SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA CHEMICKEJ A POTRAVINÁRSKEJ
TECHNOLÓGIE

Evidenčné číslo: FCHPT-10881-17549

OPTIMÁLNE PLÁNOVANIE TRASY PRE HETEROGÉNNE MULTI-VOZIDLOVÉ SYSTÉMY

DIZERTAČNÁ PRÁCA

Bratislava, 2016 **Ing. Slavomír Blažek**

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA CHEMICKEJ A POTRAVINÁRSKEJ TECHNOLÓGIE

Evidenčné číslo: FCHPT-10881-17549

OPTIMÁLNE PLÁNOVANIE TRASY PRE HETEROGÉNNE MULTI-VOZIDLOVÉ SYSTÉMY

DIZERTAČNÁ PRÁCA

Študijný program: Riadenie procesov
Číslo študijného odboru: 2621
Názov študijneho odboru: 5.2.14 Automatizácia
Pracovisko: Ústav automatizácie, informatizácie a matematiky
Vedúci dizertačnej práce: prof. Ing. Miroslav Fikar, DrSc.
Konzultant dizertačnej práce: doc. Ing. Michal Kvasnica, PhD.

Bratislava, 2016 **Ing. Slavomír Blažek**

Slovenská technická univerzita v Bratislave
Ústav informatizácie, automatizácie a matematiky

Fakulta chemickej a potravinárskej technológie
Akademický rok: 2016/2017
Evidenčné číslo: FCHPT-10881-17549

STU
FCHPT

ZADANIE DIZERTAČNEJ PRÁCE

Študent:	**Ing. Slavomír Blažek**
ID študenta:	17549
Študijný program:	riadenie procesov
Študijný odbor:	5.2.14. automatizácia
Vedúci práce:	prof. Ing. Miroslav Fikar, DrSc.
Konzultant:	doc. Ing. Michal Kvasnica, PhD.

Názov práce: **Optimálne plánovanie trasy pre heterogénne multi-vozidlové systémy**

Jazyk, v ktorom sa práca vypracuje: slovenský jazyk

Riešenie zadania práce od:	22. 08. 2011
Dátum odovzdania práce:	28. 11. 2016

L. S.

Ing. Slavomír Blažek
riešiteľ

prof. Ing. Miroslav Fikar, DrSc.
vedúci pracoviska

prof. Ing. Miroslav Fikar, DrSc.
garant študijného programu

Mojej manželke a rodine.

Touto cestou by som sa chcel poďakovať prof. Ing. Miroslavovi Fikarovi, DrSc. a doc. Ing. Michalovi Kvasnicovi, PhD. za všetky cenné rady, pripomienky a odborné vedenie pri písaní a koordinácii mojej dizertačnej práce. Ďalej ďakujem za všestrannú pomoc a podporu celému kolektívu Ústavu informatizácie, automatizácie a matematiky.

Ing. Slavomír Blažek
Bratislava, 2016

Abstrakt

Práca je venovaná problematike návrhu hľadania optimálnej cesty pre heterogénne vozidlá. Ponímané vozidlá pozostávajú z dvoch častí, ktoré sa dokážu pohybovať individuálne každé zvlášť. Jedno z nich je rýchlejšie, ale má krátky dosah, a je preto potrebné, aby sa držalo v určitej vzdialenosti od hlavného vozidla. Cieľom je nájsť optimálnu cestu prejdenú v najkratšom možnom čase za podmienky, aby rýchlejšia časť z heterogénneho vozidlového systému navštívila všetky želané body na trase práve raz. V rámci riešeného problému uvažujeme tri situácie.

V prvom prípade uvažujeme, že poradie návštev jednotlivých bodov je dané. Ukážeme si, že hľadanie optimálnej cesty v tejto situácii je možné riešiť ako celočíselné programovanie nad kužeľom druhého rádu.

V druhej verzii predpokladáme, že poradie, v ktorom chceme body na trase navštíviť, nie je dané a našim cieľom bude nájsť také poradie, pri ktorom bude trvať prejdenie trasy najkratšie. V práci prezentujeme dva prístupy ako daný problém vyriešiť, pričom pri porovnávaní výsledkov zohľadňujeme výpočtovú náročnosť.

