ZÁPADOČESKÁ UNIVERZITA V PLZNI

FAKULTA EKONOMICKÁ

Diplomová práce

Optimalizace logistických procesů na bázi spolupráce přepravních společností

Logistics optimization based on carrier cooperation

Bc. Tomáš Fiala

Plzeň 2020

ZDE SE NACHÁZÍ ORIGINÁL ZADÁNÍ KVALIFIKAČNÍ PRÁCE.

Čestné prohlášení

Prohlašuji, že jsem diplomovou práci na téma

„Optimalizace logistických procesů na bázi spolupráce přepravních společností “

vypracoval samostatně s použitím uvedené literatury a zdrojů informací.

V Plzni dne

.....

podpis autora

Obsah

Úvod

Současná doba volá po co největší optimalizaci všech odvětví především za účelem snížení nákladů, což vede ke snížení výsledné ceny produktu a konkurenceschopnosti firem. Ať už se jedná o optimalizaci operací ve výrobě s cílem co nejkratšího a nejefektivnějšího výrobního cyklu, nebo optimalizaci logistického řetězce s cílem doručit produkt zákazníkovi v co nejkratší době a za co možná nejnižší náklady.

V době moderních technologií, kdy mají přepravci volný přístup k aktuálním informacím o hustotě provozu, uzavírkách a jiných možných zdrženích na komunikacích hledají přepravci způsob, jak tyto informace nejlépe využít a mít tak konkurenční výhodu nad ostatními. A jelikož na zpracování všech informací v reálném čase lidský mozek nestačí, vybízí se do procesu zapojit počítačovou techniku. Optimalizačním algoritmem lze pak ušetřit náklady a zkrátit trasu potřebou pro obsloužení všech požadavků o desítky procent. Ani výpočetní síla moderních počítačů však nemusí být dostatečná a to v případě zpracování úloh s velkým množstvím zákazníků a zvolením nevhodného algoritmu. Optimalizace a plánování tras není nicméně nové téma a je mu věnováno již nespočetně publikací.

Práce samotná vznikla s cílem nalezení řešení, které by umožnilo lepší využití kapacit v silniční dopravě v rámci balíkové přepravy za pomoci sdílení kapacit mezi jednotlivými přepravci. V praxi by se mohlo jednat o jednotlivce vlastnící dodávku nebo o přepravní společnosti ochotné spolupracovat. Všichni účastníci by byli motivováni zvýšením vlastního zisku. Pro dosažení tohoto řešení je nezbytné nejprve splnit následující dílčí cíle:

- rozebrat možnosti spolupráce mezi přepravci
- popsat a implementovat metody pro řešení přepravních optimalizačních úloh
- použít vybrané metody pro implementaci modelu řešící sdílení kapacit
- přizpůsobit implementovaný model reálným datům

Po návrhu a implementaci řešení je posledním cílem práce zhodnocení řešení a návrh případných zlepšení.

Práce bude rozdělena na pět kapitol. V první kapitole práce bude blíže popsána problematika a možnosti spolupráce přepravců z hlediska přepravního řetězce. Dále budou popsány možné formy spolupráce mezi přepravci jako centralizovaná a decentralizovaná forma spolupráce. Ve druhé kapitole budou popsány heuristické a exaktní metody pro řešení optimalizačních úloh. Ve třetí kapitole bude práce zabývat teoretickými modely logistických úloh od jednoduchého problému obchodního cestujícího až po model řešící přidělení požadavků pomocí kombinatorické aukce. Ve čtvrté kapitole budou jednotlivé modely implementovány a ohodnocena nalezená řešení pro zadaná testovací data. Bude také popsáno prostředí ve kterém budou modely implementovány. V poslední páte kapitole bude použit implementovaný model pro reálné vstupy, získané výsledky budou zhodnoceny. Poté budou navrhnuta možná zlepšení implementovaného řešení.

1. Problematika

Velká města v současné době čelí husté a složité dopravní situaci, k tomu také velkou mírou přispívají přepravní společnosti. Mnoho přepravců nabízí přepravu ve stejné oblasti, což má za následek cestu několika přepravců na stejné místo doručení. Výsledkem jsou různé negativní následky jako vyšší cena přepravy, hustší provoz a znečistění prostředí způsobené mimo jiné následujícími faktory:

- Růst poptávky po přepravě mezi rozdílnými místy
- Růst nákladů na jednotlivé požadavky
- Problémy drobných(mikro) a malých přepravců s optimalizací [33]
- Řešení optimalizace pomocí úpravy intervalů přepravy, což nabízí pouze omezený prostor pro vylepšení zisku

Přepravní společnosti využívající silniční přepravu, jeden z hlavních producentů emisí CO_2, jsou podporovány veřejnými institucemi ke vzájemné spolupráci. Cílem není jen snížit emise škodlivých látek, ale také snížit dopravní zácpy a hluk způsobený dopravou. Navíc vzájemná spolupráce v přepravě a logistice vede k růstu úrovně služeb, zvyšování podílu na trhu, zvýšení kapacit a snižuje dopad efektu biče.[2] Efekt biče spočívá v tom, že se variabilita poptávky v dodavatelských řetězcích směrem od konečných zákazníků přes obchod až k výrobcům a jejich dodavatelům stále více zvětšuje.[3]

Z těchto důvodů není překvapivé, že v poli sdílení přepravních kapacit probíhá aktivní výzkum s možným velkým dopadem na silniční přepravu, jelikož z výzkumů vyplývá, že při spolupráci přepravců lze snížit emise skleníkových plynů až o 33% oproti situaci, kdy nespolupracují.[12] Dalším důvodem je silná konkurence mezi společnostmi, které se snaží díky tlaku zákazníků nabízet kvalitnější a levnější služby. Velké společnosti mají oproti

drobným a malým přístup k velké cenově efektivní a produktivní přepravní síti, proto se tato práce soustředí na optimalizaci operací drobných a malých podniků dle [33].

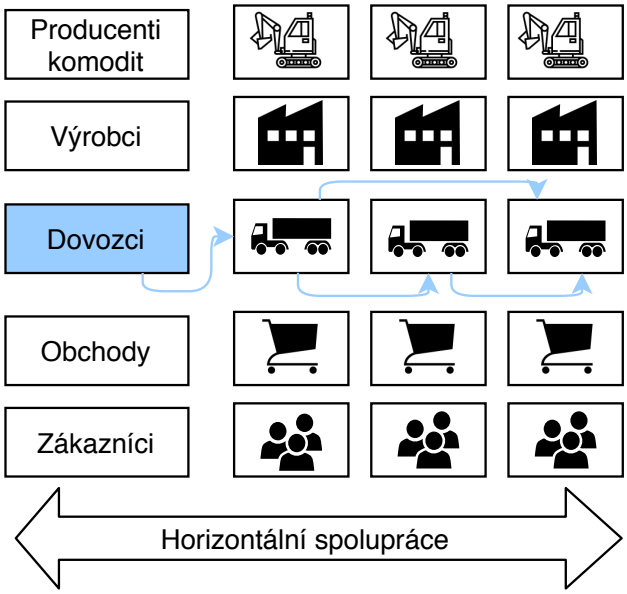
Sdílení kapacit mezi přepravci v rámci stejného časového období a místa by mohlo výrazně snížit zmíněné negativní následky. Přepravci by tak mohli obsloužit část poptávky po přepravě za ostatní přepravce bez výrazného prodloužení jejich vlastních tras. Zároveň by přepravci byli schopni lépe využít kapacitu svých vozidel a tím uspořit z hlediska počtu potřebných vozidel a ujeté vzdálenosti. Vzhledem k vysoké konkurenci v tomto odvětví je to nezbytné, pokud chtějí být přepravci efektivní a konkurenceschopní.[1] Výhody spolupráce v přepravě v posledních desetiletích přitahují pozornost zejména díky technologiím umožňujících snadnou komunikaci, která takovou spolupráci mezi dopravci umožňuje. Spolupráce mezi partnery a aliancemi je dlouho zkoumané téma. Literatura zaměřená na logistiku je však primárně zaměřena na vertikální spolupráci. Vertikální spolupráce označuje hierarchické vztahy, kdy je jedna strana klientem druhé. Spolupráce mezi společnostmi na stejné úrovni přepravního řetězce, která se označuje jako horizontální spolupráce, obdržela pouze omezenou pozornost. V horizontální spolupráci hraje velkou roli neutrální zprostředkovatel. Aby mohl zprostředkovatel efektivně rozhodovat, musí mít dostatečnou znalost zúčastněných společností a znalost jak horizontální spolupráci zahájit a dále rozvíjet. Vybrané společnosti by se měli vzájemně doplňovat. [2][5]

1.1 Horizontální spolupráce

Pokud chce společnost zavést horizontální spolupráci, musí investovat ne jen čas a úsilí, ale musí být také ochotna spolupracovat někdy i s přímým konkurentem. Tato spolupráce může být vytvořena sítí pro spolupráci přepravců (*Collaborative Carrier Network*, CCN). Analýza této sítě ukazuje, že je paretovsky účinná, kde žádný z účastníků není poškozen, ale mnoho je na tom lépe. CCN je popsána jako druh společné spolupráce více přepravců, takzvané aliance a jejím cílem je dosažení nejvyššího zisku a zároveň zvýšení zisku pro členy aliance. Je obvykle tvořena více přepravci, kteří poskytují stejné služby ve stejné geografické oblasti (obr. 1.1). Hlavní motivace společností pro spolupráci jsou peněžní odměny, vyšší zisky a reputace partnerských společností. Dále mají vliv na spolupráci externí motivy jako státní regulace nebo tlak zákazníků a konkurence. Problém aliance je především neochota sdílení informací, jelikož mají přepravci konkurenční vztah. Dále

přepravci musí řešit, jaké požadavky jsou ochotni předat ostatním přepravcům a jaké nakoupit při výměně požadavků. Aliance musí také řešit způsob rozdělení zisku získaného díky spolupráci mezi přepravci. Způsoby rozdělení přímo ovlivňují efektivitu spolupráce. [5]

Obrázek 1.1 Horizontalní spolupráce



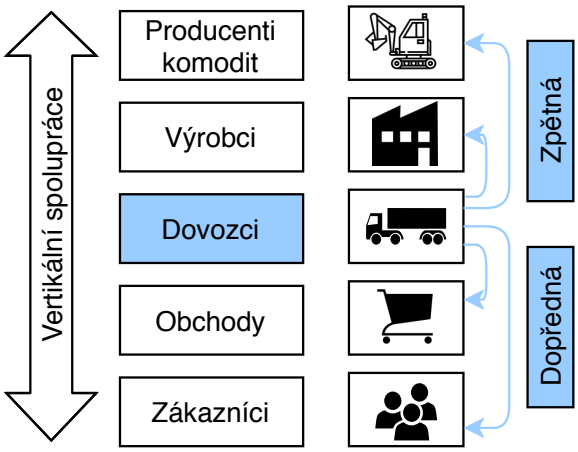
Zdroj: vlastní zpracování

Horizontální spolupráce zahrnuje dvě a více konkurenčních společností s různou podnikovou kulturou, vizí a prostředí, z kterých pocházejí. Důvěra je proto zásadní element pro efektivní a fungující horizontální spolupráci. v případě nedostatku důvěry mezi partnery roste riziko konfliktů a oportunistického jednání. Výběr partnerů je proto velmi důležitý pro úspěšné fungování spolupráce. Výsledky výzkumů ukazují, že je důležité ověřit, zdali se k sobě společnosti hodí a jsou ochotny horizontální spolupráci mezi sebou zahájit. Tato ochota spolupracovat závisí na filozofii a zkušenostech společností, ale i zájmu jednotlivců ovlivňujících chod firmy. Bylo také zjištěno, že osobní zájmy těchto jednotlivců mají klíčový vliv na úspěch spolupráce. Taková spolupráce může být jednotlivcům přínosná z pohledu sociálních vztahů, zlepšení reputace, přístup do jinak uzavřených sociálních skupin a tak dále. Jinými slovy, vztahy mezi jednotlivci z různých společností jsou zásadní pro úspěšné fungování spolupráce. [5]

1.2 Vertikální spolupráce

V případě, že je spolupráce mezi společnostmi navázána v různých etapách logistického řetězce jedná se o vertikální spolupráci. Ta je navázána za účelem získání prospěchu a je často časově omezena. Z hlediska typů spolupráce se může jednat o zpětnou (*backward*) spolupráci, kdy společnost spolupracuje se společnostmi ve směru k producentovi nebo o dopřednou (*forward*), kdy je spolupráce navázána se společnostmi blíže ke koncovému zákazníkovi (obr. 1.2). [34]

Obrázek 1.2 Vertikální spolupráce



Zdroj: vlastní zpracování

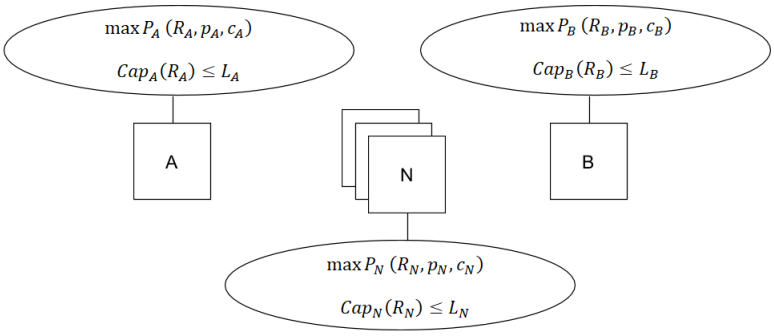
Hlavní výhody vertikální spolupráce jsou nízké investiční náklady a snadné ukončení spolupráce. Spolupráce s sebou nese rizika v podobě sdílení know-how, závislosti na silných partnerech a potencionálně vyšší náklady z důvodu absence konkurence dodavatelů. O této spolupráci se také hovoří jako o logistice třetí strany (3PL), kdy jsou kompletní logistické služby prováděny a koordinovány najatou firmou (třetí stranou). Dále existuje logistika čtvrté strany, která spočívá v provádění logistických úkonů jednou najatou firmou a administrativních úkonů jinou firmou (čtvrtá strana). Touto spoluprací se však práce do hloubky nezabývá. [34]

1.3 Formy spolupráce

Dále jsou popsány tři možné formy spolupráce mezi přepravci v rámci horizontální spolupráce. V případě, že mezi přepravci není navázána žádná spolupráce (viz obr. 1.3), snaží

se každý z přepravců $i, i \in (A, \ldots, N)$ maximalizovat svůj vlastní zisk P_i, který závisí na souboru zakázek R_i, plateb p_i a nákladů c_i. Zároveň je omezena velikost zakázky Cap_{R_i} vlastní kapacitou přepravce L_i.

Obrázek 1.3 Situace bez spolupráce



Zdroj: vlastní zpracování

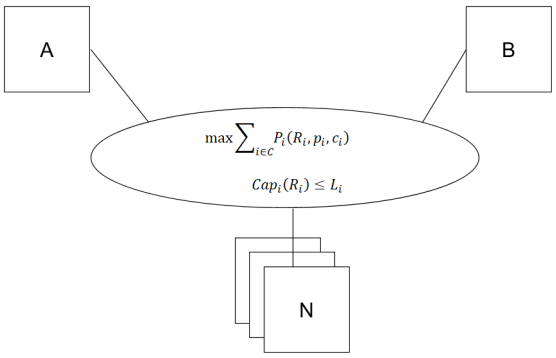
Velká část literatury a výzkumů z této oblasti se zaměřuje pouze na spolupráci mezi přepravci, tj. vlastníky a operátory dopravních prostředků. Pohled na spolupráci mezi vlastníky zásilek není brán v potaz. Z pohledu současné literatury lze pozorovat časté zaměření na centralizované plánování, tj. plánování s úplnými informacemi. Centralizované plánování je důležitý aspekt vzájemné spolupráce v přepravě, kde je centralizovaná autorita, která rozhoduje o přidělování poptávané přepravy tak, aby byly splněny požadavky všech spolupracujících přepravců. Dále spolupráci rozdělujeme na dva druhy decentralizované spolupráce a to založenou na aukci a bez aukce, které budou popsány dále.[6]

Velmi důležitý je také přístup sdílení dosažených výsledků. Nejběžnější metody jsou založeny na kooperativní teorii her, která popisuje spolupráci více hráčů, způsob spolupráce, strategie a právě rozdělování zisku. Nejběžnější metodou je Shapleyova hodnota, která říká, jak významná je pozice hráče. Metoda bere v potaz přínos hráče do koalice a z něj vycházejí hodnoty pro dělení výhry. [9] Další z metod rozdělování zisku je proporční metoda, která určuje podíl pro každého hráče na základě polohy depa, skladů a zákazníků. Výhoda této metody je rychlý výpočet pro rozsáhlé soubory dat, jelikož není potřeba předem vypočítat optimální řešení úlohy.[10] Poslední rozšířenou metodou je metoda *nucleolus*, která místo hledání férového řešení minimalizuje maximální nespokojenost hráčů tím, že nabídne určité rozdělení výhry, zjistí nespokojenost jednotlivých hráčů a největší zjištěnou nespokojenost se snaží pokaždé minimalizovat.[11]

1.3.1 Centralizovaná spolupráce

V případě centralizovaného plánování je celkový zisk maximalizován společně, tato spolupráce se nazývá také jako *centralized collaborative planning*. Na obrázku 1.4 je zobrazen mechanismus spolupráce, kdy jsou sdíleny veškeré informace s centrální autoritou.

Obrázek 1.4 Centralizovaná spolupráce



Zdroj: vlastní zpracování

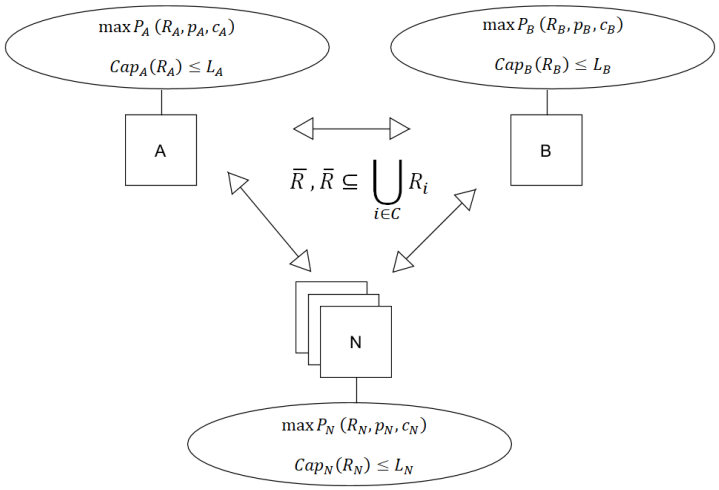
V tomto případě je rozhodování přenecháno centrální autoritě s přístupem ke všem informacím. Jako příklad může být webový portál zastávající rozhodovací funkci na základě získaných informací. Pro nalezení optimálního řešení musí být však nejprve vyřešena optimalizační úloha. I když pověřená autorita nalezne optimální řešení, nemusí mít moc toto řešení prosadit pouze výměnnou požadavků mezi členy, proto byla představena možnost outsourcingu požadavků.[1] Mnohé studie potvrzují, že spojení několika společností s různými požadavky, ale stejným distribučním centrem, má potenciál zvýšení zisku o 20-30% oproti plánování bez spolupráce. [12]

1.3.2 Decentralizovaná spolupráce

V případě, že společnosti nejsou ochotny sdílet veškeré informace s centrální autoritou je zapotřebí zavést decentralizovaný přístup, kde mohou členové spolupracovat jak mezi sebou, tak pomocí centrální autority, která nemá veškeré informace. Na obrázku 1.5 je znázorněno, že každá společnost maximalizuje svůj zisk P, jako v případě bez spolupráce s tím rozdílem, že společnosti mohou poskytnout požadavky R k výměně.

Jeden z problémů, na kterém závisí efektivita spolupráce, je výběr partnerů ke spolupráci. Potenciální partneři mohou mít různé požadavky, které je třeba zvážit a například

Obrázek 1.5 Decentralizovaná spolupráce



Zdroj: vlastní zpracování

pomocí stanovení ukazatelů jednotlivých přepravců předpovědět synergii mezi partnery. Hlavní ukazatele, které mají na fungování spolupráce vliv, jsou velikost a počet požadavků na přepravu. Menší vliv má například maximální možné zpoždění zásilek. [13]

Další problém je výběr požadavků, které by měly být nabídnuty ostatním partnerům. Obecně přepravci nechtějí nabízet veškeré své zásilky, ale využít na přepravu své vlastní kapacity. Jako logické řešení tohoto problému se naskýtá použití metody *Team orienteering problem*, který spočívá v ohodnocení cesty mezi počáteční a konečnou lokací přes zadané uzly. Cílem je určit možné cesty pro každé vozidlo a maximalizovat součet hodnocení cest. Cesty s nejmenším ohodnocením by pak přepravce nabídl ostatním přepravcům. [14] Dále se přepravci musí rozhodovat, zdali chtějí zprostředkovat požadavky za jiné přepravce. I na tento problém by bylo možné nalézt řešení pomocí metody *Team orienteering problem*, bohužel s rostoucím počtem požadavků rychle rostou výpočetní nároky. Zároveň může být požadavek hodnotnější pro další členy, a proto by centrální autorita musela najít řešení, jak požadavky přiřadit přepravcům. Možnosti řešení této úlohy budou popsány v této práci dále.

Posledním problémem, který zde bude popsán je zvolení metody pro výměnu požadavků. Poté co byli vybráni partneři a požadavky k obsloužení, je nutné stanovit, v jaké formě bude probíhat výměna požadavků. Logicky by byla lepší výměna požadavků ve formě celé obslužné trasy, než rozdělení na jednotlivé zakázky. Přepravce by tak obsluhoval již celou optimalizovanou trasu, kterou by přepravce poskytl centrální autoritě k výměně. [15]

Výměna celých tras je běžnější pro zakázky typu FTL (*full truckload*), kde zakázka zaplní celou kapacitu návěsu/nákladního vozidla. S ohledem na možnosti využití mechanizmu výměny požadavků v reálném prostředí byly vytvořeny takové systémy pro podporu rozhodování, které na základě empirické analýzy efektivity spolupráce poskytují přepravcům informace pro jejich rozhodování v reálném čase. Pokud je tento systém správně nastaven, blíží se efektivitou centralizované spolupráci.

