tras uvažovány opravdu pouze ty stanice, které si uživatel vybere při definování návštěv. Přes výhody kratších výpočtů a menší výpočetní náročnosti je však třeba upozornit na možná negativa. Vzhledem k uvažování pouze povinných stanic může `Routing Solver` zkolabovat na „nemožnosti některé úlohy vyřešit“, respektive řešení bude triviální, a trasa tak bude uvažována pouze z počáteční do koncové

polohy. Příkladem úlohy, jež může vést k takovému výsledku, je situace, kdy uživatel má rozvážet kola mezi stanicemi, ale jeho počáteční kapacita je nulová a nevybere si žádné stanice, kde by kola přebývala. V takovém případě je tedy nemožné vykonávat vyrovnávání a ze startu trasa přejde rovnou do koncové polohy.

6.6 Hodnoty pro vyrovnávání počtu kol (setpoints)

Posledním nastavením, které lze zvolit v zadávacím formuláři, je typ *setpointu* – tedy požadovaného stavu, jež má za úkol určit, kolik kol má být na jednotlivých stanicích dostupných. Jak bylo naznačeno dříve, k těmto počtům lze přistupovat ze dvou odlišných hledisek.

Prvním typem jsou hodnoty poskytované společností nextbike. Ty podle dostupných informací představují odhady návštěvnosti jednotlivých stanic v době jejich vytvoření. Přestože se může jednat o zastaralé údaje, v systému byly ponechány pro případy, kdy by prediktivní model nepodával dostatečně spolehlivé výsledky.

Druhý přístup představuje prediktivní model, který je ve schématu (Obr. 6) označen jako `ForecastService`. Jedná se o komponentu, která byla v rámci systému vyvinuta od počátku. Tato služba na základě historických dat predikuje očekávaný počet výpůjček a analyzuje trendy z posledních několika týdnů. Cílem je reagovat na dynamické změny v chování zákazníků, například zvýšenou frekvenci v určitých lokalitách, nebo vliv sezónních změn.

`ForecastService` je komplexní prediktivní systém založený na pravděpodobnostním modelování. Do svých výpočtů zahrnuje celou řadu faktorů, jako je rozlišení mezi pracovními dny a víkendy, aktuální počasí, výpůjčky za poslední týdny, roční období a další proměnné, které mohou ovlivňovat chování uživatelů a zatížení jednotlivých stanic.

Výstupem modelu je datová sada, ve které je každý den rozdělen do patnáctiminutových intervalů. Pro každý tento interval a každou stanici model predikuje očekávaný počet výpůjček a vrácení kol. Kromě toho se snaží odhadnout i celkové množství kol, která jsou v danou chvíli v oběhu, tedy právě zapůjčená, což pomáhá lépe odhadovat aktuální dostupnost kol ve městě jako celku.

6.7 Návrh neformálního modelu Rich VRP

Před samotnou implementací programu pro návrh směn servisních techniků byl sestaven neformální matematický model, jenž sloužil jako výchozí rámec pro řešení dané úlohy. Cílem tohoto modelu bylo především zpřehlednění úvah, identifikace klíčových omezení a formulace hlavního směru, kterým by se mělo při hledání řešení postupovat.

Jako výchozí rámec byl zvolen model vycházející z problematiky *Rich Vehicle Routing Problem* (Rich VRP), tedy úlohy, která rozšiřuje klasický problém rozvozu zboží

(VRP) o více praktických aspektů najednou. Některá z těchto rozšíření, jako jsou časová okna, kapacitní omezení, nebo rozdílné konce a začátky tras, byla blíže představena v kapitole 3.

$$\begin{aligned}\min & \sum_{k\in K}\sum_{i\in V}\sum_{j\in V} c_{ij}\cdot x_{ijk} \\ & + \sum_{i\in N} p_i \cdot \left(1 - \sum_{k\in K} y_{ik}\right) \\ & + \sum_{k\in K} p_{\text{cap}} \cdot \max\left(0, L^f_{e_k,k} + L^b_{e_k,k}\right)\end{aligned} \tag{53}$$

Účelová funkce (53) se skládá ze tří složek:

- **Přepravní náklady** (první člen) minimalizují celkové náklady c_{ij} spojené s přesunem z uzlu i do uzlu j, pokud tento přesun vykonává technik k (tj. $x_{ijk} = 1$).
- **Penalizace za nenavštívený uzel** (druhý člen) se uplatňuje v případě, že uzel i není navštíven žádným technikem (tj. $y_{ik} = 0$). V takovém případě se k účelové funkci přičte penalizace p_i, jejíž velikost odpovídá důležitosti neprovedených úkonů na dané stanici.
- **Penalizace za překročení kapacity** (třetí člen) zachycuje případy, kdy vozidlo k ukončuje trasu s nezpracovaným nákladem. Tento zůstatek je dán součtem funkčního ($L^f_{e_k,k}$) a rozbitého ($L^b_{e_k,k}$) nákladu na konci trasy. Penalizace je vážena koeficientem p_{cap}.

Model je podmíněn následujícími omezeními:

$$\sum_{k\in K} y_{ik} \leq 1 \qquad \forall i \in N \tag{54}$$

$$\sum_{j\in V} x_{ijk} = \sum_{j\in V} x_{jik} \qquad \forall k \in K,\ \forall i \in V \setminus \{s_k, e_k\} \tag{55}$$

$$y_{ik} = \sum_{j\in V} x_{ijk} \qquad \forall i \in V,\ \forall k \in K \tag{56}$$

$$0 \leq L^f_{ik} \leq Q_k \qquad \forall i \in V,\ \forall k \in K \tag{57}$$

$$0 \leq L^b_{ik} \leq Q_k \qquad \forall i \in V,\ \forall k \in K \tag{58}$$

$$L^b_{jk} \geq L^b_{ik} + b_j - M(1 - x_{ijk}) \qquad \forall i \in V,\ \forall j \in V \setminus D,\ \forall k \in K \tag{59}$$

$$L^b_{dk} = 0 \qquad \forall d \in D,\ \forall k \in K \tag{60}$$

$$L^f_{ik} + L^b_{ik} \leq Q_k \qquad \forall i \in V,\ \forall k \in K \tag{61}$$

$$t_i^{\text{start}} \leq T_{ik} \leq t_i^{\text{end}} \qquad \forall i \in V, \forall k \in K \tag{62}$$

$$T_{jk} \geq T_{ik} + s_i + t_{ij} - M(1 - x_{ijk}) \qquad \forall i, j \in V, \forall k \in K \tag{63}$$

$$x_{ijk} = 0 \qquad \text{pokud role}_i \notin R_k \tag{64}$$

$$\sum_{k\in K}\sum_{j\in V} x_{ijk} \geq 1 \qquad \forall i \in \text{Povinné uzly} \tag{65}$$

Každou z rovnic lze interpretovat následovně:

- Rovnice 54 zajišťuje, že každému běžnému uzlu $i \in N$ je přiřazeno nejvýše jedno vozidlo $k \in K$. K označuje množinu všech vozidel.
- Rovnice 55 zachovává tokovou rovnováhu. Technik, který navštíví uzel $i \in V \setminus \{s_k, e_k\}$, do něj musí přijet a také z něj odjet (počty příchozích a odchozích hran se rovnají). Jedinou výjimkou jsou počáteční s_k a koncové e_k uzly cesty.
- Rovnice 56 propojuje binární proměnnou y_{ik} (uzel i obsloužen vozidlem k) s tokem x_{ijk} (technik k jede z i do j).
- Rovnice 57 a 58 omezují funkční náklad L^f_{ik} a rozbitý náklad L^b_{ik} v uzlu i tak, aby obě proměnné ležely mezi 0 a maximální kapacitou vozidla Q_k. Zároveň se předpokládá, že proměnné L^f_{ik} a L^b_{ik} jsou celočíselné, neboť reprezentují počty jednotlivých kol.
- Rovnice 59 řídí tok rozbitého nákladu. Pokud technik jede z i do j, pak musí být zadní náklad v j alespoň $L^b_{ik}+b_j$ (kde b_j je množství naloženého nákladu v j). Pokud mezi uzly nevede hrana, nerovnost je deaktivována pomocí velké konstanty M.
- Rovnice 60 říká, že v každém depu $d \in D$ je náklad rozbitých kol na začátku nulový (kola se zde vykládají).
- Rovnice 61 zajišťuje, že součet funkčního ($L^f_{jk} = L^f_{ik} - f_j \quad$ pokud $x_{ijk} = 1$) a rozbitého (L^b_{ik}) nákladu nepřekračuje kapacitu Q_k.
- Rovnice 62 omezuje, aby technik k dorazil do uzlu i (čas příjezdu T_{ik}) v rámci předepsaného časového okna $[t_{\text{start}i},\ t_{\text{end}i}]$.
- Rovnice 63 zajišťuje časovou návaznost mezi dvěma po sobě jdoucími uzly – pokud technik přechází z i do j, čas příjezdu do j musí být větší nebo roven času příjezdu do i plus servisní doby s_i a cestovního času t_{ij}. Pokud však hrana není aktivní ($x_{ijk} = 0$), podmínka je deaktivována pomocí velké konstanty M.
- Rovnice 64 vylučuje nepřípustné přesuny: pokud uzel i vyžaduje roli, kterou technik k nemá, není mu dovoleno odtud vyjíždět (tedy $x_{ijk} = 0$ pro všechna j).
- Rovnice 65 zajišťuje, že každý povinný uzel (např. stanice se specifickou kampaní nebo poruchou) bude navštíven alespoň jedním technikem.

Navržený model představuje koncepční rámec zachycující základní prvky úlohy, jako je přidělení vozidel, toky nákladu a časová omezení. Není však přímo vhodný pro řešení standardním MIP solverem, protože některé vnitřní vazby a logistická omezení jsou zde zjednodušena, což může vést k nekonzistentním výsledkům.

Model slouží jako výchozí bod pro praktickou implementaci v Google OR-Tools, která umožňuje využití specializovaných mechanismů – callbacků pro tok nákladu, rozšířených dimenzí pro čas a kapacitu či přímého napojení pomocných proměnných na rozhodovací hrany. V další kapitole bude tento model převeden do procedurální formy, která zajistí přesnější a konzistentní řešení.

7 Výpočetní modul – Engine

V této kapitole je podrobně popsán návrh a implementace výpočetního modulu označovaného jako `Engine`. Modul tvoří jádro celé aplikace a slouží k tomu, aby na základě vstupních dat byla nalezena dostatečně vyhovující trasa mezi jednotlivými stanicemi při respektování všech stanovených požadavků.

7.1 Postup při řešení

Po důkladném seznámení s problematikou a identifikaci všech možných stavů, které mohou v systému nastat, bylo přistoupeno k vývoji výpočetního modulu, jenž by byl schopen tyto situace sledovat a efektivně vyhodnocovat. K tomuto účelu byl zvolen programovací jazyk Python a knihovna OR-Tools 4.3, která poskytla robustní základ pro další vývoj. Mezi hlavní důvody tohoto výběru patřila kvalitní dokumentace, široké možnosti konfigurace, bezplatná licence a silná uživatelská komunita.

Pro prvotní seznámení se s tímto nástrojem a zahájení vývoje výpočetního jádra byly využity ukázkové příklady z oficiální dokumentace společnosti Google. Tyto jednoduché úlohy představovaly vhodný výchozí bod pro pochopení základních principů knihovny. Klíčovým příkladem byl základní problém rozvozu zboží (VRP, *Vehicle Routing Problem*) a jeho rozšíření o kapacitní omezení ve formě CVRP (*Capacitated Vehicle Routing Problem*).

Na tyto základní modely byla následně navázána další rozšíření, která byla doplněna o nové definice, další dimenze pro sledování různých parametrů a implementace pokročilejších metodik. V průběhu vývoje byly experimentálně ověřovány různé přístupy, které jsou podrobně popsány v následujících podkapitolách. U jednotlivých metod je vždy uvedeno zdůvodnění jejich výběru, předcházející úvahy i konkrétní implementační kroky.

V období před zpřístupněním reálných dat od společnosti byla testování prováděna na modelovém příkladu vytvořeném speciálně pro účely této úlohy. Tato data byla generována uměle a sloužila výhradně k validaci výpočetních postupů a ověření funkčnosti navrženého řešení.

