Semestrální práce z KIV/PRO

Heuristický algoritmus pro vytvoření efektivní školní autobusové dopravy

Lukáš Runt (A20B0226P)

7. listopadu 2021

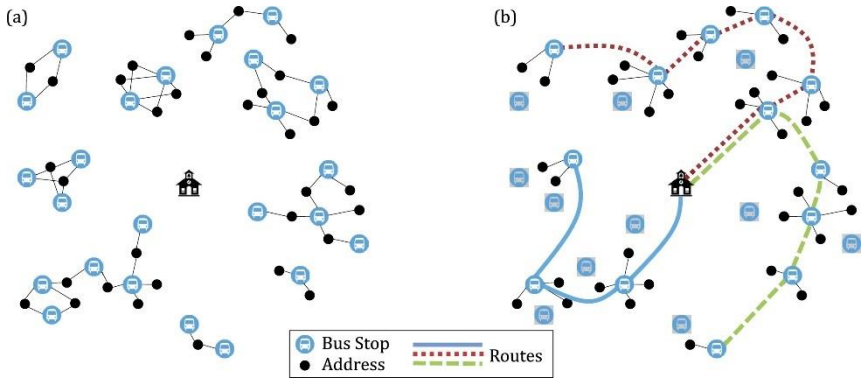
Obsah

Zadání

Najděte v anglicky psané odborné literatuře článek v délce alespoň 5 stránek o nějakém algoritmu řešícím libovolný problém. Algoritmus popište do českého referátu tak, aby ho podle vašeho názoru pochopil běžný student 2. ročníku informatiky. Váš text musí svědčit o tom, že algoritmu rozumíte, a musí ho z něj pochopit i nezasvěcený čtenář.

Úvod a formulace problému

V zemích jako je Wales, Anglie nebo Austrálie musí stát zajistit pro každého žáka, který bydlí dál než 4,8 km (3,2 km) od školy, odvoz autobusem do školy. Každý rok úředníci vytvoří seznam potencionálních autobusových zastávek a adres žáků, kteří mají nárok na dopravu do školy. Z těchto dat se musejí vytvořit trasy tak, aby doba jízdy netrvala déle než 60 minut (45 minut), zastávka byla v přijatelné vzdálenosti od domu žáka, a to při co nejmenším počtu tras, protože každý autobus stojí ročně od 25 000£ do 35 000£. Úkolem je tedy sestavit trasy tak, aby byli obslouženi všichni žáci a zároveň by bylo řešení co nejlepší z finančního hlediska (co nejméně tras, které jsou co nejkratší).



Obr. 1 (a) Příklad problému (b) Příklad řešení s použitím 3 tras, šedé zastávky nejsou použity

Tabulka 1 – Vysvětlení zkratek použitých v dokumentu

Zkratka	Popis
m_w	Maximální možná vzdálenost bydliště studenta od zastávky
m_t	Maximální čas školní trasy (45 nebo 60 minut)
m_e	Minimální vzdálenost adresy studenta od školy, aby mu byla přidělena doprava
Q	Kapacita autobusu (počet sedadel)
V_1	Množina vrcholů, které představují autobusové zastávky a školu (v_0)
$V_1 - \{v_0\}$	Množina všech autobusových zastávek, které mohou být potencionálně použity
V_2	Množina adres studentů, kterým má být přiřazena doprava
t(u, v)	Označení času mezi vrcholy u a v
d(u, v)	Označení vzdálenosti mezi vrcholy u a v
E_1	Množina hran obsahující hrany mezi každým vrcholem množiny V_1 v obou směrech
E_2	Množina hran obsahující hrany mezi adresou studenta a zastávkou < m_w
R	Množina tras autobusů
V_1'	Množina zastávek použitých v řešení R
$V_2(v)$	Podmnožina adres přiřazených k zastávce $v \in V_1'$
s(v)	Označuje počet studentů přiřazených k jedné adrese nebo zastávce
s(R)	Celkový počet studentů přiřazených k 1 trase
s(v, R)	Počet studentů potřebných k nastoupení na trase R na zastávce $v \in V_1'$
t(R)	Celkový čas cesty R se zpožděním
A	Množina řešení použitých více objektivním algoritmem optimalizace
δ	Diskrétní parametr používaný s více objektivním algoritmem optimalizace
$\lfloor . \rfloor_\delta$	Označuje hodnotu zaokrouhlenou dolů k násobku δ
f_1, f_2	Funkce k optimalizování (průměrný čas strávený chůzí a průměrný čas jízdy autobusu)

Notace a definice

Problém tras školních autobusů (SBRP) lze stanovit pomocí dvou grafů. První graf $G_1(V_1, E_1)$ se skládá z množiny vrcholů V_1, která je množinou školy v_0 a všech potenciálně použitelných autobusových zastávek $v_1, \ldots, v_n$. Množina E_1 je pak množinou orientovaných ohodnocených hran mezi všemi vrcholy V_1, funkcemi t(u, v) a d(u, v) je přiřazena hranám váha. Druhý graf $G_2(V_1-\{v_0\}, V_2, E_2)$ je definován množinou vrcholů V_2, která obsahuje všechny adresy studentů s váhou $s(v) \in Z^+$, která definuje počet studentů na jedné adrese. Hrany z množiny E_2 se pak vytvářejí mezi vrcholy z množiny V_1 a V_2, které jsou od sebe v maximální vzdálenosti m_w: $E_2 = \{\{u, v\}: u \in V_2 \wedge v \in (V_1-\{v_0\}) \wedge d(u, v) \leq m_w\}$.