1.3.3 Decentralizovaná spolupráce založená na aukci

Decentralizovaná výměna požadavků může být řízena pomocí aukčního systému, kdy partneři přidají své zakázky do sdíleného prostoru. Díky mechanismu aukce lze potenciálně získat informace o prioritách přepravců ať jako poptávajících, tak jako nabízejících. Centrální autorita je v tomto případě dražitel.

V případě kombinační aukce účastníci seskupují více požadavků do kombinací a těmto kombinacím přiřazují své nabídky. Nabídky jsou pak vyhodnoceny a vybrány tak, aby maximalizovali konečnou částku zaplacenou za prodané kombinace. Účastník aukce pak získá buď celou kombinaci zakázek, nebo žádnou. Tato spolupráce tak funguje v následujících pěti krocích:

1. Každý přepravce učiní výběr požadavků pro sdílení.
2. Vytvoření kombinací požadavků z výběru požadavků.
3. Každý přepravce určí marginální zisk pro každou z vytvořených kombinací požadavků
4. Přiřazení kombinací požadavků přepravcům na základě maximalizace součtu marginálních zisků určených v bodě 3. V případě více, že je požadavek součástí více než jedné kombinace, pouze jedna z těchto kombinací je přidělena přepravci.
5. Rozdělení zisku, pokud přiřazení požadavku jinému přepravci přineslo nějaký výnos.

Tyto kroky jsou cyklicky opakovány dokud je výsledný zisk vyšší než zisk v předchozí iteraci. Kroky 1 a 3 jsou automaticky prováděny přepravci a kroky 2, 4 a 5 jsou prováděny centrální autoritou. V tomto typu aukce jsou přepravci jak v roli nabízejících, tak v roli kupujících, kteří mezi sebou obchodují kombinace požadavků. [17]

2. Metody řešení úloh

V této kapitole jsou popsány některé algoritmy pro řešení optimalizačních úloh. Tyto algoritmy se dělí na tři hlavní kategorie: exaktní, heuristické a metaheuristické. Z pohledu získaného řešení se u heuristických a metaheuristických metod setkáváme s pojmem suboptimální řešení, což znamená nejlepší nalezené řešení, u kterého nejsme schopni dokázat, jestli je optimální či nikoliv. Optimální řešení můžeme potvrdit pouze v případě prohledání celého stavového prostoru úlohy, což provádíme pouze při hledání řešení exaktními algoritmy.

2.1 Exaktní metody

Exaktní metody, jak už bylo řečeno, zkoumají celý stavový prostor za účelem nalezení globálního optima. Při používání exaktních metod je třeba brát ohled na to, že při zvětšující se velikosti problému, roste často exponenciálně i stavový prostor. Z tohoto důvodu je jeho prohledávání pro rozsáhlé úlohy velmi časově náročné. Pro úlohy menšího rozsahu jsou exaktní algoritmy vhodné pro ověření nalezení skutečného optimálního řešení. [8]

Existuje mnoho různých optimalizačních problémů a pro každý z nich existují různé postupy a algoritmy pro nalezení optimálního řešení. Před tím, než můžeme začít psát program řešící optimalizační problém, musíme identifikovat typ zadaného problému a zvolit vhodný algoritmus pro nalezení optimálního řešení.

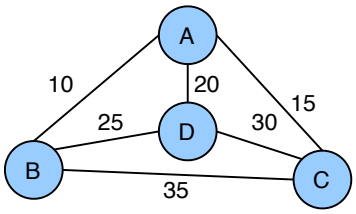
Základní typ problému je lineární optimalizační úloha. Tento typ úloh je definován lineární optimalizační funkcí a lineárními omezujícími podmínkami, které jsou následně použity v cenové optimalizační funkci. Pro nalezení řešení lze použít například simplexovou metoda, pomocí které je možné nalézt řešení v polynomiálním (omezeném) čase. V případě, že rozhodovací proměnné modelu mohou nabývat celočíselných hodnot,

jedná se o celočíselné programování (*integer programming*). Tento typ úloh je mimořádně užitečný v praxi, ale je nutné zmínit, že může být velmi výpočetně náročný a optimální řešení tak nemusí být nalezeno v rozumném čase. Pro řešení těchto problémů je často zvolena metoda větví a mezí (*branch and bound*) nebo metoda řezných nadrovin (*cutting plane*). [8]

2.1.1 Metody větví a mezí

Princip metody větví a mezí (*Branch and bound*) spočívá v systematickém procházení potenciálních řešení. Ze začátku je prohledávaný strom tvořen pouze kořenovým uzlem. Poté je určena hranice (*bound*) reprezentující hodnotu zatím nejlepšího nalezeného řešení. Počáteční hodnota je určena pomocí jiné heuristické metody. V každé iteraci je pro další prohledávání vybrán uzel reprezentující neprozkoumanou část stavového prostoru již nalezených řešení. Vybraný uzel je poté rozvětven (*branch*). Vygenerované uzly jsou ohodnoceny a v případě, že je hodnota nalezených uzlů lepší, než aktuální hranice, je hodnota hranice (*bound*) nahrazena. V případě, že je hodnota hranice horší, je celý uzel zahozen. Předpoklad je totiž takový, že nebude možné nalézt lepší řešení z jeho podprostoru, než je ohodnocení daného uzlu. V případě, že nejsou již žádné uzly k prohledávání, je algoritmus ukončen a jako optimální řešení je stanoveno řešení s nejlepším ohodnocením. [18]

Obrázek 2.1 Příklad grafu se čtyřmi uzly



Zdroj: vlastní zpracování

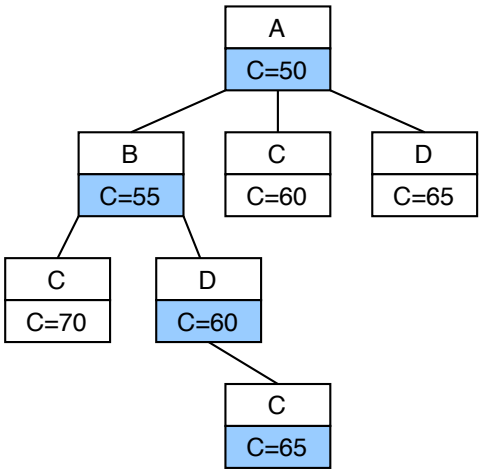
Pomocí této metody může být například vyřešena úloha obchodního cestujícího. Uvažujme neorientovaný graf (obr. 2.1) se čtyřmi uzly a ohodnocenou hranou vedoucí mezi každým uzlem. Nejdříve je nutné vypočítat dolní mez pro jakoukoliv možnou cestu.

Pomocí vzorce (2.1) je učena dolní mez pro tento příklad. Pro každý uzel u je třeba nalézt přilehlou hranu $e, e \in E_u$ s nejnižším ohodnocením a sečíst nalezená minimální ohodnocení. Tím dostaneme ohodnocení počátečního uzlu A.

$$C = \sum_{u \in V} \min_{e \in E_u} e \tag{2.1}$$

Počáteční hodnota C je vypočtena z celé množiny uzlů V. Pro následující uzly je hodnota C vypočtena pouze pro uzly, které ještě nebyli navštíveny. Na obrázku 2.2 jsou modře znázorněny uzly s nejnižším ohodnocením. Z výsledného grafu je možné odvodit nejkratší možnou cestu $A \to B \to D \to C \to A$. Celková vzdálenost této trasy je tedy $10 + 25 + 30 + 15 = 80$, což je optimální řešení této úlohy.

Obrázek 2.2 Průchod stromu pomocí metody větví a mezí.



Zdroj: vlastní zpracování

Časová náročnost tohoto algoritmu zůstává v nejhorším případě stejná, jako řešení hrubou silou. Ten nastane v případě, že algoritmus nebude moci omezit průchod žádným uzlem. Dále záleží na zvolené metodě pro výpočet hodnoty mezí C, jelikož na základě těch je určeno, které uzly budou omezeny.[18]

2.1.2 Metoda větví a řezů

Metodu větví a řezů *branch and cut* získáme kombinací metody větví a mezí s metodou sečných nadrovin (*cutting plane*), která je založena na simplexové metodě. Metoda větví a řezů se používá pro zúžení prohledávaného prostoru. Princip metody spočívá v relaxaci celočíselné úlohy pro nalezení neceločíselného řešení. Nejprve je provedena LP relaxace a následně použita standardní simplexová metoda pro nalezení řešení pro neceločíselné hodnoty jinak celočíselných proměnných. Poté je použita metoda sečných

nadrovin s cílem najít splněné podmínky celočíselnosti, které jsou porušeny prozatímním řešením a odříznout část množiny přípustných řešení, ve kterém neleží žádný celočíselný bod. Tímto procesem není vyloučeno žádné přípustné řešení. Následně je započata metoda větví a mezí. Problém je obvykle rozdělen na dvě a více variant, které jsou vyřešeny simplexovou metodou. Neceločíselná řešení LP slouží jako horní mez a celočíselná řešení jako dolní mez. Uzel může být odříznut v případě, že je horní mez nižší než existující dolní mez. Dále je provedena LP relaxace a mohou být opět použity sečné nadroviny.[29]

2.2 Heuristické metody

Heuristické, nebo také aproximační metody, prohledávají oproti exaktním metodám pouze část stavového prostoru. Díky tomu nemusí najít optimální řešení, ale snižuje se tím výrazně časová náročnost vyhledávání. Heuristické metody mohou být použity samostatně, ale zvýšení efektivity bývají často použity v kombinaci s metaheuristickými algoritmy. Heuristické metody bývají často navrženy pro konkrétní problémy, proto jsou v metodách níže uvedeny problémy, které se dají danou metodou řešit.[8]

2.2.1 Algoritmus nejbližšího souseda

Touto metodou je platné řešení hledáno tak, že se v každém kroku rozhodne pro nejvýhodnější variantu pokračování. Konkrétně pro případ problému obchodního cestujícího je nejprve zvoleno první město a poté hledáme další nejbližší. Takto postupně projdeme všechna města a po navštívení posledního se vrátíme na začátek. Tento algoritmus je velmi jednoduchý a generuje platná řešení, ale nemusí být blízká optimu. Tento algoritmus je možné použít i pro problém okružních jízd, ale musí být zaručeno, že nebude překročena kapacita nebo délka trasy.The traveling salesman problem: An overview of exact and approximate algorithms.[8]

2.2.2 Algoritmus vkládání

Algoritmus začíná vytvořením počáteční cesty mezi dvěma zvolenými uzly. Postupně jsou přidávány ještě nepoužité uzly dle předem specifikovaného kritéria. Uzly mohou být vybírány podle následujících kritérií:

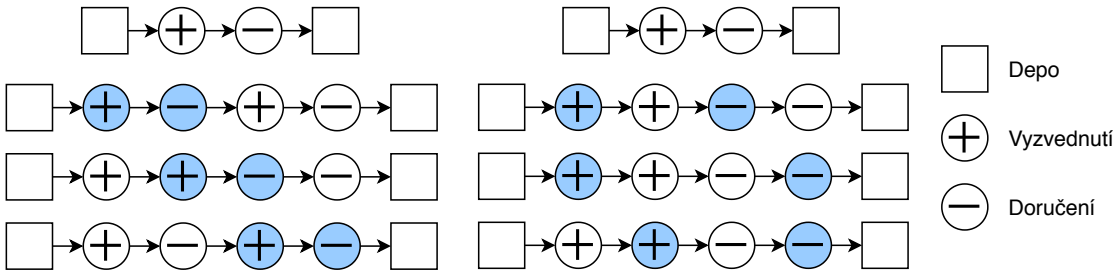
- uzel způsobující nejmenší přírůstek vzdálenosti

- nejbližší uzel k dosavadní trase
- nejvzdálenější uzel k dosavadní trase
- uzel tvořící největší úhel se dvěma po sobě jdoucími vrcholy

Po vybrání uzlu je nutné určit místo, na jaké má být uzel vložen. Místo je vybráno na základě minimálního přírůstku po vložení mezi dva již vložené uzly. Algoritmus končí po vložení všech uzlů do vytvořené cesty. [8]

Tuto metodu lze rozšířit na algoritmus dvojitého vkládání, který je použit při vkládání dvou na sobě závislých uzlů, například v případě nutnosti navštívení uzlu pro vyzvednutí před navštívením uzlu pro doručení.

Obrázek 2.3 Algoritmus dvojitého vkládání. Uzly nového požadavku označeny modře



(a) Možnosti vložení po sobě jdoucího požadavku (b) Možnosti vložení nového požadavku bez návaznosti

Zdroj: vlastní zpracování

Nejprve je nalezen požadavek s nejdelší trasou pro splnění ze souboru dostupných požadavků pomocí rovnice (2.2). Tím je určena počáteční trasa $H = \{1, i, j, 1\}$ a soubor uzlů pro vyzvednutí neobsloužených požadavků $P^{'} = P \setminus \{i\}$.

$$\max\{c_{1i} + c_{ij} + c_{j1}\} \quad i \in P, j \in d(i) \tag{2.2}$$

V dalším kroku jsou iterativně ohodnoceny zbylé páry uzlů pro aktuální trasu H dokud $P^{'} \neq \emptyset$. Ohodnocení je vypočteno jako délka nejkratší možné trasy při vložení dalšího požadavku z množiny $P^{'}$ do stávající trasy H. Cesta s nejnižším ohodnocením Δ_i, určeném minimem dvojice hodnotících funkcí (2.3) a (2.4), je vložena do trasy H. První funkce vyjadřuje vážené náklady na vložení dvou po sobě jdoucích uzlů požadavku (obr. 2.3(a)). Druhá funkce vyjadřuje vážené náklady při vložení uzlu pro vyzvednutí mezi jakékoliv existující uzly bez přímé návaznosti na uzel pro doručení (obr. 2.3(b)).[28]

$$\Delta_i = \min \Big\{ \min_{k,l \in E'} \{\alpha c_{ki} + c_{ij} + (2-\alpha)c_{jl} - c_{kl}\}, \tag{2.3}$$

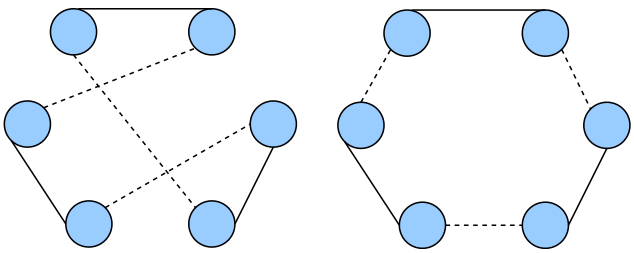
$$\min_{(k,l),(s,t) \in E'} \{\alpha(c_{ki} + c_{il} + c_{kl}) + (2-\alpha)(c_{sj} + c_{jt} - c_{st})\} \Big\} \tag{2.4}$$

Spuštění algoritmu vícekrát se změnou váhy α, $0 < \alpha < 2$ může vést k nalezení lepšího řešení. Výsledkem je subotimální řešení průchodu všemi uzly grafu, které lze například použít jako výchozí řešení pro metodu větví a mezí. [28]

2.2.3 Metoda 3-opt

Metoda 3-opt funguje na základě výměny tří hran v každé iteraci s cílem snížit celkovou vzdálenost na trase. Nejprve je nalezena počáteční trasa což může být provedeno náhodně, nebo například pomocí algoritmu vkládání(2.2.2). Následně je vygenerován soubor obsahující unikátní trojice hran pro zadanou trasu. Hrany mezi uzly ve vybrané trojici jsou poté kombinovány a v případě nalezení kratší trasy v rámci trojice, jsou hrany vyměněny a algoritmus pokračuje od začátku s novou trasou. V případě, že pro všechny trojice hran není nalezena úprava k vylepšení trasy, algoritmus končí. [27]

Obrázek 2.4 Změna hran pomocí metody 3-opt.



(a) Trasa před nahrazením hran

(b) Trasa po nahrazení hran

Zdroj: vlastní zpracování

Výsledné řešení pro konkrétní trojici hran je zobrazeno na obrázku 2.4, kde jsou provedeny výměny hran mezi všemi uzly. V praktickém použití hrany netvoří uzavřenou trasu mezi všemi uzly, ale je zadána posloupnost hran, které na sebe navazují. V této posloupnosti je hledáno lepší řešení a v případě nalezení jsou hrany vyměněny.

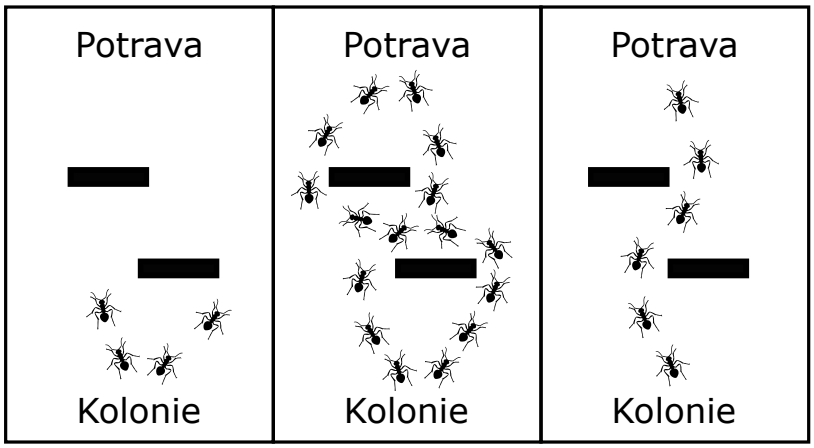
2.3 Metaheuristické algoritmy

Metaheuristické algoritmy jsou vhodné pro nalezení dostatečného dobrého řešení optimalizačního problému, které by při řešení exaktní metodou nebylo možné nalézt ve schůdném čase. Tyto algoritmy jsou založeny na různých konceptech prohledávání prostoru daného problému.

2.3.1 Mravenčí kolonie

Metoda mravenčí kolonie je založena na pozorování reálných mravenčích kolonií. Mravenci při hledání potravy značí cestu vylučováním feromonů a v závislosti na množství vyloučeného feromonu mravenci získají informaci o délce trasy a kvalitě nalezené potravy. Ostatní mravenci tyto feromony sledují a časem se kratší trasy vedoucí k lepšímu zdroji potravy stanou více frekventované (viz obr. 2.5). Díky tomu se na dané trase feromony akumulují rychleji a trasa tak přitahuje více mravenců. Princip této optimalizace je využitý v této metodě.[19]

Obrázek 2.5 Princip optimalizace v mravenčí kolonii



Zdroj: vlastní zpracování

Vlastní metoda pak zkoumá stavový prostor, kde optimalizační funkce představuje kvalitu potravy, a feromony jsou reprezentovány adaptivní pamětí. Při prohledávání je zohledněno i rozhodování jakou z následujících cest je nejlepší zvolit s ohledem na délky cest. Algoritmus je vhodný pro hledání řešení grafových úloh, jako například problém obchodního cestujícího (travelling salesman problem) nebo problém okružních jízd (vehicle routing problem). [19]

Pro názornost je zvolen postup pro problém obchodního cestujícího. Je zadáno n měst a TSP může být definován, jako problém hledání nejkratší cesty na které obchodní cestující navštíví všechna města právě jednou. Euklidovskou vzdálenost mezi městy i a j, značíme jako d_ij.

Euklidovská vzdálenost je definována vzorcem (2.5).

$$d_ij = \sqrt{((x_i - x_j)^2 + ((y_i - y_j))^2)} \tag{2.5}$$

Vlastní graf je definován seznamem uzlů N reprezentující města a hranami E, reprezentující cesty mezi městy. Jednotlivé mravence ve městě i a čase t značíme jako $b_i(t)(i = 1, \ldots, n)$ a celkový součet mravenců je tedy $m = \sum_{i=1}^{n}(b_i(t))$. Každý mravenec má následující vlastnosti:

- Město, do kterého mravenec půjde, vybírá v závislosti na vzdálenosti města a množství stop na dané hraně.
- Cestování do již navštívených uzlů je povoleno v jen případě, že už všechny uzly navštívil.
- Po dokončení celé cesty je každé hraně, po které mravenec cestoval, přidána stopa vyjádřena vzorcem (2.6).

$$\tau_{ij}(t + n) = \rho\tau_{ij}(t) + \Delta\tau_{ij} \tag{2.6}$$

$$\Delta\tau_{ij} = \begin{cases} \frac{Q}{L_k}, & \text{když mravenec } k \text{ navštíví hranu } (i, j) \text{ na jeho trase} \\ 0, & \text{v ostatních případech} \end{cases} \tag{2.7}$$

V rovnici (2.6) je $\Delta\tau_{ij}$ množství stop na jednotku délky mravence $k, k \in N$, Q konstanta a L_K délka cesty mravence k. Stopa je zároveň snížena koeficientem odpařování ρ Pro první vlastnost je třeba vypočítat pravděpodobnost navštívení města j mravencem k v případě že se nachází ve městě i vyjádřenou vzorcem (2.8).

$$p_{ij}^{k} = \begin{cases} \frac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\sum_{k \in N \setminus \{T\}} (\tau_{ik})^{\alpha}(\eta_{ik})^{\beta}}, & \text{pro } j \in N \setminus \{T\} \\ 0 & \text{v ostatních případech} \end{cases} \tag{2.8}$$

Parametry α a β určují váhu vzdálenosti a síly stopy při výpočtu pravděpodobnosti. Samotný algoritmus začíná vybráním náhodného vrcholu jako počátku. V konstrukční fázi si mravenec postupně vybírá město, které ještě nenavštívil a následně se do něj přesune. Pokud neexistuje žádné nenavštívené město, vrací se mravenec do původního vrcholu a uchová si svoji trasu T. V případě, že se mravenec nachází ve městě i, je vypočtena pravděpodobnost, že se mravenec vydá do města j pomocí rovnice (2.8). Po ukončení konstrukční fáze všemi mravenci, je spočtena feromonová stopa, kterou za sebou každý mravenec zanechal pomocí vztahu (2.6). V další fázi jsou aktualizovány hodnoty feromonu. V případě, že hrana používána mnoha mravenci, hodnota se zvyšuje a v opačném případě se hodnota snižuje. Takto získané řešení zahrnuje zapomínání nepoužívaných stop a tím zabraňuje předčasné konvergenci do lokálních optim. [20]

2.3.2 Simulované žíhání

Metoda simulovaného žíhání (*Simulated annealing*) je optimalizační metoda prohledávání stavového prostoru a je založena na modelu vycházejícího ze simulace metalurgického postupu žíhání. Postup se skládá ze zahřátí materiálu a postupného zchlazení. Metoda spočívá v tom, že při počátečních iteracích děláme změny k horšímu (zvyšujeme teplotu) z toho důvodu, aby algoritmus neuvázl pouze v lokálním minimu. Oproti obyčejnému gradientnímu algoritmu jsou přijímána i horší nalezená řešení. Pravděpodobnost přijmutí takového řešení je závislá na teplotě a velikosti zhoršení. V průběhu výpočtu je teplota postupně snižována na základě konvergence k cíli. Při rychlé konvergenci je také rychle snižována teplota a algoritmu je bráněno jít do hůře hodnocených stavů. Konverguje-li algoritmus pomalu (hodnocení stavů moc neklesá), zpomalí se snižování teploty, aby se případně podařilo vyprostit z lokálního minima. Pokud nejsme spokojeni s výsledkem, tak teplotu lze i zvyšovat. V případě růstu teploty pravděpodobnost, že bude algoritmus pokračovat do hůře hodnoceného stavu, stoupá. [25]

2.3.3 Tabu prohledávání

V metodě tabu prohledávání (*Tabu search*) je v každé iteraci nalezeno k současnému řešení x několik sousedních, díky lokálnímu vyhledávání. Po nalezení sousedních řešení je x nahrazeno nejlepším ze sousedních řešení i v případě, že je současné řešení lepší než nově nalezené. Již prozkoumaná řešení se ukládají do *tabu* seznamu na určitý počet iterací. Pro šetření paměti nejsou do seznamu ukládány celé seznamy řešení, ale pouze některé jejich atributy sloužící k porovnání řešení. Algoritmus prohledávání se skládá z následujících kroků: [24]

1. Vyber počáteční řešení $x \in X$ pro začátek zpracování a nastav tabu seznam $T = \emptyset$
2. Nechť $N(x)$ je množina sousedních řešení x, najdi nejlepší řešení $x' \in N(x) \setminus T$
3. Pokud neexistuje žádné x', x je lokálním optimem a metoda je zastavena
4. Jinak označ x' jako nové x, aktualizuj tabu seznam T a jdi zpět do bodu 2.