V treťom prípade uvažujeme, že body, ktoré chceme na trase navštíviť, sú v pohybe zároveň s vozidlom.

Kľúčové slová: multivozidlový systém, hľadanie optimálnej cesty, celočíselné programovanie, programovanie nad kužeľom druhého rádu

Abstract

This thesis considers a path planning problem for heterogeneous vehicles. Such vehicle consist of two parts which have the ability to move individually. One of them is faster, but has shorter range and is therefore required to keep in a close distance to the main vehicle. The objective is to devise an optimal path of minimal length under the condition that the faster part of the heterogeneous system visits all desired waypoints exactly once. Three versions of the problem are considered.

One assumes that the order in which the waypoints are visited is known a-priori. In such a case we show that the optimal path can be found by solving a mixed-integer second-order cone problem.

The second version assumes that the order in which the waypoints are visited is not known a-priori, but can be optimized as to shorten the length of the path. Two approaches to solve this problem are presented and evaluated with respect to computational complexity.

In the third situation we consider that visited points are moving in real time along with the vehicle.

Keywords: multi-vehicle system, path planning, mixed-integer programming, second order cone programming

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam symbolov

A	koeficient ohraničenia (matica)
a_{ij}	koeficient ohraničenia
b	koeficient pravej strany ohraničenia (vektor)
b_i	koeficient pravej strany ohraničenia
B	matica stĺpcov bázy
$\mathcal{B}$	Báza matice
c	koeficient účelovej funkcie (vektor)
C	koeficient účelovej funkcie (matica)
c_j	koeficient účelovej funkcie
D	aktívne ohraničenie
F	množina prípustných riešení
f_0	účelová funkcia
f_i	funkcie ohraničenia
H	symetrická kladne semidefinitná matica
i	pomocné počítadlo
j	pomocné počítadlo
k	pomocné počítadlo
M	koeficient hornej hranice prípustných hodnôt
m	počet ohraničení

n	pomocné počítadlo
$\mathbb{N}$	množina prirodzených čísel
$\mathcal{Q}$	kužeľ druhého rádu
$\mathbb{R}$	množina reálnych čísel
$\mathbb{R}^n$	priestor n-rozmerných reálnych vektorov
$\mathbb{R}^{n \times m}$	priestor $(n \times m)$-rozmerných reálnych matíc
s	doplnková premenná
S	doplnková premenná - matica
$\mathbb{S}^{n \times n}$	priestor symetrických $n \times n$-rozmerných matíc
$\mathbb{U}$	množina prípustných akčných zásahov
v	doplnková premenná
$\mathbb{X}$	množina prípustných stavov
x	optimalizovaná premenná
x_i	rozhodovacia premenná
W	množina aktívnych ohraničení
z	optimalizovaná premenná

Grécke symboly

α	pomocný parameter
δ	pomocný parameter
Δ	pomocný parameter
γ	pomocný parameter
λ	pomocný parameter
θ	pomocný parameter
τ_{set}	množina koncových stavov

Dolný index

eq	koeficient v tvare rovnosti

Horný index

$*$	optimálna hodnota
T	transpozícia

Zoznam skratiek

CARIMA	Controlled Auto-Regressive Integrated Moving Average
DMC	Dynamic Matrix Control
ILP	celočíselné lineárne programovanie
LP	lineárne programovanie
MILP	zmiešané celočíselné programovanie (angl.: *mixed integer linear programming*)
MI-NLP	zmiešané celočíselné nelineárne programovanie (angl.: *mixed integer nonlinear programming*)
MI-SOCP	zmiešané celočíselné programovanie nad kužeľom druhého rádu (angl.: *mixed integer second order cone programming*)
MPC	prediktívne riadenie (angl.: *model predictive control*)
PID	proporcionálne integračne derivačný
PWA	po častiach afínna (angl.: *piecewise affine*)
QP	kvadratické programovanie (angl.: *quadratic programming*)
RHC	riadenie s posuvným horizontom (angl.: *receding horizont control*)