Definice 1. Realizovatelné řešení SRBP je množinou tras $R = \{R_1, \ldots, R_k\}$, ve které je každá trasa $R \in R$ jednoduchá cesta obsluhována jedním autobusem s kapacitou Q, kde každý autobus musí přijet do školy při navštívení posledního vrcholu jeho trasy. Zároveň je potřeba splnit následující omezení:

$$\bigcup_{i=1}^{k} R_i = V_1' \tag{1}$$

$$\forall u \in V_2 \quad \exists v \in V_1' : \{u, v\} \in E_2 \tag{2}$$

$$s(R) \leq Q \quad \forall R \in \mathcal{R} \tag{3}$$

$$t(R) \leq m_t \quad \forall R \in \mathcal{R} \tag{4}$$

Kde $V_1' \subseteq (V_1-\{v_0\})$ je podmnožina zastávek použitých v řešení V_1', které splňují výše zmíněné omezení.

Vzhledem k finanční náročnosti, je cílem algoritmu minimalizovat počet tras. Když je nalezeno řešení s minimálním počtem tras pokračuje se s hledáním nejkratších tras a vzdáleností adres studentů od zastávek. Definice 1 také povoluje stavění více autobusů na jedné zastávce. Tyto zastávky pak nazýváme vícenásobné.

Při definování problému v reálném světě musíme brát ještě v úvahu:

- Studenti jsou přiřazeni nejbližší zastávce z množiny V_1'
- Studentovi je přiřazena jedna konkrétní trasa, aby se zabránilo přeplnění autobusů v případě zastávky, ke které je přiřazeno více tras.
- Je hledáno řešení problému pro rozvoz do školy. V případě rozvozu studentů ze školy budou zastávky navštíveny v opačném pořadí.
- Do celkového času trasy musí být připočteno zpoždění. Zpoždění je čas, který je zabere zpomalování, otevírání dveří, nastupování studentů, zařazování se zpátky do provozu. Zpoždění je sice ovlivněno více faktory, běžně se ale počítá s lineárním modelem y = a + bx, kde y je zpoždění, x je počet nastupujících pasažérů, b označuje čas potřebný k nastoupení jednoho pasažéra, a zaznamenává všechny zbývající zpoždění.

Definice 2. Nechť $s(u_i, R)$ označuje počet studentů nastupující do autobusu na trase R na zastávce u_i. Čas projetí trasy $t(R)$ trasy $R = (u_1, u_2, \ldots, u_i) \in R$ je vypočítán jako:

$$t(R) = \left(\sum_{i=1}^{l-1} t(u_i, u_{i+1})\right) + t(u_l, v_0) + \left(\sum_{i=1}^{l} a + b \cdot s(u_i, R)\right). \tag{5}$$

V našem případě budeme počítat s a = 15, b = 5 (vteřin). Tyto hodnoty vycházejí z výzkumů [3, 4, 5].

Věta 1. Nalezení realizovatelného řešení patří do třídy složitosti NP

Důkaz Nechť $G(V_1, V_2, E_2)$ je graf, kde každý vrchol z množiny V_2 má stupeň 1. To znamená, že pro každou autobusovou zastávku $v \in (V_1-\{v_0\})$, je v povinný a musí být alespoň v jedné trase. Počet

nastupujících studentů je pevně stanovený, což znamená, že i zpoždění na každé zastávce je pevně stanoveno.

Primárním cílem SBRP je minimalizovat počet tras autobusů používaných v řešení. Je proto žádoucí plnit autobusy efektivně. Jelikož jsou povoleny vícenásobné zastávky, může se stát, že 2 studenti, kteří bydlí na jedné adrese budou sice nastupovat na stejné zastávce, ale pojedou do školy jinou trasou (autobusem). Je tedy potřeby rozhodnout, kteří studenti budou přiřazeni, do které trasy. Dalším problémem je, kterou autobusovou zastávku přiřadit do řešení a kterou ne. To souvisí s problémem pokrytí. Pokrytí množin zahrnuje množinu označovanou jako vesmír U = {1, 2, ..., n}, jejíž prvky jsou podmnožinami vesmíru. Poté musíme identifikovat nejmenší podmnožinu S'⊆S, jejíž sjednocení se rovná vesmíru. Neboli S' potřebuje pokrýt U. Např.: Pokud máme množiny U = {1, 2, 3, 4} a S = {{1}, {1, 2}, {1, 3}, {3, 4}, {4}}, bude optimální pokrytí S' = {{1, 2}, {3, 4}}.

Definice 3. Máme dáno U a S, S'⊆S je kompletně pokrývající pokud $\bigcup_{s\in S'}$=U. Minimální pokrytí je kompletní, jestliže při odebrání kteréhokoliv prvku z S' způsobí nekompletní pokrytí. Optimální pokrytí je takové, které má minimální kardinalitu mezi všemi pokrytími.

Vytvoření optimálního porytí patří do třídy složitosti NP, avšak lze nalézt přibližné řešení použitím Chvantalova greedy algoritmu [6], který vybírá prvky z S obsahující co největší počet nepokrytých elementů, které přidává do S' dokud není dosaženo pokrytí. Vytvoření minimálního pokrytí je pak zajištěno tak, že se pokoušíme postupně odebírat prvky z množiny S' tak, aby bylo zachováno úplné pokrytí.