Tabu vyhledávání může být využito pro prohledání grafu a nalezení přibližného řešení problému obchodního cestujícího. To znamená, že nalezené řešení není zaručeně nejkratší cesta, která prochází všemi městy, ale řešení je nalezeno v relativně krátkém čase.[24]

3. Analýza možností řešení

Následující kapitola se zabývá popsáním teoretických modelů pro logistické úlohy, které řeší jednotlivé problematiky související s tématem práce. Při řešení spolupráce přepravců pomocí výměny požadavků založené na aukci je nutné nejdříve vyřešit následující problémy:

1. Přepravci se musí rozhodnout, jaké požadavky chtějí obsloužit sami a jaké by měli být nabídnuty ostatním přepravcům.
2. V průběhu aukce musí být přepravci schopni určit hodnotu příhozů, jejichž hodnota bude vyjadřovat jejich ochotu nabízený požadavek odkoupit.
3. Vozidla musí mít přiřazené takové trasy, aby byly všechny požadavky obslouženy.

Jelikož první dva body jsou vázané na nalezení optimálních tras, nejprve jsou popsány problémy, které tuto problematiku řeší. Dále je popsána strategie výběru požadavků a jako poslední je v této kapitole blíže popsána výměna požadavků založená na aukci.

3.1 Problém optimalizace tras

Problémem optimalizace tras jsou myšleny úlohy v oblasti logistiky, u kterých se snažíme najít řešení, které je nejlépe ohodnocené účelovou funkcí. Účelová funkce logistické úlohy může hodnotit například délku trasy, náklady nebo časovou náročnost. V úloze je pak cílem najít minimum takové funkce. V případě, že účelová funkce hodnotí dosažený zisk, je cílem najít naopak maximum účelové funkce.

V následující části jsou popsány základní problémy optimalizace tras, tj. problém obchodního cestujícího (TSP) a problém okružních jízd (VRP). Ty tvoří základ pro problém okružních jízd s časovými okny (VRPTW), který je popsán z důvodu znázornění práce s časovým omezením požadavků. Dále je popsán problém okružních jízd při spolupráci

přepravců (CCNVRP), který řeší distribuci požadavků mezi jednotlivé přepravce. Na základech těchto problémů jsou postaveny modely praktické části pro sdílení požadavků. Jako poslední je popsán problém spolupráce přepravců (CCPLTL), který je založen na principu kombinatorické aukce. Součástí tohoto problému je určení ceny požadavků za kterou je přepravce požadavek obsloužit, bez znalosti kompletních informací a následná úprava ceny dražitelem. [7]

3.1.1 Problém obchodního cestujícího

Problém obchodního cestujícího, tedy *Traveling Salesman Problem* (TSP), je asi nejznámější kombinatorická úloha, kterou se matematici zabývají již od 18. století. Problém spočívá v tom, že obchodní cestující musí na své cestě navštívit každé přidělené město právě jednou a poté se vrátit do počátečního bodu. Cílem je optimalizace vzdálenosti a s tím souvisejících nákladů a času cesty. [8]

Notace:

- $N = \{0, 1, 2 \ldots, n\}$ množina uzlů, kde je počátek umístěn v uzlu 0
- S množina nekompletních tras (*subtours*)
- c_{ij} náklady přesunu mezi uzly i a j
- x_{ij} binární rozhodovací proměnná udávající, jestli obchodník uskuteční trasu mezi uzly i a j.

Formálně uvažujeme plný graf $G = (N, E)$ tedy každý uzel spojený se všemi ostatními.
Účelová funkce:

$$min \sum_{i \in V} \sum_{j \in N} c_{ij} x_{ij} \tag{3.1}$$

Omezující podmínky:

$$\sum_{j \in N} x_{ij} = 1, \forall i \in N \tag{3.2}$$

$$\sum_{i \in N} x_{ij} = 1, \forall j \in N \tag{3.3}$$

$$\sum_{i,j \in S} x_{ij} \leq |S| - 1, S \subset N, 2 \leq |S| \leq n - 2 \tag{3.4}$$

$$x_{ij} \in \{0, 1\}, \forall i, j \in V, i \neq j \tag{3.5}$$

V tomto modelu účelová funkce (3.1) minimalizuje výdaje na cestu. Omezující podmínky (3.2) a (3.3) zaručují, že každý uzel bude navštíven právě jednou. Podmínka (3.4) zaručuje nedělitelnost trasy na více podskupin (*subtours*), kde je počet navštívených uzlů na trase menší, než celkový počet uzlů. V případě, že je taková trasa nalezena, byla by pravá strana podmínky (3.4) rovna $|S|$. Jelikož v TSP nejsou definovány omezující podmínky kapacity, každé výsledné řešení tohoto exaktního modelu je tedy platné. [8]

Omezující podmínky (3.2) a (3.3) sice zaručují, že každý uzel je navštíven pouze jednou, avšak ke splnění stačí vytvoření více cest mezi nejbližšími uzly na místo jedné spojité cesty. Eliminace rozdělení trasy na podskupiny může být také vyřešeno nahrazením omezující podmínky (3.4) za omezující podmínku MTZ (zkratka odvozená od jmen autorů Miller, Tucker a Zemlin).

$$u_i - u_j + (n - 1)x_{ij} \leq n - 2 \quad i, j \in N \setminus \{0\}, i \neq j \tag{3.6}$$

Princip omezující podmínky MTZ spočívá v omezení počtu hran pro předem vybraný uzel. Nejprve je nutné v modelu definovat rozhodovací proměnnou $u_i \in R$, která určuje pořadí, ve kterém bude každý uzel navštíven a přidat omezující podmínku (3.6). [30]

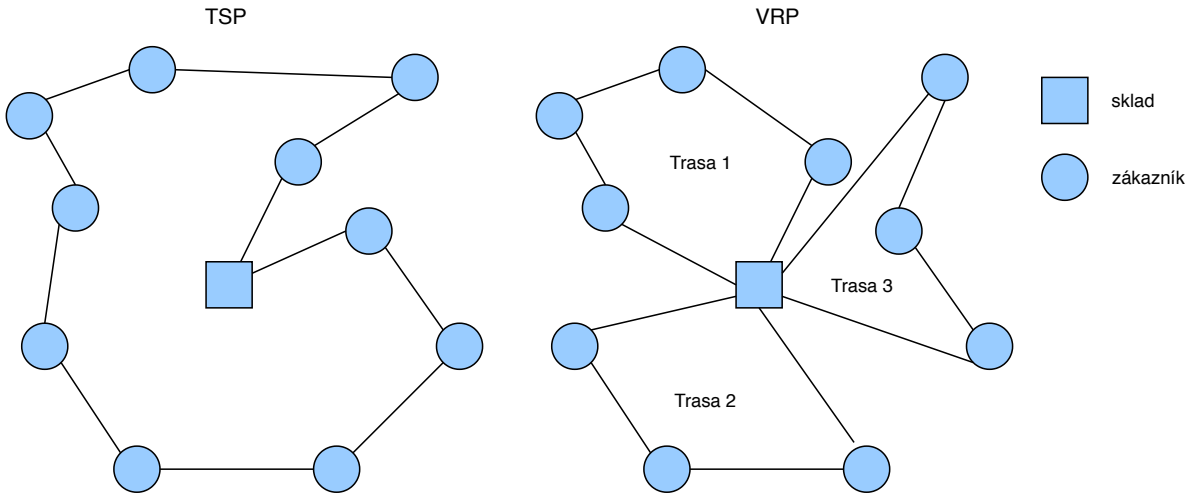
Jelikož nezáleží na směru cesty, jedná se o takzvaný symetrický problém. To znamená, že náklady na cestu jsou mezi uzly i a j stejné, bez ohledu na směr cesty. Dále u problému obchodního cestujícího nezáleží, z jakého uzlu je cesta započata.[8]

3.1.2 Problém okružních jízd

Problém okružních jízd neboli *Vehicle Routing Problem* (VRP) je logistická úloha podobná problému obchodního cestujícího. Pro ilustraci je zobrazeno porovnání řešení těchto modelů na obrázku 3.1. Vlevo je znázorněna trasa obchodního cestujícího z počátečního bodu přes všechny uzly. Vpravo je zobrazeno ilustrativní řešení problému okružních jízd, kdy je obsloužení rozděleno na 3 trasy.[21]

V této úloze jsou uzly chápány jako zákazníci a jeden nebo více uzlů jako sklad přepravce, který slouží jako výchozí a koncový bod všech vozidel. Cílem modelu mini-

Obrázek 3.1 Porovnání řešení TSP a VRP



Zdroj: vlastní zpracování

malizovat náklady na obsloužení zákazníků, tj. najít optimální trasu pro daný počet vozidel. Optimální trasa může znamenat trasu s nejmenší celkovou ujetou vzdáleností. Bez příslušných omezujících podmínek, by bylo optimální řešení obsloužení všech uzlů jedním vozidlem stejné jako v případě problému obchodního cestujícího. Všechna vozidla musejí opustit sklad a na konci cesty se do něj musejí zase vrátit. Uvažujme podobnou notaci jako pro problém obchodního cestujícího:

- N množina uzlů, kde sklad je umístěn v uzlu 0
- V množina vozidel
- k index vozidla, každé vozidlo má přiřazenu právě jednu trasu
- R_k trasa vozidla k, zadaná množinou uzlů, která začíná a končí v uzlu 0
- c_{ij} značí náklady na přesun mezi zákazníky i a j, někdy chápáno jako vzdálenost mezi zákazníky nebo doba jízdy
- x_{ijk} binární rozhodovací proměnná udávající uskutečnění trasy mezi zákazníky i a j vozidlem k.

Model získáme rozšířením modelu obchodního cestujícího přidáním dimenze vozidel k rozhodovací proměnné x. Tím vznikne účelová funkce (3.7) a omezující podmínky (3.8),(3.9) a (3.10).

Účelová funkce:

$$min \sum_{k \in V} \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ijk} \tag{3.7}$$

Omezující podmínky:

$$\sum_{k \in V} \sum_{j \in N} x_{ijk} = 1, \forall i \in N \setminus \{0\} \tag{3.8}$$

$$\sum_{k \in V} \sum_{j \in N} x_{ijk} = 1, \forall i \in N \setminus \{0\} \tag{3.9}$$

$$u_{ki} - u_{kj} + (n-1)x_{ijk} \leq n-2 \quad i,j \in N \setminus \{0\}, i \neq j, k \in V \tag{3.10}$$

$$x_{iik} = 0, \forall i \in N, \tag{3.11}$$

$$\sum_{i \in N} x_{0ik} = 1, \forall k \in V \tag{3.12}$$

$$\sum_{i \in N} x_{i0k} = 1, \forall k \in V \tag{3.13}$$

$$\sum_{i \in N} x_{ihk} - \sum_{j \in N} x_{hjk} = 0, \forall h \in N \setminus \{0\}, \forall k \in V \tag{3.14}$$

V tomto modelu, stejně jako v modelu TSP, účelová funkce (3.7) minimalizuje výdaje na cestu. Omezující podmínky (3.8) a (3.9) zaručují, že každý uzel bude navštíven vozidlem k právě jednou. Podmínka MTZ (3.10) zaručuje nedělitelnost trasy na více podskupin. Dále byly přidány nové omezující podmínky omezující pohyb vozidel. Podmínka (3.11) omezuje cesty v rámci stejného uzlu. Omezující podmínka (3.12) a (3.13) zaručuje výjezd každého vozidla a jeho návrat do depa. Omezující podmínka (3.14) zaručuje, že vozidlo musí po navštívení uzlu h tento uzel opustit.[21]

Pro určení celkových nákladů každého vozidla, které jsou definovány jako součet vzdáleností uzlů j na trase vykonané vozidlem k, je definován vzorec (3.15).

$$Cost(R_k) = \sum_{j \in R_k} c_{k_j, k_{(j+1)}} \tag{3.15}$$

Celková cena je značena jako S a je určena součtem tras všech vozidel k dle vzorce (3.16).

$$Cost(S) = \sum_{k \in V} Cost(R_k) \tag{3.16}$$

Tento model lze použít na určení nejkratší možné trasy pro množinu vozidel V, avšak bez ohledu na kapacitu či časovou náročnost. V případě, že množina V obsahuje pouze jedno vozidlo, je tento problém redukován na problém obchodního cestujícího.[21]

3.1.3 Problém okružních jízd s časovými okny

Problém okružních jízd s časovými okny (VRPTW) je model s cílem minimalizovat počet potřebných vozidel, ale také celkový čas a vzdálenost vynaložených vozidly přepravce. Tento model může být použit pro řešení reálných problémů jako hledání cesty s nejnižšími náklady kurýrních služeb nebo určení skladů a zákazníků jednotlivým vozidlům. Obsloužení zákazníka musí proběhnout v předem daném časovém okně a proto je nutné v případě vyšších časových nároků požadavků využití více než jednoho vozidla za cenu delší celkové trasy. Uvažované jsou pouze variabilní náklady vozidla v závislosti na ujeté vzdálenosti.[21]

- i, j index uzlu, každý uzel má přiřazenou souřadnici, v seznamu uzlů jsou zahrnuty uzly zákazníků a uzly skladů
- k index vozidla
- V soubor homogenních vozidel
- N soubor zákazníků
- c_{ij} značí náklady na přesun mezi uzly i a j
- x_{ijk} binární rozhodovací proměnná udávající jestli vozidlo k uskuteční trasu mezi zákazníky i a j.
- s_{jk} rozhodovací proměnná udávající čas, kdy vozidlo k začne s obsluhou zákazníka j

Účelové funkce minimalizující celkovou vzdálenost jsou pro tento a předchozí model problému okružních jízd identické:

$$min \sum_{k \in V} \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ijk} \tag{3.17}$$

Pro definování problému okružních jízd s časovými okny jsou využity omezující podmínky z modelu bez časových oken, které jsou dále rozšířeny o omezující podmínky časových oken a kapacity:

$$\sum_{k \in V} \sum_{j \in N} x_{ijk} = 1, \forall i \in N \tag{3.18}$$

$$\sum_{j \in N} x_{0jk} = 1, \forall k \in V \tag{3.19}$$

$$\sum_{i \in N} x_{i0k} = 1, \forall k \in V \tag{3.20}$$

$$\sum_{i \in N} x_{ihk} - \sum_{j \in N} x_{hjk} = 0, \forall h \in C, \forall k \in V \tag{3.21}$$

$$\sum_{k \in V} d_i \sum_{j \in N} x_{ijk} \leq q, \forall k \in V \tag{3.22}$$

$$s_{jk} + t_{ij} + K(1 - x_{ijk}) \leq s_{jk}, \forall i, j \in N, \forall k \in V \tag{3.23}$$

$$a_i \leq s_{jk} \leq b_i, \forall i \in N, \forall k \in V \tag{3.24}$$

$$\sum_{k \in V} \sum_{j \in N} x_{k0j} \leq v, \forall j \in N, \forall k \in V \tag{3.25}$$

$$x_{ijk} \in 0, 1, \forall i, j \in N, \forall k \in V \tag{3.26}$$

Omezující podmínka (3.18) zaručuje, že každý zákazník bude navštíven právě jednou. Následující tři podmínky (3.19), (3.21) a (3.20) zaručují, že každé vozidlo, které opustí depo se po doručení zásilky od zákazníka vrátí do depa zpět. Tyto omezující podmínky jsou shodné s předchozím modelem. Pro přidání dimenze času a kapacity vozidel jsou formulovány následující omezující podmínky. Podmínkou (3.22) je zaručeno, že vozidlo nepřekročí svou kapacitu. Podmínka (3.23) zaručuje, že vozidlo navštíví zákazníka v požadovaném časovém okně. Podmínka (3.24) kontroluje čas odjezdu vozidla od zákazníka do dalšího uzlu. Podmínka (3.25) zaručuje, že počet vozidel jedoucích ze skladu není větší než počet dostupných vozidel. (3.26) je podmínka integrity rozhodovací proměnné x_{ij}.

Dle takto definovaných omezujících podmínek je řešení platné pouze v případě, pokud je požadavek obsloužen do horní hranice jeho časového okna. V případě, že vozidlo dorazí do uzlu pod spodní hranicí jeho časového okna, je nuceno čekat. Každé vozidlo musí také začít a skončit svou cestu během časového okna určeného pro depo. Je také možné definovat časová okna jako volná, kdy sice vozidlo může obsloužit uzel později, ale je mu

v takovém případě přičtena penalizace zhoršující celkovou hodnotu účelové funkce. Tento model slouží jako základ pro komplexnější problémy pomocí přidání dalších omezujících podmínek. Tím můžeme model upravit a přidat například vozidla s různou kapacitou, možnost obsloužení zákazníka ve více časových oknech nebo přidání vyzvednutí zakázky mimo sklad.[21]

3.1.4 Problém okružních jízd za spolupráce přepravců

Jedním z řešení optimálního přiřazení požadavků přepravcům je model *Collaborative Carrier Vehicle Routing Problem* (CCVRP). Tento model je založen na skupině přepravních společností, které zákazníkům nabízejí standardní přepravní službu. Předpokládáme, že každý přepravce má určitý počet požadavků, které je třeba v určitém čase obsloužit. Operační plánování vozidel přepravců je prováděno periodicky. Spolupráce přepravců funguje na principu výměny požadavků mezi přepravci a modelem pro rozhodování o tom, zdali má požadavek sám přepravce nebo má být předán jinému. [22]

Cílem modelu je maximalizace celkového zisku. Budeme uvažovat následující scénář, kdy přepravce i získal zakázku na přepravu za cenu r_j a přepravce k, který může tuto zakázku obsloužit za nižší náklady. Pro vylepšení celkového zisku všech přepravců v rámci aliance, přepravce i může předat zakázku a obdržet za ní od přepravce k kompenzaci v_j. Minimální výše této kompenzace je vázaná na marginální zisk přepravce i.

V případě, že by přepravci sdílely veškeré informace o zakázkách, jednalo by se o problém okružních jízd s více depy. Nicméně v konkurenčním prostředí nejsou přepravci ochotní sdílet soukromé informace o zákaznících, marginálních výdajích a zisku. Z tohoto důvodu nelze problém vyřešit pomocí centralizované spolupráce.

- M spolupracující přepravci
- N požadavky, které mají být v daný čas splněny
- N_i požadavky přepravce $i, i \in M, N_i \subseteq N$
- C_i náklady přepravce i
- P_i původní zisk přepravce i
- r_j cena požadavku j

- v_j kompenzace za předání požadavku přepravce j
- x_{ij} binární rozhodovací proměnná udávající předání požadavku j přepravci i

Účelová funkce:

$$max\left(\sum_{j\in N} r_j - \sum_{i\in M} C_i\right) \tag{3.27}$$

Omezující podmínky:

$$1 = \sum_{i\in M} x_{ij}, \forall j \in N \tag{3.28}$$

$$C_i = \sum_{j\in N} \beta_1 \cdot x_{ij} + L(\forall j \in N | x_{ij} = 1)\beta_2, \forall i \in M \tag{3.29}$$

$$P_i \leq \sum_{j\in N} r_j \cdot x_{ij} + \sum_{j\in N_i} v_j(1 - x_{ij}) - \sum_{j\in NN_i} .v_j \cdot x_{ij} - C_i, \forall i \in M \tag{3.30}$$

$$x_{ij} \in 0, 1, \forall i \in M, \forall j \in N \tag{3.31}$$

$$v_j \in R, \forall j \in N \tag{3.32}$$

Účelová funkce (3.27) vyjadřuje maximalizaci zisku jako celkový zisk bez výdajů všech přepravců. Omezující podmínka (3.28) zajišťuje, že každý požadavek je přiřazený přesně jednomu přepravci. Pomocí podmínky (3.29) jsou vypočteny přepravní náklady. Parametr β_1 zde označuje náklady na zastavení a parametr β_2 náklady na kilometr přepravy. Zároveň je zde zahrnuta omezující podmínka pro nepřekročení maximální vzdálenosti na jedno vozidlo při výpočtu vzdálenosti L. Podmínka (3.30) zaručuje, že se zisk přepravců nesníží kvůli výměně požadavků. Pro každého přepravce je vypočten zisk P_i pomocí součtu získaných zisku ze získaných požadavků a kompenzací a odečtení platby kompenzací a ztraceného zisku z předaných požadavků.