7.1.1 Rebalanc

Problematika distribuce kol mezi jednotlivými stanicemi byla uvažována jako klíčová součást celého projektu. Z tohoto důvodu byl směr vývoje výpočetního modulu už z úvodu zaměřen primárně na řešení této oblasti.

Základní definice vyrovnávání se zpočátku jevila jako poměrně jednoduchá – stanice byly rozděleny na dva typy podle hodnoty získané jako rozdíl mezi aktuálním počtem kol a přednastaveným žádaným stavem. V případě záporné hodnoty bylo třeba na dané stanici kolo vyložit, zatímco při kladné hodnotě šlo o přebytek, který mohl být odebrán.

Knihovna OR-Tools dokáže v základní konfiguraci s tímto typem úlohy efektivně pracovat. Stanice, u nichž je požadováno doplnění kol (záporná hodnota), nejsou v rámci optimalizace navštěvovány dříve, než vozidlo disponuje dostatečným množstvím nákladu. Obdobně nejsou zařazovány ani stanice, jejichž obsluha by po naložení vedla k překročení maximální kapacity vozidla.

Tato přístupová logika však v reálných podmínkách představuje poměrně striktní omezení. Z tohoto důvodu bylo nutné přistoupit k alternativnímu způsobu modelování, který by umožnil flexibilnější přístup k obsluze jednotlivých stanic. Vzhledem k tomu, že OR-Tools ve svém výchozím nastavení neumožňuje částečné plnění požadavků na jedné stanici, bylo rozhodnuto o rozdělení stanic na dílčí jednotky odpovídající jednotlivým kolům.

Tímto způsobem bylo umožněno realizovat i částečné vyrovnání stavů. V praxi to znamenalo, že každá stanice byla duplikována tolikrát, kolik kol se na daném místě nacházelo. V případě stanic s přebytkem byla hodnota kol přiřazena jako kladná, zatímco u stanic s nedostatkem byla hodnota záporná.

7.1.2 Rozbitá kola, funkční kola a sdílená kapacita

Distribuce funkčních kol představovala pouze jednu ze dvou kategorií nákladů, které byly v rámci řešené úlohy zohledňovány. Ačkoli byla problematika vyrovnávání již implementována, bylo nutné dále rozšířit model o nakládání a vykládání rozbitých kol. Tento aspekt s sebou přinesl řadu dodatečných komplikací, a to jak z hlediska sdílené kapacity vozidel pro oba typy nákladu, tak i z pohledu možnosti vykládky vadných kol na speciálně označených stanicích typu *depo* (resp. *workshop*).

Každé vozidlo, jak bylo dříve uvedeno, disponuje pevně stanovenou maximální kapacitou, kterou nelze překročit. Tato kapacita byla upravena tak, aby byla sdílena mezi oběma typy nákladu – tedy funkčními i rozbitými koly. Matematicky je tato podmínka popsána rovnicí (66), kde $d_{functional}$ označuje aktuálně přepravovaný počet funkčních kol, d_{broken} odpovídá množství rozbitých kol a Q udává celkovou kapacitu vozidla:

$$d_{functional} + d_{broken} \leq Q \tag{66}$$

Na základě této skutečnosti bylo nezbytné vytvořit tři samostatné dimenze pro sledování nákladu. První dimenze zaznamenávala množství funkčních kol, druhá se vztahovala k rozbitým kolům a třetí sloužila ke kontrole celkové zatíženosti vozidla tak, aby nedošlo k překročení kapacitního omezení.

Přestože byly dimenze vytvořeny a připraveny k použití, bylo zapotřebí upravit i samotné uzly (stanice), neboť dosud neexistovalo oddělení obou typů nákladu na jejich úrovni. Nejjednodušší cestou se ukázalo vytvoření samostatných položek pro každý typ kol v rámci dané stanice. Aby však nedocházelo ke stejným problémům jako v případě redistribuce funkčních kol, bylo i zde nutné rozdělit jednotlivá kola na samostatné (dílčí)

uzly. Každá z těchto instancí pak zaznamenávala svůj typ nákladu samostatně, a tím bylo zajištěno správné mapování do odpovídajících dimenzí.

Další specifickou situací, kterou bylo nutné řešit, bylo vykládání rozbitých kol na stanici označené jako workshop. Vzhledem k tomu, že běžně bývá požadované množství nákladu stanoveno předem, nebylo možné použít jednoduchý přístup se zápornou hodnotou nákladu jako indikátorem vykládky. Cílem bylo umožnit vozidlu navštívit depa kdykoli v průběhu trasy a zároveň při každé návštěvě dynamicky vyprázdnit zásobu rozbitých kol.

V rámci návrhu řešení bylo přistoupeno k úpravě chování těchto speciálních uzlů. Při průchodu algoritmem stanice standardně vracela hodnotu odpovídající změně nákladu, přičemž u běžných stanic šlo o hodnotu zodpovídající přepravovaným kolům. U stanic typu workshop, identifikovaných pomocí jejich indexu, však bylo při každé návštěvě dynamicky invertováno množství přepravovaných rozbitých kol – výsledná hodnota tak byla záporná o velikosti maximální kapacity vozidla. Tímto mechanismem bylo dosaženo kompletního vyložení všech vadných prostředků, zatímco funkční kola zůstávala v nákladovém prostoru pro další rozvoz.

7.1.3 Úkony na stanicích, servisní časy

Předchozí řešení zatím popisovalo pouze dva z pěti úkonů, které je možné na stanicích vykonávat. Otázkou tak stále zůstávaly opravy drobných závad na místě, stahování kampaně a průběžná kontrola stavu kol.

Kontroly a opravy kol, které mohou být realizovány přímo na stanici, byly do modelu začleněny podobným způsobem jako redistribuce funkčních a vážně poškozených kol. I zde dochází k rozdělení na samostatné uzly, avšak s jedním klíčovým rozdílem - každý vrchol se pro danou činnost rozděluje pouze jednou. Počítá se totiž s tím, že servisní technik provede všechny úkony při jediné návštěvě stanice. Pokud je například na stanici potřeba zkontrolovat sedm kol, jsou všechny tyto kontroly provedeny najednou. Tento přístup se osvědčil z několika důvodů:

- Kontrola jednoho kola v průměru nezabere více než půl minuty, neboť spočívá pouze ve skenování QR kódu a případném stručném komentáři k jeho stavu.
- V případě oprav je výhodné provést všechny zásahy najednou, když už se technik na stanici zdržuje a má s sebou potřebné vybavení.
- Není třeba uměle navyšovat počet uzlů dalšími kopií stanic, což by vedlo ke zbytečnému růstu složitosti úlohy.
- Rozdíl mezi uzly tvoří pouze rozdílné výpočty servisních časů.

Pro zahrnutí časové náročnosti těchto úkonů do optimalizace bylo třeba definovat tzv. servisní časy. Ty představují dobu, kterou vozidlo (respektive technik) na stanici stráví. Každý typ úkonu má přiřazenou svou průměrnou časovou konstantu, vyjádřenou

v sekundách. Například pro naložení rozbitého kola je servisní čas součtem doby potřebné pro jeho identifikaci a fyzické umístění do vozidla (obdobně u redistribuce).

U uzlů zajišťujících opravy a kontroly jsou servisní časy počítány odlišně. Vstupní datové balíčky obsahují počet kol, na které se daný úkon vztahuje. Tento počet se vynásobí příslušnou časovou konstantou a výsledek tvoří celkový servisní čas pro daný uzel.

Zvláštní roli v tomto ohledu hrají stanice označené jako workshop. U těchto stanic je servisní čas přednastaven jako konstantní hodnota, aplikovaná bez ohledu na aktuální stav nákladu. Tento přístup byl zvolen z důvodu, že během výpočtu nelze dynamicky měnit dobu strávenou na stanici v závislosti na aktuálním množství rozbitých kol určených k vyložení.

Další specifickou kategorií úkonů jsou kampaně. Ačkoli se kola určená ke kampani obvykle nacházejí v provozuschopném stavu, je jejich cílem odstranění ze systému. Tato kola se tedy musí ze stavu stanice odečíst, jelikož je nelze dále redistribuovat. Ve zbytku výpočtového modelu se s těmito koly pak nakládá obdobně jako s rozbitými - ať už při tvorbě klonovaných stanic, nebo při vyhodnocování typu nákladu.

7.1.4 Startovní a koncové polohy

Implementace startovních a koncových poloh představovala relativně jednodušší část celého procesu, neboť je tato funkcionalita přímo podporována základními možnostmi použité knihovny.

Přesto bylo nezbytné pro každého řidiče vytvořit speciální samostatné uzly pro start i konec trasy, zejména v případě otevřených konců. Tento přístup umožňoval zahájit trasu prakticky odkudkoli - tedy buď na existujících stanicích, nebo mimo ně, například na aktuální pozici uživatele v okamžiku spuštění výpočtu.

V situacích, kdy ale uživatel na své aktuální pozici nevykonával žádné činnosti, nebyla tato poloha reprezentována samostatnou stanicí. Toto řešení způsobovalo závažné problémy, neboť stanice s aktuální polohou technika se ve výpočtu nemusely objevit. Výsledkem bylo, že technik neměl definované místo, kde by mohl začít či ukončit směnu, a úloha tak nemohla být správně vyřešena.

Pro otevřené konce celodenních směn vznikla navíc potřeba zavedení tzv. *dummy node*. Tento speciální uzel lze navštívit za nulovou cenu (časovou i nákladovou), avšak nelze z něj dále vyjet. Dummy node tedy plní funkci fiktivního ukončení trasy na kterékoliv ze stanic.

7.1.5 Povinné a zakázané stanice

Nutnost výběru stanic, na které by byly do výpočtu zařazeny s určitou prioritou, byla stejně důležitým prvkem, jako možnost některé stanice z výpočtu vyřadit. Tyto problémy daly také za vznik dvou odlišných přístupů k zadání a výsledkům.

Jak bylo uvedeno v kapitole 6.5, existují dva typy výpočtů. Uzavřený typ pracuje pouze se stanicemi zařazenými mezi *povinné* stanice. Z praktického hlediska to znamená, že pokud se žádá výpočet směny pouze pro jediného servisního technika, do výpočtu se dostanou pouze stanice, které byly definovány jako požadované. Tato úloha se tak v základu řeší relativně snadno. Problém však nastává, pokud se má do jednoho výpočtu zahrnout řešení více řidičů, nebo pokud je úloha označena za *otevřenou*.

Otevřený výpočet je jednodušší formou a je proto rozumnější si základní logiku představit právě na něm. Do Solveru se v takovém případě dostanou všechny stanice, které nebyly vyřazeny při předchozí filtraci během vytváření zadání. Pro Brno to znamená, že po uvolnění všech stanic je k dispozici přes 250 odběrových míst s přibližně 1000 koly mezi nimi.

Pokud mezi těmito stanicemi chceme některé preferovat, mohou být při vstupu označeny jako *fixní* (`fixed_locations`). Tyto stanice jsou poté přiřazeny k jednotlivým řidičům „napevno“. V překladu to znamená, že návštěva těchto stanic je vynucená a nelze je ze výpočtu vyřadit. Jedinou výjimkou je situace, kdy se již návštěva do plánu nevejde z důvodu časové náročnosti.

Naopak *zakázaná* místa (`banned_locations`) představují přesný opak - ve výsledné trase se nesmí vyskytovat za žádnou cenu. Oproti povinným stanicím, kde se jejich vynechání penalizuje takovou hodnotou, aby se Solveru jejich vynechání nevyplatilo, mají zakázané stanice elegantnější řešení.

OR-Tools totiž umožňuje ke každému uzlu jasně definovat, které vozidlo může uzel navštívit. V případě, že žádné vozidlo není ke stanici přiřazeno, nebo je ke stanici přiřazena prázdná množina povolených vozidel pro návštěvu, stává se tato pro daný výpočet nedosažitelnou.

Zbytek stanic, které nejsou ani v jedné z výše zmíněných kategorií, spadá do skupiny volitelných. Fungují jako běžné uzly, u nichž je přiřazena drobná penalizace podle typu úkonu, který reprezentují. Ze řešení je tedy lze vyloučit, ale pro minimalizaci celkové nákladové funkce se Solver v případě volně dostupného času po návštěvě všech fixních uzlů bude nadále snažit tyto volitelné stanice navštívit.