Věta 2. Zvážíme-li SBRP, kde nejsou povoleny zastávky, kterými prochází více tras ($R_i \cap R_j = \emptyset\ \forall R_i, R_j \in R$)**,** nechť G($V_1$, E_1) je graf, jehož hrany splňují trojúhelníkovou nerovnost. Nechť *R* = {R_1, ..., R_k} je řešení, které splňuje omezení (1) až (3), který má minimální celkový čas tras $\Sigma_{i=0}^{k} t(R_i)$. Potom množina autobusových zastávek V_1' použitých v *R* odpovídá minimálnímu pokrytí S'.

Důkaz Odstranění jakékoliv zastávky $v \in V_1'$ odpovídá odstranění prvku *v* množiny S', což podle definice vede k neúplnému pokrytí a porušení omezení (2). Naopak přidání dalšího prvku *v* do S' bude mít za následek úplné, ale ne minimální pokrytí, a to odpovídá přidání jedné zastávky do *R*, a to prodlouží dobu jízdy jedné trasy z *R*. Toto neplatí, pokud řešení obsahuje vícenásobné zastávky, neboť se může stát, že přidání zastávky povede ke zkrácení jedné nebo více tras.

Jednotlivé Algoritmy

Úkolem programu je najít heuristické realizovatelné řešení, které používá co nejmenší počet tras. Prvotní strategií algoritmu bude najít pevně daný počet tras *k*, kde budeme zanedbávat omezení (4). Další algoritmus se pak bude snažit zkrátit trasu tak, aby bylo omezení (4) dodrženo. Když se to nepodaří, zvýší se počet povolených tras *k* o jednu a algoritmus se poté opakuje.

Počáteční řešení

Počáteční řešení *R* = {R_1, ..., R_k} je vytvořeno vygenerováním podmnožiny autobusových zastávek V_1' odpovídající kompletnímu pokrytí. V našem případě se tohoto dosáhne pomocí Chvantalovo greedy algoritmu [6]. Po vygenerování této množiny mohou být některé zastávky odstraněny, aby bylo dosaženo minimálního úplného pokrytí.

Když máme vygenerovanou množinu V_1', je každý student přiřazen k nejbližší zastávce. Poté musíme zastávky přiřadit do tras tak, že žádná trasa není přeplněna. Z čehož získáváme bin packing problém [7] s množinou položek s váhami a *k* autobusy s kapacitou Q. Postupně je vybrána zastávka *v* s největším počtem pasažérů, která je přiřazena na konec vhodné trasy. Trasa je vybrána, pokud *R* obsahuje *v* a zároveň má dostatečně velkou volnou kapacitu pro *v*. Pokud tato kritéria splňuje více tras, je vybrána trasa s méně studenty. Pokud se nezdá, že by nějaká trasa vyhovovala těmto kritérium. V tom případě

je vybrána trasa s největší volnou kapacitou. To má za následek vzniknutí vícenásobné zastávky, protože část studentů se bude muset přiřadit k jiné trase.

Lokální vyhledávání

Lokální vyhledávání udělá pár změn, aby se snížily náklady a nebyla by porušena omezení (1) až (3). Budeme tedy vyměňovat sousedy uvnitř i mezi dvěma trasami R_1, $R_2 \in R$, předpokládejme že $R_1 = (u_1, u_2, ..., u_n)$ a $R_2 = (v_1, v_2, ..., v_n)$.

Výměna sekce: Vezmeme dva vrcholy z obou tras, které nejsou počáteční ani koncové. Následně vezmeme sekce (sekce začíná prvním vybraným vrcholem a končí druhým, do sekce patří vše, co je mezi těmito vrcholy) a vyměníme je mezi trasami. Sekce můžeme zkusit i invertovat, pokud to bude vést k lepším výsledkům.
Vložení sekce: Vezměme sekci z R_1 společně s vrcholem v_j z R_2. Následně sekci z první trasy odstraníme a vložíme jí před vrchol v_j.
Vytvoření vícenásobné zastávky: Vezmeme vrchol u_i z trasy R_1, který nepatří do trasy R_2 a přiřadíme pár studentů z vrcholu u do trasy R_2.

Při prvních dvou operací se může stát, že se poruší některá omezení. Jelikož jsou povoleny vícenásobné zastávky, může se stát, že trasa bude obsluhovat vrchol vícekrát. Tyto nedostatky lze ošetřit odstraněním příslušného vrcholu a přiřazením studentů do jiné trasy.

Výměna: Vezmeme dva nekoncové vrcholy z jedné trasy a prohodíme je.
2-opt [8]**:** Vezmeme dva nekoncové vrcholy a v celé sekvenci prohodíme pořadí.
Prodloužený Or-opt: Vezmeme sekvenci jako v metodě 2-opt, tentokrát ji ale odstraníme a vložíme na jiné místo v trase.

Lokální vyhledávání se pak bude řídit sestupnou metodou (od největších po nejmenší změny). V každém cyklu se proházejí sousedé, přičemž se uloží krok, který nejvíce sníží náklady. Procedura se vykonává, tak dlouho dokud nedojde ke žádnému zlepšení. Počet tahů v každé cyklu je $O(m^4)$, kde m je velikost stávajícího řešení. Záměrem je rychle identifikovat lokální optimum, které je pak předáno další části programu.

Generování nového pokrytí pomocí vyřazovacího operátora

I když lokální vyhledávání dokáže nalézt a zlepšit náklady řešení, vůbec nemění podmnožinu V_1' používaných autobusových zastávek. Jedním ze způsobů, jak změnit podmnožinu je vyměnit nějakou zastávku za neaktivní, nebo nějako zastávku úplně odstranit, avšak odstranění zastávky se moc nedoporučuje, protože může být porušeno omezení (2).