3.1.5 Problém spolupráce přepravců

Dalším popsaným modelem je *Carrier Collaboration Problem in Less then Truckload Transportation* (CCPLTL), ve kterém přepravci vytvoří přepravní alianci za účelem sdílení kapacit jejich vozidel a požadavků na přepravu. Pro určení ceny a přesun požadavků mezi přepravci je zvolen aukční přístup. Dražitel má za úkol určit a udržovat aktuální

cenu pro každý požadavek. Cílem je maximalizace zisku celé aliance. Každý přepravce vybírá požadavky s cílem maximalizace vlastního zisku dle cen stanovených dražitelem. V případě, že po ukončení aukce zůstanou nějaké požadavky přiděleny více přepravcům najednou, přidělí se požadavek náhodně jednomu z nich.[7]

V modelu jsou definovány dvě role, dražitel a zájemce. Dražitel je virtuální koordinátor a má za úkol určovat ceny pro každý požadavek. Jeho cílem je maximalizovat celkový zisk aliance s tím omezením, že každý požadavek je přiřazen maximálně jednomu přepravci. V roli zájemců jsou přepravci, kteří vybírají nabízené požadavky s cílem maximalizace individuálního zisku na základě ceny stanovené dražitelem. Proces se skládá z následujících kroků:

1. Před aukcí odešle přepravce požadavky k aukci. Tyto požadavky mají cenu, která byla za přepravu nabídnuta odesílatelem. Ostatní přepravci tuto cenu neznají.

2. Dražitel určí počáteční cenu pro každý požadavek, která je nižší nebo rovna ceně nabídnuté za přepravu.

3. Zájemci vyjádří svůj zájem o nabízené požadavky za stanovené ceny. Každý zájemce může vybrat jakýkoliv požadavek pro maximalizaci svého zisku. Rozhodnutí, které požadavky má přepravce vybrat, řeší *bidding problem*, popsaný dále.

4. Dražitel upraví cenu požadavků. Zrušením (relaxací) omezující podmínky, která zaručuje, že je každý požadavek obsloužen nejvíce jedním zájemcem. Ceny jsou upraveny na základě porušení zrušených omezujících podmínek při aktuálně vybraných požadavcích.

5. Opakování kroků 3. a 4. dokud nejsou všechny požadavky vybrány právě jedním zájemcem a přerozdělení nemůže být vylepšeno v daném počtu dalších iterací nebo je předem daného počtu iterací dosaženo.

6. V případě, že je požadavek vybrán více než jedním zájemcem, je tento požadavek náhodně přiřazen jednomu z nich.

7. Po dokončení aukčního procesu a nabití zisku každým přepravcem, dražitel rozdělí zbylý zisk mezi všechny zájemce tak, aby zisk každého přepravce nebyl menší než v případě, kdy by nebyl členem aliance.

Pro zjednodušení má každý uzel maximálně jeden přiřazený požadavek.
Značení:

- $i, j, m = 1, \ldots, N$ značí index uzlu, kde N je počet uzlů. Uzly N zahrnují pozice všech odesílatelů, zákazníků a depa přepravců.
- $k = 1, \ldots, K$ značí index přepravce, kde K je celkový počet přepravců
- $l = 1, \ldots, L$ značí index požadavků, kde L celkový počet požadavků

Parametry modelu:

- P_i požadavky s místem odeslání v uzlu i
- D_i požadavky s místem doručení do uzlu i
- d_l množství doručené s požadavkem l
- p_l cena zaplacená odesilatelem za požadavek l
- C kapacita vozidla
- W_k počet vozidel přepravce k
- o_k depo přepravce k
- c_{ij} cena přepravy mezi uzly i a j pro všechna vozidla
- t_{ij} doba přepravy mezi uzly i a j
- a_i nejbližší možný čas obsloužení uzlu i
- b_i nejpozdější možný čas obsloužení uzlu i
- T_{ij} vysoké číslo, $T_{ij} = b_j - a_i$

Proměnné modelu:

- q_{ij}^k přepravené množství přes hranu (i, j) přepravcem k
- x_{ij}^k počet přejezdů přes hranu (i, j) vozidly přepravce k
- $y_l k$ binární proměnná nabývající hodnotu 1, pokud je požadavek l obsloužen přepravcem k

- t_i^k okamžik v čase, kdy vozidlo přepravce k, opustí uzel i

S použitím tohoto značení je optimalizace celkového zisku aliance formulován jako MIP model.

Účelová funkce:

$$max\left(\sum_{k\in K}\sum_{l\in L} p_l y_{jk} - \sum_{k\in K}\sum_{i\in N}\sum_{j\in N} c_{ij}x_{ijk}\right) \tag{3.33}$$

Omezující podmínky:

$$\sum_{j\in N, i\neq j} x_{ijk} = \sum_{j\in N, i\neq j} x_{jik} = 1, \forall i \in N, i \neq o_k, \forall k \in K \tag{3.34}$$

$$q_{ij}^k \leq C \cdot x_{ij}^k, \forall i, j \in N, i \neq j, \forall k \in K \tag{3.35}$$

$$\sum_{j\in N, i\neq j} q_{ij}^k - \sum_{j\in N, i\neq j} q_{ji}^k = \sum_{l\in P_i} d_l \cdot y_{lk} - \sum_{l\in D_i} d_l \cdot y_{lk}, \forall i, j \in N, \forall k \in K \tag{3.36}$$

$$\sum_{j\in N, j\neq o_k} x_{o_k j}^k = \sum_{j\in N, j\neq o_k} x_{j o_k}^k, \forall k \in K \tag{3.37}$$

$$\sum_{j\in N, j\neq o_k} x_{o_k j}^k \leq W_k, \forall j \in N, \forall k \in K \tag{3.38}$$

$$\sum_{k\in K} y_{lk} \leq 1, \forall l \in L \tag{3.39}$$

$$\sum_{j\in N, i\neq j} x_{ij}^k \leq 1, \forall i \in N, i \neq o_k, \forall k \in K \tag{3.40}$$

$$t_j^k \geq t_i^k + t_{ij}^k \cdot x_{ij}^k - T_{ij} \cdot (1 - x_{ij}^k), \forall i, j \in N, i \neq j, j \neq o_k, \forall k \in K \tag{3.41}$$

$$\sum_{j\in N, i\neq j} q_{o_k i}^k = \sum_{l\in P_{o_k}} d_l \cdot y_{lk} - \sum_{l\in D_{o_k}} d_l \cdot y_{lk}, \forall k \in K \tag{3.42}$$

$$x_{ij}^k \geq 0, x_{ij}^k \in Z, \forall i, j \in N, i \neq j, \forall k \in K \tag{3.43}$$

$$y_{lk} \in 0, 1, y_{lk} \in Z, \forall l \in L, \forall k \in K \tag{3.44}$$

$$0 \leq a_i \leq t_i^k \leq b_i, \forall i \in N, i \neq o_k, \forall k \in K \tag{3.45}$$

Účelová funkce (3.33) vyjadřuje celkový zisk aliance. Omezující podmínka (3.34) zajišťuje, že vozidla, které z uzlu odjeli, do nich předtím také přijeli. Podmínka (3.35) zajišťuje nepřekročení kapacit vozidel. Podmínka (3.36) zajišťuje, že v každém uzlu je vyrovnané požadované a doručené množství. Podmínka (3.37) zajišťuje, že počet vozidel, která vyjela z depa přepravce je roven počtu vozidel, co do depa přijela. Podmínka (3.38) říká, že

přepravcem mohou být využita pouze jeho vozidla. Podmínka (3.39) zajišťuje, že je každý požadavek přiřazen pouze jednomu přepravci. Podmínka (3.40) říká, že každý uzel pro doručení/vyzvednutí je navštíven každým přepravcem nanejvýše jednou. Podmínka (3.41) určuje vztah mezi časy odjezdu. Podmínka (3.42) říká, že se do depa vrací pouze prázdná vozidla. Podmínka (3.45) omezuje časová okna pro vyzvednutí a doručení požadavku.

3.1.6 Určení požadavků na obsloužení přepravci

Na základě předchozího modelu je vytvořen iterační model určení ceny za účelem maximalizace zisku aliance. Ceny v této aukci jsou Langrangeovi multiplikátory pro relaxovanou omezující podmínku (3.39). Tyto multiplikátory $\lambda = \{\lambda_l \geq 0, l \in L\}$ určené dražitelem jsou různé pro každý požadavek l.

Pro vyřešení problému přihazování je potřeba změnit předchozí účelovou funkci (3.33).

$$Z_{LR}^k = Max(\sum_{l \in L}(p_l - \lambda_l) \cdot y_{lk} - \sum_{i \in N} \sum_{j \in N, j \neq i} c_{ij} \cdot x_{ij}^k) \tag{3.46}$$

Tento model je využit poté, co dražitel určí cenu pro každý požadavek. Po určení požadavků na obsloužení dražitel upraví ceny požadavku změnou λ_l a tento postup se opakuje iterativně do doby než je porušena nějaká z omezujících podmínek.

3.1.7 Úprava ceny dražitelem

Jak už bylo zmíněno, pro úpravu ceny je použit Langrangeův multiplikátor vypočten rovnicí (3.47).

$$\lambda_l^{(m+1)} = Max\{\lambda_l^m + t_m \cdot (\sum_{k \in K} y_{lk} - 1), 0\}, l \in L \tag{3.47}$$

Aukce bude ukončena v případě, že bude porušena podmínka (3.39), počet iterací přesáhne předem daný limit nebo pokud se nezvýší zisk vypočtené z účelové funkce (3.46) minimálně o předem daný krok oproti předchozí iteraci.

3.2 Strategie vyhodnocení požadavků

Cílem přepravců je najít takové požadavky, které je pro přepravce výhodné nabídnout ostatním. Zároveň však nechtějí zveřejnit citlivá data o jejich podnikání. Z toho důvodu je složité najít efektivní strategii vyhodnocení požadavků. V této strategii musí být zohledněny následující faktory:

1. Marginální zisk přepravce
2. Vzdálenost do vlastního depa
3. Vzdálenost do depa konkurenčních přepravců
4. Vzdálenost k ostatním požadavkům

Dále jsou popsány vybrané strategie pro vyhodnocení požadavků založené na jednotlivých faktorech.

3.2.1 Výběr na základě míst pro vyzvednutí a doručení

Ze všech požadavků $r \in R_a$, kde R_a je soubor požadavků držených přepravcem a. Přepravce vybere soubor požadavků $B \subseteq R_a$. Každý přepravce může poskytnout dané množství svých požadavků do aukce. Dále přepravce vypočte ohodnocení e_B^1 pomocí hodnotící funkce:

$$e_B^1 = \sum_{r,q \in B} (t_{p_r p_q} + t_{d_r d_q}) \tag{3.48}$$

Kde $r, q \in B$ je dvojice vzdáleností mezi místy vyzvednutí požadavků p_r a p_q a míst pro doručení d_r a d_q. Vzdálenost mezi dvěma uzly je značena t_{ij}. Soubor s nejnižší hodnotou je poté poslán do aukce v případě FTL, kdy nelze obsloužit požadavky jednou cestou. [23]

3.2.2 Výběr na základě vzdálenosti od depa

Přepravce a vybere požadavky $r \in R_a$, které mají nejmenší vzdálenost k depu jiného přepravce. Vzdálenost požadavku r do $o_c, c \in C$, kde C je soubor všech spolupracujících přepravců, je určena $dmax$ kritériem, tj. maximální vzdálenosti mezi místem vyzvednutí

p_r a místem doručení d_r a depem o_c. Vypočtené ohodnocení e_r^2 je hodnota minimální vzdálenosti do depa jiného přepravce o_c:

$$e_r^2 = min_{c \in C}\{max\{t_{p_r o_c}, t_{d_r o_c}\}\} \tag{3.49}$$

Přepravce vybírá n_a požadavků s nejnižším ohodnocením a pošle je do aukce. [23]

3.2.3 Výběr na základě marginálního zisku

Požadavky jsou vybrány v závislosti na jejich marginálním zisku. Každý přepravce vybere požadavky s nejnižším marginálním ziskem a poskytne je ostatním přepravcům. Každý přepravce vypočítá svůj marginální zisk pro nabízené požadavky bez ohledu na kompenzaci. Poté jsou požadavky (nebo soubory požadavků) přiřazeny přepravcům tak, aby byl celkový součet marginálních zisků co nejvyšší. V případě snížení celkových nákladů a tím zvýšení zisku je tento zisk rozdělen mezi přepravce. Výběr tedy můžeme rozdělit do pěti kroků:

1. Vytvoření souboru požadavků pro dražbu
2. Vytvoření svazku požadavků obdržených od přepravců
3. Určení marginálních zisků pro každý svazek požadavků
4. Přiřazení svazků přepravcům
5. Rozdělení vygenerovaného zisku mezi přepravce

Tyto kroky jsou opakovány, dokud se výsledný zisk zvyšuje. Kroky 1 a 3 jsou automaticky prováděny přepravci a kroky 2,4 a 5 jsou prováděny centrální autoritou. [17]

3.2.4 Generování nabídek

Po odeslání všech požadavků k aukci jsou vytvořeny svazky požadavků centrální autoritou, která vybírá za všech možných kombinací požadavků. Z toho důvodu roste počet možných svazků požadavků exponenciálně $(2^n - 1)$, kde n je počet požadavků, které jsou nabízeny. Cena, za kterou se nabídky nebo celé svazky draží je vypočtena na základě marginálního zisku, který je vypočten jako množství příjmů za splnění požadavku bez nákladů na

daný požadavek. Předpokládáme, že se přepravci budou při dražbě chovat férově a budou nabízet skutečně vypočtené hodnoty. V případě, že se přepravce pokusí zvýšit svůj zisk přihazováním vyšší než skutečné ceny na své vlastní požadavky, může tím zvýšit riziko, že svůj nabízený požadavek neprodá. V případě, že bude nabízet nižší, než skutečnou cenu za cizí požadavky, vystavuje se riziku, že bude ostatními přeplacen. Neférové jednání není tedy vyloučené, ale zároveň se může stát, že díky tomu přepravce nezvýší svůj zisk. V modelech však je kontrola kapacity, a tak nemůže přepravce přihazovat na svazky požadavků, které nemůže obsloužit. [23]

3.3 Výměna požadavků založená na aukci

Pro metodu výměny požadavků uvažujeme dva druhy aukcí, aukci jednotlivých požadavků neboli *Single Request Reassignment* (SRRA) a aukci svazku požadavků neboli *Bundle request reassignment* (BRRA), které jsou popsány dále. První ze zmíněných algoritmů je jednodušší a počítá pouze s jedním požadavkem k dražbě. Jelikož přepravci neznají soubor všech požadavků, které budou k dražbě nabízeny, musí přepravci přistupovat k těmto požadavkům postupně bez závislosti na ostatních. V případě BRRA jsou informace o všech nabídkách známé před začátkem aukce. [17]

3.3.1 Aukce jednotlivých požadavků

V případě SRRA přidává přepravce pouze jeden požadavek do aukce. V každém aukčním kole je prodáván celkově nejméně výdělečný požadavek a za vyvolávací cenu je vybrán marginální zisk vlastníka požadavku. Ostatní přepravci za požadavek nabídnou jejich vypočtený marginální zisk pro daný požadavek. V případě, že je některá nabídka vyšší než vyvolávací cena aukce je úspěšná a vlastník nejvyšší nabídky získá požadavek. Jako kompenzace je vyplacena původnímu vlastníkovi druhá nejvyšší nabídka výhercem aukce. Poté je zahájeno další kolo aukce. Tento typ aukce se nazývá *second price sealed-bid* (také Vickeryova aukce), kde se oproti obálkové metodě neplatí nejvyšší částka. Tento typ aukce podporuje přepravce, aby během aukce využili skutečně získané hodnoty. Nevýhoda tohoto postupu je typicky malé množství vydražených požadavků. [23]

3.3.2 Kombinatorické aukce

Kombinatorická aukce je typ aukce, kde účastníci mohou dražit současně více než jednu položku ve svazcích (*bundles*) na bázi všechno nebo nic. Z pohledu BRRA tedy přepravci mohou dražit skupinu požadavků najednou. Mechanismus aukce je založen na metodě *first price sealed-bid*, která spočívá ve výhře nejvyšší nabídky a zaplacení celé hodnoty. Je povoleno dražit veškeré kombinace nabízených požadavků najednou. Přepravci mohou každé kolo odeslat jednu nabídku ve výši jejich marginálního zisku. Centrální autorita poté musí najít nejlepší řešení pro přiřazení nabídek přepravcům za účelem maximalizace zisku. Tento problém je znám jako *Winner determination problem*. Formulace matematického modelu BRRA: [23]

- C soubor přepravců
- R soubor požadavků
- B soubor svazků $B \subseteq R$
- P_{bc} cena, jakou je přepravce c ochotný zaplatit za svazek b
- W_{br} parametr určující zdali je požadavek r zahrnut v svazku b
- Q_{bc} parametr určující zdali přepravce c odeslal nabídku na svazek b
- y_{bc} binární rozhodovací proměnná označující, že je svazek b připsán přepravci c

Účelová funkce:

$$max \sum_{c \in C} \sum_{b \in B} P_{bc} y_{bc} \tag{3.50}$$

Omezující podmínky:

$$\sum_{b \in B} y_{bc} \leq 1, \forall c \in C \tag{3.51}$$

$$\sum_{c \in C} y_{bc} \leq 1, \forall b \in B \tag{3.52}$$

$$y_{bc} \leq Q_{bc}, \forall b \in B, \forall c \in C \tag{3.53}$$

$$\sum_{b \in B} \sum_{c \in C} y_{bc} \cdot W_{br} = 1, \forall r \in R \tag{3.54}$$

$$y_{bc} \in 0, 1, \forall b \in B, \forall c \in C \tag{3.55}$$

Účelová funkce (3.50) maximalizuje celkové cashflow. Každý přepravce může vydražit pouze jeden svazek v daném kole, což je zajištěno podmínkou (3.51). Podmínka (3.52) zaručuje, že každý svazek vydražen pouze jednou. Přepravce může získat svazek pouze v případě že náleží do jeho trasy (3.52). Dle podmínky (3.54) může být každý požadavek obsloužen právě jednou. Podmínka (3.55) zaručuje binární hodnoty v rozhodovací proměnné y_{bc}.

V případě, že je při dokončení kola aukce nalezeno řešení s vyšším celkovým ziskem než v předchozím kole, je zahájeno další kolo. Počáteční hodnota pro celkový zisk je vypočítán ze situace kdy každý přepravce vydraží své požadavky za vyvolávací cenu. To je zaručeno tím, že přepravci draží i vlastní požadavky. Z toho důvodu se také nemůže stát, aby celkový zisk aliance klesl. Z toho také pro přepravce vyplývá, že získá nejhůře tento počáteční zisk, i v případě neúspěšné aukce.

4. Implementace modelů

V této kapitole je popsáno prostředí použité pro implementaci modelů a poté je provedena analýza jednotlivých modelů. Jako zdroj testovacích dat byl zvolen soubor 101 uzlů R101 sloužící pro srovnávání optimalizačních modelů VRPTW(viz příloha A). Nejprve jsou analyzovány modely problému obchodního cestujícího a problému okružních jízd, na kterých jsou složitější modely založeny. Poté je analyzován model centralizované spolupráce přepravců a porovnán s modelem decentralizované spolupráce přepravců z pohledu výpočetní složitosti a optimality výsledného řešení. Naměřená délka řešení úloh je relativní a je závislá na rychlosti procesoru, veškeré simulace v této práci byly řešeny na počítači vybaveném dvou jádrovým procesorem Intel Core i5-4210U 2.40GHz.

4.1 Nástroje pro řešení úloh

Pro možnost analyzovat vybrané modely byl vybrán komerční nástroj IBM ILOG CPLEX (dále referováno jako CPLEX). CPLEX je optimalizační softwarový balík původně vyvinutý americkým matematikem Robertem Bixbym. Od roku 1988 byl komerčně prodáván firmou CPLEX Optimization Inc. Ta byla v roce 1997 koupena společností ILOG, která byla koupena v lednu 2009 firmou IBM. Tento program je dostupný pro akademické účely zdarma. Zároveň lze stáhnout grafické vývojové rozhraní, které usnadňuje práci s tímto nástrojem. Důležitá je také dostupná dokumentace, kde jsou popsány všechny funkce, a je nezbytná pro porozumění programu a jeho následného použití. [31]

4.1.1 CPLEX

CPLEX je softwarový balík obsahující sadu nástrojů, algoritmů a poskytuje také rozhraní pro komunikaci s dalšími aplikacemi jako MS Excel nebo programovacími jazyky Python, C++ a Java. Pomocí balíku CPLEX je možné řešit následující problémy matematického

programování.

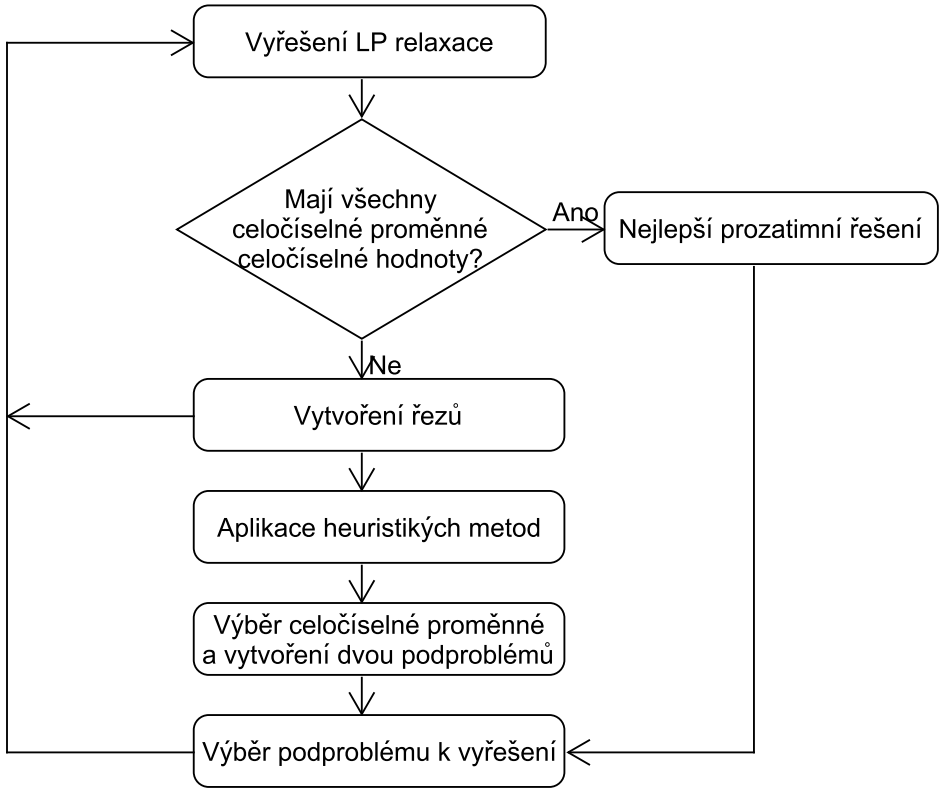
- lineární programování (LP)
- smíšené celočíselné programování (MIP)
- kvadratické programování (QP)
- smíšené celočíselné kvadratické programování (MIQP)

Při řešení MIP CPLEX nejprve model předzpracuje s cílem zmenšit velikost a zpřesnit formulaci problému. Poté je provedena LP relaxace úlohy, což znamená uvolnění rozhodovacích proměnných například z nabývání celočíselných hodnot na proměnnou nabývající reálné hodnoty. Dále je provedena kontrola, zda-li má problém řešení a jsou upřesněny meze problému. Po předzpracování jsou použity heuristické metody prohledávání stromu do hloubky a prohledávání okolí. Díky heuristických metodám lze nalézt řešení složitého MIP problému v krátkém čase a tím pomoci rychleji ověřit optimalitu výsledného řešení před heuristické metody.