SDP	semidefinitné programovanie
SOCP	programovanie nad kužeľom druhého rádu (angl.: *second order cone programming*)
TSP	problém cestujúceho obchodníka (angl.: *travelling salesman problem*)
VRP	smerovanie dopravných trás (angl.: *vehicle routing problem*)

Kapitola 1

Úvod

Problémy plánovania trasy sa stávajú v dnešnej dobe čoraz viac oblasťou výskumu a štúdia. Hlavný záujem v týchto problémoch pramení z potreby znižovania nákladov na celkovú logistiku. Potrebné šetrenie sme schopní dosiahnuť optimalizovaním trasy vozidiel operujúcich v teréne. Výhodou skrátenia dĺžky trasy je nie len šetrenie paliva, ale najmä času, ktorý má v súčasnosti stále vyššiu cenu. Zisk je pritom na oboch stranách – dodávateľa i odberateľa. Nehovoriac o včasnom príchode záchranárskych zložiek, kedy ušetrený čas môže mať i cenu života.

1.1 Problém smerovania dopravných trás

Logistika ako problém sa rieši v oblasti operačného výskumu už dlhú dobu. Formulácia problému súvisí s problémom obchodného cestujúceho (angl.: *Traveling Salesman Problem – TSP*), ktorým sa zaoberali matematici už v 19. storočí. Ako optimalizačná úloha bol tento problém sformulovaný okolo roku 1930 s cieľom minimalizovať dĺžku trasy pri prechádzaní medzi jednotlivými vrcholmi grafu. Vo vedeckej publikácii ho ale ako prví uverejnili Dantzig a Ramser (1959). Vo svojej práci sformulovali prvý algoritmický prístup, ktorý bol využitý pri distribúcii paliva. V roku 1964 Clarke a Wright vylepšili Dantzigov a Ramserov algoritmus novým prístupom, nazývaným ukladací algoritmus (angl.: *saving algorithm*) (Clarke a Wright, 1964), využívajúcim tzv. pažravú metódu. Nájdenie minimálnej dĺžky trasy prehľadaním všetkých možností je úloha, ktorej zložitosť narastá s pribúdajúcim počtom vrcholov superpolynomiálne, čo označujeme v kombinatorickej optimalizácii ako nedeterministický polynomiálne ťažký (alebo len NP-ťažký) problém (Toth a Vigo, 2001). Preto sa pri riešení uplatňujú heuristické metódy. Tieto metódy je možné použiť aj na riešenie problému smerovania dopravných trás (angl.: *Vehicle Routing Problem*

– *VRP*), ktorý predstavuje zovšeobecnenie problému obchodného cestujúceho pre skupinu vozidiel s určitými obmedzeniami. Podľa charakteru obmedzení rozoznávame ďalšie varianty problému smerovania dopravných trás, ako napr. problém s obmedzenými kapacitami áut, problém s časovými oknami (kedy je presne určený časový interval, v ktorom musia byť jednotlivé miesta obslúžené), problém so zberom a doručovaním a podobne. Vo väčšine podobných prípadov sa v praxi na hľadanie riešenia používa TSP, ktoré je riešené ako problém zmiešaného celočíselného programovania (Applegate, 2006; Miller et al., 1960).

V poslednej dobe je ale riešenie problému plánovania trasy pre jedno vozidlo nedostačujúce. Veľa firiem a korporácií vo väčšine prípadov operuje s niekoľkými vozidlami, ktorých pohyb a trasy je potrebné efektívne skoordinovať. V mnohých prípadoch sú tieto vozidlá usporiadané do jedného heterogénneho systému. Takéto heterogénne vozidlo potom pozostáva z niekoľkých častí, ktoré môžu byť oddelené od hlavného vozidla a vykonávať úlohy separátne. Po uskutočnení jednotlivých úloh sa tieto vozidlá vracajú naspäť na hlavné vozidlo, či už kvôli načerpaniu paliva alebo kvôli opätovnému zásobovaniu (Hoff et al., 2010).

VRP rieši mnoho aplikácii v priemysle. Faktom je, že použitie optimalizačných programov môže ušetriť spoločnosti až 5% nákladov (Hasle et al., 2007), keďže preprava je zvyčajne významnou zložkou z ceny produktu a tvorí obyčajne aj 10% (Rodrigue et al., 2009) – prepravný sektor činí v Európskej únii 10% z HDP (hrubého domáceho produktu). Preto akékoľvek šetrenie v oblasti VRP, dokonca i menej ako 5%, má veľký význam (Hasle et al., 2007).