Problematika se rozšiřuje zejména v případě více řidičů, a to bez ohledu na typ výpočtu. V takovém případě je totiž nutné sledovat a penalizovat jednotlivé stanice vzhledem k vozidlu. Navíc neexistují povolené a zakázané uzly pouze pro každého řidiče zvlášť, ale byla přidána i mechanika pro výběr fixních a zamítnutých stanic nezávisle na jejich prioritě.

Opět je vhodné začít od jednodušší varianty, tedy od společných typů stanic. Ty jsou řešeny právě pomocí výše uvedených metod – tedy penalizací v případě požadovaných, u zamítnutých přiřazením prázdné množiny vozidel, které ji mohou navštívit (případně úplným vyřazením z poskytovaných dat). V případě fixních stanic není třeba mít obavy

o jejich návštěvu více techniky, jelikož Solver povoluje vždy pouze jednu návštěvu uzlu v rámci celé trasy bez ohledu na vozidlo.

Poněkud složitější byla logika u stanic povolených nebo zakázaných jen pro jednoho konkrétního zaměstnance. U fixních stanic bylo třeba tato místa přiřadit pouze vozidlu, pro které byla definována. To však znamenalo jejich vyřazení z množiny povolených míst pro ostatní.

7.1.6 Více směn a přestávky

Tato problematika představovala největší úskalí celého modelování. Běžně se totiž tyto situace, kdy zaměstnanec může mít dvě časová okna, řeší za pomocí naplánování tras dvou samostatných vozidel s posunutým startem rozvozu. Ačkoli se jedná o nejvíce elegantní metodu, v tomto případě ji nešlo využít. Při takovém plánování totiž nelze snadno sledovat fixní stanice a vynucovat tak jejich návštěvu. Dalším důvodem zamítnutí tohoto přístupu byly nastavení konců a začátku pauz, které v rámci 'pauz na oběd' mohly být definovatelné dvěma různými způsoby:

- Uživatel si mohl vybrat, kde pauzu začne a skončí.
- Na volbě mu nezáleželo – předpokládá se jeho pohyb v okolí poslední navštívené stanice, ze které po ukončení mezery přejíždí dále.

V případě jasně definovaných začátku a konců pauz by model sice šlo (při ignorování problematiky s fixními stanicemi) použít, selhal by však v prvním momentě, kdy by uživatel tyto informace nezadal. Solver by totiž nebyl schopen vytvořit počáteční řešení, a tak by automaticky celou úlohu označil za nemožnou.

První myšlenka, jak tuto problematiku vyřešit, byla vytvoření samostatné stanice, do které se vozidlo uchýlilo v čase pauzy, a přetrvalo v ní do doby jejího skončení. Ta byla z počátku nastavena v podobě způsobem, že čas její návštěvy byl specifikován na časové okno rozmezí pěti minut okolo počátku konce první směny. Servisní čas, tedy doba, kterou mělo vozidlo v tomto uzlu vykonávat, pak byla nastavena na délku celé pauzy. Vzdálenost této matice pak byla od všech ostatních nulová - pro možnost okamžitého přesunu do tohoto stavu, jestliže nastal čas.

Tento systém vypadal na první pohled jako správný směr pro způsob modelování, opak však byl pravdou. Umělá stanice sice fungovala správně při přesunu, její nevýhody však vyplynuly na povrch po skončení pauzy - vozidlo se následně, vzhledem k nulovým vzdálenostem, dokázalo 'teleportovat' na jakékoli místo úplně zadarmo. To bylo pro řešení nepřípustné. Znamenalo by to totiž, že by pracovník musel tuto cestu absolvovat v rámci pauzy, což nelze vyžadovat.

Z tohoto důvodu bylo přistoupeno k jiné metodě. Pauza byla do modelu vložena přímo jako součást časové dimenze pomocí funkce `AddFixedDurationBreak`, která umožňuje definovat přerušení trasy vozidla v přesně určeném čase a s pevně danou délkou.

Takto vložená pauza se následně automaticky promítá do plánování tras, aniž by bylo nutné vytvářet umělé stanice.

Tato metoda umožnila zároveň zachovat informaci o návaznosti tras před a po přerušení a mezitím řešila situaci, kdy nejsou definovány konkrétní začátky a konce pauzy. Pokud nebyly zadány, vozidlo mohlo přerušení provést v libovolné vhodné chvíli v rámci stanoveného časového intervalu. Díky tomu Solver stále dokázal nalézt realizovatelné počáteční řešení i v případě, že pauza nebyla striktně svázaná s konkrétními uzly.

7.1.7 Přiřazení úkonů podle rolí zaměstnanců

Kromě přiřazení základních parametrů bylo nezbytné také zajistit, aby na jednotlivých stanicích byly povoleny výhradně ty úkony, které měl daný technik dle zadání vykonávat.

Pro lepší ilustraci lze uvést následující příklad. Při vyplňování formuláře si uživatel zvolí, že chce během směny provádět pouze úkony typu `FIX` a `REBALANC`. Všechny ostatní činnosti, jako například `CHECK`, `BROKEN` nebo `CAMPAIGN`, by mu tedy měly být na stanicích nepřístupné.

Z předchozích částí řešení však vyplývá, že tato logika dosud zohledněna nebyla. Výpočetní model předpokládal, že všechny typy úkonů jsou přístupné všem technikům bez rozdílu. Z tohoto důvodu bylo třeba doplnit filtr, který by zakázal vykonání nevybraných úkonů na příslušných stanicích.

Jedním z důvodů, proč byly do výpočtu zahrnuty všechny úkony, byla povaha dat přicházejících z databáze. Ta poskytovala veškeré informace o stavu jednotlivých stanic, nikoli pouze ty, které byly relevantní pro konkrétního technika. V důsledku toho byly pro každou stanici generovány tzv. podstanice, a to pro všechny úkony, jejichž hodnota nebyla rovna nule. Tento přístup sice nebyl chybný, avšak mohl vést k neefektivnímu zvětšování výpočetní úlohy.

V aktuálním stavu tedy byly stanice již rozděleny dle typů úkonů. Každé takto vzniklé podstanici byla přiřazena informace o reprezentované činnosti, čímž vznikl prostor pro jejich filtrování a selektivní povolování. Zároveň tímto krokem došlo i ke snížení velikosti modelu.

Pro omezení nežádoucích návštěv stanic byly navrženy dva přístupy. V prvním z nich byla množina zakázaných uzlů rozšířena o ty, které reprezentovaly nepovolené úkony. Tento postup byl využit zejména v situacích, kdy bylo plánováno více techniků s rozdílným rozsahem oprávnění. Při vytváření podstanic se v takovém případě generovaly pouze úkony, které byly obsaženy v roli alespoň jednoho z techniků. Následně byly tyto uzly iterativně procházeny a pokud se ukázalo, že daný technik nemá pro daný úkon oprávnění, byl mu příslušný uzel zakázán. Tento způsob sice není zcela efektivní z hlediska výpočetního výkonu, jeho výhodou je však snadná implementace.

Druhý způsob spočíval v úplném vynechání stanic, které by žádný z techniků nemohl navštívit. K tomu byla využita filtrace přes všechny úkony uvedené ve vstupním

zadání. Tento přístup se ukázal být efektivní nejen díky tomu, že znemožnil návštěvu nevhodných míst (jelikož tyto stanice ve vstupních datech vůbec nefigurovaly), ale rovněž vedl k výraznému zmenšení celkové velikosti úlohy tím, že eliminoval zbytečné větvení.

7.1.8 Počáteční a koncový stav vozidla

Jak bylo už zmíněno výše, podle servisního manuálu společnosti nextbike by měla být každá směna servisního technika ukončena s prázdným vozidlem. Toto omezení však bylo následně definováno jako volitelné, neboť se mohou vyskytnout speciální případy, ve kterých není požadavek na prázdné vozidlo na konci směny relevantní.

Vzhledem k tomu, že optimalizační knihovna OR-Tools tuto podmínku nativně nezohledňuje, bylo nutné navrhnout metodiku, která by systém při hledání řešení nutil vyprazdňování vozidel zohlednit. Byly zvažovány dva základní přístupy:

1. **Měkká omezení pomocí penalizace**
   V rámci tohoto přístupu bylo povoleno, aby vozidlo směnu ukončilo s nenulovým množstvím nákladu, avšak tento stav byl penalizován. Do hodnotící funkce byla započítávána penalizace za každou jednotku nákladu, která zůstala ve vozidle po ukončení trasy. Penalizace byla určena konstantní hodnotou násobenou počtem zbývajících jednotek nákladu. Tento přístup tak směřoval k minimalizaci zbytkového nákladu, avšak v případech, kdy byla ponechání nákladu výhodnější z hlediska celkových nákladů, bylo toto řešení akceptováno.
2. **Tvrdé omezení pomocí vysoké penalizace**
   V tomto případě bylo dosaženo prakticky tvrdého omezení tím, že penalizace za jakýkoli zbylý náklad na konci směny byla nastavena na extrémně vysokou hodnotu. Tato penalizace byla zvolena tak, aby její započítání do cílové funkce bylo vždy méně výhodné než úplné vyprázdnění vozidla, bez ohledu na okolnosti. Tím bylo zajištěno, že řešení neobsahovala žádná vozidla s nenulovým nákladem na konci směny.

Oba přístupy umožňují určitou míru flexibility podle požadavků konkrétního zadání. V případě standardního provozu je preferována druhá varianta, která odpovídá firemnímu postupu. Naopak při testování nebo řešení nestandardních případů je možné využít první metodiku, jež umožňuje kompromis mezi nákladovou efektivitou a provozními požadavky.

Testována byla rovněž metodika založená na nastavení pevného omezení na nulovou hodnotu nákladu v okamžiku, kdy vozidlo dorazí na koncový uzel své trasy. Tento přístup se v průběhu několika experimentů ukázal jako efektivní, nicméně v některých případech byl optimalizačním nástrojem ignorován.

Předpokládá se, že důvod tohoto chování spočíval v nevhodném způsobu, jakým knihovna OR-Tools pracovala s takzvanou sdílenou dimenzí, jež byla využívána pro reprezentaci více typů nákladu zároveň. Bylo pozorováno, že při použití pevného omezení

byla knihovna schopna správně vynulovat pouze jeden z nákladových typů, zatímco druhý zůstal nenulový. Tato skutečnost vedla k porušení požadovaného omezení.

Lze tedy konstatovat, že ačkoliv se daný přístup v určitých scénářích osvědčil, jeho spolehlivost nebyla zcela zaručena a bylo doporučeno využívat jej s opatrností, případně kombinovat s penalizační metodikou popsanou výše.

7.1.9 Penalizace

Pro správné nastavení povolení stanic, možnosti jejich vynechání, a rozlišení priorit jednotlivých úkonů, bylo nutné navrhnout systém penalizací, který by umožňoval reflektovat všechny tyto aspekty. Vývoj této části modelu musel být opakovaně upravován, a to jak z hlediska poměru jednotlivých hodnot, tak i z důvodu nesprávně navržené a pochopené logiky v počáteční fázi implementace.

V úvodních verzích modelu bylo usilováno o pokrytí veškerých omezení výhradně prostřednictvím funkce `Disjunction`, která je v knihovně OR-Tools určena k nastavení penalizace za vynechání jednotlivých uzlů. Tento přístup je v obdobných optimalizačních modelech běžně využíván, avšak klíčovým aspektem jeho správného fungování je způsob aplikace.

V počátečním návrhu byly hodnoty penalizací přiřazovány k uzlům individuálně při každé změně modelu. Předpokládalo se, že interní logika Solveru tyto hodnoty automaticky kumuluje. Tato domněnka se však ukázala jako chybná. Na základě analýzy nesprávně generovaných výsledků a opětovného prostudování dokumentace bylo zjištěno, že funkce `Disjunction` žádnou automatickou agregaci penalizací neprovádí. Dokonce nebyla brána ani nejvyšší hodnota, nýbrž pouze ta nejnižší, která umožňovala vynechání uzlu s co nejmenší penalizací. Ačkoli tento mechanismus dává z hlediska optimalizace smysl, vedl v daném kontextu k nefunkčnímu modelu, a celá koncepce penalizací tak musela být přepracována.

Na základě této zkušenosti bylo navrženo nové řešení, které penalizuje stavy na dvou úrovních — zaprvé prostřednictvím `Disjunction`, zadruhé přímou úpravou hodnot v matici vzdáleností, tedy záměrně umělým zdražením. Pro účely nastavení pokut za vynechání stanic byla zavedena samostatná proměnná, v níž byly uchovávány příslušné hodnoty penalizací pro jednotlivé stanice. Při průchodu modelem pak byly tyto hodnoty dále modifikovány dle aktuálních požadavků, a teprve poté hromadně aplikovány. Tím bylo dosaženo jejich správné aktivace ve výpočetním jádře modelu.