Vyřazovací operátor funguje tak, že se vytvoří nová množina $V_1'' \neq V_1'$, která splňuje omezení. K vytvoření V_1''se náhodně vybere nepovinná zastávka z množiny V_1', která je odstraněna. Následuje odstranění dalších x nepovinných zastávek (počet odebraných zastávek je zvolen náhodně). Tímto se dosáhne částečného pokrytí, takže budou muset být přidány další zastávky pomocí Chvantalova algoritmu, který vybere a přidá náhodně zastávku, která obslouží studenty, kteří nejsou pokryti. Přidávání se opakuje, dokud nejsou pokryti všichni studenti.

Po vytvoření je každá adresa studenta přiřazena k nejbližší zastávce v V_1''. Zastávky, které nemají přiřazené žádné studenty budou odstraněny. Nakonec se spustí „Aktualizace řešení“ (Pseudokód 1).

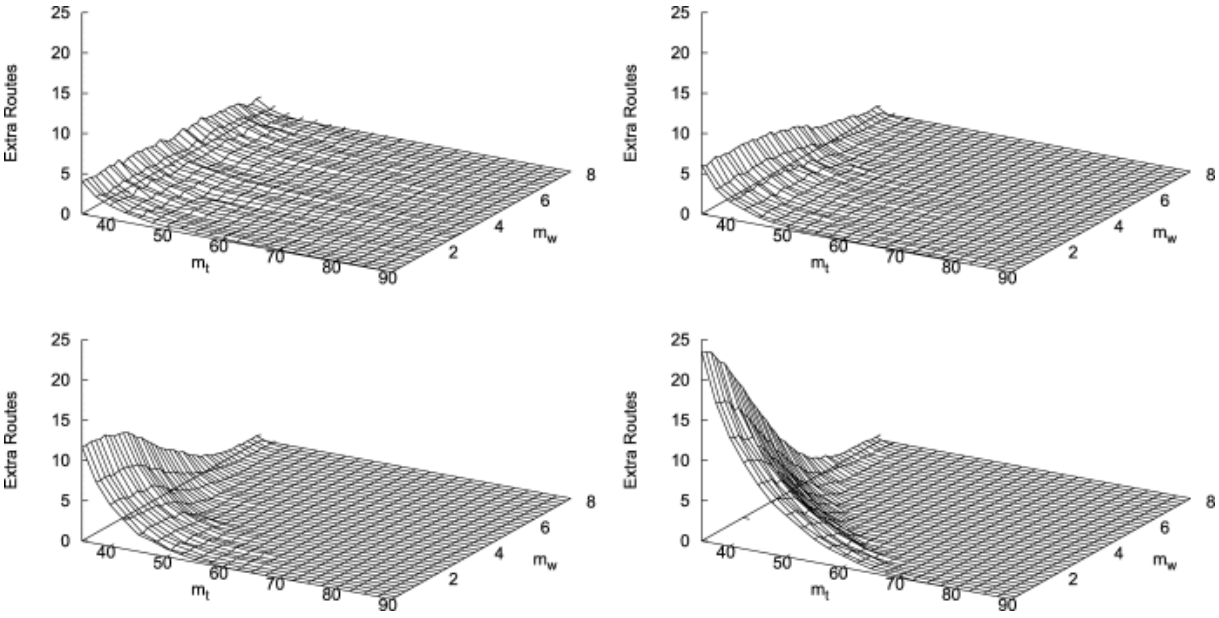
Pseudokód 1: **Aktualizace řešení (R, V_1', V_1'')**

(1) **pro každou** autobusovou zastávku v ∈ (V_1–{v_0})
(2) **jestliže** v ∈ V_1''
(3) nechť s(v)‘ je počet studentů nastupující na v na trase V_1'
(4) nechť s(v)‘‘ je počet studentů nastupující na v na trase V_1''
(5) **jestliže** s(v)‘ < s(v)‘‘ **potom**
(6) přidej v, společně s s(v)‘ - s(v)‘‘ studenty do seznamu zastávek, které mají být přidány do množiny R
(7) **jinak jestliže** s(v)‘ > s(v)‘‘ **potom**
(8) odstraň s(v)‘ - s(v)‘‘ studentů z počtu na zastávce v z množiny R, jestliže toto vede k tomu, že k zastávce nebude nikdo přiřazen, v se odstraní z R
(9) **jinak jestliže** v ∈ V_1' **potom**
(10) odstraň všechny studenty přiřazené k v z množiny R

Primární funkce aktualizace řešení je to, že odstraní všechny zastávky z *R*, které nejsou zahrnuty v V_1'' a k vytvoření seznamu zastávek a studentů, které je potřeba přidat do řešení. Po skončení algoritmu (Pseudokód 1) se tyto zastávky přidají do *R*, stejně jako v sekci počáteční řešení.