1.2 Prediktívne riadenie

Plánovanie trasy je iba jednou zo zložiek logistiky. Čoraz častejšie sa vyskytujú v logistike autonómne vozidlá, či drony (Ackerman (2013), Miah (2016)). V tejto práci sa pozrieme na tento typ problému komplexnejšie. Naplánujeme trajektóriu, ale zároveň zabezpečíme lokálne riadenie vozidla tak, aby túto trajektóriu sledoval. Na tento účel využijeme prediktívne riadenie.

Prediktívne riadenie (angl.: *Model Predictive Control – MPC*) je v dnešnej dobe veľmi rozšírené. Je používané v rôznych odvetviach priemyslu po celom svete a keďže sa v poslednej dobe stala spotreba energie zariadení vážnym problémom, jeho popularita stále rastie. Je to najmä vďaka jeho schopnosti zlepšiť ekonomické náklady na bezpečnú prevádzku. Aby sme boli schopní viesť bezpečnú prevádzku, musíme dodržať kritéria bezpečnosti, ako dodržanie ohraničení a spätnoväzbovej stability, ale aj kritéria výkonnosti. Tými sú minimalizácia spotreby suroviny a energie, maximalizácia výťažku a čistoty produktov, mini-

malizácia fyzickej záťaže (napr. zabránenie veľkým zmenám vstupov), presnosť a rýchlosť sledovania žiadanej hodnoty, atď. MPC sa využíva v rôznych priemyselných procesoch od roku 1980, napr. v chemických prevádzkach a ropných rafinériách. Už v týchto začiatkoch používania sa vytvorilo veľa úspešných priemyselných MPC aplikácií (Qin a Badgewell, 1997).

V 70. rokoch bol vyvinutý ropnými inžiniermi v Shell prvý MPC algoritmus známy ako metóda Dynamic Matrix Control (DMC) (Cutler a Ramaker, 1979). Tento algoritmus bol predstavený začiatkom roka 1980 a bol určený pre použitie v ropných rafinériách. Táto osvedčená metóda je schopná fungovať po dlhú dobu takmer bez akéhokoľvek významného ľudského zásahu, pričom neklesá jej výkonnosť. Prediktívne riadenie je technika, ktorá je schopná zvážiť obmedzenia modelu a predpovedať odozvy zariadenia. DMC algoritmus bol určený práve predovšetkým na predpovedanie odozvy zariadenia. Metóda DMC sa používa dodnes takmer vo všetkých komerčných priemyselných distribuovaných riadiacich systémoch a softvérových balíčkoch simulačných procesov. Aplikácie, v ktorých je implementovaná, sa nachádzajú v celej rade oblastí, vrátane chemického a petrochemického priemyslu, spracovania potravín, v automobilovom priemysle a leteckých aplikácií. Podobnou metódou, ktorá je úspešne aplikovaná vo veľkých priemyselných procesoch je atraktívne a široko používané prediktívne heuristické riadenie (Richalet et al., 1978). Silné stránky tejto metódy spočívajú a v jej robustnosti a v jednoduchosti jej vykonávania. Aj tento algoritmus má vcelku veľkú hospodárnosť pre použitie v praxi, kde sa vie oceniť jeho účinnosť.

Prediktívne riadenie sa využíva aj v oblasti riadenia robotických vozidiel (Borrelli et al., 2005; Corona a De Schutter, 2008). V histórii sa pokus o vytvorenie kompletne autonómneho vozidla vyskytol už v roku 1926 (Sentinel, 1926). V dnešnej dobe už sú autonómne vozidlá v oblasti výskumu realitou, pričom v posledných rokoch je snaha zaviesť i autonómnu prepravu. V Európe, mestá v Belgicku, Francúzsku, Taliansku a Veľkej Británii majú v plánoch zaviesť transportný systém pre vozidlá bez vodiča (CitynetMobil, 2013; News, 2014).