Dále bylo implementováno i nastavení penalizací pro stanice, které neměly být za žádných okolností navštíveny. Požadovaného efektu bylo dosaženo přiřazením extrémně vysokých hodnot do příslušných prvků matice časů přejezdů pro dané vozidlo. Tímto způsobem bylo znemožněno efektivní využití daných tras, čímž bylo zajištěno, že nežádoucí úkony nebudou zahrnuty do výsledného řešení optimalizačního modelu.

7.2 Konečný model – rozbor kódu

Celý výpočetní `Engine` byl rozdělen do několika samostatných funkcí. Tato podkapitola je zaměřena na jejich detailní popis, způsob implementace a vzájemné propojení v rámci finálního řešení. U jednotlivých částí budou uvedeny také ukázky zdrojového kódu, na jejichž základě bude demonstrováno konkrétní provedení jednotlivých funkcionalit a jejich role v celkovém výpočetním postupu.

7.2.1 Čtení vstupních dat – init_data_model

Funkce `init_data_model` je navržena pro přípravu a strukturování vstupních dat před jejich použitím v optimalizačním výpočtu. Jsou jí předávány dva slovníky: `inputForm`, který obsahuje informace o řidičích, vozidlech a jejich preferencích, a `matrices`, v němž se nachází matice vzdáleností a časů mezi jednotlivými lokalitami a informace o těchto lokalitách.

V rámci funkce jsou vstupní data rozbalena a převedena do jednotného formátu, který je následně uložen do globální proměnné. Jsou zpracovány údaje o lokalitách (např. jejich identifikátory, požadavky na přesun kol, množství nefunkčních kol, potřebný čas na kontrolu nebo opravu apod.), informace o jednotlivých řidičích (např. jejich pracovní doba, přiřazené a zakázané lokality, počáteční a koncové pozice, případně plánované přestávky), a dále také informace o vozidlech (včetně aktuálního i maximálního nákladu a počátečního stavu).

Pro všechny lokality jsou rovněž nastaveny výchozí časové intervaly dostupnosti, a to jednotně jako celodenní okno v rozsahu `(0, 86400)` sekund. Výsledná datová struktura slouží jako základní vstup pro navazující optimalizační algoritmy.

7.2.2 Rozšíření původních dat – expand_data

Ačkoli je tato funkce vnořena do hlavní funkce na přípravu dat `create_data_model` (popsané v následující podkapitole), její detailní popis je zařazen samostatně a dříve, protože zásadním způsobem ovlivňuje strukturu dat vstupujících do optimalizačního modelu.

Primárním cílem funkce `expand_data` je transformace vstupních dat o stanicích do podrobnější reprezentace, která reflektuje specifické činnosti vykonávané na jednotlivých místech. Zatímco původní vstupní data popisují každou stanici jediným uzlem, funkce `expand_data` umožňuje rozpad stanic na tzv. "podstanice", z nichž každá reprezentuje konkrétní kolo, případně typ operace, která má na dané stanici proběhnout (např. oprava, kontrola, naložení či vyložení kol apod.).
Kromě toho tato funkce:

- filtruje nepotřebné stanice podle přiřazených rolí pracovníků;
- rozlišuje typy nákladu (funkční kola, rozbitá kola, kampaně);
- bere v potaz rozvrh pracovníků (začátek, konec směny, "pauzy na oběd");

- zohledňuje omezení daná kompetencemi jednotlivých techniků (`REBALANCE, FIX, REMOVE, CHECK, CAMPAIGN`);
- vytváří nové vzdálenostní a časové matice odpovídající nové struktuře uzlů;
- uchovává vazby mezi původními stanicemi a novými poduzly pomocí mapovacích struktur.

Vstupní proměnné:

- `dist_matrix` – čtvercová matice vzdáleností mezi stanicemi, slouží k vyhodnocení nákladů na přejezd mezi místy.
- `time_matrix` – matice jízdních časů mezi stanicemi, používá se pro modelování časových dimenzí v optimalizaci.
- `depos` – indexy stanic, které slouží jako depa (workshopu) pro opravy rozbitých kol.
- `walk_to_station` – doby docházky (např. z místa zaparkování ke stanici), přičítají se k obslužnému času na místě.
- `service_time` – délka trvání servisního zásahu (opravy) na stanici, vztahuje se k uzlům typu `REPAIR`.
- `check_time` – délka trvání kontroly stanice, vztahuje se k uzlům typu `CHECK`.
- `time_windows` – seznam dvojic (od, do) určujících časová okna, kdy lze stanici obsloužit.
- `functional_cargo` – požadavky na manipulaci s funkčními koly: kladné hodnoty značí odběr, záporné dodání.
- `broken_cargo` – požadavky na manipulaci s rozbitými koly: kladné hodnoty značí odběr, záporné dodání (do depa).
- `campaign` – počet kol na stanici pro odběr v rámci kampaně.
- `start` – indexy stanic reprezentujících začátky směn jednotlivých techniků.
- `end` – indexy stanic reprezentujících konce směn jednotlivých techniků.
- `lunch_start`, `lunch_end` – indexy stanic označujících začátek a konec pauzy na oběd, je-li definována.
- `driver_roles` – seznam rolí přiřazených jednotlivým technikům; určuje, jaké typy úkonů mohou vykonávat.

Na základě těchto vstupů je každá stanice rozšířena do více uzlů, pokud se na ní má vykonávat více operací. Příkladem může být stanice, na níž probíhá jak oprava, tak kontrola, případně je zde zadáno více typů nákladu. Taková stanice je rozdělena na více podstanic, které jsou následně zpracovávány samostatně v optimalizačním modelu.

Součástí procesu je i přizpůsobení časových oken a servisních časů každému nově vzniklému uzlu, stejně jako úprava matice vzdáleností a časových přechodů tak, aby nové podstanice byly integrovány konzistentně.

Výstupní proměnné:

- `index_mapping` – Slovník, který ke každé původní stanici přiřazuje seznam indexů jejích rozpadlých uzlů.
- `ex_dist_matrix` – Nová matice vzdáleností mezi všemi rozšířenými uzly (včetně nových aktivit).
- `ex_time_matrix` – Analogická matice časové náročnosti pro přesun mezi jednotlivými uzly.
- `ex_walk_to_station` – Časy potřebné na pěší přesun k jednotlivým stanicím od míst nejbližších možných míst pro parkování.
- `ex_service_time` – Servisní časy specifikované pro každý nový uzel.
- `ex_time_windows` – Upravená časová okna podle nových požadavků na jednotlivé aktivity.
- `new_functional_cargo` – Počet funkčních kol přiřazený jednotlivým podstanicím (kladné hodnoty pro vyzvednutí, záporné pro vykládku).
- `new_broken_cargo` – Stejně jako předchozí, ale pro rozbitá kola a kampaně, které jsou do této kategorie zařazeny taktéž.
- `roles` – Typ operace, která má být na daném uzlu vykonána (`CHECK`, `FIX`, `REBALANCE`, `REMOVE`, `CAMPAIGN`, `START`, `END`, `LUNCH_START`, `LUNCH_END`, `DEPO`).

Tato transformace vytváří jednotné rozhraní pro optimalizační model, který pak může efektivně plánovat trasy a úlohy více vozidel a techniků, přičemž respektuje časová i kapacitní omezení, typ nákladu i specializaci pracovníků. Hlavní výhodou tohoto systému je především možnost částečného plnění úkolu typu `REMOVE`, `REBALANCE` a `CAMPAIGN`.

7.2.3 Příprava dat pro výpočet – create_data_model

Klíčovým krokem při formulaci optimalizačního problému je vytvoření rozšířeného datového modelu, který reflektuje všechny požadavky úlohy, včetně časových oken, typů nákladu, rolí jednotlivých stanic a specifických omezení pro vozidla. K tomu slouží funkce `create_data_model`, která vstupně přijímá strukturovaný slovník `inputForm` a předvypočítané matice vzdáleností a časů v parametru `matrices`.

Funkce nejprve načte základní data pomocí volání `init_data_model` a následně provede jejich rozšíření pomocí funkce `expand_data`. Tento krok zajišťuje, že každá stanice je rozdělena do více uzlů dle typu operace, která na ní probíhá (např. nakládka, vykládka, kontrola, kampaň atd.).

Následně dochází k přemapování klíčových indexů, jako jsou startovní a koncové uzly tras (`starts`, `ends`) či body přestávky na oběd (`lunch_start`, `lunch_end`) tak, aby odpovídaly nově rozšířenému modelu. Každý z těchto bodů je vybrán na základě specifické role, která je přiřazena vektoru `roles`.

Časová okna pro oběd jsou nastavena dynamicky podle pracovní doby řidiče s tolerancí ±5 minut. Tím je zajištěna dostatečná flexibilita plánování přestávek, aniž by docházelo k narušení ostatních operací.

Funkce také inicializuje několik pomocných struktur, jako jsou penalizace uzlů (`penalty_to_nodes`) a disjunkce (`disjunctions`), které jsou připraveny pro každý konkrétní vůz.

Zvláštní důraz je kladen na přiřazení povinně navštívených uzlů pro každé vozidlo, které se odvozují od `fixed_location`. V tomto kroku jsou vybírány takové uzly, jejichž role odpovídá požadovaným činnostem přiřazeným jednotlivým řidičům. Stejně tak se vylučují uzly, které nesmí být navštíveny konkrétními vozidly (`banned_locations`). Konečný seznam zakázaných uzlů vzniká spojením uzlů, které jsou mimo definovaný rozsah daného vozidla, s povinně navštívenými uzly jiných vozidel.

Dále se sestavuje seznam všech speciálních uzlů, které tvoří počátky, konce a body přestávek všech tras (`all_start_end`), a samostatně se identifikují uzly s nulovým servisním časem mimo tyto speciální body (`no_service_time`).

Výsledný datový model je bohatě strukturovaný a obsahuje veškeré informace potřebné pro formulaci a řešení úlohy pomocí OR-Tools. Vrací se jako slovník `data`, který zahrnuje mimo jiné i rozšířené matice vzdáleností a časů, role, náklady, časová okna a logiku pro více vozidel a směn.

7.2.4 Výpočetní jádro – main

Stěžejní součástí implementace modelu pro optimalizaci rozvozu jízdních kol je funkce `main()`, která zajišťuje sestavení a konfiguraci celého optimalizačního modelu pomocí knihovny Google OR-Tools. Cílem modelu je nalézt trasu jednoho či více vozidel přepravujících dva typy nákladu – funkční a rozbitá kola – mezi jednotlivými stanicemi v síti.

Výsledná trasa musí splňovat řadu omezení, zejména omezenou kapacitu vozidel, časová omezení (například pracovní dobu řidičů či otevírací dobu stanic) a specifické požadavky na jednotlivé typy stanic (například stanice určené výhradně k opravám nebo překládce).

Inicializace datového modelu a základního objektu OR-Tools

V rámci řešení optimalizační úlohy vozidlového rozvozu se jako první krok v hlavní funkci `main()` připraví datový model, který obsahuje veškeré vstupní informace potřebné k definici problému – funkce `create_data_model`. Tyto informace zahrnují například matici vzdáleností mezi jednotlivými uzly, počet vozidel, startovní a koncové pozice každého vozidla, kapacity, časová okna a další parametry specifické pro danou úlohu. Datový model je reprezentován ve formě slovníku, který umožňuje centralizovanou správu všech vstupních dat.

Následuje vytvoření objektu `RoutingIndexManager`, který slouží jako převodník mezi indexy uzlů, jak jsou definovány ve vstupních datech, a interními indexy používanými optimalizační knihovnou Google OR-Tools. Tento krok je zásadní zejména v případech, kdy je řešeno více vozidel s různými startovními a koncovými body, protože umožňuje správně mapovat externí reprezentaci uzlů na interní systém optimalizace. Konkrétně `RoutingIndexManager` přijímá jako vstup počet všech uzlů, počet vozidel a seznam startovních a koncových uzlů.