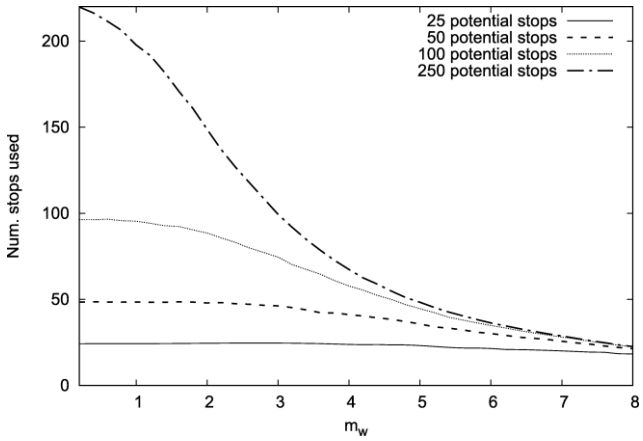
Experimentování

Za účelem experimentování byly generovány varianty s 25, 50, 100 a 250 autobusovými zastávkami s m_w od 0,2 do 8 km s krokem po 0,2 km. Pro každou variantu by vytvořeno 20 případů (celkem 3200 různých případů). V každém případě byly adresy studentů generovány v rádiusu 25 km, kde bylo vygenerováno 585 pro přibližně 1000 studentů. Kapacita autobusů byla nastavena na 70 sedadel a hodnota m_e na 5 km od školy. Při pokusu se bude počítat s 30 vteřinami zpoždění na každé zastávce. Počet *v* se pak v každém případě porovnává s minimálním možným počtem tras (Nejmenší počet autobusů tak, aby byl počet sedadel větší nebo roven počtu studentů).



Obr. 2 Počet požadovaných nadbytečných tras v závislosti na různých hodnotách m_t a m_w pro případy, kde |V_1–{v_0}| = 25, 50, 100 a 250 autobusových zastávek. Každý bod je průměrem 20 náhodně vygenerovaných problémů.

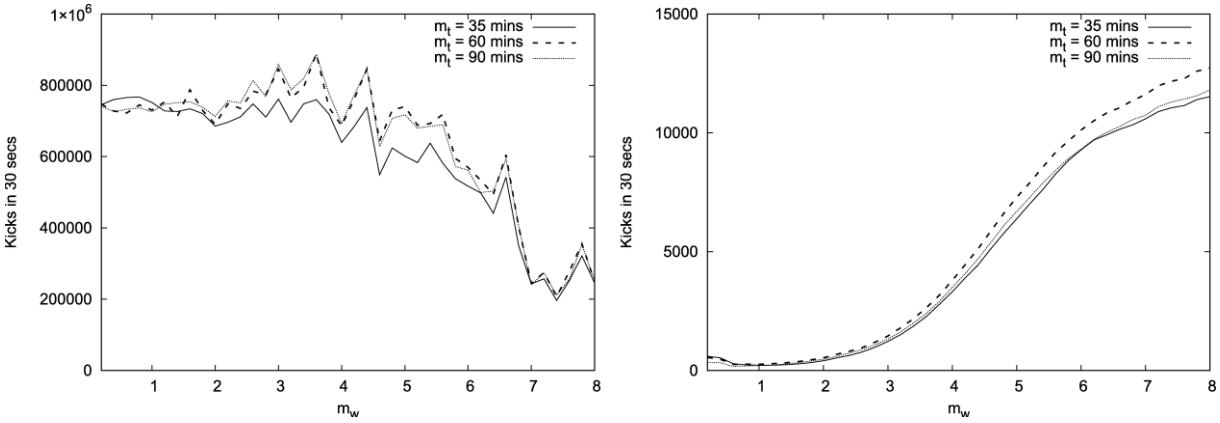
Na obrázku 2 lze vidět závislosti počtu tras při různých omezeních. Můžeme například pozorovat, že při přísných limitech bude potřeba více tras.



Obr. 3 Počet autobusových zastávek použitých v řešeních pro případy, kde $|V_1-\{v_0\}| = 25, 50, 100$ a 250 potencionálních autobusových zastávek v závislosti na různých hodnotách m_w. Každý bod je průměrem 20 náhodně vygenerovaných případů.

Z grafu na obrázku 3 můžeme vyvodit, že pokud bude maximální povolená vzdálenost adresy studenta od zastávky malá, bude to znamenat, že se bude muset použít více zastávek, které nebude možno vynechat, a to bude vést k většímu počtu tras.

Čím bude m_w větší, tím budeme moci vyřadit více zastávek, což zapříčiní více studentů na jednotlivých zastávkách a nutnost rozdělit studenty do více tras (tj. přes zastávku povede více tras).



Obr. 4 Počet vyřazení v závislosti na m_w za 30 vteřin pro případy s 25 (vlevo) a 250 (vpravo) potenciálními zastávkami, pro 3 různé hodnoty m_t. Hodnoty jsou zprůměrovány z 20 náhodných dat.

Grafy na obrázku 4 označují počet vykonání lokálního vyhledávání a vyřazovacího operátora během 30 vteřin po které byl program spuštěn. Pro případy s 25 autobusovými zastávkami je největší počet provedení, když jsou adresy blízko zastávkám. V těchto případech jsou skoro všechny zastávky povinné, operátor tak může dělat jen malé a rychlé změny. Výsledek je pak blízko optimálnímu řešení, což znamená, že lokální vyhledávání bude dokončeno poměrně rychle. Naopak v případě 250 zastávek je počet řešení mnohem větší, což zapříčiní vysokou časovou náročnost lokálního vyhledávání a tím pádem i menší počet použití vyřazovacího operátora.

Výsledky tedy ukazují, že optimální výsledky lze najít pomocí minimálního počtu tras. To platí hlavně, když lze využít jen malý počet tras. Dále bylo zjištěno, že v případech, kdy jsou skoro všechny zastávky povinné, nemá vyřazovací operátor moc velký vliv a bylo by lepší se zaměřit na lokální vyhledávání.