Dále CPLEX využívá metodu větví a řezů (viz kapitola 2.1.2) pro nalezení optimálního řešení zadaného modelu (obr. 4.1),tj. nalezení hodnot rozhodovacích proměnných modelu, pro které jsou splněny všechny omezující podmínky.

V tomto nástroji je také možné vytvořit IBM ILOG skript v jazyce OPL, který vytvořené modely umožňuje spouštět opakovaně s jinými parametry k získání věrohodnějších dat pro analýzu modelu. CPLEX také umožňuje takzvaný teplý start, kdy je možné zadat modelu předem nalezené lokální optimum dosažené s vlastní heuristickou metodou. Tím může být urychleno nalezení optimálního řešení pro daný model. [31]

Obrázek 4.1 Medota větví a řezů



Zdroj: vlastní zpracování

4.1.2 Jazyk OPL

Optimalizační programovací jazyk IBM ILOG OPL slouží k modelování a řešení optimalizačních problémů. Poskytuje možnost modelování lineárních, kvadratických a celočíselných programů. OPL model se skládá z následujících částí

- definice datových struktur a deklarace dat
- předzpracování dat (nepovinné)
- definice modelu (rozhodovací proměnné, účelová funkce, omezující podmínky)
- zpracování výsledků (nepovinné)
- kontrola běhu programu (nepovinné)

Jedním ze základů OPL je datový typ *range* obsahující posloupnost celých čísel kde člen $a_{n+1} = a_n + 1$. Tento typ je používaný pro deklaraci polí, proměnných a iteraci přes data. Datové struktury jsou definovány pomocí klíčového slova *tuple*. Stejně jako databázové tabulky mohou mít deklarovaný unikátní klíč pomocí klíčového slova *key*, který umožňuje přistup k datům souboru *tupple* objektů pomocí unikátních identifikátorů. $Tuple$ může obsahovat primitivní datové typy *int*, $float$ a jednorozměrné pole tvořené těmito typy, *string* nebo *tuple*. Ze všech těchto typů je také možné vytvořit soubory dat.

Rozhodovací proměnné modelu jsou definovány pomocí klíčového slova *dvar*. Tyto proměnné mohou mít typ *int*, $float$ a *boolean* a nebo mohou být vícerozměrné pole proměnných daného typu. K definování složitějších rozhodovacích proměnných slouží výrazy, které se definují pomocí klíčového slova *dexpr*. Pro maximalizaci a minimalizaci výrazu účelové funkce jsou definována klíčová slova *maximize* a *minimize*. Omezující podmínky lze definovat pomocí klíčového slova *subjectto*. Samotný zdrojový kód je psán do souboru s koncovkou *.mod* a data pro daný model do souboru s koncovkou *.dat*.

Data je možné načíst v datovém souboru *.dat* z tabulky ve formátu MS Excel pomocí instrukce $SheetConnectiontabulka(\text{"}soubor.xls\text{"});$, která vytvoří proměnnou *tabulka*, přes kterou lze přistupovat k datům uloženým v souboru $soubor.xls$. Data jsou vyčtena pomocí instrukce $DataSheetRead(tabulka, \text{"“}data!A2 : A4\text{"});$, která vrátí data z buněk $A2 - A4$ do proměnné $Data$. Tu je potřeba dále inicializovat v souboru modelu *.mod* s očekávaným datovým typem $\{int\}Data = \ldots$, kde "..."znamená načtení hodnoty z datového souboru. [32]

4.2 Model problému obchodního cestujícího

Nejprve byl implementován model pro řešení problému obchodního cestujícího (TSP), popsaného v kapitole 3.1.1. Pro ilustraci je zde ukázka implementace modelu v jazyce OPL. Celá implementace modelu viz příloha B. Účelová funkce tohoto modelu je definována pomocí příkazu *minimize* a minimalizuje součet vzdáleností mezi uzly navštívené obchodním cestujícím.

```
minimize sum(i,j in CustomersAndDepo : i != j) (Distance[i][j]*x[i][j]);
```

Poté jsou definovány omezující podmínky první pro zajištění, že je každé město obchodním cestujícím navštíveno a také opuštěno právě jednou.

```
forall (i in CustomersAndDepo)
  sum(j in CustomersAndDepo : i != j) x[i][j] == 1;
  sum(j in CustomersAndDepo : i != j) x[j][i] == 1;
```

Pro eliminaci nekompletních cest je využita omezující podmínka MTZ (rov. (3.6)). Pole *Customers* neobsahuje první uzel ve kterém je umístěno depo. Omezující podmínka definována v jazyce OPL následovně:

```
forall(i,j in Customers : j != i)
    u[i] - u[j] + (CustomersNumber)*x[i][j] <= (CustomersNumber-1);
```

Po vložení testovacích dat o velikosti $n = 26, 51, 101$ byl vyřešen model úlohy. Z tabulky 4.1 je možné vidět exponenciální nárůst časové náročnosti při použití exaktního algoritmu. Při použití heuristického algoritmu je možné dosáhnout lokální optimum mnohem rychleji, ale není garantováno nalezení optimálního řešení.

Tabulka 4.1 Výsledky optimalizace modelu TSP se souborem testovacích dat.

Počet uzlů n	Hodnota účelové funkce	Trvání výpočtu (s)
26	308,4	1,17
51	454,9	23,59
101	635,9	155,7

Zdroj: vlastní zpracování

4.3 Model problému okružních jízd

Dále byl analyzován model problému obchodního cestujícího (VRP) s účelovou funkcí, která minimalizuje celkovou vzdálenost. Tento model je možné rozšířit přidáním omezujících podmínek a proto slouží jako výchozí bod pro další modely. Implementace modelu viz příloha C.
Pro analýzu byly zvoleny následující parametry:

- kapacita vozidel $C = 200$
- $N = 1, 2, \ldots, n$ soubor všech uzlů
- $V = 1, 2, \ldots, v$ soubor vozidel
- Depo v uzlu 0
- souřadnice měst a doba obsloužení ze souboru testovacích dat R101

Účelová funkce minimalizující celkovou ujetou vzdálenost vozidel k, oproti problému obchodního cestujícího je však přidána proměnné x dimenze vozidel k.

```
minimize sum(k in Vehicles,i,j in CustomersAndDepo : i != j)
  (Distance[i][j]*x[k][i][j]);
```

Navíc jsou rozšířeny také omezující podmínky modelu. První dvě podmínky zaručují opuštění všech vozidel k z depa a třetí podmínka zajišťuje opuštění uzlu po jeho navštívení.

```
forall(k in Vehicles)
    sum (j in CustomersAndDepo)x[k][0][j] == 1;


  forall(k in Vehicles)
    sum (i in CustomersAndDepo) x[k][i][0] == 1;


  forall(h in Customers, k in Vehicles)
    sum(i in CustomersAndDepo)x[k][i][h]
    - sum(j in CustomersAndDepo)x[k][h][j] == 0;
```

Po vložení testovacích dat o velikosti $n = 26, 51, 71$ a $v = 4$ byl vyřešen model úlohy obdobně jako pro problém obchodního cestujícího. V tabulce 4.2 je opět vidět nárůst časové náročnosti s přibývajícím množstvím uzlů a vozidel. Úloha s počtem uzlů $n = 100$ nemohla být vyřešena z důvodu nedostatku paměti RAM pro velikost vyhledávacího stromu, jehož velikost přesahovala 3,5GB. Z tohoto důvodu byl pro třetí výpočet snížen počet uzlů na $n = 71$. Prakticky zde lze vidět nárůst zkoumaného prostoru nejen zvýšením počtu uzlů, ale také přidáním dimenze úlohy.

Tabulka 4.2 Výsledky optimalizace modelu TSP se souborem testovacích dat.

Počet uzlů n	Počet vozidel v	Hodnota účelové funkce	Trvání výpočtu (s)
26	4	348,2	9,41
51	4	476,8	186,91
71	4	587,2	1645,06

Zdroj: vlastní zpracování

4.4 Model problému okružních jízd s časovými okny

Model *Vehicle Routing Problem with Time Windows* je rozšíření modelu VRP o časová okna, kdy má vozidlo obsloužit požadavek. VRP můžeme rozšířit přidáním následujících dvou omezujících podmínek pro zajištění časového okna navštívení uzlu a včasného odjezdu do následujícího.Implementace modelu viz příloha D.

```
forall(i in CustomersAndDepots, k in Vehicles)
  LBTW[i] <= s[k][i] <= UBTW[i];

forall(i,j in AllNodes, k in Vehicles)
  (x[k][i][j] == 1) => (s[k][i] + Time[i][j]  <= s[k][j]);
```

Dále je model rozšířen omezujícími podmínkami zaručující nepřekročení maximální kapacity vozidla *Capacity* a omezující podmínkou zaručující nepřekročení počtu dostupných vozidel v.

```
forall(k in Vehicles)
  sum(i in Customers, j in CustomersAndDepots)(Demand[i] * x[k][i][j])
    <= Capacity;


forall(k in Vehicles, j in CustomersAndDepots)
  sum (k in Vehicles, j in CustomersAndDepots) x[k][0][j] <= v;
```

Takto definovaný model již obsahuje kontrolu kapacit vozidel a v případě jejich překročení využije další vozidlo v případě, že je dostupné. Tím zaručuje využití optimálního počtu vozidel. Dále generuje řešení bez překročení časových oken požadavků. Tento model však nereaguje na výnosnost požadavku vzhledem k jeho nákladům.

Pro analýzu modelu byly použity následující hodnoty:

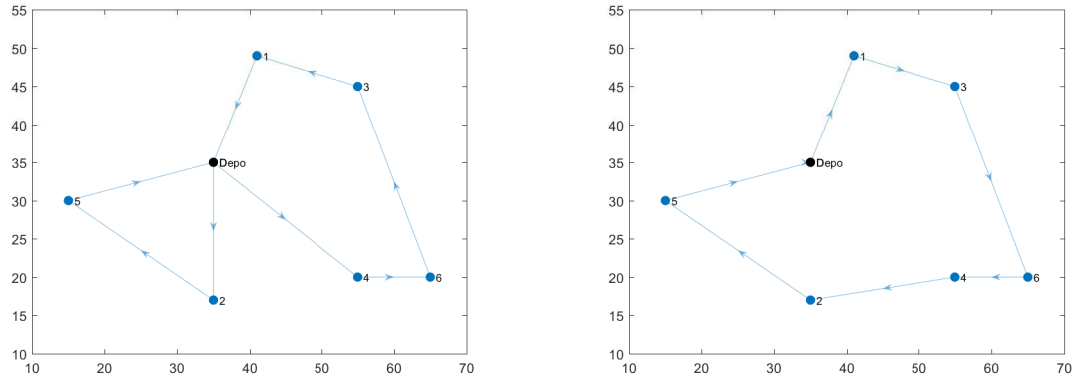
- počet vozidel $v = 2$
- kapacita vozidel $C = 200$
- souřadnice měst a doba obsloužení ze souboru testovacích dat R101

Účel modelu je simulovat reakci na různé vstupní data. Nejprve byla pro porovnání nákladů při použití modelu VRPTW použita základní sada dat obsahující 3 požadavky na obsluhu a dvě depa. Zároveň byla zohledněna časová okna jednotlivých požadavků, kapacita vozidel a velikost nákladu. Pro základní sadu dat bylo nalezeno optimální řešení (obr. 4.2), kde všechny jsou všechny požadavky obslouženy jedním přepravcem.

Cílem modelu je minimalizovat celkovou vzdálenost a proto je výsledná trasa závislá na velikosti požadavků a časových oken. Na obrázku 4.2 jsou zobrazeny dvě situace. První situace, kdy má vozidlo nedostatečnou kapacitu na obsloužení všech požadavků v jedné trase a proto musí být použito i druhé vozidlo a druhá situace kdy má vozidlo dostatečnou kapacitu na obloužení všech požadavků. Výsledná vzdálenost varianty (a) má hodnotu 154 zatímco pro variantu (b) je hodnota nižší a to rovna 131. Porovnání těchto hodnot ukazuje, že v případě, že je možné použít menší množství vozidel, bude výsledná vzdálenost kratší. To jest za předpokladu, že vozidla mají depo ve stejném uzlu.

V případě takto definovaných omezujících podmínek hledá model řešení s co nejkratší trasou, což znamená řešení s minimálním množstvím co nejvíce využitých vozidel jak z hlediska kapacity, tak z hlediska časové vytíženosti vozidla. Tento princip můžeme převést

Obrázek 4.2 Porovnání tras pro různé velikosti obsluhovaných požadavků



(a) VRPTW za nedostatečné kapacity vozidla. (b) VRPTW za dostatečné kapacity vozidla.

Zdroj: vlastní zpracování

také na případ více přepravců s jední vozidlem, kdy tato pravidla platí obecně také. Tento případ je popsán v následující části.

4.5 Model problému spolupráce přepravců

Pro implementaci modelu spolupráce přepravců (CCPLTL) byl rozšířen předchozí model VRPTW o dimenzi přepravců a dále byl každému požadavku přiřazen uzel pro doručení, ale také uzel pro vyzvednutí. Implementace modelu viz příloha E.

Pro analýzu modelu byly zvoleny následující parametry:

- Počet uzlů $N = 17$
- Počet přepravců $K = 3$
- Počet vozidel přepravce $W_k = 2$
- Počet požadavků $L = 5$

Účelová funkce modelu maximalizuje zisk všech přepravců na základě ceny za realizované požadavky a nákladů na jejich obsloužení. Oproti předchozímu modelu VRPTW je nutné předem vypočítat cenu pro každý požadavek.

```
dexpr float revenue = sum(k in carriers, r in requests)
```

```
                        price_p[r]*y[r][k];
dexpr float cost = sum(v in vehicles, i in nodes, j in nodes : i != j)
                        cost_c[i][j]*x[v][i][j];
maximize revenue - cost;
```

Modelu bylo nutné přidat omezující podmínku pro navštívení uzlu pro vyzvednutí stejným přepravcem jako uzlu pro doručení a zaručit časovou posloupnost vyzvednutí a doručení pro každou dvojici.

```
forall(k in carriers, i,j in nodes){
  if(pickup_P[i] != -1
  && delivery_D[j] != -1
  && pickup_P[i] == delivery_D[j]){
      sum(n in nodes)x[k][n][i] == sum(n in nodes)x[k][n][j];
      t[k][i] <= t[k][j];
  }
}
```

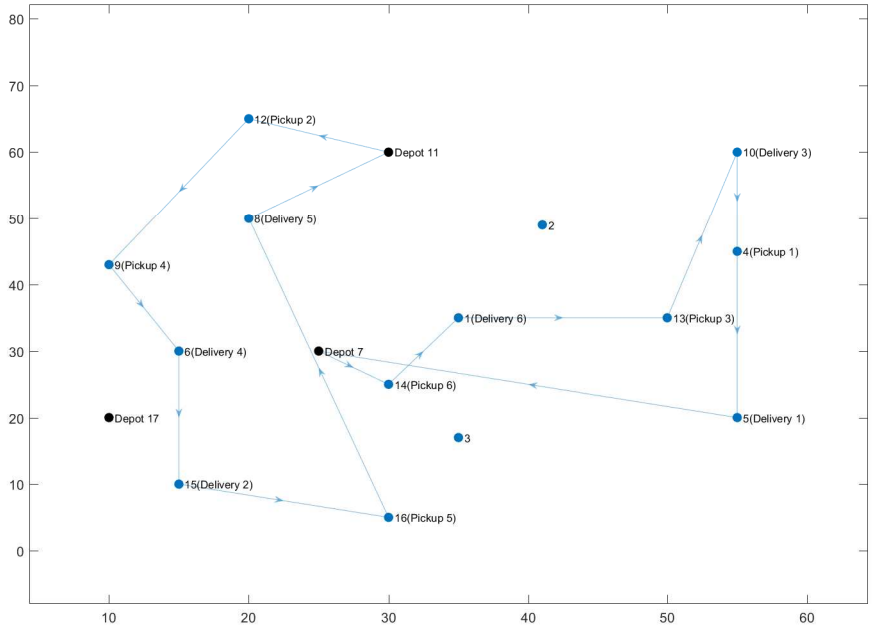
Dále bylo nutné přidat omezení zaručující splnění všech požadavků alespoň jedním přepravcem. Upravením následující nerovnice na rovnici lze docílit obsloužení všech požadavků. V případě, že je tato omezující podmínka definována nerovnicí $y[l][k] <= 1$, nemusí být obslouženy všechny požadavky, ale pouze požadavky, které zvýší hodnotu účelové funkce, tj. celkový zisk. Tím je také zajištěna celková bezztrátovost všech obsloužených požadavků.

```
forall(l in requests)
  sum(k in carriers) y[l][k] <= 1;
```

Jednotlivé požadavky mohou být obslouženy libovolným přepravcem, a tak lze tento model považovat za centralizovaný. Na obrázku 4.3 je znázorněna výsledná trasa, kde jsou požadavky obslouženy dvěma přepravci ze tří, za použití dvou vozidel. Celkově je obslouženo šest požadavků. V tomto případě jsou obslouženy všechny požadavky.

Ve výsledném řešení jsou zahrnuta časová okna, ve kterých jsou požadavky obslouženy a které nesmí být překročeny. Zároveň nesmí být překročena kapacita vozidla aktuálně

Obrázek 4.3 Spolupráce přepravců (6 požadavků)



Zdroj: vlastní zpracování

naloženým nákladem. Dále je zaručeno, že je uzel pro vyzvednutí požadavku vždy navštíven před uzlem pro doručení. V případě použití více vozidel je zajištěno obsloužení vyzvednutí a doručení jedním a tím samým vozidlem. V případě, že by požadavek nebyl pro žádného přepravce rentabilní, nebude obsloužen.

4.6 Centralizovaný model

Nejdříve byl analyzován centralizovaný model, kde všechny požadavky přepravců mohly být sdíleny s cílem maximalizovat celkový zisk aliance (více kapitola 1.3.1). Vstupní data jsou vygenerována do testovacích souborů dat. Pro ilustraci je zde první soubor rozebrán. Před spuštěním modelu jsou nejdříve načteny souřadnice uzlů, ze kterých jsou vypočteny vzdálenosti mezi jednotlivými uzly. Vzdálenost je počítána jako euklidovská. Dále je nutné vypočítat cenu a čas strávený vozidlem pro každou trasu. Všechny testovací sety zahrnují 3 přepravce, jejichž vozidla mají kapacitu 10. V prvním souboru dat je analyzováno 6 požadavků, tady 12 přepravních uzlů, které musejí být navštíveny. Tyto požadavky jsou generovány náhodně vybráním místa pro vyzvednutí a místa doručení

z předem definovaném souboru uzlů. Každý uzel může být přiřazen maximálně jednomu požadavku. Každý požadavek má náhodně přiřazenou velikost 2 (20% kapacity vozidla),5 (50% kapacity vozidla), nebo 10 (100%kapacity vozidla) a požadavky nemohou být rozděleny na více menších částí. Cena za přepravu požadavku je vypočtena jako $p_l = \alpha * (1 + \beta) \cdot d \cdot (c_{0i} + cij + c_{j0})/C$, kde α značí výšku marže přepravce a β míru využití vozidla (čím více je vozidlo využité, tím vyšší je stanovena cena) a d je velikost požadavku. Časová okna pro obsloužení požadavku jsou převzata ze souboru testovacích dat R101 (příloha 5.4). Ne všem přepravcům musely být ve výsledném řešení přiřazeny požadavky, jelikož pro obloužení nebylo nutné využít všech kapacit.

Tabulka 4.3 Doba běhu se soubory testovacích dat

Data	Centralizovaný model	Centralizovaný model s heuristikou	Zlepšení
A	1027,45s	316,55s	69%
B	503,92s	306,30s	39%
C	975,42s	492,53s	49%

Zdroj: vlastní zpracování

Ačkoliv byl centralizovaný model schopen najít optimální řešení pro vytvořené soubory dat, celková doba pro vyřešení problému se pohybovala v řádech desítek minut až několika hodin pro vyšší počet požadavků (viz tabulka 4.3). S cílem snížit výpočetní čas byla implementována heuristická metoda dvojitého vkládání (kapitola 2.2.2) pro nalezení počátečního řešení a vylepšení tohoto řešení pomocí metody 3-opt (kapitola 2.2.3). Nalezené řešení bylo použité pro teplý start modelu s cílem usnadnit programu prořezávání stromu. Po přidání heuristiky byl snížen výpočetní čas pro nalezení optimálního řešení v průměru o 52,3%.