1.3 Súčasný stav riešenej problematiky

Ako sme spomínali už v úvode, hlavný zámer tejto práce spočíva v optimalizácii trajektórie pre heterogénne vozidlo a v riadení tohto vozidla, aby túto trajektóriu sledoval. V nasledujúcej časti si rozoberieme, v akej miere sa rieši táto problematika vo svete.

Ako prvý príklad bola v rámci úvodu spomínaná publikácia Hoff et al. (2010). V tejto referencii autori spomínajú stratégiu dodávky potravín do ostrovných krajín prostredníctvom lode, ktorá má na palube zásobovacie vozidlá. Tieto nákladné autá sa vylodia na os-

trov, aby zásobili jednotlivé potravinové centrá a potom sa vrátili späť na loď pre opätovné zásobenie. Táto istá publikácia skúma aj distribúciu nealkoholických nápojov spoločnosti Coca-Cola. Táto spoločnosť používa veľkokapacitné vozidlá v kombinácii s menšími vozidlami. Kým malé vozidlá dodávajú tovar priamo do jednotlivých lokácií, veľkokapacitné prinášajú nápoje iba na miesto, odkiaľ sa tovar presunie do menších vozidiel. Správne zosúladenie tohoto heterogénneho systému ukazuje výrazné zníženie celkových nákladov na distribúciu. Podobná aplikácia je opísaná aj v Fagerholt (1999), len s tým rozdielom, že sa zameriava na námorný priemysel.

Iný prípad aplikácie pre riadenie heterogénneho multivozidlového systému sa zaoberá zbieraním odpadu. Štúdia bola publikovaná v Tung a Pinnoi (2000), kde sa rozoberá zbieranie odpadu v meste Hanoi. Multivozidlový systém obsahuje vozidlá s rozdielnymi kapacitami, ktoré musia byť skoordinované spoločne tak, aby sa minimalizovali výdaje na zozbieranie odpadu v jednotlivých oblastiach mesta.

Plánovanie trasy pre heterogénne vozidlo bolo rozoberané tiež v práci Mathew et al. (2014). Autor riešil problém špecifickej distribúcie tovaru v mestských lokalitách, kde nákladné auto nieslo kvadrikoptéru. Tá odnášala jednotlivé balíčky na želané miesta, čím výrazne urýchľovala dodávkový proces. V tomto prípade autori ukázali, že riešenie takéhoto problému je výpočtovo netriviálne, a tak sa priklonili k riešeniu s použitím heuristiky na koordináciu trasy nákladného auta a kvadrikoptéry.

Vo všetkých hore uvedených odkazoch sa autori rozhodli zamerať na heuristické prístupy pre nájdenie trasy pre heterogénne vozidlá. Heuristické riešenia sa v mnohých prípadoch nájdu pomerne ľahko. Ich nevýhodou však je, že nedávajú záruku, že výsledné riešenie je skutočne optimálne, a teda najlepšie možné. Ba čo viac, nevie sa pri nich určiť ani ako ďaleko je nájdené riešenie s heuristikou od skutočne optimálneho riešenia.

Práca, ktorá sa zaoberala nájdením optimálneho riešenia problému plánovania trasy pre heterogénne vozidlo, bola publikovaná v Garone et al. (2012). Autori uvažovali heterogénne vozidlo pozostávajúce z dvoch zložiek: lode a helikoptéry. V tomto usporiadaní sa berie loď ako vozidlo s neobmedzeným dojazdom a nízkou rýchlosťou v porovnaní s helikoptérou. Tá je na druhej strane rýchla, avšak jej dolet je časovo obmedzený vzhľadom na veľkosť nádrže. Cieľom je nájsť optimálnu trasu pre heterogénne vozidlo, v rámci ktorej je potreba stanoviť miesta vzletov a pristátí helikoptéry, ktorej cieľom je navštíviť želané stanovištia. Autori navrhli formuláciu zmiešaného celočíselného nelineárneho problému (MI-NLP) na nájdenie optimálnej cesty. Zamýšľaná MI-NLP formulácia má niekoľko nevýhod. Prvou je predpoklad, že poradie návštev jednotlivých stanovíšť je predom dané. Druhou je, že riešenie MI-NLP optimalizačného problému je výpočtovo veľmi náročné, pričom táto náročnosť výrazne rastie s pribúdajúcim množstvom stanovíšť. Autori v publikácii ukázali, že dokonca už pri malom množstve stanovíšť (rádovo 7) dosahuje čas na