Poté se inicializuje hlavní optimalizační objekt `RoutingModel`, který je jádrem řešení problému. Tento objekt přijímá jako parametr právě vytvořený `RoutingIndexManager` a slouží k definování samotné optimalizační úlohy (Obr. 7). Do tohoto modelu budou následně přidána různá omezení, jako jsou kapacity vozidel, časová okna, požadavky na nakládku a vykládku, stejně jako cílová funkce optimalizace, například minimalizace celkové délky tras nebo doby jízdy.

Tímto způsobem je zajištěna správná struktura dat a příprava modelu pro další kroky optimalizace, které budou provedeny v následujících částech funkce `main()`.

```
# Create the routing index manager
manager = pywrapcp.RoutingIndexManager(
    len(data['distance_matrix'][0]),
    data['num_vehicles'],
    data['starts'],
    data['ends'],
)

# Create Routing Model
routing = pywrapcp.RoutingModel(manager)
```

Obr. 7: Vytvoření modelu

Sledování kapacity vozidel podle typů nákladu

Pro správné řízení nákladů a kapacit vozidel je nezbytné v rámci optimalizačního modelu definovat jednotlivé nákladové dimenze. Tyto dimenze reprezentují různé typy nákladu, které mohou být během rozvozu manipulovány – například funkční a rozbitá kola, nebo jiné specifické kategorie nákladu.

V objektu `RoutingModel` byly proto vytvořeny dimenze, které odpovídají kapacitním omezením jednotlivých typů nákladu. Každá dimenze sleduje množství nákladu určitého typu, které je aktuálně ve vozidle, a zároveň zajišťuje, že během trasy nebude překročena maximální kapacita vozidla pro daný náklad.

Definice těchto dimenzí zahrnuje nastavení horního limitu kapacity, tedy maximálního možného množství nákladu, které může být ve vozidle současně. Dále lze modelovat změny množství nákladu při průjezdu jednotlivými uzly, což umožňuje simulovat procesy nakládky a vykládky.

Díky těmto dimenzím je možné přesně kontrolovat stav vozidla po průjezdu každým uzlem, zabránit přetížení a zároveň zajistit, že požadavky jednotlivých stanic na

množství vyloženého či naloženého zboží budou splněny v souladu s kapacitními omezeními. Nákladové dimenze tak tvoří klíčovou součást modelu pro realistické a efektivní plánování tras.

V tomto konkrétním modelu byly definovány tři dimenze – dvě samostatné pro `functional_cargo` (funkční kola určená k redistribuci) a `broken_cargo` (rozbitá kola a kola s kampaní), a jedna společná, která sleduje součet obou typů nákladu a slouží jako kontrola, aby nedošlo k překročení celkové kapacity.

Jako ukázka implementace dimenze a jejího nastavení slouží dimenze sledující celkovou kapacitu vozidla. Pro získání hodnot nákladu v jednotlivých uzlech je využita tzv. *callback funkce* `combined_capacity_callback`. Tato funkce přijímá index aktuálního uzlu na trase, který pomocí správce indexů (`manager.IndexToNode`) překládá na odpovídající uzel v datovém modelu. Poté načte množství funkčního i rozbitého nákladu přiřazeného tomuto uzlu a vrátí jejich součet jako požadavek na kapacitu.

Callback je registrován v rámci routing modelu jako jednorozměrný transit callback, který poskytuje informace o požadavku na kapacitu pro každý navštívený uzel.

Dále je pomocí funkce `AddDimensionWithVehicleCapacity` do modelu přidána dimenze pojmenovaná `Capacity`. Tato dimenze využívá registrovaný callback pro vyhodnocení aktuálního stavu nákladu ve vozidle. Parametry této funkce specifikují, že kapacita nemá žádnou rezervu (slack = 0), maximální kapacita vozidla je nastavena hodnotou `data['vehicle_capacity']`, a počáteční kumulativní hodnoty nejsou vynulovány (start cumul = False).

Nakonec je instance této dimenze získána pomocí `GetDimensionOrDie`, což umožňuje pozdější přístup k této dimenzi pro další omezení nebo analýzu (Obr. 8).

```python
# Controling of the combined capacity
def combined_capacity_callback(from_index):
    node = manager.IndexToNode(from_index)
    return (data['functional_cargo'][node] + data['broken_cargo'][node])

demand_callback_index = routing.RegisterUnaryTransitCallback(combined_capacity_callback)

# Add capacity dimension
routing.AddDimensionWithVehicleCapacity(
    demand_callback_index,
    0,                                  # null capacity slack
    data['vehicle_capacity'],           # vehicle maximum capacity
    False,                              # start cumul to zero
    'Capacity')                         # name of dimension

capacity_dimension = routing.GetDimensionOrDie("Capacity")
```

Obr. 8: Ukázka nastavení dimenze (společná dimenze pro celkovou kapacitu)

Dalším důležitým prvkem pro správné fungování modelu bylo nastavení výchozího stavu nákladu ve vozidlech na počátku trasy. V některých situacích totiž může vozidlo začínat trasu již částečně naložené – například pokud převzalo část nákladu z předchozí

směny, nebo pokud bylo ručně doplněno. Pro tyto případy bylo do modelu implementováno omezení počátečního stavu jednotlivých dimenzí (Obr. 9).

Nastavení probíhá přímo nad kumulativní proměnnou (`CumulVar`) příslušné dimenze v místě startu trasy daného vozidla. Pro každé vozidlo je tak zajištěno, že jeho počáteční množství funkčního i rozbitého nákladu odpovídá předem zadaným hodnotám. Tyto hodnoty jsou v aktuální implementaci získávány ze vstupního formuláře, kde je uživatel může zadat manuálně na základě reálné situace.

```
# Set cargo into vehicle before start
for vehicle_id in range(data['num_vehicles']):
    start_id = routing.Start(vehicle_id)
    functional_load_dimension.CumulVar(start_id).SetValue(data['current_functional'[vehicle_id])
    broken_load_dimension.CumulVar(start_id).SetValue(data['current_broken'][vehicle_id])
```

Obr. 9: Nastavení počátečního nákladu

Kromě nastavení výchozího stavu nákladu ve vozidlech je pro realistické plánování nutné také specifikovat požadovaný stav na konci trasy. V některých případech je například žádoucí, aby vozidlo dorazilo zpět na základnu prázdné, případně aby s sebou vezlo maximálně omezené množství konkrétního typu nákladu. To může být důležité z hlediska plánování další směny nebo kvůli logistickým omezením skladování.

Tento požadavek je v modelu řešen pomocí tzv. měkkých horních limitů (*soft upper bounds*), které umožňují určit maximální požadovanou hodnotu dané dimenze na konci trasy – a zároveň definovat penalizaci za její překročení. Model se tak může rozhodnout omezení porušit, ale pouze za cenu výrazného zhoršení výsledné optimalizační funkce.

Následující kód (Obr. 10) popisuje, jak je toto omezení implementováno pro dimenzi celkové kapacity. V případě, že má být vozidlo na konci prázdné, je pro příslušný konečný uzel nastavena horní mez s penalizací. Pokud je ponecháno právo skončit s určitou zbytkovou kapacitou, je použita nižší penalizace.

```
#Penalize the capacity violation on the end of the route (broken cargo)
for vehicle_id in range(data['num_vehicles']):
    if data['finish_empty'][vehicle_id]:
        end_index = routing.End(vehicle_id)
        penalty_per_bike = 10000000 if data["finish_empty"][vehicle_id] else 35000
        capacity_dimension.SetCumulVarSoftUpperBound(
            end_index,                      # index of the end node
            0,                              # wanted remaining capacity
            penalty_per_bike)               # penalty coefficient (for each unit over limit)
```

Obr. 10: Nastavení penalizace za překročení limitu na konci trasy

Penalizace podle typu uzlu a role vozidla

Pro dosažení požadovaného chování modelu je klíčové vhodně nastavit ohodnocení za nenavštívení určitých typů uzlů. Každý uzel může mít přiřazenu specifickou roli (`CHECK`, `FIX`, `REBALANCE`, `CAMPAIGN`, `REMOVE`), a každé vozidlo má předem definováno, které role má být schopno obsloužit.

U uzlů typu `LUNCH_START` a `LUNCH_END`, které značí začátek a konec pauzy, se přidává penalizace, pro motivaci modelu tato místa řadit do trasy, pokud k nim existuje relevantní časový úsek. Toto pokutování je individuální pro každé vozidlo (Obr. 11).

```
for vehicle_id in range(data['num_vehicles']):
    for node in range(len(data['functional_cargo'])):
        if (node not in data['starts'] and
            node not in data['ends'] and
            node not in data['depos'] and
            data['roles'][node] == 'LUNCH_START' or
            data['roles'][node] == 'LUNCH_END'):
            data['disjunctions'][vehicle_id][node] += 5000000
```

Obr. 11: Penalizace uzlů pro obědové pauzy

Hlavní logika přidělování penalizací podle rolí uzlů a kompetencí vozidel probíhá následovně (Obr. 12). Nejprve jsou definovány úrovně penalizací pro jednotlivé typy úkonů. Poté se kontroluje, zda je daný uzel přístupný danému vozidlu – pokud ano, je přiřazena mírná penalizace, pokud ne, je uzel označen jako prakticky nedosažitelný.

```
roles = ['CHECK', 'FIX', 'RE_BALANCE', 'CAMPAIGN', 'REMOVE']
penalty_values = {'CHECK': 30000, 'FIX': 50000, 'RE_BALANCE': 75000,
                  'CAMPAIGN': 100000, 'REMOVE': 750000}

for vehicle_id in range(data['num_vehicles']):
    for role in roles:
        for node, node_role in enumerate(data['roles']):
            if node in data['starts'] or node in data['ends'] or node in
data['depos']:
                continue
            if node_role == role:
                if role in data['role'][vehicle_id]:
                    data['disjunctions'][vehicle_id][node] += penalty_values[role]
                else:
                    data['penalty_to_nodes'][vehicle_id][node] = INF
```

Obr. 12: Penalizace jednotlivých uzlů (podstanic)

Zvláštní případ tvoří stanice, u kterých se výslovně neuvažuje doba servisu – tzv. `no_service_time` uzly. Ty jsou rovněž trestány nekonečnou hodnotou, čímž se zamezuje jejich zařazení do trasy, jelikož tato místa nereprezentují žádné relevantní aktivity (Obr. 13).

```
for node in data['no_service_time']:
    data['penalty_to_nodes'][vehicle_id][node] = INF
```

Obr. 13: Penalizace uzlů s nulovými úkony

Naopak uzly reprezentující workshopy (`DEPOS`) nejsou penalizovány vůbec. Jejich nenavštívení model nijak neomezuje, pokud to není v rozporu s jinými podmínkami úlohy.

Díky tomuto přístupu lze velmi flexibilně řídit, které typy uzlů budou preferovány či naopak vyloučeny z trasy jednotlivých vozidel. Tímto způsobem je možné model přizpůsobit jakýmkoliv provozním omezením nebo požadavkům na kompetence jednotlivých pracovníků.

Časová dimenze a pracovní okna

Časová dimenze modelu slouží k řízení posloupnosti událostí v čase. Zohledňuje jak cestovní časy mezi jednotlivými uzly, tak i náročnosti obsluhy strávené na stanicích. Tento přístup umožňuje věrně modelovat provozní realitu, kde určitou dobu zabírá nakládka, vykládka nebo jiná činnost.

Součástí časové dimenze je také možnost tzv. čekání, kdy vozidlo může na stanici dorazit dříve a vyčkat na začátek platného časového okna — např. z důvodu přestávky nebo otevírací doby. Časová dimenze není navázána na pevný začátek (čas nula), ale respektuje individuální striktní startovní čas každého vozidla, který odpovídá reálnému výjezdu jeho naplánované směny. Stejně tak je pevně určen konec trasy, typicky omezen pracovní dobou. Tyto časy jsou nastavovány dále ještě explicitně na počátečních a koncových uzlech.

U všech běžných uzlů (s výjimkou počátečních a koncových) jsou tak v modelu definována časová okna, ve kterých je možné stanici obsloužit. Tím je možné zohlednit např. otevírací dobu, dostupnost pracovníků nebo technologická omezení na pořadí operací.

Díky zavedení časové dimenze je možné model rozšířit i o další pokročilé prvky, jako jsou pauzy (např. obědové přestávky), synchronizace více vozidel na určitém uzlu, nebo cílení návštěv v konkrétních časech během dne.