4.7 Decentralizovaný model

Dále byl analyzován decentralizovaný model(1.3.2),pro který byla použita stejná sada dat jako pro centralizovaný model. Navíc bylo nutné určit výchozí hodnoty multiplikátoru λ_l pro každý požadavek, jelikož na přesnosti jeho počátečního výběru se odvíjel počet iterací potřebných k nalezení optimálního řešení. V případě, že byl požadavek z důvodu snížení zisku o hodnotu λ_l pro všechny přepravce nevýhodný a tudíž by snížil jejich

vlastní zisk, byla λ_l v následujícím kroku snížena o 1% původní stanovené ceny. Naopak, v případě, že byl požadavek alokován více než jedním přepravcem, byla hodnota zvýšena o $0,5\%$ původní ceny. V případě, že byl požadavek alokován právě jedním přepravcem, zůstávala hodnota λ_l nezměněna. Na základě experimentů s daty byla upravena funkce pro výpočet λ_l multiplikátoru. V případě nenalezení řešení po 40 iteracích je o polovinu snížen krok t, což mělo v případech přiřazení požadavku více přepravcům za následek nalezení výsledného řešení pro zadaná data. Snížení kroku již pro počáteční iterace mělo negativní efekt na rychlost nalezení řešení. Pro nalezení výsledků kombinatorické aukce byl nastaven krok t na hodnotu $t = 0,02$ s pozitivními výsledky na rychlost nalezení řešení. S vyšší hodnotou kroku byla některá řešení přeskočena, a tak nebylo nalezeno nejlepší možné řešení. Snížení kroku však zvýšilo počet iterací, a tím i celkovou čas potřebný pro nalezení řešení. V případě požadavku alokovanému dvěma nebo více přepravcům se navrhované řešení dle [7], pomocí přiřazení požadavku náhodnému přepravci neukázalo jako vhodné, z důvodu nepredikovatelné změny výsledné trasy a tudíž by nebylo zajištěno nalezení optimálního řešení. Pokud nastane případ, že se hodnota λ_l dostane do záporných hodnot, znamená to, že žádný z přepravců není schopný doručit požadavek za cenu, která byla původně stanovena. Jelikož má model za cíl obsloužit všechny požadavky, je v tomto případě alokován přepravci s nejvyšší hodnotou λ_l. V důsledku by to znamenalo, že by požadavek byl obsloužen se ztrátou, avšak nejmenší možnou v rámci spolupracujících přepravců. Tento případ nastává převážně pro požadavky, které využívají malou část kapacity vozidla. To způsobí stanovení nižší ceny požadavku. Výsledná hodnota účelové funkce centralizovaného a decentralizovaného modelu se pro zkoumané soubory dat shodují, což prokazuje validitu decentralizovaného modelu.

Celkový čas na vyřešení úlohy pomocí decentralizovaného modelu je dán počtem iterací a dobou běhu jednotlivých podúloh. Nelze však předpokládat, že s vyšším počtem iterací nutně poroste celkový čas vyřešení problému decentralizovanou metodou, jelikož čas na vyřešení podúloh může být v některých případech kratší s vyšším počtem iterací a v některých delší s nižším počtem iterací. Dále je nutné si uvědomit, že CPLEX nalezne řešení prvních podúloh rychleji, než podúlohy blížící se k optimálnímu řešení ke konci řešení modelu. Na počet iterací mají vliv také vložená data, nastavený krok t změny koeficientu λ_l, ale také výchozí nastavení koeficientů λ_l.

Tabulka 4.4 Hodnoty účelové funkce centralizovaného a decentralizovaného modelu

	A	B	C	D	E
Náklady	534,8	237,2	252,5	320.15	223,6
Výnosy	2179,87	408,38	394,99	214,80	245,37
Doba běhu	316,55s	306,3s	492,53s	80,91	24,36s
Náklady	534,8	237,2	252,5	214,80	223,6
Výnosy	2179,87	408,38	394,99	320,15	245,37
Doba běhu	632,45s	246,25s	1244,74s	222,7	194,56s
Iterace	21	53	48	57	33

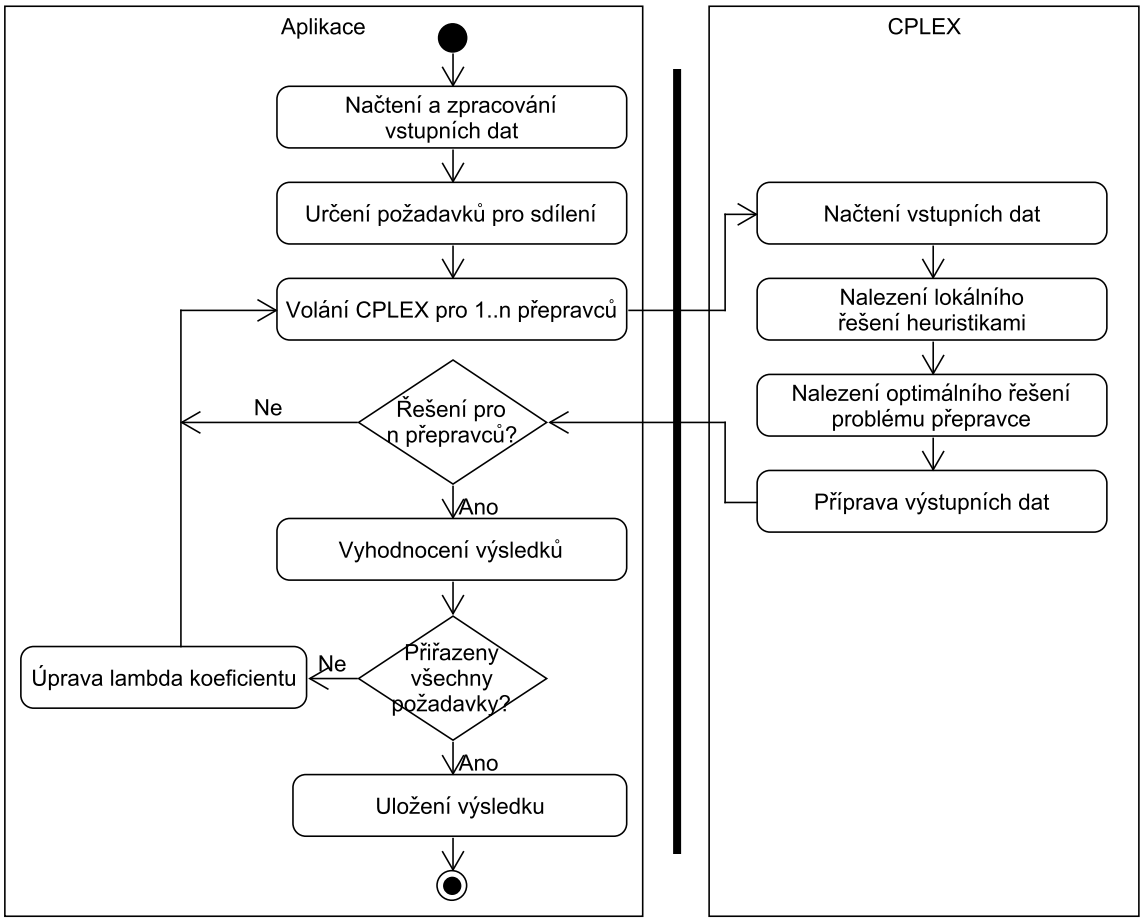
Zdroj: vlastní zpracování

4.8 Decentralizovaný model s výběrem požadavků

Pro případ, že přepravci nejsou ochotni sdílet veškeré své požadavky, byl decentralizovaný model rozšířen funkcionalitou, která umožňuje výběr požadavků pro aukci. Aby bylo možné s modelem pracovat i v další kapitole, kde bude potřeba pracovat i s jinými zdroji dat, než které OPL podporuje (CSV, databázová rozhraní, rozhraní externí aplikace nebo HTTP protokol), bylo nutné model spouštět z jiného prostředí, které tyto zdroje dat podporuje. Jádro modelu (definované v souboru *.mod*) je možné spustit pomocí programovacích jazyků Java, C++ nebo Python. Pro tento účel byl zvolen interpretovaný jazyk Python s knihovnou *doopl* a celkový algoritmus je zobrazen na diagramu aktivit 4.4.

Nejprve jsou načteny vstupní data, případně vygenerovaná náhodná. Dále je možné vyřešit globální optimalizační problém pro porovnání optimálního řešení a řešení dosažené decentralizovaným modelem. Poté je zavolána zvolená metoda pro určení požadavků pro sdílení. Následně je zavolán CPLEX model, který pro zadaná data nalezne optimální řešení. Model je zavolán pro všechny přepravce zvlášť a po získání optimálního řešení všech přepravců je zkontrolováno, zda jsou veškeré požadavky obslouženy. V případě, že nejsou, je upraven koeficient pro výpočet multiplikátoru λ_l a model zavolán s upravenými hodnotami. Celý výpočet se opakuje do přiřazení všech požadavků nebo překročení předem daného počtu iterací. Byly implementovány dvě metody výběru sdílených požadavků. Pro testování byla vybrána metoda výběru na základě vzdálenosti od depa (kapitola 3.2.2), kdy každý přepravce vybere právě jeden požadavek ke sdílení.

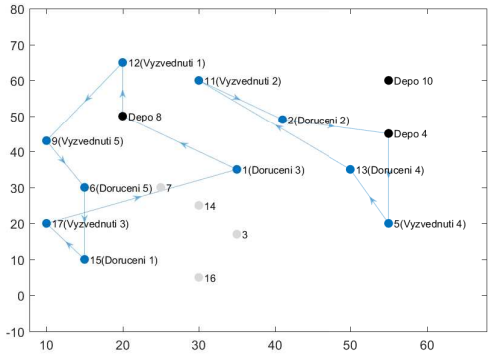
Obrázek 4.4 Diagram aktivit decentralizovaného modelu s výběrem požadavků



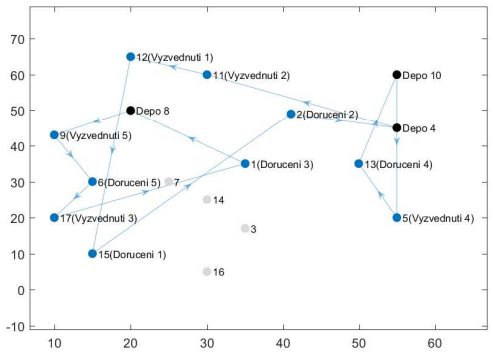
Zdroj: vlastní zpracování

Pro porovnání efektivity výběru požadavků byly nalezeny hodnoty účelové funkce bez sdílení požadavků mezi přepravci a poté byl vyřešen ten samý problém za použití zmíněné metody výběru požadavků ke sdílení. Pro ilustraci je na obrázku 4.5 (a) zobrazeno optimální řešení při sdílení všech požadavků pro soubor dat B. Na obrázku 4.5 (b) řešení modelu s náhodně přiřazenými požadavky a na obrázku 4.5 (c) každý dopravce sdílí jeden vybraný požadavek na základě vzdálenosti od depa. V tomto případě byl realokován pouze požadavek č. 4, který model přiřadil jinému přepravci, s cílem zvýšení celkového zisku přepravců. Z toho důvodu jednomu z přepravců nebyl přiřazen žádný požadavek k obloužení, stejně jako v případě optimálního řešení.

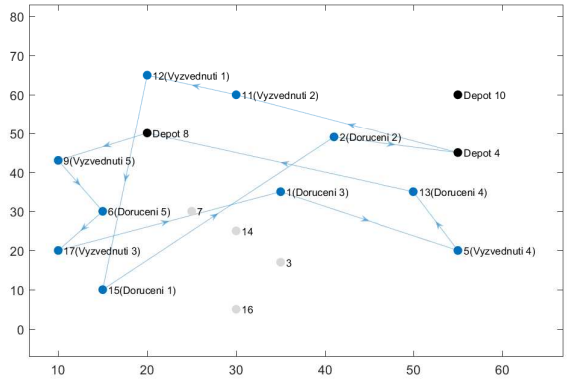
Obrázek 4.5 Porovnání tras s různými metodami sdílení požadavků.



(a) Optimální řešení centralizovaného modelu



(b) Řešení bez sdílení požadavků mezi přepravci



(c) Řešení decentralizovaného modelu se sdílením požadavků na základě vzdálenosti od depa

Zdroj: vlastní zpracování

Mimo matice přechodů mezi uzly jsou výstupem modelu také časová okna v zadaném

rozmezí. Ta určují kdy má být uzel navštíven přepravcem. Pro soubor dat B bylo zadáno časové rozmezí 0-200, pro ilustraci jsou výsledná časová okna přepravců stanovena pro optimální řešení následovně:

Přepravce 1: Depo $4(t_4 = 0) \rightarrow$ Uzel $5(t_5 = 35,0) \rightarrow$ Uzel $13(t_{13} = 60,8) \rightarrow$ Uzel $11(t_{11} = 102,8) \rightarrow$ Uzel $2(t_2 = 128,3)$

Přepravce 2: Depo $4(t_4 = 0) \rightarrow$ Uzel $12(t_{12} = 25,0) \rightarrow$ Uzel $9(t_9 = 59,1) \rightarrow$ Uzel $6(t_6 = 83,0) \rightarrow$ Uzel $15(t_{15} = 139,8) \rightarrow$ Uzel $17(t_{17} = 160,9) \rightarrow$ Uzel $1(t_1 = 200,0)$

Hodnota t_i značí čas, kdy má přepravce vyrazit z uzlu i do uzlu následujícího. Čas potřebný obsloužení požadavku (vyložení či naložení) je v čase již zahrnut.

Pro porovnání efektivity decentralizovaného modelu s výběrem požadavků byly použity soubory dat A-E. Ty byly vyřešeny centralizovaným modelem s cílem najít optimální řešení pro zadaná data. Požadavky byly náhodně přiřazeny přepravcům a bez sdílení požadavků byl vyřešen decentralizovaný model. Poté byly vybrány požadavky ke sdílení na základě vzdálenosti od depa ostatních přepravců a vyřešen decentralizovaný model. Ve sloupci Z_{opt} je výsledný zisk centralizovaného modelu, tj. optimální řešení při sdílení všech požadavků. Ve sloupci $Z_{\text{bez sdileni}}$ je celkový zisk decentralizovaného modelu bez sdílení požadavků mezi přepravci. Ten vyjadřuje dosažený zisk přepravců bez vzájemné spolupráce. Ve sloupci Z_{sdileni} je celkový zisk dosažený decentralizovaným modelem s výběrem požadavků pro sdílení.

Tabulka 4.5 Vliv metod sdílení požadavků na celkový zisk přepravců

Data	Z_{opt}	$Z_{\text{bez sdileni}}$	Z_{sdileni}	$u_1 = \frac{Z_{\text{bez sdileni}}}{Z_{opt}}$	$u_2 = \frac{Z_{\text{sdileni}}}{Z_{opt}}$	$u_3 = \frac{Z_{\text{sdileni}} - Z_{\text{bez sdileni}}}{Z_{\text{sdileni}}}$
A	1645,72	1095,58	1390,19	67%	84%	21%
B	171,18	-111,08	-82,98	-65%	-48%	33%
C	142,49	23,95	105,76	17%	74%	77%
D	105,35	-0,05	97,25	0%	92%	100%
E	21,77	-94,63	21,77	-435%	100%	534%
Průměr:				-83%	61%	153%

Zdroj: vlastní zpracování

Z naměřených dat v tabulce 4.5 byly vypočteny tři ukazatele. První ukazatel $u_1 = \frac{Z_{\text{bez sdileni}}}{Z_{opt}}$ vyjadřuje podíl zisku bez sdílení požadavků mezi přepravci a optimálního zisku při sdílení všech požadavků. Obdobně druhý ukazatel $u_2 = \frac{Z_{\text{sdileni}}}{Z_{opt}}$ vyjadřuje podíl zisku dosaženého pomocí sdílení požadavků metodou založené na vzdálenosti od depa přepravců a optimálního zisku. Odečtením ukazatelů $\sum u_2 - u_1$ získáme procentuální nárůst mezi $Z_{\text{bez sdileni}}$ a Z_{sdileni} oproti Z_{opt}. V průměru je tato hodnota rovna 144%. V případě souboru dat E ja tato hodnota dokonce 535%. Z toho lze usuzovat, že v případě, kdy je nějaký požadavek pro vlastníka ztrátový, může přidělení tohoto požadavku přepravci, pro kterého ztrátový není velice pomoci přiblížení se optimálnímu řešení, jako právě pro soubor dat E.

Ukazatel u_3 vyjadřuje o kolik procent je $Z_{\text{bez sdileni}}$ nižší, než Z_{sdileni}. Průměr hodnot je opět umocněn nárůstem Z_{sdileni} souboru dat E. Naopak nejnižší nárůst je naměřen v souboru dat A (21%). V žádném případě nenastalo zhoršení celkového Z_{sdileni} oproti $Z_{\text{bez sdileni}}$, což potvrzuje teorii z podkapitoly 3.3.2. V průměru by přepravci při nesdílení požadavků přicházely o 153% z hodnoty zisku. Hodnoty ukazují, že je metoda nejen schopna zvýšit zisk, ale také omezit ztrátu v případě ztrátových požadavků.

5. Využití modelu pro reálné vstupy

Tato kapitola se zabývá zjištění reálných silničních tras a jejich využití v decentralizovaném modelu z předchozí kapitoly. Pro předchozí simulace byla využita data R101 obsahující následující:

- Číslo požadavku (0-101)
- Souřadnice X (0-77)
- Souřadnice Y (0-77)
- Okno pro obsloužení požadavku (0-240 minut)
- Čas potřebný k obloužení požadavku (0-10 minut)

Pro reálné použití je nutné nahradit souřadnice X a Y skutečnými souřadnicemi zeměpisné šířky a délky. Časové okno je ponecháno jako rozmezí čtyř hodin (0-240 minut). Kapacita vozidel je ponechána jako procentuální využití vozidla v rozmezí 1-10, kde hodnota 10 značí 100% kapacitu vozidla. Dále je nutné zajistit výpočet reálné délky trasy a čas potřebný k přejezdu mezi uzly.

5.1 Vyhledávání reálné trasy pomocí OSRM

Projekt *Open Source Routing Machine* (OSRM) poskytuje výkonný mechanismus pro vyhledávání nejkratší trasy v dopravní síti po celém světě. Projekt je dostupný k volnému užití pod licencí BSD a je podporován operačními systémy Windows, Linux, OS X a FreeBSD. Pro výpočet trasy používá data projektu OSM (*Open Street Map*). Jelikož výpočet nejkratší trasy může zabrat v reálné dopravní síti i několik vteřin, OSRM využívá techniku kontrakční hierarchie.

Metoda kontrakční hierarchie je používána pro vyhledávání nejkratší trasy mnohem efektivněji než například Djikstrův algoritmus. Algoritmus se dělí v zásadě na tři hlavní části, předzpracování zadaného grafu, fáze přizpůsobení a zpracování požadavku na nejkratší trasu.

Předzpracování, které může zabrat i několik hodin, vygeneruje graf s jednotlivými uzly. Ve fázi přizpůsobení, která trvá rámcově několik vteřin, jsou vypočteny váhy hran, které mohou být později upraveny, například kvůli změně dopravní situace. Poslední fáze slouží pro vyhledávání nejkratší trasy v řádu milisekund.

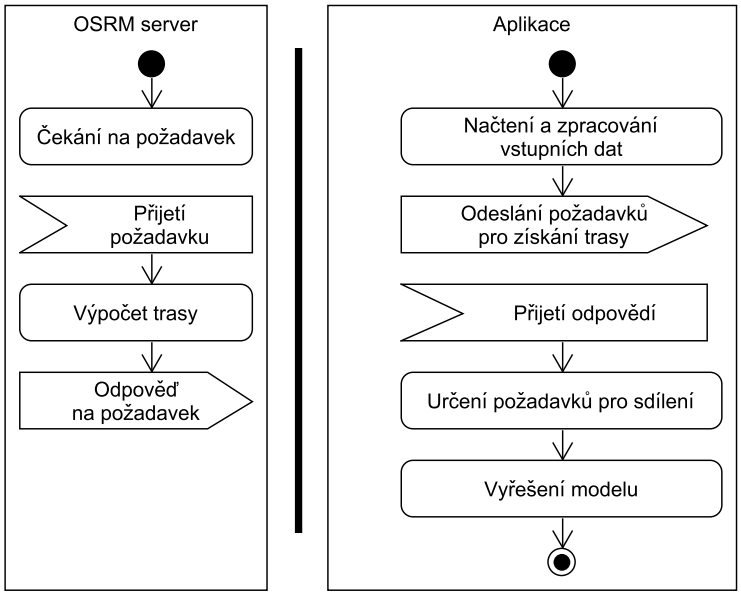
Jelikož je OSRM poskytován jako *open source*, je možné spouštět výpočty přímo na lokálním prostředí. To představuje výhodu nejen v rychlejší odezvě dotazování, ale i kvůli možnosti konfigurace přímo pro daný účel, pro který bude OSRM použito. Všechny mapy jsou shromažďovány komunitou a dostupné pod *open source* licencí podobně, jako portál Wikipedia shromažďuje články. Jeden z portálů s mapami ve formátu osm.pbf je například *Geofabrik.de*. Při stahování map je nutné brát v potaz vysoké nároky na kapacitu zařízení, na kterém budou mapy předzpracovány, zejména pak na velikost paměti RAM. Po importování vybrané mapy lze spustit server vracející informace o trase pro zadané body. Dotazování komunikuje přes *http* protokol a vrací standardně data ve formátu $JSON$.

5.2 Integrace OSRM

Pro získání reálné vzdálenosti mezi dvěma body na mapě lze použít například euklidovskou vzdálenost, což je dostačující pro teoretické výpočty. V reálné situaci však taková vzdálenost neodpovídá skutečné dopravní síti. Nesprávná vzdálenost by měla za následek nevalidní řešení optimalizačního problému. Dále není možné počítat s konstantní časovou náročností, jelikož se přepravce ve městě pohybuje pomaleji než mimo město a podobně. Použitý nástroj OSRM poskytuje informaci nejen o vzdálenosti, ale také o časové náročnosti dané trasy. Díky tomu je možné zachovat omezující podmínky modelu pro časová okna. *OSRM routing* funguje na principu klient-server pomocí HTTP protokolu. Server odpovídá na požadavek obsahující zeměpisnou šířku a zeměpisnou délku po sobě jdoucích uzlů ve formátu JSON. Pro účely této práce je potřeba nalézt vzdálenost mezi všemi kombinacemi dvou uzlů. Integrace OSRM byla provedena do řešení z předchozí kapitoly (viz obr. 5.1). Po načtení vstupních dat jsou odeslány požadavky na OSRM ser-

ver, který tyto požadavky zpracuje a vrátí požadované informace. Pro detailní zobrazení aktivity *Vyřešení modelu* viz obr. 4.4.