Vylepšení finančního hlediska a kvality služby

Jakmile bude dosaženo smysluplného řešení, využívajícího co nejmenší počet tras, je v našem zájmu využít dalších úprav, aby byla služba výhodná z finančního i komfortního hlediska. Z finančního hlediska máme zájem, aby trasy byly co nejkratší. Studenti zase chtějí, aby měli zastávku co nejblíže k domovu. Tyto cíle jsou ovšem v rozporu. Budeme tedy muset dojít ke kompromisu obou stran. Naším cílem je tedy optimalizovat dvě funkce, a to průměrný čas strávený chůzí každým studentem

$$f_1(\mathcal{R}) = \frac{\sum_{u \in V_2} s(u) \cdot t(u, v^*)}{\sum_{u \in V_2} s(u)} \tag{6}$$

(kde $v^* \in V_1'$ je nejbližší zastávka od adresy studenta) a průměrný čas jízdy autobusu.

$$f_2(\mathcal{R}) = \frac{\sum_{i=1}^{k} t(R)}{k} \tag{7}$$

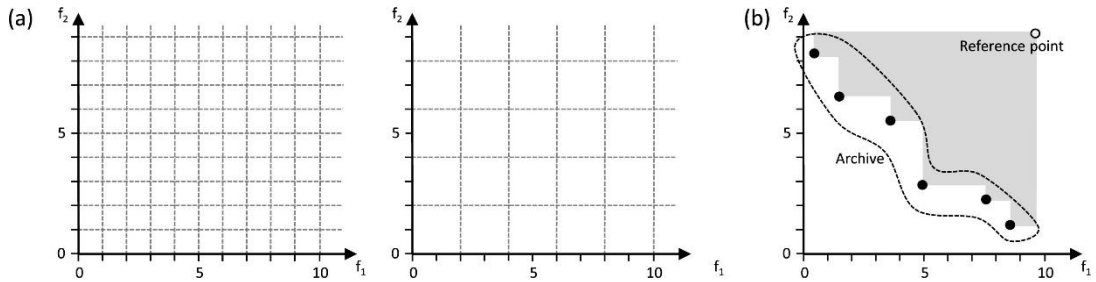
Strategií je provést postupně úpravy podmnožiny V_1', čímž se změní rozdělení studentů, k jednotlivým zastávkám a dojde ke změně funkce f_1. Řešení je poté optimalizováno pomocí lokálního vyhledávání, s cílem snížit funkci f_2.

Pseudokód 2: **Více objektivní lokální vyhledávání (A)**

(1) **když** existuje R ∈ A, který nebyl navštíven
(2) Označ R jako navštívený
(3) **pro každé** v ∈ (V_1−{v_0})
(4) R‘ <- R
(5) **jestliže** $v \notin V_1'$
(6) vlož v do R‘ a spusť lokální vyhledávání
(7) **jinak jestliže** $v \in V_1'$ **a zároveň** v není povinný
(8) odstraň v z R‘ a spusť lokální vyhledávání
(9) Aktualizuj archiv (R‘, A)

Pseudokód 3: **Aktualizuj archiv (R‘, A)**

(1) **pro každé** R ∈ A
(2) **jestliže** R‘ dominuje R
(3) odstraň R z A
(4) **jinak jestliže** $\lfloor f_1(R') \rfloor_\delta \geq \lfloor f_1(R) \rfloor_\delta$ **a zároveň** $\lfloor f_2(R') \rfloor_\delta \geq \lfloor f_2(R) \rfloor_\delta$
(5) **konec**
(6) označ R za neoznačený a přidej ho do A



Obr. 5 (a) Znázornění čtvercové sítě pro hodnoty δ = 1 a δ = 2. Archivy umožňují jedno řešení na čtverec. Při δ = 0 je velikost archivu neomezena. (b) Demonstrace S-metriky pomocí archivní sady vzájemně nedominujících řešení a vhodného referenčního bodu.

Algoritmus (Pseudokód 2) začíná archivem řešení A, kde je každé řešení označeno jako nenavštívené. V každém cyklu se pak vybere nenavštívené řešení *R*, ze kterého jsou pak generována

nová řešení přidáváním nebo ubíráním zastávek a následnou optimalizací. Tato řešení jsou přidávána do archivu, pokud nedominuje řešení z archivu A. Pokud nové řešení dominuje nějakému řešení z archivu, pak je toto řešení z archivu odstraněno. Výsledkem procedury je pak množina vzájemně nedominujících řešení.

V řádku (6) pseudokódu 2 je zastávka přidána do R' tak, že jsou jí přiřazeni studenti, pro které je tato zastávka nejbližší. Pokud se nějaká zastávka v tomto důsledku zůstane bez přiřazených studentů, je odstraněna z R'. Nakonec se v přiřadí do jedné nebo více tras pomocí počátečního řešení. Podobným způsobem se postupuje při odstranění zastávky na řádce (8), kde se adresy z odstraněné zastávky přiřadí k nejbližším pozůstalým zastávkám a zároveň se kontroluje, jestli nedošlo k porušení omezení.

Tyto algoritmy jsou výpočetně složité, neboť každé řešení je potřeba navštívit a řešení můžou stále přibývat. V některých případech může trvat běh programu dokonce celé dny. Aby se docílilo snížení času vykonání programu, zaokrouhlují se na řádku (4) pseudokódu 3 dolů na nejbližší násobek δ, kde $\delta \geq 0$ je parametr specifikovaný uživatelem. Toto znamená, že nové řešení je přidáno pouze v případě, pokud je dostatečně odlišné od řešení, které již v množině A je. Tento archiv si lze představit jako síť čtverců, kde je pro každý čtverec povoleno jen jedno řešení, jako na obrázku 5. Protože je povoleno jen jedno řešení na čtverec, sníží se počet řešení v archivu a tím i potřebný čas na běh programu, což může potenciálně vést k nepřesným výsledkům (jako pro data města Adelaide pro δ = 10 na obrázku 6.