vyriešenie takéhoto MI-NLP problému niekoľko hodín. Preto sa nakoniec uchýlili k použitiu niekoľkých heuristických pravidiel, čím však získali opäť len suboptimálne riešenie. Navyše, dané heuristické pravidlá opäť predpokladali, že poradie návštev jednotlivých stanovíšť bude pevne dané.

Na základe analýzy súčasného stavu riešenej problematiky sme zistili, že autori v spomínaných prácach implementovali optimalizáciu trasy pre heterogénne vozidlo buď s použitím heuristických metód, alebo so stanovenými obmedzeniami pri hľadaní optimálneho riešenia. Výsledky práce Garone et al. (2012) neboli uplatniteľné v praxi pre vysokú výpočtovú zložitosť. Preto sa v tejto práci zameriame na hľadanie optimálnej trasy pre heterogénne vozidlo tak, aby sa nám podarilo výpočtovú zložitosť riešenia znížiť na takú úroveň, aby sme mohli riešenie uplatniť v praxi aj pri väčšom množstve stanovíšť, ktoré potrebuje heterogénne vozidlo navštíviť.

1.4 Ciele dizertačnej práce

Hlavným cieľom tejto práce je navrhnúť formuláciu optimalizačného problému plánovania trasy pre heterogénne vozidlo. Pritom chceme zabezpečiť, aby sme riešenie našli bez použitia heuristických metód a teda, aby nájdené riešenie bolo skutočne optimálne. Zároveň by sme chceli výpočtovú zložitosť znížiť do takej miery, aby bola softvérová aplikácia použiteľná v praxi. Z hľadiska stanovenia problému sa zameriame na heterogénne vozidlo pozostávajúce z dvoch častí: pomalšieho vozidla s neobmedzeným dojazdom, ktoré zároveň slúži ako nosič druhého, rýchleho, tzv. agilného vozidla, ktoré má časovo obmedzený dojazd. Cieľom bude navštíviť vopred dané stanovištia agilným vozidlom. Ďalej sa venujeme sledovaniu trasy heterogénnym vozidlom prostredníctvom prediktívneho riadenia.

Čiastkové ciele tejto dizertačnej práce by sme mohli zhrnúť v niekoľkých bodoch:

- Navrhnúť výpočtovo jednoduchšiu formuláciu optimalizácie trasy pre problém riešený v práci Garone et al. (2012). V spomínanej publikácii autori navrhli na riešenie problému formuláciu zmiešaného celočíselného nelineárneho problému. Našim cieľom je navrhnúť výpočtovo výhodnejšiu formuláciu zmiešaného celočíselného programovania nad kužeľom druhého rádu (MI-SOCP). Formulácia MI-SOCP je oveľa priaznivejšia pre výpočtový čas, čo umožní vytvoriť optimálnu trasu aj pre väčší počet stanovíšť.

- Navrhnúť formuláciu pre rozšírené riešenie vyššie spomínaného problému, tak aby sme boli schopní optimalizovať aj poradie návštev jednotlivých stanovíšť. Keďže sa riešenie problému hodnotí vopred ako výpočtovo veľmi náročné, volíme dva prístupy na dosiahnutie cieľa:

  - získať poradie návštev stanovíšť vyriešením TSP problému, ktoré opomína skutočnosť, že vozidlo je heterogénne a následne navrhnutou formuláciou v prvom bode získať trasu pre heterogénne vozidlo. Treba podotknúť, že získané riešenie je iba suboptimálne, ale hľadanie riešenia je výpočtovo jednoduchšie ako v nasledujúcom prístupe popísanom nižšie.
  - Nájsť skutočne optimálne riešenie prostredníctvom rozšírenej MI-SOCP formulácie, ktorá by bola schopná optimalizovať aj poradie bodov.