Obrázek 5.1 Diagram aktivit po přidání OSRM



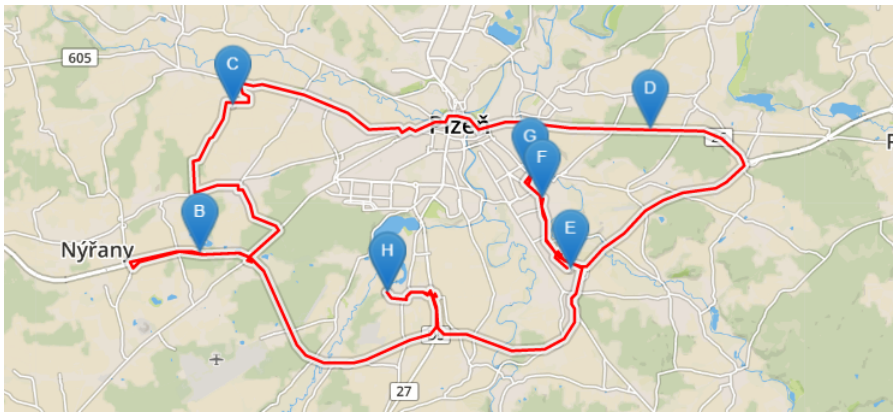
Zdroj: vlastní zpracování

Po vyřešení modelu je každému přepravci stanovena trasa a časová okna pro obsloužení požadavků. Pro ilustraci je na na obrázku 5.2 zobrazeno nalezené řešení pro jednoho ze tří přepravců. Trasa byla zobrazena na mapě pomocí nástroje *Leaflet open machine*. Tento nástroj umožňuje vložení trasy pomocí souřadnic a využívá OSRM pro výpočet trasy. Trasa je vygenerována z výstupu modelu ve formátu po sobě jdoucích souřadnic jednotlivých požadavků. Navigace není součástí implementovaného řešení, proto nelze zajistit přesná trasa po které by přepravce požadavky obsloužil. To by v reálném případě mohlo znamenat rozdíl mezi skutečnou délkou trasy a délkou trasy použitou pro nalezení řešení.

Obslouženy jsou tři požadavky umístěné náhodně v okolí města Plzně. Opět bylo nutné dodržet veškeré omezující podmínky modelu jako dodržení časových oken, kapacit vozidla a následnost uzlů pro vyzvednutí a doručení.

Depo $H(t_H = 0) \rightarrow$ Uzel $B(t_B = 25, 5) \rightarrow$ Uzel $C(t_C = 44, 3) \rightarrow$ Uzel $D(t_D = 75, 5) \rightarrow$ Uzel $E(t_E = 97, 6) \rightarrow$ Uzel $F(t_F = 113, 0) \rightarrow$ Uzel $G(t_G = 125, 1) \rightarrow$ Depo $H(t_H = 133, 9)$

Obrázek 5.2 Příklad výsledné trasy přepravce



Zdroj: vlastní zpracování s využitím *Leaflet open machine*

Hodnota t_i značí čas, kdy má přepravce vyrazit z uzlu i do uzlu následujícího. Čas potřebný obsloužení požadavku je v čase již zahrnut. Každý požadavek má předem dané dvojice uzlů. Na mapě jsou uzly pro první požadavek značeny (B, F), pro druhý požadavek (C, D) a pro třetí požadavek (E, G). Depo se nachází v uzlu H. Celá trasa i s desetiminutovými okny pro naložení a vyložení nákladu v každém uzlu zabere tedy 133,9 minut.

5.3 Zhodnocení výsledků

Pro analýzu byly generovány náhodné požadavky které byly vyřešeny pomocí centralizovaného modelu s cílem najít optimální řešení pro zadaná data. Poté byly přiřazeny požadavky náhodným přepravcům a vyřešen decentralizovaný model bez sdílení požadavků. Nakonec byl pro stejná data vyřešen model s výběrem požadavků ke sdílení na základě vzdálenosti od depa ostatních přepravců. Od každého přepravce byl vybrán nejvýše jeden požadavek ke sdílení.

Ve sloupcích Z_{opt}, $Z_{\text{bez sdileni}}$ a Z_{sdileni} jsou naměřené hodnoty pro soubory dat A-E. Hodnoty vyjadřují celkový zisk všech přepravců za obloužení zadaných požadavků. Ukazatel $u_2 = \frac{Z_{\text{sdileni}}}{Z_{opt}}$ vyjadřuje přiblížení řešení decentralizovaného modelu k optimu dosaženého sdílením vybraných požadavků. V průměru má hodnotu 68% optimálního řešení čím se přiblížil o 45% $(\sum u_2 - u_1)$ optimálnímu řešení oproti hodnotám řešení bez sdílení, zobrazených ve sloupci $u_1 = \frac{Z_{\text{bez sdileni}}}{Z_{opt}}$. V průměru byla pro všechny soubory dat sdílena jedna třetina dostupných požadavků. To potvrzuje, že při správném výběru

Tabulka 5.1 Vliv metod sdílení požadavků na celkový zisk přepravců

Data	Z_{opt}	$Z_{\text{bez sdileni}}$	Z_{sdileni}	$u_1 = \frac{Z_{\text{bez sdileni}}}{Z_{opt}}$	$u_2 = \frac{Z_{\text{sdileni}}}{Z_{opt}}$	$u_3 = \frac{Z_{\text{sdileni}} - Z_{\text{bez sdileni}}}{Z_{\text{sdileni}}}$
A	44,85	21,55	21,55	48%	48%	0%
B	36,42	3,01	24,39	8%	67%	88%
C	60,48	28,42	47,80	47%	79%	41%
D	36,62	-14,09	28,53	-38%	78%	149%
E	87,19	43,04	58,29	49%	67%	26%
Průměr:				23%	68%	61%

Zdroj: vlastní zpracování

požadavků pro sdílení lze nepřímo úměrně zvýšit celkový zisk přepravců. Poslední ukazatel $u_3 = \frac{Z_{\text{sdileni}} - Z_{\text{bez sdileni}}}{Z_{\text{sdileni}}}$ vyjadřuje o kolik procent je $Z_{\text{bez sdileni}}$ nižší, než Z_{sdileni} To znamená o kolik procent možného zisku by přepravci přicházeli za nesdílení požadavků. V průměru by si pohoršili o 61%.

Ve všech případech byl zisk dosažený sdílením požadavků stejný nebo vyšší, než při nesdílení požadavků. Pro soubor dat A nastala situace, kdy byly přiřazeny přepravcům jejich vlastní požadavky. Tato situace nastala z důvodu nenalezení lepšího řešení pomocí kombinatorické aukce. To může být způsobené výběrem nevhodných požadavků ke sdílení.

5.4 Návrhy na zlepšení

V následující části jsou popsány návrhy na zlepšení vytvořeného řešení, možnosti dalšího rozvoje a využití. Nejprve je provedena diskuse na snížení složitosti problému.

Ačkoliv řešení bylo ve všech případech schopné nalézt řešení daného problému, pro vyšší počet uzlů ($10 < n < 15$) se čas potřebný k nalezení řešení pohyboval v řádech desítek minut (viz výsledky naměřené v tabulce 4.3). Pro reálné využití by bylo nutné výpočetní náročnost snížit. Toho by bylo možné docílit rozdělením implementovaného modelu na dvě části. První část hledající optimální trasu pro obsloužení všech kombinací zadaných požadavků a druhá část řešící optimální přiřazení kombinatorickou aukcí. Pro zlepšení efektivity by dále bylo vhodné vylepšit navrhované heuristické řešení o časová a kapacitní omezení, což by mělo za následek nalezení lepšího suboptimálního řešení. Takto nalezené řešení by mohlo být využito jako počáteční řešení pro nalezení optima,

nebo dokonce jako výsledné řešení problému. Na složitost problému má také vliv velikost vstupních dat, zejména počet požadavků a počet přepravců. Pro zmenšení složitosti by bylo možné rozdělit jak požadavky, tak přepravce podle jejich lokality a tím zmenšit prohledávaný prostor. Rozdělením celého prostoru na několik podprostorů by vzniklo několik výpočetně méně náročných úloh (viz výsledky naměřené v tabulce 4.1). V případě snížení složitosti by bylo mnohem méně výpočetně náročné aktualizování data s právě dostupnými požadavky.

Pro rozšíření navrhované formy spolupráce je další aspekt vstřícnost zákazníků a jejich zájem o přepravu za pomoci sdílení kapacit. Co se týče motivace, přepravci jsou motivováni zvýšením zisku, zákazníci však takto motivováni v navrhovaném řešení nejsou. Pro zvýšení ochoty zákazníků by mohlo být implementováno snížení ceny na základě snížení nákladů na přepravu a poskytnout zákazníkům transparentní informace o rozdílu ve výsledné ceně. Tato práce se nevěnuje právním aspektům výměny požadavků, ale bylo by nutné, aby takto vytvořená spolupráce splňovala veškerá právní náležitosti (viz vyhláška o přepravním řádu pro veřejnou drážní a silniční osobní dopravu č. 175/2000 Sb.).

Pro reálné použití by bylo třeba rozšířit současné řešení o uživatelské rozhraní pro zadávání požadavků a zobrazení požadavků přidělených přepravci. Veškerá data by musela být ukládána do databáze a zpracována na vzdáleném serveru. Pro lepší využití přepravci by dále bylo vhodné provést analýzu využívaných systémů a umožnit automatické vložení požadavků z těchto systémů.

Závěr

V této práci byly zkoumány možnosti optimalizace logistických procesů se zaměřením na možnost využití aukce požadavků. Hlavním cílem bylo vytvoření řešení, které by zlepšilo využití kapacit přepravců. Pro splnění tohoto cíle bylo nutné nejprve popsat možnosti spolupráce mezi přepravci. V kapitole 1 - Problematika byla popsána problematika spolupráce přepravců z hlediska logistického řetězce. Dále byly v této kapitole popsány formy řízení spolupráce přepravců. Na základě tohoto členění byly v kapitole 4 - Implementace modelů, implementovány modely horizontální spolupráce pro centralizovaný a decentralizovaný přístup a zároveň zde byl popsán nástroj použitý pro implementaci modelů, IBM ILOG CPLEX.

Pro umožnění implementace těchto modelů bylo nutné nejdříve popsat vhodné optimalizační problémy a metody řešení, což bylo provedeno v kapitole 3 - Analýza možností řešení. Centralizovaný a decentralizovaný model byl testován na pěti souborech náhodně generovaných dat s cílem ověřit efektivitu decentralizovaného modelu. Pro všechny soubory dat byla metoda schopna nelézt optimální řešení, avšak z důvodu omezeného počtu měření, nelze s jistotou zaručit nalezení optimálního řešení ve všech možných případech. S cílem snížit výpočetní náročnost modelu byly použity vybrané heuristické metody popsané v kapitole 2 - Metody řešení úloh. Implementace tohoto řešení měla za následek snížení času potřebného k nalezení optimálního řešení modelu.

Dále byl implementován decentralizovaný model s výběrem požadavků k aukci. Pro výsledný model bylo opět provedeno měření za použití vygenerovaných souborů náhodných dat. Výsledkem byl nárůst celkového zisku spolupracujících přepravců ve všech případech. V kapitole 5 - Využití modelu pro reálné vstupy, byl model přizpůsoben reálným datům a bylo provedeno měření s pěti náhodně generovanými soubory dat, pro které model opět nalezl řešení zvyšující původní zisk. Byl tak splněn hlavní cíl práce, nalezení řešení umožňující lepší využití kapacit v silniční dopravě v rámci balíkové přepravy.

Vyšší výsledný zisk znamená nižší náklady na požadavky, které jsou ve vytvořeném řešení přímo úměrné ujeté vzdálenosti. Pro ověření validity zvoleného přístupu by bylo vhodné provést rozsáhlejší měření s vyšším počtem souborů dat. V kapitole 5 - Využití modelu pro reálné vstupy, jsou blíže popsány návrhy na zlepšení pro další vývoj.

Při porovnání stanovených cílů diplomové práce, které měly být dosaženy s výsledky diplomové práce, je možné konstatovat, že ačkoliv byly cíle splněny a výsledné řešení bylo ověřeno testovacími daty, je pro jeho použití ve skutečném prostředí nutné odpovědět na mnoho dalších otázek.

Seznam zkratek

CCN	Collaborative Carrier Network. Síť pro spolupráci přepravců.
CCNVRP	Collaborative Carrier VRP. Problém okružních jízd při spolupráci přepravců.
CCPLTL	Carrier Collaboration Problem in Less than Truckload Transportation. Spolupráce přepravců s nákladem menším než kapacita vozidla
CTM	Collaborative Transportion Management. Kolaborativní řízení dopravy.
MIP	Mixed Integer Programming. Řešení problému kde je jedna a více proměnných celočíselného typu.
MTZ	Miller, Tucker, Zemlin formulation. Formulace omezující podmínky pro eliminaci dělení tras.
OPL	Optimization Programming Language. Jazyk použitý pro modelování v CPLEX studiu.
TSP	Traveling salesman problem. Problém obchodního cestujícího.
VICS	Voluntary Inter-industry Commerce Solutions Association. Dobrovolná asociace pro meziodvětvové obchodní normy.
VRP	Vehicle routing problem. Problém okružních jízd.

Seznam literatury

[1] FERNÁNDEZ, Elena, Mireia ROCA-RIU a M. Grazia SPERANZA. *The Shared Customer Collaboration Vehicle Routing Problem. European Journal of Operational Research* [online]. 2018, 265(3), 1078-1093 [cit. 2019-04-21]. DOI: 10.1016/j.ejor.2017.08.051. ISSN 03772217. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S0377221717307828

[2] GANSTERER, Margaretha a Richard F. HARTL. *Collaborative vehicle routing: A survey. European Journal of Operational Research* [online]. 2018, 268(1), 1-12 [cit. 2019-04-21]. DOI: 10.1016/j.ejor.2017.10.023. ISSN 03772217. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S0377221717309360

[3] WILMJAKOB, Herlyn. *Innovative Methods in Logistics and Supply Chain Management: The Bullwhip Effect in Expanded Supply Chains and the Concept of Cumulative Quantities* [online]. 2014, 26 [cit. 2019-04-21]. Dostupné z: https://www.researchgate.net/publication/313239968_The_Bullwhip_Effect_in_Expanded_Supply_Chains_and_the_Concept_of_Cumulative_Quantities

[4] BLECKER, Thorsten a Wolfgang KERSTEN, RINGLE, Christian, ed. *Innovative Methods in Logistics and Supply Chain Management: Current Issues and Emerging Practices*. Berlin: epubli, 2014. ISBN 978-3-7375-0341-9

[5] BRØDE JEPSEN, Lisbeth. *Innovative Methods in Logistics and Supply Chain Management: Critical Success Factors for Horizontal Logistic Collaboration* [online]. 2014 [cit. 2019-04-21]. Dostupné z: https://www.researchgate.net/publication/264541570_Critical_Success_Factors_for_Horizontal_Logistic_Collaboration

[6] CRUIJSSEN, Frans, Olli BRÄYSY, Wout DULLAERT, Hein FLEUREN a Marc SALOMON. *Joint route planning under varying market conditions: Critical Success Factors for Horizontal Logistic Collaboration* [online]. 2007, 2014, 37(4), 287-304 [cit. 2019-04-21]. DOI: 10.1108/09600030710752514. ISSN 0960-0035. Dostupné z: http://www.emeraldinsight.com/doi/10.1108/09600030710752514

[7] DAI, Bo a Haoxun CHEN. *PRICE-SETTING BASED COMBINATORIAL AUCTION APPROACH FOR CARRIERS' COLLABORATION IN LESS THAN TRUCKLOAD TRANSPORTATION* [online]. 2011 [cit. 2019-04-21]. Dostupné z: https://www.researchgate.net/publication/221539571_Price-setting_based_Combinatorial_Auction_Approach_for_Carriers'_Collaboration_in_Less_than_Truckload_Transportation

[8] LAPORTE, Gilbert. *The traveling salesman problem: An overview of exact and approximate algorithms. European Journal of Operational Research* [online]. 1992, 59(2), 231-247 [cit. 2019-04-21]. DOI: 10.1016/0377-2217(92)90138-Y. ISSN 03772217. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/037722179290138Y

[9] GUAJARDO, Mario a Mikael RÖNNQVIST. *A review on cost allocation methods in collaborative transportation: An overview of exact and approximate algorithms. International Transactions in Operational Research* [online]. 2016, 23(3), 371-392 [cit. 2019-04-21]. DOI: 10.1111/itor.12205. ISSN 09696016. Dostupné z: http://doi.wiley.com/10.1111/itor.12205

[10] ÖZENER, Okan Örsan, Özlem ERGUN a Martin SAVELSBERGH. *Allocating Cost of Service to Customers in Inventory Routing. Operations Research* [online]. 2013, 61(1), 112-125 [cit. 2019-04-22]. DOI: 10.1287/opre.1120.1130. ISSN 0030-364X. Dostupné z: http://pubsonline.informs.org/doi/abs/10.1287/opre.1120.1130

[11] FERGUSON, Thomas. Game Theory: Games in Coalitional Form. [online]. 2014 [cit. 2019-04-22]. Dostupné z: https://www.math.ucla.edu/ tom/-Game_Theory/Contents.html

[12] SOYSAL, Mehmet, Jacqueline M. BLOEMHOF-RUWAARD, Rene HAIJEMA a Jack G.A.J. VAN DER VORST. *Modeling a green inventory routing problem for perishable products with horizontal collaboration* [online]. 2018, 89, 168-182 [cit. 2019-04-22]. DOI: 10.1016/j.cor.2016.02.003. ISSN 03050548. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S030505481630020X

[13] PALHAZI CUERVO, Daniel, Christine VANOVERMEIRE a Kenneth SÖRENSEN. *Determining collaborative profits in coalitions formed by two partners with varying characteristics. Transportation Research Part C: Emerging Technologies* [online]. 2016, 70, 171-184 [cit. 2019-04-22]. DOI: 10.1016/j.trc.2015.12.011. ISSN 0968090X. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S0968090X15004271

[14] CHAO, I-Ming, Bruce L. GOLDEN a Edward A. WASIL. *The team orienteering problem. European Journal of Operational Research* [online]. 1996, 88(3), 464-474 [cit. 2019-04-22]. DOI: 10.1016/0377-2217(94)00289-4. ISSN 03772217. Dostupné z: http://linkinghub.elsevier.com/retrieve/pii/0377221794002894

[15] WANG, Xin a Herbert KOPFER. *Collaborative transportation planning of less-than-truckload freight. OR Spectrum* [online]. 2014, 36(2), 357-380 [cit. 2019-04-22]. DOI: 10.1007/s00291-013-0331-x. ISSN 0171-6468. Dostupné z: http://link.springer.com/10.1007/s00291-013-0331-x

[16] SPRENGER, Ralf a Lars MÖNCH. *A decision support system for cooperative transportation planning: Design, implementation, and performance assessment. Expert Systems with Applications* [online]. 2014, 41(11), 5125-5138 [cit. 2019-04-22]. DOI: 10.1016/j.eswa.2014.02.032. ISSN 09574174. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S0957417414001080

[17] BERGER, Susanne a Christian BIERWIRTH. *Solutions to the request reassignment problem in collaborative carrier networks. Transportation Research Part E: Logistics and Transportation Review* [online]. 2010, 46(5), 627-638 [cit. 2019-04-22]. DOI: 10.1016/j.tre.2009.12.006. ISSN 13665545. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S1366554509001562

[18] CLAUSEN, Jens. *Branch and Bound Algorithms: Principles and Examples* [online]. 1999, , 30 [cit. 2019-04-22]. Dostupné z: https://imada.sdu.dk/Employees/jbj/heuristikker/TSPtext.pdf

[19] BIANCHI, Leonora, Marco DORIGO, Luca Maria GAMBARDELLA a Walter J. GUTJAHR. *A survey on metaheuristics for stochastic combinatorial optimization. Natural Computing* [online]. 2009, 8(2), 239-287 [cit. 2019-04-22]. DOI: 10.1007/s11047-008-9098-4. ISSN 1567-7818. Dostupné z: http://link.springer.com/10.1007/s11047-008-9098-4

[20] DORIGO, M., V. a A. COLORNI. *Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)* [online]. 26(1) [cit. 2019-04-22]. DOI: 10.1109/3477.484436. ISSN 10834419. Dostupné z: http://ieeexplore.ieee.org/document/484436/

[21] EL-SHERBENY, Nasser A. *Vehicle routing with time windows. Journal of King Saud University - Science* [online]. 2010, 22(3), 123-131 [cit. 2019-04-22]. DOI: 10.1016/j.jksus.2010.03.002. ISSN 10183647. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S1018364710000297

[22] BERGER, Susanne a Christian BIERWIRTH. *The Collaborative Carrier Vehicle Routing Problem for Capacitated Traveling Salesman Tours.* [online]. Bremen, 2007, , 4 [cit. 2019-04-22]. Dostupné z: http://cs.emis.de/LNI/Proceedings/Proceedings109/gi-proc-109-013.pdf

[23] GANSTERER, Margaretha a Richard F. HARTL. *Request evaluation strategies for carriers in auction-based collaborations. OR Spectrum* [online]. 2016, 38(1), 3-23 [cit. 2019-04-22]. DOI: 10.1007/s00291-015-0411-1. ISSN 0171-6468. Dostupné z: http://link.springer.com/10.1007/s00291-015-0411-1

[24] GLOVER, Fred. *Tabu search - Part I* [online]. 1990, , 32 [cit. 2019-04-22]. Dostupné z: https://www.researchgate.net/publication/220668740_Tabu_search_-_Part_I

[25] BALAPRAKASH, Prasanna. *Estimation-based metaheuristics for stochastic combinatorial optimization.* ACM SIGEVOlution [online]. 2010, 5(1), 18-19 [cit. 2019-04-22]. DOI: 10.1145/1811155.1811157. ISSN 19318499. Dostupné z: http://portal.acm.org/citation.cfm?doid=1811155.1811157

[26] BACHIR, Benhala, Ahaitouf ALI a Mechaqrane ABDELLAH. *Multiobjective Optimization of an Operational Amplifier by the Ant Colony Optimisation Algorithm. Electrical and Electronic Engineering* [online]. 2012, 2(4), 230-235 [cit. 2019-04-22]. DOI: 10.5923/j.eee.20120204.09. ISSN 2162-9455. Dostupné z: http://article.sapub.org/10.5923.j.eee.20120204.09.htm

[27] LIN, Shen. Computer *Solutions of the Traveling Salesman Problem. Bell System Technical Journal* [online]. 1965, 44(10), 2245-2269 [cit. 2020-03-20]. DOI: 10.1002/j.1538-7305.1965.tb04146.x. ISSN 00058580. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6767727