V tomto typu dimenze lze také efektivně využít principu "zamezení návštěvy nevhodných uzlů"prostřednictvím manipulace s časem. Konkrétně lze nastavit velmi vysoké hodnoty cestovních nebo servisních časů tak, aby se uzel stal pro vozidlo časově nedosažitelným. Pokud by v takovém případě návštěva uzlu vedla k překročení časového horizontu trasy nebo porušení časového okna, stane se tento uzel pro model neproveditelný (*infeasible*).

Tento přístup je široce využíván pro selektivní vypínání uzlů, které nemají být zahrnuty do výsledného řešení, aniž by bylo třeba explicitně omezovat rozhodovací prostor modelu. Oproti jiným metodám, jako jsou penalizace neobsloužení, nabízí tento přístup

vyšší kontrolu nad dosažitelností stanic a přirozeněji reflektuje reálné omezení — např. kvůli vzdálenosti, dostupnosti nebo pracovní době.

Ve výsledku se tak jedná o flexibilní a výkonný mechanismus, který je dobře integrován do časové dimenze bez nutnosti dalších strukturálních úprav modelu.

7.2.5 Přestávky během směny (Breaks)

Pokud má vozidlo definované dvě pracovní části směny (například dopolední a odpolední blok), je mezi těmito částmi automaticky vkládána pevná přestávka. Tato pauza je realizována jako pevně daný interval, během kterého vozidlo nemůže vykonávat žádnou činnost. Pauza je modelována pomocí objektu `FixedDurationIntervalVar` a následně přiřazena do časové dimenze každého vozidla.

Výpočet intervalu pauz probíhá za kontroly, zda má zaměstnanec definované dvě pracovní časová okna. Pokud ano, vypočítá se volný časový interval mezi nimi a určí se začátek přestávky. Tento interval je pak uložen do datové struktury modelu (Obr. 14).

```python
data['vehicle_break_intervals'] = {}
for v in range(data['num_vehicles']):
    if len(data["work_hours"][v]) == 2:
        vehicle_break_window = data['work_hours'][v]
        time_or = max(vehicle_break_window[1][0] - vehicle_break_window[0][1], 0)
        start_time = max(vehicle_break_window[0][1] - 10, 0)
        data['vehicle_break_intervals'][v] = (start_time, start_time + time_or)
    else:
        data['vehicle_break_intervals'][v] = None
```

Obr. 14: Kontrola, zda má vozidlo dvě časová okna pro směny

Zavedení takto získaných přestávek je pak do modelu vloženo pomocí metody `SetBreakIntervalsOfVehicle` (Obr. 15), která očekává seznam časových intervalů pro každé vozidlo. Pauza má pevně daný čas začátku a délku, přičemž model ji považuje za povinnou součást trasy vozidla.

```python
def add_breaks(routing, manager, data):
    node_visit_transit = {index: int(data['service_time'][manager.IndexToNode(index)]) for index in range(routing.Size())}
    for v in range(data['num_vehicles']):
        if len(data["work_hours"][v]) == 2:
            vehicle_break_window = data['work_hours'][v]
            start_time = vehicle_break_window[0][1]
            end_time = vehicle_break_window[1][0]
            break_duration = end_time - start_time

            break_interval = routing.solver().FixedDurationIntervalVar(
                start_time, start_time, break_duration, False, f'Break for vehicle {v}'
            )

            time_dimension.SetBreakIntervalsOfVehicle([break_interval], v, node_visit_transit.values())
```

Obr. 15: Vložení pauz do modelu

Jak bylo zmíněno výše, model umožňuje kromě definování přestávek jako pevného časového intervalu také nasměrovat vozidla tak, aby tyto pauzy strávila v konkrétních lokalitách. Toho je dosaženo využitím umělých stanic, typicky označených jako `LUNCH_START` a `LUNCH_END`, jejichž časová okna jsou nastavená těsně v oblasti počátku pauz. Spolu s vysokou hodnotou penalizace je tak tento systém skvělým způsobem, jak definovat strávení přestávek i v daných oblastech.

Modelování povinných a zakázaných návštěv uzlů

Model umožňuje velmi flexibilně řídit, které uzly mají, nebo naopak nesmí, být navštíveny jednotlivými vozidly. K tomu slouží mechanismus tzv. *disjunkcí* (Obr. 16), které definují penalizace za nenavštívení daného uzlu. Tento přístup umožňuje:

- **Vynutit návštěvu uzlu** nastavením vysoké penalizace – model se pokusí takový uzel zařadit do trasy, aby minimalizoval celkové náklady.
- **Zakázat návštěvu uzlu** nastavením nulové penalizace – uzel je volitelný, ale vzhledem k jiným omezením (např. zákaz vozidla na uzlu, navýšením časové náročnosti nad rámec úlohy) bude vynechán.
- **Rozlišit mezi obecnými a individuálními požadavky** – obecně požadované návštěvy mají nižší penalizaci, než ty specifické pro konkrétní vozidlo.

Zároveň lze nastavovat zákaz přístupu konkrétního vozidla do určitých lokalit pomocí metody `SetAllowedVehiclesForIndex`, čímž se zabrání vytvoření nepřípustného řešení s nedovolenou návštěvou.
Pro jemnější ovlivnění návštěvnosti lze navíc modifikovat i samotné distanční, nebo časové matice:

- Uzel, který má být penalizován, má zvýšené náklady na příjezd i odjezd pomocí `penalty_to_nodes`.
- Speciální pozornost je věnována kontrole indexů, aby nedošlo k chybám při úpravách matic.

```python
for vehicle_id in range(data['num_vehicles']):
    for node, penalty in enumerate(data['disjunctions'][vehicle_id]):
        if node not in data['starts'] and node not in data['ends']:
            routing.AddDisjunction([manager.NodeToIndex(node)], penalty)
```

Obr. 16: Nastavení disjunkcí pro stanice

Výsledný harmonogram

K nalezení výsledného řešení je využit konfigurovatelný vyhledávací algoritmus, jehož chování lze ovlivnit pomocí uživatelsky definovaných parametrů. Nejprve je zvolena počáteční strategie (viz Tab. 1), která slouží k nalezení výchozího řešení jako základu pro následnou optimalizaci. Poté může být aktivována pokročilá metoda lokálního prohledávání (viz Tab. 2), která v rámci zadaného časového limitu postupně vylepšuje stávající řešení. Volitelně lze také zobrazit logování průběhu vyhledávání, což umožňuje detailní sledování procesu a vyhodnocení dosažených výsledků. Samotné vyhledání řešení je realizováno voláním metody s příslušně nastavenými parametry, přičemž výsledkem je také návratový status informující o úspěšnosti výpočtu. Více o způsobu volby tohoto nastavení je popsáno v kapitole 7.3.

7.2.6 Zpracování výsledku – data_into_json

Výsledné řešení bylo nutné dále distribuovat do systému. Nejvhodnějším formátem pro přenos dat se opět ukázal být formát JSON. Funkce `data_into_json` slouží k převedení výstupu do navržené struktury, která je uzpůsobena jak pro uložení do databáze, tak pro další zpracování na straně frontendu.

Struktura celého objektu vychází ze stejného principu jako vstupní data ve formátu `InputForm` – tedy jako slovník (*dictionary*). Návrh obou forem JSON objektů byl rovněž součástí této práce. Jejich výsledná podoba se v průběhu vývoje několikrát měnila s ohledem na potřeby frontendu a pro lepší přehlednost a uchopitelnost dat.

Při návrhu exportní logiky bylo zároveň nutné vyřešit i některé specifické problémy, které nelze přímo řešit v rámci výstupu z OR-Tools a vyžadují vlastní metodiku zpracování.

Jedním z klíčových problémů bylo rozdělení směn v případech, kdy bylo plánováno více pracovních oken. Výstup OR-Tools totiž nerozlišuje jednotlivé úseky práce, a proto bylo potřeba sledovat časové průběhy tras. Pokud došlo k překročení časové hranice směny nebo k návštěvě speciální stanice ohraničující pauzu, bylo nutné výstup ručně rozdělit do odpovídajících směnových částí.

7.3 Kontrola výsledků a výběr nejlepších nastavení

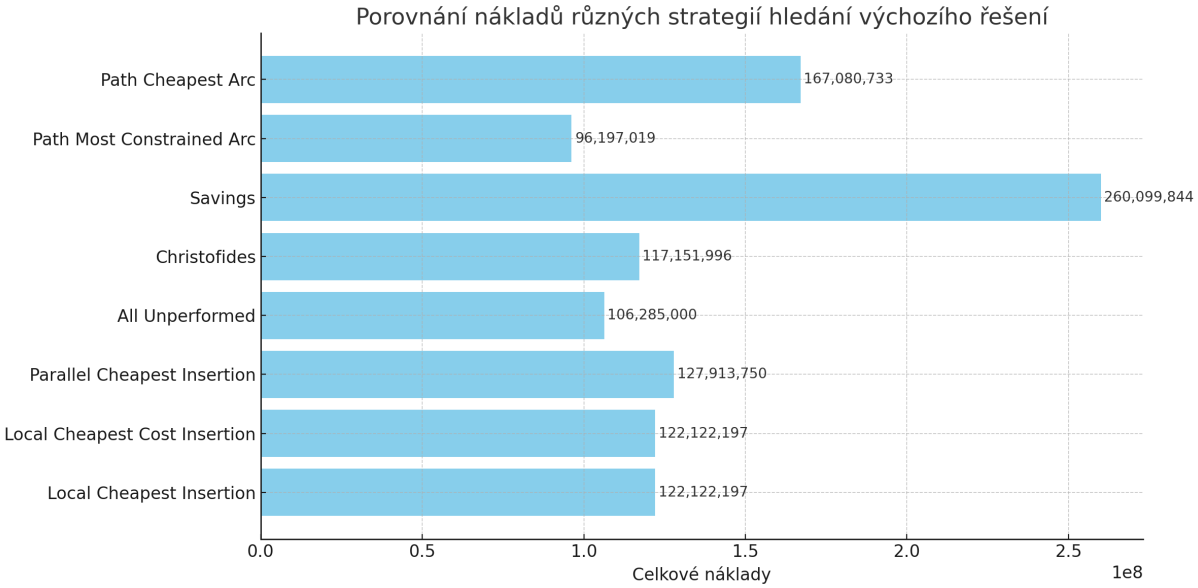
Pro ověření funkčnosti a dosažení co nejlepších výsledků byly prováděny různé testy v rámci validace a kontroly kvality modelu. Testovací úloha byla navržena pro jedno vozidlo a obsahovala následující specifikace:

- 214 stanic na území města Brna, přičemž všechny byly nastaveny jako povinné (s cílem ověřit schopnost modelu naplnit všechny požadavky a v případě překročení časového limitu některé stanice vynechat),

- aktuální stav stanic ke dni 16. 4. 2025, včetně informací o dostupných kolech (zajištění opakovatelnosti experimentu),
- jedno vozidlo s maximální kapacitou 20 jednotek nákladu, s nulovým nákladem na začátku i na konci trasy,
- časová a vzdálenostní matice přejezdů mezi odpovídajícími stanicemi,
- přiřazení všech typů úkonů jednotlivým stanicím (např. redistribuce, svoz rozbitých kol apod.),
- maximální čas pro výpočet nastavený na jednu minutu (odpovídající předpokládanému omezení při nasazení do reálného provozu),
- počáteční, koncová i přestávkové stanice umístěny do uměle vytvořeného depa v lokalitě u Technického muzea (tuto stanici si musí uživatel v aplikaci nastavit v době psaní této práce manuálně, jelikož o ní v databázi prozatím neexistují data).

Cílem těchto testů bylo dosáhnout co nejnižší hodnoty nákladové funkce (*cost function*), přičemž výstup musel být zároveň logický a smysluplný i při vizuálním ověření výsledné trasy a provedených úkonů.

Pro testování byly využity všechny metody pro hledání počátečních řešení (*first solution strategies*) dostupné v knihovně OR-Tools. Některé strategie se však nedostaly do finálního grafického přehledu (viz obr. 17), neboť za stanovený časový limit nebyly schopny vytvořit žádné validní počáteční řešení.



Obr. 17: Porovnání strategií pro hledání počátečních řešení

Z výsledků je patrné, že volba strategie pro počáteční řešení má zásadní vliv na kvalitu finálního výsledku. Rozdíly mezi jednotlivými metodami byly často výrazné, a proto byly pro další testování vybrány pouze dvě nejperspektivnější strategie:

- `PATH_MOST_CONSTRAINED_ARC` – metoda s nejnižší dosaženou hodnotou nákladové funkce,

- `PARALLEL_CHEAPEST_INSERTION` – metoda, která sice nedosahovala nejlepší hodnoty, avšak generovala vizuálně rozumné trasy s logickým sledem úkonů.