- Návrh prediktívneho regulátora na sledovanie optimálnej trajektórie a určenie vplyvu voľby účelovej funkcie na kvalitu sledovania.

- Rozšíriť formuláciu hľadania optimálnej trasy pre heterogénne vozidlo tak, aby sme zvážili možnosť pohybu jednotlivých stanovíšť v reálnom čase, behom navigácie heterogénneho vozidla.

1.5 Členenie práce

V práci sa budeme zaoberať predovšetkým problematikou optimalizácie trasy pre heterogénne vozidlo. V kapitole 2 spomíname štandardné typy optimalizačných problémov, na ktoré sa dá úloha hľadania optimálnej trajektórie pre heterogénne vozidlo previesť. Keďže základom výskumu je efektívne definovanie daného optimalizačného problému, venujeme sa tu úlohám programovania od lineárnych, kvadratických až po semidefinitné programovanie a programovanie nad kužeľom druhého rádu. K niektorým optimalizačným problémom si uvedieme aj metódy a techniky riešenia.

Kapitola 3 ukazuje základné fundamenty, pomocou ktorých budeme sledovať optimálnu trajektóriu pre heterogénne vozidlo. Povieme si o prediktívnom riadení vo všeobecnosti – čo vlastne znamená, z čoho sa skladá a ako funguje. Predstavíme si formuláciu MPC ako optimalizačného problému, ktorý sa skladá z účelovej funkcie a ohraničení. Sformulujeme problém MPC pre stavové modely a opíšeme si jeho implementáciu. Rozoberieme si ohraničenia systémov a spôsoby, ako sa s nimi vysporiadať tak, aby sme dodržali základné kritéria bezpečnosti.

Kapitolou 4 otvárame problém s hľadaním trajektórie pre heterogénne vozidlo, ktorého úlohou je navštíviť stanovené cieľové body, tak aby bola dĺžka trasy minimálna. Na problém sa dá pozerať z viacerých uhlov, preto podrobne rozoberáme riešenia pri rôznych podmienkach. V prípade, že máme poradie návštev cieľových bodov dané, opisujeme optimálne riešenie v kapitole 4.2. Hneď v nasledujúcej kapitole 4.3 rozoberáme situáciu, kedy potrebujeme optimalizovať aj poradie bodov. Tu si ukážeme dva prístupy, pričom jeden nám ponúkne časovo efektívnejšie, avšak subooptimálne riešenie, kým druhý výpočtovo

náročnejšie, ale optimálne riešenie. Najzaujímavejší problém ale riešime v kapitole 4.4, kde sa zaoberáme simuláciou a uplatnením optimalizačného riešenia v praxi. Ukážeme si jednoduché prediktívne riadenie autonómneho vozidla podľa získanej trasy v reálnom čase, pričom uvažujeme aj prípad, kedy sa cieľové body v čase pohybujú. Technické riešenie prezentované v kapitolách 4.2 a 4.3 vychádza z našich prác:

- Klaučo, M. – Blažek, S. – Kvasnica, M.: An Optimal Path Planning Problem for Heterogeneous Multi-Vehicle Systems. International Journal of Applied Mathematics and Computer Science, 26(2), 297–308, 2016.
- Klaučo, M. – Blažek, S. – Kvasnica, M. – Fikar, M.: Mixed-Integer SOCP Formulation of the Path Planning Problem for Heterogeneous Multi-Vehicle Systems. In European Control Conference, Strasbourg, France, 1474–1479, 2014.

V prípadovej štúdii si porovnáme výsledky jednotlivých získaných riešení, výpočtovú náročnosť a zanalyzujeme dopad suboptimality pri optimalizovaní poradia bodov. Na záver zhrnieme prínos práce a výsledok výskumu v oblasti optimalizácie trasy pre heterogénne vozidlo. Výsledky kapitol 4.4 a 4.5 neboli doteraz publikované a predstavujú rozšírenie predchádzajúcich prác.

Kapitola 2

Prehľad optimalizačných problémov

V matematike a počítačovej vede sa hľadanie najlepšieho zlúčiteľného riešenia zo všetkých považuje za optimalizačný problém. Riešenie je možné hľadať pomocou rozličných metód. Medzi najpoužívanejšie patria metódy matematického programovania. Úloha matematického programovania pozostáva z cieľov a ohraničujúcich podmienok.