Tabulka 2 - 10 případů z reálného světa, seřazeno podle potencionálních autobusových zastávek

Location	Country/State	$\|V_1\| - 1$	$\|V_2\|$	Students	m_e	m_w	Stops per address	Addresses per stop
Brisbane	Queensland	1817	438	757	3.2	1.6	79.2	19.1
Adelaide	South Australia	1118	342	565	1.6	1.6	100.0	28.8
Edinburgh-1	Scotland	959	409	680	1.6	1.6	84.6	36.1
Edinburgh-2	Scotland	917	190	320	1.6	1.6	256.1	53.1
Bridgend	Wales	633	221	381	4.8	1.6	45.3	15.9
Milton Keynes	England	579	149	274	4.8	1.6	64.5	16.6
Cardiff	Wales	552	90	156	4.8	1.6	63.9	10.4
Canberra	ACT	331	296	499	4.8	1.0	13.3	11.9
Suffolk	England	174	123	209	4.8	1.6	10.4	7.4
Porthcawl	Wales	153	42	66	3.2	1.6	53.5	14.8

Tabulka 3 - Výsledky využití algoritmu na případech z reálného světa

Location	k	$\delta = 1$			$\delta = 10$			$\delta = 30$		
		Time (s)	$\|\mathcal{A}\|$	$S \pm CV$	Time (s)	$\|\mathcal{A}\|$	$S \pm CV$	Time (s)	$\|\mathcal{A}\|$	$S \pm CV$
Brisbane	11	31444.0	399.3	256.2 ± 1.0%	2329.1	20.2	217.4 ± 3.4%	539.3	2.9	150.6 ± 7.3%
Adelaide	9	9326.8	350.5	341.3 ± 0.5%	1191.5	22.5	315.8 ± 1.3%	125.1	3.5	250.7 ± 4.2%
Edinburgh-1	10	16155.9	314.9	337.5 ± 0.5%	906.5	19.9	308.8 ± 1.4%	159.1	3.2	257.0 ± 3.2%
Edinburgh-2	5	582.5	313.0	431.6 ± 0.4%	135.7	38.1	421.3 ± 0.3%	17.7	11.1	394.5 ± 1.7%
Bridgend	6	2743.7	296.2	140.5 ± 2.5%	518.3	24.1	128.2 ± 3.2%	332.9	4.8	103.0 ± 7.3%
Milton Keynes	5	491.6	318.9	303.9 ± 0.6%	108.8	34.9	293.0 ± 1.0%	36.2	10.2	257.0 ± 2.3%
Cardiff	3	7.1	172.4	308.1 ± 0.4%	2.4	38.1	305.8 ± 0.6%	1.2	13.4	299.8 ± 0.9%
Canberra	8	356.0	150.8	233.4 ± 0.6%	32.8	8.8	224.3 ± 0.9%	11.6	3.3	215.5 ± 1.2%
Suffolk	4	28.6	127.5	185.4 ± 2.7%	8.8	17.6	180.0 ± 2.7%	5.9	5.6	171.4 ± 3.0%
Porthcawl	1	4.4	127.5	152.6 ± 1.7%	0.9	43.4	149.1 ± 2.9%	0.5	18.1	141.7 ± 3.3%

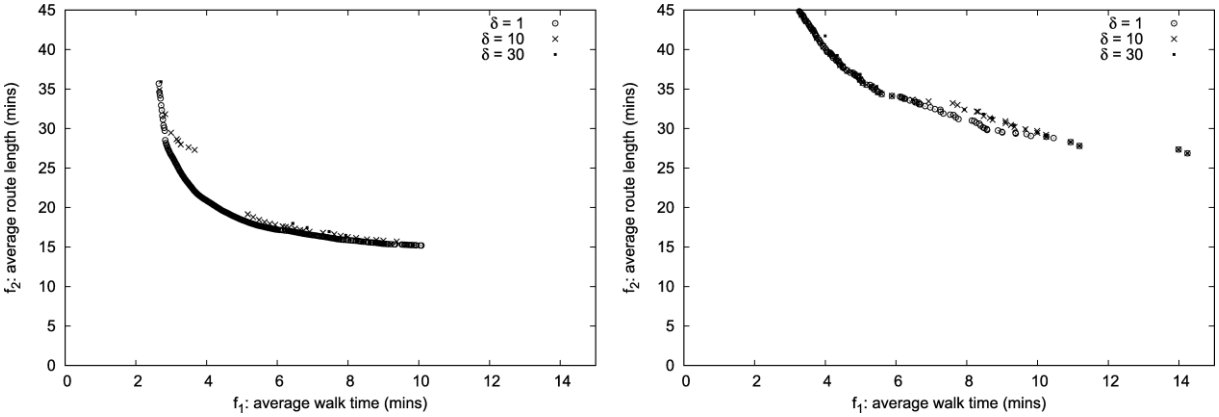
Testování výsledků

K testování algoritmu posloužilo 10 reálných případů (tabulka 2). Počítalo se s kapacitou autobusů Q = 70 a maximální jízdou autobusem m_t = 45 minut. Pro 9 z 10 případů bylo počáteční řešení, tedy řešení s minimálním počtem autobusů, nalezeno do 3 sekund. V Suffolku bylo totiž potřeba o 1 autobus navíc.