Následně bylo provedeno další testování, zaměřené na volbu vhodného algoritmu pro lokální prohledávání. Pro obě vybrané strategie počátečního řešení byly otestovány všechny dostupné metody lokálního prohledávání. Výsledky byly překvapivě konzistentní – při stejném čase běhu modelu vykazovaly všechny strategie stejnou hodnotu nákladové funkce. Z toho lze usuzovat, že hlavní rozdíly ve výsledcích jsou primárně dány volbou počátečního řešení.

Z tohoto důvodu byly pro další použití preferovány dvě metody:

- `AUTOMATIC` – univerzální volba, která umožňuje OR-Tools zvolit optimální metodu na základě vstupních dat,
- `GREEDY_DESCENT` – metoda, která jako jediná ukončuje běh v okamžiku, kdy není možné nalézt další zlepšení, čímž zajišťuje rychlé konvergence bez zbytečných iterací.

`GREEDY_DESCENT` nedisponuje pamětí navštívených stavů, a proto je náchylnější k uvíznutí v lokálním minimu. V rámci testovaných scénářů však tento problém nenastal. Přesto nelze vyloučit, že při budoucím nasazení v provozu bude třeba zvážit úpravu metodiky podle specifických podmínek dané úlohy.

7.4 Zpracování dat z ForecastService

Úvod k této službě je obsažen v kapitole 6.6. Pro připomenutí, služba zpracovává historická data výpůjček kol a na jejich základě vytváří odhad poptávky a nabídky kol pro jednotlivé stanice v rámci časových úseků během dne.

Model slouží jako klíčový nástroj pro efektivnější plánování distribuce kol mezi stanicemi. Jeho implementace a zpracování dat představují nezbytný krok pro zlepšení plánování směn a řízení redistribuce v systému sdílení kol. Pro správnou interpretaci výsledků bylo nutné detailně porozumět povaze a struktuře vstupních i výstupních dat.

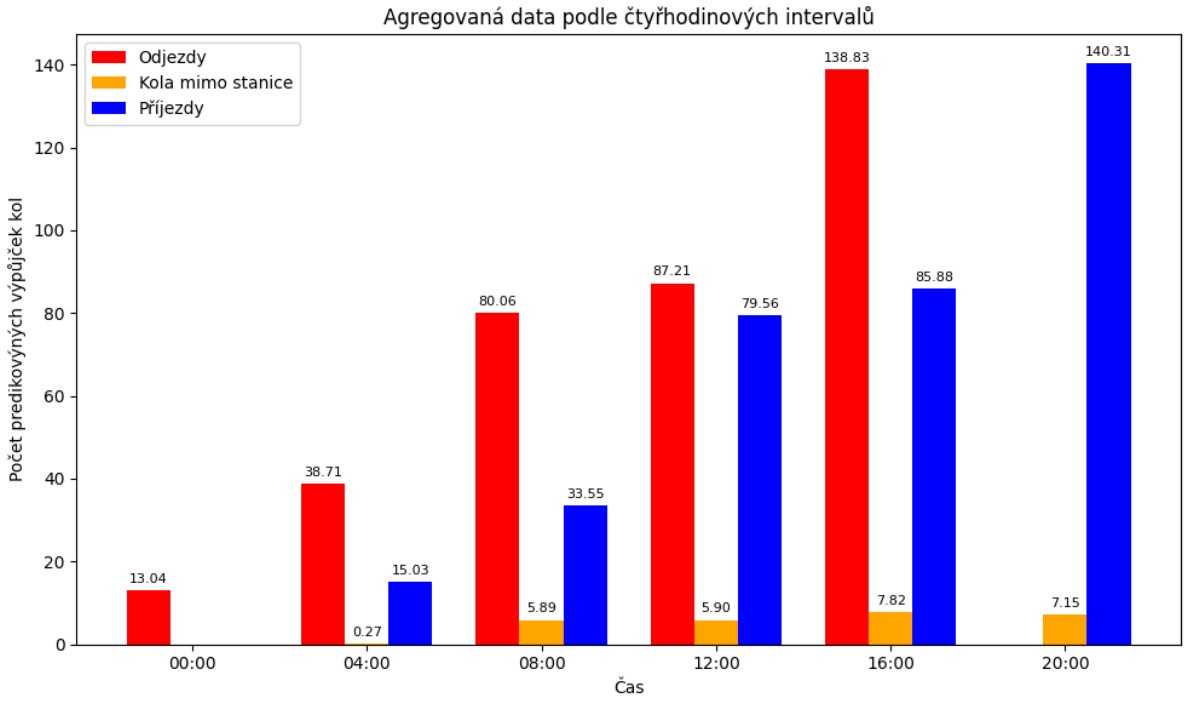
Výstupem této služby je slovník obsahující čtyři klíče, jejichž význam a struktura jsou následující:

- **pole `o` (odjezdy)** – Tato třírozměrná matice popisuje predikovaný počet kol, která by měla být v jednotlivých časových intervalech vypůjčena z daných stanic. Rozměry pole jsou `[224, 7, 96]`, kde:
  - 224 odpovídá počtu sledovaných stanic,
  - 7 reprezentuje dny v týdnu (pondělí až neděle),
  - 96 značí počet časových intervalů během dne, přičemž den je rozdělen do 15minutových úseků.
- **pole `p` (příjezdy)** – Pole se stejnou strukturou jako `o`, ale popisuje predikovaný počet kol, která by měla v jednotlivých intervalech na stanice přijít (být vrácena).

- **pole `mis` (kola mimo stanice)** – Jednorozměrné pole o délce 96, které udává odhad počtu kol, která jsou v daném časovém intervalu momentálně mimo stanice, tedy vypůjčená a nedostupná pro další distribuci či výpůjčky.
- **slovník `seznamv`** – Obsahuje mapování identifikátorů stanic na jejich indexy v ostatních polích (`o` a `p`), čímž umožňuje jednoznačnou identifikaci dat příslušejících ke konkrétní stanici.

Díky tomuto modelu lze tedy získat detailní predikci, kdy a kde je potřeba zvýšit nebo naopak snížit počet kol, což umožňuje plánovat efektivnější rozvoz a vyvažování kol mezi stanicemi. Predikce zároveň reflektují sezónní i denní variabilitu v poptávce a nabídce, což je nezbytné pro robustní a flexibilní plánování provozu.

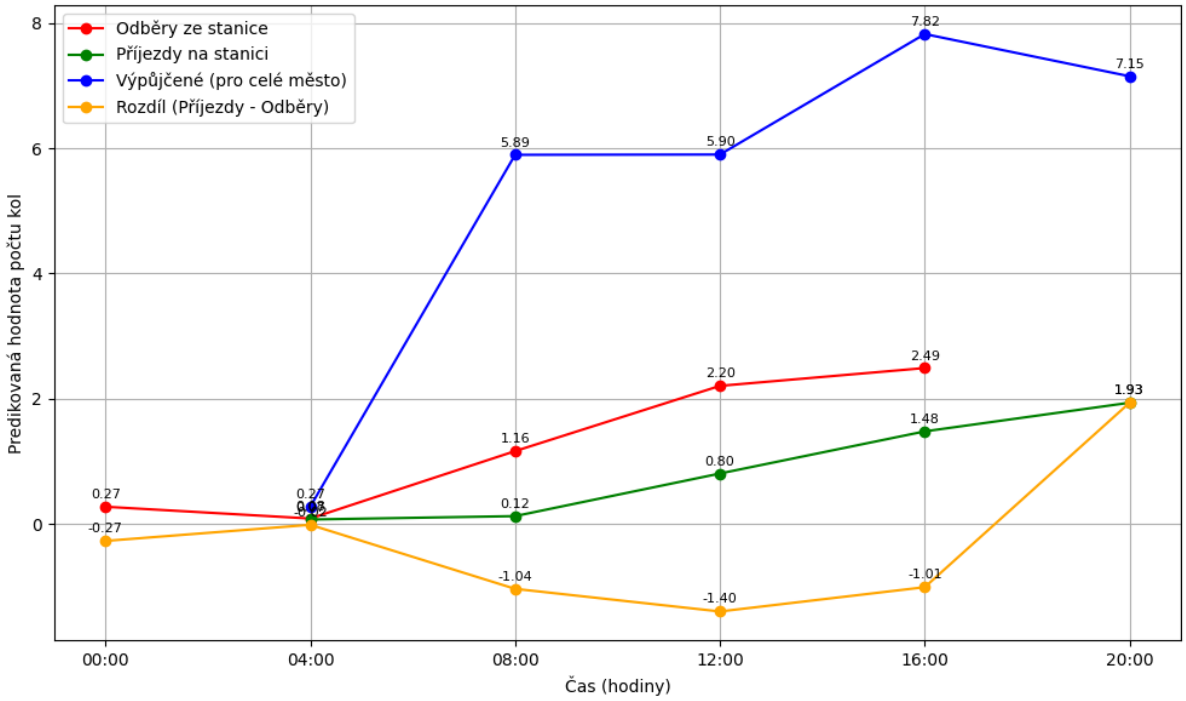
V rámci práce byla data o výpůjčkách a vrácení kol agregována do čtyřhodinových časových intervalů, což umožnilo podrobnější sledování časových trendů při současném snížení výpočetní náročnosti zpracování. V grafu (Obr. 18) jsou zobrazeny souhrnné hodnoty těchto agregovaných dat za všech 224 stanic na území města Brna. Po sečtení hodnot za jednotlivé časové intervaly lze konstatovat, že průměrně za celý den proběhne přibližně 380 výpůjček. Počet vrácených kol zpět na stanice je obdobný, rozdíly vznikají v důsledku výpůjček přesahujících půlnoc. Průměrná hodnota výpůjček je tedy 15 kol za hodinu. Zároveň lze pozorovat, že průběh vracení kol je oproti výpůjčkám posunut o určitý časový interval, což odpovídá reálnému toku pohybu kol v systému. Tento posun je klíčovým prvkem pro analýzu dynamiky distribuce a slouží jako důležitý podklad pro plánování převozů kol mezi stanicemi.



Obr. 18: Predikce výpůjček v rámci Brna pro 15. 5. 2025

Pro získání časových oken určených pro dovoz kol je nutné převážně sledovat rozdíl odjezdů a příjezdů, jako je tomu v grafu (Obr. 19) znázorněno pro stanici na náměstí Svobody.

Jak lze pozorovat právě na zmíněné stanici, v případě, že by byl dostupný počet kol přesně roven predikované hodnotě výpůjček, došlo by při skutečném vývoji poptávky ke stavu, kdy by stanice nebyla schopna obsloužit všechny požadavky. Výsledkem by byl negativní stav kol, což je z hlediska reálného provozu nepřípustné, neboť by tím společnost přicházela o potenciální zisky z neuskutečněných výpůjček.



Obr. 19: Průběh denní predikce (15. 5. 2025) pro stanici na náměstí Svobody

Z tohoto důvodu bylo nezbytné zapracovat i potenciální nedostatek kol do výpočetního modelu (Enginu). Pro řešení této problematiky byla navržena metodika, která je v současnosti popsána primárně teoreticky. Přestože je její část již implementována, zatím nebylo možné ověřit její plnou funkčnost vzhledem k problémům s aktualizací hodnot a jejich správným načítáním v závislosti na zadání výpočtu.

Navržený přístup spočívá ve sledování očekávaných stavů kol v čase a identifikaci kritických bodů, ve kterých by mohlo dojít k překročení dolní hranice dostupnosti kol (tj. méně než nula). V těchto případech se vytvoří nové, pomocné uzly („podstanice“ – `REBALANCE = 1`) reprezentující potřebu dovozu kola do příslušné stanice alespoň jednu hodinu před tím, než k tomuto nedostatku může dojít. Hodnoty těchto dovozů jsou odvozeny z časových oken rozdílů mezi příjezdy a odjezdy, jak je zobrazeno v grafu (Obr. 19).

Podobný mechanismus se uplatňuje i v opačném případě – tedy pokud je na stanici po celý den predikováno nadbytečné množství kol (typicky více než jedno kolo, které nebude v daném dni využito). V takové situaci se toto kolo označí jako „dostupné k odvozu“ (`REBALANCE = -1`), přičemž jeho přesný čas odběru není pevně stanoven. Kolo tak může být v rámci plánování převozu redistribuováno na jinou, zatíženější stanici, kde je jeho využití pravděpodobnější.