[28] RENAUD, Jacques, Fayez F. BOCTOR a Jamal OUENNICHE. *A heuristic for the pickup and delivery traveling salesman problem. Computers & Operations Research* [online]. 2000, 27(9), 905-916 [cit. 2020-04-10]. DOI: 10.1016/S0305-0548(99)00066-0. ISSN 03050548. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S0305054899000660

[29] PADBERG, Manfred a Giovanni RINALDI. *A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems. SIAM Review* [online]. 1991, 33(1), 60-100 [cit. 2020-05-04]. DOI: 10.1137/1033004. ISSN 0036-1445. Dostupné z: http://epubs.siam.org/doi/10.1137/1033004

[30] PATAKI, Gábor. *Teaching Integer Programming Formulations Using the Traveling Salesman Problem. SIAM Review* [online]. 2003, 45(1), 116-123 [cit. 2020-05-05]. DOI: 10.1137/S00361445023685. ISSN 0036-1445. Dostupné z: http://epubs.siam.org/doi/10.1137/S00361445023685

[31] IBM Support [online]. USA, 2020 [cit. 2020-05-06]. Dostupné z: https://www.ibm.com/support/home/

[32] IBM *ILOG CPLEX Optimization Studio: OPL Language Reference Manual* [online]. IBM, 2017 , 146 [cit. 2020-05-10]. Dostupné z: https://www.ibm.com/support/knowledgecenter/SSSA5P_12.7.1/ilog.odms.studio.help/pdf/opl_langref.pdf

[33] *Uživatelská příručka k definici malých a středních podniků* [online]. 2017-02-16, 60 [cit. 2020-05-8]. DOI: 10.2873/244305. Dostupné z: https://op.europa.eu/en/publication-detail/-/publication/79c0ce87-f4dc-11e6-8a35-01aa75ed71a1/language-cs

[34] *Vertical integration and horizontal integration* [online]. 2018 [cit. 2020-05-08]. Dostupné z: https://www.mbacrystalball.com/blog/strategy/vertical-horizontal-integration-strategy/

Seznam obrázků

Seznam tabulek

Seznam příloh

Příloha A: Soubor testovacích dat

i	X	Y	DEMAND	READY	DUE	TIME	i	X	Y	DEMAND	READY	DUE	TIME
1	35	35	0	0	230	0	52	49	58	10	88	98	10
2	41	49	10	161	171	10	53	27	43	9	52	62	10
3	35	17	7	50	60	10	54	37	31	14	95	105	10
4	55	45	13	116	126	10	55	57	29	18	140	150	10
5	55	20	19	149	159	10	56	63	23	2	136	146	10
6	15	30	26	34	44	10	57	53	12	6	130	140	10
7	25	30	3	99	109	10	58	32	12	7	101	111	10
8	20	50	5	81	91	10	59	36	26	18	200	210	10
9	10	43	9	95	105	10	60	21	24	28	18	28	10
10	55	60	16	97	107	10	61	17	34	3	162	172	10
11	30	60	16	124	134	10	62	12	24	13	76	86	10
12	20	65	12	67	77	10	63	24	58	19	58	68	10
13	50	35	19	63	73	10	64	27	69	10	34	44	10
14	30	25	23	159	169	10	65	15	77	9	73	83	10
15	15	10	20	32	42	10	66	62	77	20	51	61	10
16	30	5	8	61	71	10	67	49	73	25	127	137	10
17	10	20	19	75	85	10	68	67	5	25	83	93	10
18	5	30	2	157	167	10	69	56	39	36	142	152	10
19	20	40	12	87	97	10	70	37	47	6	50	60	10
20	15	60	17	76	86	10	71	37	56	5	182	192	10
21	45	65	9	126	136	10	72	57	68	15	77	87	10
22	45	20	11	62	72	10	73	47	16	25	35	45	10
23	45	10	18	97	107	10	74	44	17	9	78	88	10
24	55	5	29	68	78	10	75	46	13	8	149	159	10
25	65	35	3	153	163	10	76	49	11	18	69	79	10
26	65	20	6	172	182	10	77	49	42	13	73	83	10
27	45	30	17	132	142	10	78	53	43	14	179	189	10
28	35	40	16	37	47	10	79	61	52	3	96	106	10
29	41	37	16	39	49	10	80	57	48	23	92	102	10
30	64	42	9	63	73	10	81	56	37	6	182	192	10
31	40	60	21	71	81	10	82	55	54	26	94	104	10
32	31	52	27	50	60	10	83	15	47	16	55	65	10
33	35	69	23	141	151	10	84	14	37	11	44	54	10
34	53	52	11	37	47	10	85	11	31	7	101	111	10
35	65	55	14	117	127	10	86	16	22	41	91	101	10
36	63	65	8	143	153	10	87	4	18	35	94	104	10
37	2	60	5	41	51	10	88	28	18	26	93	103	10
38	20	20	8	134	144	10	89	26	52	9	74	84	10
39	5	5	16	83	93	10	90	26	35	15	176	186	10
40	60	12	31	44	54	10	91	31	67	3	95	105	10
41	40	25	9	85	95	10	92	15	19	1	160	170	10
42	42	7	5	97	107	10	93	22	22	2	18	28	10
43	24	12	5	31	41	10	94	18	24	22	188	198	10
44	23	3	7	132	142	10	95	26	27	27	100	110	10
45	11	14	18	69	79	10	96	25	24	20	39	49	10
46	6	38	16	32	42	10	97	22	27	11	135	145	10
47	2	48	1	117	127	10	98	25	21	12	133	143	10
48	8	56	27	51	61	10	99	19	21	10	58	68	10
49	13	52	36	165	175	10	100	20	26	9	83	93	10
50	6	68	30	108	118	10	101	18	18	17	185	195	10
51	47	47	13	124	134	10							

Zdroj: http://w.cba.neu.edu/~msolomon/r101.htm

Příloha B: TSP model

```
// Customers
int     CustomersNumber      = ...;
range   CustomersAndDepo     = 0..CustomersNumber;
range   Customers            = 1..CustomersNumber;
int   XCoord[CustomersAndDepo] = ...;
int   YCoord[CustomersAndDepo] = ...;
// Cost or distance between i and j
float Distance[CustomersAndDepo][CustomersAndDepo];

execute INITIALIZE {
  for(var i in CustomersAndDepo) {
    for (var j in CustomersAndDepo){
      if (i == j) {
        Distance[i][j] = 0;
        Time[i][j] = 0;
      } else {
        Distance[i][j] = Math.floor(Math.sqrt(Math.pow(XCoord[i]-XCoord[j], 2)
        + Math.pow(YCoord[i]-YCoord[j], 2))*10)/10;
        Time[i][j] = Distance [i][j];
      }
    }
  }
}
// Decision variables
// 1 if a vehicle drives directly from vertex i to vertex j
```

```
dvar boolean x[CustomersAndDepo][CustomersAndDepo];
// MZP constrain variable
dvar float u[CustomersAndDepo];

/*********
TSP MODEL
*********/

minimize sum(i,j in CustomersAndDepo : i != j) (Distance[i][j]*x[i][j]);;
subject to {
  forall(i in CustomersAndDepo)
    x[i][i] == 0;
    // Each customer is visited exactly once
  forall (i in CustomersAndDepo)
    sum(j in CustomersAndDepo : i != j) x[i][j] == 1;
  forall (i in CustomersAndDepo)
    sum(j in CustomersAndDepo : i != j) x[j][i] == 1;
  //subtour elimination MTZ constraint
  forall(i in Customers)
    forall(j in Customers : j != i)
      u[i] - u[j] + (CustomersNumber)*x[i][j] <= (CustomersNumber-1);
};
```

Příloha C: VRP model

```
// Vehicles
int     v           = ...;
range   Vehicles    = 1..v;
// Customers
int     CustomersNumber      = ...;
range   CustomersAndDepo     = 0..CustomersNumber;
range   Customers            = 1..CustomersNumber;
int   XCoord[CustomersAndDepo] = ...;
int   YCoord[CustomersAndDepo] = ...;
// Cost or distance between i and j
float Distance[CustomersAndDepo][CustomersAndDepo];
execute INITIALIZE {
  for(var i in CustomersAndDepo) {
    for (var j in CustomersAndDepo){
      if (i == j) {
        Distance[i][j] = 0;
      } else {
          Distance[i][j] =
            Math.floor(Math.sqrt(Math.pow(XCoord[i]-XCoord[j], 2)
            + Math.pow(YCoord[i]-YCoord[j], 2))*10)/10;
  }}}
}
// Decision variables
// 1 if a vehicle drives directly from vertex i to vertex j
dvar boolean x[Vehicles][CustomersAndDepo][CustomersAndDepo];
```

```
dvar float u[Vehicles][CustomersAndDepo]; // MZP constrain variable
/*** VRP MODEL ****/
dexpr float distance = sum(k in Vehicles,i,j in CustomersAndDepo : i!=j)
  (Distance[i][j]*x[k][i][j]);
minimize distance;
subject to {
  forall(k in Vehicles,i in CustomersAndDepo)
    x[k][i][i] == 0;
    // Each customer is visited exactly once s
  forall (i in Customers)
    sum(k in Vehicles,j in CustomersAndDepo : i != j) x[k][i][j] == 1;
    // Each customer is visited exactly once s
  forall (i in Customers)
    sum(k in Vehicles,j in CustomersAndDepo : i != j) x[k][j][i] == 1;
    // Each vehicle must leave the depot 01
  forall(k in Vehicles)
    sum (j in CustomersAndDepo)x[k][0][j] == 1;
    // All vehicles must arrive at the depo
    forall(k in Vehicles)
      sum (i in CustomersAndDepo) x[k][i][0] == 1;
    // After a vehicle arrives at a customer
    it has to leave for another destination
    forall(h in Customers, k in Vehicles)
      sum(i in CustomersAndDepo)x[k][i][h]
      - sum(j in CustomersAndDepo)x[k][h][j] == 0;
    //subtour elimination MTZ constraint
    forall(k in Vehicles,i in Customers)
    forall(j in Customers : j != i)
      u[k][i] - u[k][j] + (CustomersNumber)*x[k][i][j]
        <= (CustomersNumber-1);
};
```

Příloha D: VRPTW

```
// Vehicles
int     v          = ...;
range   Vehicles   = 1..v;
// Customers
int     CustomersNumber      = ...;
range   Customers            = 1..CustomersNumber;
// includes the starting depot and the returning depot
range   CustomersAndDepots   = 0..(CustomersNumber+1);
// Capacity
int Capacity = ...;
// Demand
int Demand[Customers] = ...;
// Time windows
int LBTW[CustomersAndDepots] = ...; // Lower Bound of the Time Window
int UBTW[CustomersAndDepots] = ...; // Upper Bound of the Time Window
int   ServiceTime[CustomersAndDepots] = ...;
int   XCoord[CustomersAndDepots] = ...;
int   YCoord[CustomersAndDepots] = ...;
// Cost or distance between i and j
float Distance[CustomersAndDepots][CustomersAndDepots];
// Cost or distance between i and j
float Time[CustomersAndDepots][CustomersAndDepots];
execute INITIALIZE {
for(var i in CustomersAndDepots) {
for (var j in CustomersAndDepots){
```

```
if (i == j) {
Distance[i][j] = 0;
Time[i][j] = 0;
} else {
          Distance[i][j] =
          Math.floor(Math.sqrt(Math.pow(XCoord[i]-XCoord[j], 2)
          + Math.pow(YCoord[i]-YCoord[j], 2))*10)/10;
        Time[i][j] = Distance [i][j] + ServiceTime[i];
        }
     }
     }
}
// Decision variables
// 1 if a vehicle drives directly from vertex i to vertex j
dvar boolean x[Vehicles][CustomersAndDepots][CustomersAndDepots];
// the time a vehicle starts to service a customer
dvar int s[Vehicles][CustomersAndDepots];
dexpr float maxTimeSpentBetweenTwoCustomers = max(a,b in CustomersAndDepots)
  (UBTW[a] + Time[a][b] - LBTW[b]);

/**********
 VRPTW
***********/
dexpr float distance = sum(k in Vehicles, i,j in CustomersAndDepots)
   (Distance[i][j]*x[k][i][j]);
minimize distance;
subject to {
forall(i in CustomersAndDepots, k in Vehicles)
  x[k][i][i] == 0;
    // Each customer is visited exactly once 1)
forall (i in Customers)
sum(k in Vehicles, j in CustomersAndDepots) x[k][i][j] == 1;
```

```
    // Each vehicle must leave the depot 0 2)
forall(k in Vehicles)
      sum (j in CustomersAndDepots)x[k][0][j] == 1;
    // All vehicles must arrive at the depot n + 1 3)
    forall(k in Vehicles)
      sum (i in CustomersAndDepots) x[k][i][CustomersNumber+1] == 1;
     // After a vehicle arrives at a customer
     //it has to leave for another destination
    forall(h in Customers, k in Vehicles)
       sum(i in CustomersAndDepots)x[k][i][h]
       - sum(j in CustomersAndDepots)x[k][h][j] == 0;
    // A vehicle can only be loaded up to it's capacity 5)
forall(k in Vehicles)
       sum(i in Customers, j in CustomersAndDepots)
       (Demand[i] * x[k][i][j]) <= Capacity;
    // Vehicle departure time from a customer and its immediate successor 6)
    forall(i,j in CustomersAndDepots, k in Vehicles)
      s[k][i] + Time[i][j] - maxTimeSpentBetweenTwoCustomers*(1 - x[k][i][j]) <= s[k]
    // The time windows are observed 7
    forall(i in CustomersAndDepots, k in Vehicles)
      LBTW[i] <= s[k][i] <= UBTW[i];
    // From depot departs a number of vehicles equal to or smaller than v 8)
    forall(k in Vehicles, j in CustomersAndDepots)
      sum (k in Vehicles, j in CustomersAndDepots) x[k][0][j] <= v;
};
```

Příloha E: CCPLTL

```
int nodes_N = ...;
int carriers_K = ...;
int requests_L = ...;
int capacity_C = ...; //C vehicle capacity
float cost_per_distance = ...;
range nodes = 1..nodes_N;
range requests = 1..requests_L;
range carriers = 1..carriers_K;
int pickup_P[nodes];   //Pi the set of requests whose pickup site is node i
int delivery_D[nodes]; //Di the set of requests whose delivery site is node i
int quantity_d[requests] = ...;        //dl quantity delivered on request l
float price_p[requests];         //pl price paid by a shipper to serve request l
int vehicles_W[carriers] = ...;  //Wk the number of vehicles owned by carrier k
int depots_o[carriers] = ...;           //ok the depot of carrier k
float alfa[carriers] = ...; //utilization rate of vehicles
float beta[carriers] = ...; //profit margin of vehicles
int XCoord[nodes] = ...;
int YCoord[nodes] = ...;
float Distance[nodes][nodes];
//tij travelling time of a vehicle from node i to node j
float Time[nodes][nodes];
//cij shipping cost from node i to j for each vehicle
float cost_c[nodes][nodes];
int LBTW[nodes] = ...;      //a the earliest service time at node i
int UBTW[nodes] = ...; //b the latest service time at node i
```

```
int ServiceTime[nodes] = ...; // servise time at node
execute INITIALIZE {
  // calculate distances
for(var i in nodes) {
for (var j in nodes){
if (i == j) {
Distance[i][j] = 0;
Time[i][j] = 0;
       } else {
        Distance[i][j] =
          Math.floor(Math.sqrt(Math.pow(XCoord[i]-XCoord[j], 2)
          + Math.pow(YCoord[i]-YCoord[j], 2))*10)/10;
        Time[i][j] = Distance[i][j] + ServiceTime[i];
        cost_c[i][j] = Distance[i][j]*cost_per_distance;
      }
    }
  }
  var car_tmp = 1;
    for(var l in requests){
    for(var i in nodes){
      if(pickup_P[i] == l){
        for(var j in nodes){
          if(delivery_D[j] == l){
            price_p[l] =
            (alfa[car_tmp] * (1 + beta[car_tmp]) * quantity_d[l]
            * (cost_c[depots_o[car_tmp]][i] + cost_c[i][j]
             + cost_c[j][depots_o[car_tmp]]))/capacity_C;
          }
        }
      }
    }
  }
```

```
}

//decision variables
//q quantity transported through arc (i, j) by carrier k
dvar int+ q[carriers][nodes][nodes];
//the number of times that arc (i, j) is visited by vehicles of carrier k
dvar boolean x[carriers][nodes][nodes];
// y 1 if request l is served by carrier k; otherwise 0
dvar boolean y[requests][carriers];
// the time at which a vehicle of carrier k leaves node i
dvar float t[carriers][nodes];

dexpr float revenue = sum(k in carriers, r in requests) price_p[r]*y[r][k];
dexpr float cost = sum(k in carriers, i in nodes, j in nodes : i != j)
    cost_c[i][j]*x[k][i][j];
minimize -(revenue - cost);
//maximize revenue - cost;

subject to {
// 1) no cyclic path in one node
forall(k in carriers)
  forall(i in nodes)
x[k][i][i] == 0;
// 2) leave node after arrival
forall(k in carriers)
  forall(i in nodes : i != depots_o[k])
sum(j in nodes : i != j) x[k][i][j] == sum(j in nodes : i != j)x[k][j][i];
// 3) capacity constrain
forall(k in carriers, i in nodes, j in nodes)
   q[k][i][j] <= capacity_C*x[k][i][j];
  // 4) quantity picked up - delivered is equal quantity from pickup node
  //- quantity from delivery node
```

```
forall(k in carriers, i in nodes)
      (sum(j in nodes : i != j) q[k][i][j])
      - (sum(j in nodes : i != j) q[k][j][i])
      == (sum(l in (pickup_P[i] .. pickup_P[i]) : l != -1) quantity_d[l]*y[l][k])
      - (sum(l in (delivery_D[i] .. delivery_D[i]) : l != -1) quantity_d[l]*y[l][k]);
// 5) all vehivles which left depot must return to depot
forall(k in carriers)
      sum(j in nodes : j != depots_o[k]) x[k][depots_o[k]][j]
      == sum(j in nodes : j != depots_o[k]) x[k][j][depots_o[k]];
// 6) number of vehicles leaving depots must be smaller or equal to number of vehicles
forall(k in carriers)
   sum(j in nodes : j != depots_o[k]) x[k][depots_o[k]][j] <= vehicles_W[k];
// (new condition) vehicle must leave depo before serving requests
forall(k in carriers)
      (sum (i,j in nodes)x[k][i][j]>=1)
      => (sum (j in nodes : j != depots_o[k])x[k][depots_o[k]][j] == 1);
// 7) all requests must be served only once
forall(l in requests)
   sum(k in carriers) y[l][k] <= 1;
// 8) all nodes must be visited max once
forall(k in carriers, i in nodes : i != depots_o[k])
sum(j in nodes)  x[k][i][j] <= 1;
// 9) Vehicle departure time from a customer and its immediate successor
// change of 9)
forall(k in carriers)
    forall(i,j in nodes : j != depots_o[k])
     (x[k][i][j] == 1) => (t[k][i] + Time[i][j] <= t[k][j]);
// vehicle needs to visit delivery node after pickup
   forall(k in carriers, i,j in nodes)
    forall(p in (pickup_P[i] .. pickup_P[i]), d in (delivery_D[j] .. delivery_D[j])
    : p != -1 && d != -1 && p == d){
   sum(n in nodes)x[k][n][i] == sum(n in nodes)x[k][n][j];
```

```
   t[k][i] <= t[k][j];
}
// 10) quantity from depot is equal to
forall(k in carriers)
      sum(j in nodes : j != depots_o[k])
        q[k][depots_o[k]][j] ==
          (sum(l in (pickup_P[depots_o[k]] .. pickup_P[depots_o[k]]) : l != -1)
            quantity_d[l]*y[l][k])-
          (sum(l in (delivery_D[depots_o[k]] .. delivery_D[depots_o[k]]) : l != -1)
           quantity_d[l]*y[l][k]);
//11) 12) and 13) set by variable type not needed
// 14 updated) The time windows are observed
    forall(i in nodes, k in carriers)
      LBTW[i] <= t[k][i] <= UBTW[i];
}
```

Abstract

Fiala, Tomáš. *Optimalizace logistických procesů na bázi spolupráce přepravních společností* . Diplomová práce. Plzeň: Fakulta ekonomická ZČU v Plzni, 80s, 2020.

Předložená diplomová práce se zabývá metodami pro optimalizaci logistických úloh. Práce se skládá ze pěti kapitol. První kapitola se věnuje problematice současné logistiky a možnostem využití vertikální nebo horizontální spolupráce mezi přepravci. Dále jsou popsány formy spolupráce mezi přepravci a možnosti koordinace této spolupráce. Druhá kapitola se zabývá metodami používanými pro řešení optimalizačních úloh. Třetí kapitola je zaměřena na analýzu modelů a je dále rozdělena na tři části. První část popisuje rozšířené metody používané při hledání optimálních tras. Druhá část popisuje strategie výběru požadavků pro aukci. Třetí část této kapitoly obsahuje popis dvou metod výměny požadavků založených na aukci. Dále je ve čtvrté kapitole stručně popsán nástroj CPLEX, použitý pro řešení optimalizačních úloh a poté jsou popsané modely implementovány a zhodnoceny. V poslední páté kapitole je implementovaný model upraven pro reálné vstupy. Nakonec jsou získané výsledky zhodnoceny.

Klíčová slova

optimalizace, logistika, aukce, CPLEX, VRP

Abstract

Fiala, Tomáš. *Logistics optimization based on carrier cooperation.* Master thesis. Faculty of Economics, University of West Bohemia, 80s,2020

This thesis describes methods used for optimization of logistic problems. Thesis consists of five chapters. In the first chapter, current logistics issues are discussed and the possibilities of horizontal and vertical cooperation in that matter. In the same chapter are described possible forms of cooperation along with description of coordination methods. The second chapter consists of methods used for solving the optimization problems. The third chapter is focused on model analysis and is further divided onto three parts. Methods used for route optimization are described in the first part. Strategies for request selection are described in the second part. The third and also last part of this chapter contains the description of two methods used for auction-based request exchange. Next, the tool CPLEX, used for calculating the problems, is described. In the same chapter are the previously described models implemented and analysed further. Lastly, the implemented model is used with real data (eg. coordinates) and the solution is discussed.

Keywords

optimalization, logistics, auction, CPEX, VRP