Pro pokusy se na začátku vytvořila 2 řešení, první řešení se vytvořilo pomocí algoritmu počátečního řešení a pro druhé se vzala množina zastávek nejbližších k adresám studentů, která byla upravena pomocí lokálního vyhledávání a vyhazovacího operátoru. Tato dvě řešení budou reprezentovat dva extrémy aproximace Paretovy množiny [9]. První řešení zahrnuje krátké trasy s málo zastávkami, ale

s dlouhými časy chůze studentů k zastávce. Druhé pak nabízí zastávky blízko k adrese studentů a hodně dlouhých tras autobusů s mnoha zastávkami, které dodržuje všechna, omezení kromě času jízdy. Taková řešení budou uchovávána v archivu během běhu programu, avšak po dokončení jsou odstraněna a nejsou zohledněna ve statistikách.

Výsledky pokusů jsou zaznamenány v tabulce 3, kde jsou použity 3 hodnoty δ v sekundách. Kontrolujeme 3 různé údaje výstupu programu, a to dobu běhu programu, počet uložených řešení problému a S-metriku archivu (obr 9b). Výsledky jasně ukazují, že časy běhu programu klesají se zvyšujícím se δ, bohužel ale klesá přesnost Paretovy množiny. Můžeme také vidět různé časy běhu s různými daty. Porovnání výstupních sad archivů a jejich zkreslení můžeme pozorovat na obrázku 6. Velké zkreslení lze zaznamenat u města Adelaine, kde můžeme pozorovat velkou mezeru mezi řešeními při δ = 10. Tyto mezery jsou způsobeny v případě, když je jedna zastávka poměrně daleko od ostatních zastávek a zároveň je blízko velkému počtu adres. Přidání, nebo odebrání této zastávky pak způsobí velké změny ve vzdálenostech tras a času docházení studentů na zastávku. Všechny archivy řešení s vizualizacemi lze naleznout online na [2].



Obr 6. Příklady množin archivů pro města Adelaide (vlevo) a Porthcawl (vpravo)

Závěr

Problém tras školních autobusů je velmi podobný problému obchodního cestujícího. Problém tedy patří do třídy NP, takže je velmi těžké určit optimální řešení, které je lepší než ostatní. Myslím si, že jelikož jsou trasy generovány jednou ročně, nemusí být kladen důraz na rychlost algoritmu, ale na optimalizování výsledného řešení. Nelze jednoznačně určit, které řešení je jednoznačně lepší, neboť se vytvoří množina řešení, kde je každé řešení lepší v něčem jiném a je tedy na výše postavených lidech, kteří rozumí problému, čemu dají přednost. Řešení algoritmu se mi, ale zdá trochu neefektivní, hlavně vytváření počátečního řešení. Myslím, že by se dal vymyslet lepší algoritmus na počáteční vybírání zastávek. Například na začátku se tvoří trasy z celé množiny autobusových zastávek, a proto mě napadá, jestli by nebylo efektivnější rozdělit zastávky do jednotlivých čtvrtí (nebudu hledat zastávky na druhém konci města), které se můžou prolínat, podle minimálního potřebného počtu autobusů a vytvářet prvotní trasy v těchto čtvrtích a dále je optimalizovat podle algoritmů lokálního vyhledávání a vyřazovacího operátoru. Dále připadá v úvahu, napsat nový algoritmus, který by další rok, provedl jen drobné změny v předcházejícím řešení, za předpokladu, že se adresy studentů rok od roku nebudou velmi měnit, což by nejspíš bylo pohodlnější pro studenty, kterým by se neměnily místa a časy odjezdu. Poslední věc, která by se dala vylepšit je maximální čas jízdy m_t, kde se za cenu o trochu delší jízdy dá ušetřit například na celé jedné trase autobusu (například maximální povolit m_t ze 45 na 50 minut).

Bibliografie

1. R. Lewis, K. Smith-Miles / Journal of Discrete Algorithms 52–53 (2018) 2–17
   https://www.sciencedirect.com/science/article/pii/S1570866718301503#fg0010
2. Rhyd Lewis, Zdrojový kód a příklady řešení [Online]
   http://www.rhydlewis.eu/sbrp/
3. R. Bertini, A. El-Geneidy, Modeling transit trip time using archived bus dispatch system data, J. Transp. Eng. 130 (1) (2004)
   https://ascelibrary.org/doi/abs/10.1061/(ASCE)0733-947X(2004)130%3A1(56)
4. Transit Cooperative Research Program, Transit Capacity and Quality of Service Manual, Technical report, Transit Research Board, US, ISBN 978-0-309-28344-1, 2017.
   https://www.worldtransitresearch.info/research/4941/
5. C. Wang, Y. Zhirui, W. Yuan, X. Yueru, W. Wei, Modeling bus dwell time and time lost serving stop in China, J. Public Transp. 19 (3) (2016) 55–77
   https://digitalcommons.usf.edu/jpt/vol19/iss3/4/
6. V. Chvatal, A greedy heuristic for the set-covering problem, Math. Oper. Res. 4(3) (1979) 233–235.
   https://pubsonline.informs.org/doi/abs/10.1287/moor.4.3.233
7. Bin packing problém, Wikipedia, The Free Encyklopedia [Online]
   https://en.wikipedia.org/wiki/Bin_packing_problem
8. 2-Opt, Wikipedia, The Free Enciklopedia [Online]
   https://en.wikipedia.org/wiki/2-opt
9. Pareto front, Wikipedia, The Free Enciklopedia [Online]
   https://en.wikipedia.org/wiki/Pareto_front