7.5 Možnosti rozšíření

Pro zvýšení flexibility a přesnost modelu je možné uvažovat o několika modifikacích celého výpočetního procesu. Změny lze realizovat jak na úrovni datového vstupu, tak samotné struktury modelu či způsobu optimalizace. Taková rozšíření by mohla reflektovat specifické požadavky provozu systému sdílených kol, a tím lépe přispívat k efektivnímu plánování operací v reálném čase.

Jednou z možností je zavedení dalšího typu nákladu, konkrétně elektrických kol. Pro jejich začlenění je nutné přidání dvou nových kapacitních dimenzí – pro funkční a rozbitá elektrická kola, a úprava callbacku pro dimenzi sdílenou. Tato funkcionalita zatím nebyla implementována z důvodu nedostatku informací o konkrétních typech kol a stanicích, na kterých je požadována jejich priorita pro distribuce.

Další možností rozšíření je zahrnutí informací o aktuálním počasí a dopravní situaci, případně jejich predikcí. Tyto faktory mohou významně ovlivnit výběr tras či dobu potřebnou k provedení jednotlivých úkonů.

Za perspektivní lze považovat i úpravu modelu směrem k větší dynamice návrhu úlohy v průběhu jejího řešení. Uvažováno může být například druhé, menší výpočetní jádro, které by v reálném čase provádělo aktualizace na základě ručních vstupů od techniků, jako jsou změny stavů kol či přidávání nových stanic do již vygenerované trasy. Tímto způsobem by bylo možné zajistit plynulé plnění původní trasy bez jejího úplného přeplánování.

8 Závěr

Cílem této diplomové práce bylo navrhnout a implementovat flexibilní výpočetní modul pro optimalizaci tras servisních techniků v systému sdílených kol, a to na základě reálných požadavků společnosti nextbike Czech Republic s.r.o. Při návrhu řešení byly zohledněny nejen klasické logistické aspekty, jako je kapacita vozidel nebo minimalizace nákladů, ale i řada dalších praktických omezení, jako jsou časová okna, role techniků, specifické úkony na stanicích, možnost více směn nebo práce s různými typy nákladu. Výsledný model tak představuje tzv. „Rich VRP“, jehož struktura vychází z pokročilých variant klasických trasovacích problémů.

Výpočetní modul byl implementován v jazyce Python s využitím knihovny Google OR-Tools, přičemž důraz byl kladen na čitelnost, rozšiřitelnost a možnost integrace do existujícího systému. Součástí práce bylo rovněž vytvoření rozhraní pro předzpracování vstupních dat, napojení na databázi, využití prediktivního modulu pro odhad poptávky a zajištění výstupního formátu ve standardizovaném JSON zápisu.

Funkčnost modelu byla ověřena na reálných instancích z města Brna, které poskytly dostatečně robustní datový základ. Výsledky ukázaly, že navržený systém je schopen generovat reálně využitelné trasy s ohledem na komplexní okrajové podmínky, a zároveň přispět k efektivnějšímu plánování provozu.

Z výsledků testování vyplynulo, že na kvalitu finálního řešení má zásadní vliv volba strategie pro hledání počátečního řešení. Mezi testovanými možnostmi se jako nejefektivnější ukázaly dvě metody:

- `PATH_MOST_CONSTRAINED_ARC` – dosáhla nejnižší hodnoty nákladové funkce,
- `PARALLEL_CHEAPEST_INSERTION` – poskytla stabilní a logicky uspořádané trasy.

Dále bylo prokázáno, že vliv volby algoritmu pro lokální prohledávání je v tomto konkrétním nastavení menší. Všechny testované metody dosahovaly za stejný čas podobných výsledků. Přesto byly pro budoucí použití doporučeny:

- `AUTOMATIC` – univerzální volba, která nechává optimalizační jádro zvolit nejlepší metodu podle dat,
- `GREEDY_DESCENT` – rychlá konvergující metoda, vhodná zejména pro úlohy s přísným časovým limitem.

Výsledné řešení respektovalo veškerá omezení a pravidla úlohy, a přitom dokázalo vygenerovat smysluplné a provozně použitelné trasy. Model byl navržen s ohledem na budoucí rozšíření a modifikaci podle měnících se potřeb provozu.

Konzultace se servisními techniky potvrdila, že při reálném nasazení by bylo dosaženo jak snížení počtu nutných přejezdů, tak i zkrácení délky směn, bez snížení kvality pokrytí stanovených požadavků.

Navržený systém tak může sloužit jako základ pro další rozvoj nástrojů v oblasti inteligentní městské logistiky. Mezi nejvýznamnější směry budoucího vývoje patří zejména:

- rozšíření dimenzí úlohy o elektrokola ve dvou režimech s potenciálně jiným přístupem v datovém modelu,
- zahrnutí dynamického přeplánování v reálném čase (např. nenadálá změna počtu kol na jednotlivých stanicích),
- změna způsobu integrace prediktivního modulu na základě pokročilejších statistických metod.

Tato práce tak představuje nejen konkrétní implementaci výpočetního nástroje, ale i příspěvek k aplikovanému výzkumu v oblasti plánování tras a městské mobility, s potenciálem dalšího využití v praxi.

SEZNAM POUŽITÉ LITERATURY

[1] *Bike sharing for your city with nextbike - Find out more now -*. Dostupné z: https://www.nextbike.net/en/pt-cities/.

[2] *O nás - nextbike Czech Republic*. Srpen 2021. Dostupné z: https://www.nextbikeczech.com/o-nas/.

[3] *TSP History Home*. Dostupné z: https://www.math.uwaterloo.ca/tsp/history/.

[4] Boyd, S. a Mattingley, J. Branch and bound methods. *Notes for EE364b, Stanford University*. Citeseer, 2007, sv. 2006, s. 07.

[5] Matai, R.; Singh, S. a Mittal, M. *Traveling Salesman Problem: an Overview of Applications, Formulations, and Solution Approaches*. InTech, 2010. ISBN 978-953-307-426-9.

[6] Hosseininia, M. a Dadgostari, F. Hamiltonian Paths and Cycles. *Graph Theory for Operations Research and Management: Applications in Industrial Engineering*, leden 2012, s. 96–105. ISBN: 9781466626928.

[7] Bektas, T. The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 2006, sv. 34, č. 3, s. 209–219. ISSN 0305-0483. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0305048304001550.

[8] Wenyi, L. Chapter Three - Research on vehicle routing problem and application scenarios. In: Zhang, H., ed. *Handbook of Mobility Data Mining*. Elsevier, 2023, s. 63–88. ISBN 978-0-323-95892-9. Dostupné z: https://www.sciencedirect.com/science/article/pii/B9780323958929000061.

[9] Liong, C. Y.; Ismail, W. R.; Omar, K. a Zirour, M. Vehicle routing problem: models and solutions. In:. 2008. Dostupné z: https://api.semanticscholar.org/CorpusID:49977872.

[10] Lim, A. a Wang, F. Multi-depot vehicle routing problem: a one-stage approach. *IEEE Transactions on Automation Science and Engineering*, 2005, sv. 2, č. 4, s. 397–402.

[11] Toth, P. a Vigo, D., ed. *Vehicle Routing: Problems, Methods, and Applications*. 2. vyd. SIAM - Society for Industrial and Applied Mathematics, 2014. Dostupné z: https://epubs.siam.org/doi/book/10.1137/1.9781611973594.

[12] Bräysy, O. a Gendreau, M. Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms. *Transportation Science*, Únor 2005, sv. 39, s. 104–118.

[13] Desaulniers, G.; Desrosiers, J.; Erdmann, A.; Solomon, M. M. a Soumis, F. VRP with Pickup and Delivery. *The vehicle routing problem*. Philadelphia, 2002, sv. 9, s. 225–242.

[14] Malandraki, C. a Daskin, M. Time Dependent Vehicle Routing Problems: Formulations, Properties and Heuristic Algorithms. *Transportation Science*, Srpen 1992, sv. 26, s. 185–200.

[15] Batsyn, M. V.; Batsyna, E. K.; Bychkov, I. S. a Pardalos, P. M. Vehicle assignment in site-dependent vehicle routing problems with split deliveries. *Operational Research*. Springer, 2021, sv. 21, s. 399–423.

[16] Cattaruzza, D.; Absi, N. a Feillet, D. Vehicle routing problems with multiple trips. *4or*. Springer, 2016, sv. 14, č. 3, s. 223–259.

[17] Aksen, D.; Özyurt, Z. a Aras, N. Open vehicle routing problem with driver nodes and time deadlines. *Journal of the Operational Research Society*. Taylor & Francis, 2007, sv. 58, č. 9, s. 1223–1234.

[18] Toth, P. a Vigo, D. An Exact Algorithm for the Vehicle Routing Problem with Backhauls. *Transportation Science*, Listopad 1997, sv. 31, s. 372–385.

[19] Chacon, S. a Straub, B. *Pro git*. Springer Nature, 2014.

[20] *The most-comprehensive AI-powered DevSecOps platform*. Dostupné z: `https://about.gitlab.com/`.

[21] An, H.-C.; Kleinberg, R. a Shmoys, D. B. Improving Christofides' Algorithm for the s-t Path TSP. *J. ACM*. New York, NY, USA: Association for Computing Machinery, listopad 2015, sv. 62, č. 5. ISSN 0004-5411. Dostupné z: `https://doi.org/10.1145/2818310`.

[22] Hernando, L.; Mendiburu, A. a Lozano, J. A. Hill-Climbing Algorithm: Let's Go for a Walk Before Finding the Optimum. In: *2018 IEEE Congress on Evolutionary Computation (CEC)*. 2018, s. 1–7.

[23] Alsheddy, A.; Voudouris, C.; Tsang, E. P. a Alhindi, A. *Guided Local Search*. 2018.

[24] Van Laarhoven, P. J.; Aarts, E. H.; Laarhoven, P. J. van a Aarts, E. H. *Simulated annealing*. Springer, 1987.

[25] Prajapati, V. K.; Jain, M. a Chouhan, L. Tabu search algorithm (TSA): A comprehensive survey. In: IEEE. *2020 3rd International Conference on Emerging Technologies in Computer Engineering: Machine Learning and Internet of Things (ICETCE)*. 2020, s. 1–8.

[26] Kallab, C.; Haddad, S.; El Zakhem, I.; Sayah, J.; Chakroun, M. et al. Generic Tabu Search. *Journal of Software Engineering and Applications*. Scientific Research Publishing, 2022, sv. 15, č. 7, s. 262–273.

[27] Peng, D.; Cao, L. a Xu, W. Using JSON for data exchanging in web service applications. *Journal of Computational Information Systems*, 2011, sv. 7, č. 16, s. 5883–5890.

SEZNAM ZKRATEK

API Application Programming Interface

aTSP Asymmetric Traveling Salesman Problem

CVRP Capacitated Vehicle Routing Problem

ČR Česká republika

GA Genetic Algorithm – genetický algoritmus

GPS Global Positioning System

HFVRP Heterogeneous Fleet Vehicle Routing Problem

id identifikátor

JSON JavaScript Object Notation

MDVRP Multi-Depot Vehicle Routing Problem

MIP Mixed Integer Programming – smíšené celočíselné programování

MTVRP Multi-Trip Vehicle Routing Problem

MTZ Miller–Tucker–Zemlin

mTSP Multiple Travelling Salesman Problem

NP Nondeterministic Polynomial-time

NP-těžký NP-hard, problém těžší nebo stejně těžký jako problémy v NP

OR-Tools Operations Research Tools

OVRP Open Vehicle Routing Problem

PSO Particle Swarm Optimization – optimalizace pomocí rojů částic

QR Quick Response kód – dvourozměrný čárový kód pro rychlé čtení a zpracování dat

SDVRP Site-dependent Vehicle Routing Problem

sTSP Symmetric Traveling Salesman Problem

TDVRP Time-dependent Vehicle Routing Problem

TSP Travelling Salesman Problem

VCS Version Control System – systém pro správu verzí

VRP Vehicle Routing Problem

VRPB Vehicle Routing Problem with Backhauls

VRPPD Vehicle Routing Problem with Pickup and Delivery

VRPTW Vehicle Routing Problem with Time Windows

SEZNAM OBRÁZKŮ

SEZNAM TABULEK