Ciele predstavujú optimalizačné kritéria, ktoré sú maximalizačného alebo minimalizačného charakteru. Vyjadrujú sa pomocou jednej alebo viacerých funkcií, ktoré nazývame účelové funkcie.

Ohraničenia predstavujú podmienky a obmedzenia, vzhľadom ku ktorým hľadáme optimum k účelovej funkcii. Vyjadrujú sa pomocou rovníc, nerovníc alebo ich sústav.

Všeobecný optimalizačný problém je stanovený ako

$$\min_{x} \quad f_0(x) \tag{2.1a}$$

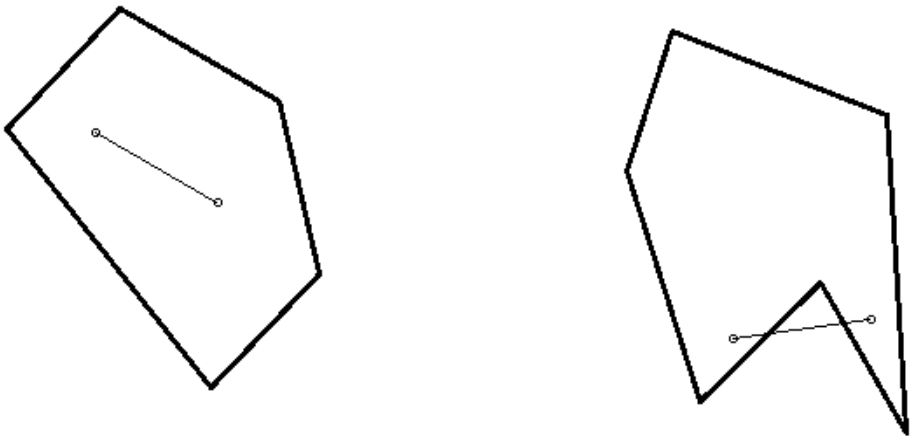
$$\text{vzhľadom na} \quad f_i(x) \leq 0, \quad i = 1, \ldots, m. \tag{2.1b}$$

Vektor $x \in \mathbb{R}^n$ je optimalizovaná premenná, $f_0 : \mathbb{R}^n \to \mathbb{R}$ je účelová funkcia a $f_i : \mathbb{R}^n \to \mathbb{R}$ predstavujú funkcie ohraničení. Optimálne riešenie k (2.1) označujeme $x^\star$, pre ktoré platí $f_0(x^\star) \leq f_0(x)$, $\forall x \in \{x \mid f_i(x) \leq 0\}$.

2.1 Konvexné optimalizačné problémy

V práci sa budeme venovať konvexným optimalizačným problémom, ktoré sú charakteristické tým, že majú konvexnú účelovú funkciu a ohraničenia tvoria konvexné množiny.

Definícia 2.1.1 (Konvexná funkcia)**.** Nech f je funkcia spojitá na intervale $[a, b]$. Potom hovoríme, že funkcia f je na konvexná práve vtedy, keď pre všetky hodnoty $\lambda \in [0, 1]$ a



Obr. 2.1: Príklad konvexnej (vľavo) a nekonvexnej (vpravo) množiny v $\mathbb{R}^2$

pre všetky body x, y z definičného oboru funkcie f platí

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda)f(y). \tag{2.2}$$

Definícia 2.1.2 (Konvexná množina)**.** Neprázdna množina $\mathcal{M} \in \mathbb{R}^n$ sa nazýva konvexná, ak platí

$$\theta x + (1-\theta)y \in \mathcal{M}, \tag{2.3}$$

pre všetky $x, y \in \mathcal{M}$ a pre všetky $\theta \in [0, 1]$.

Množina $\mathcal{M}$ je konvexná, ak s každými dvomi bodmi obsahuje aj celú ich spojnicu a platí, že množina $\mathcal{M}$ je podmnožinou $\mathbb{R}^n$. Príklad konvexnej a nekonvexnej množiny je na Obr. 2.1.

Vlastnosti účelovej funkcie a funkcií ohraničení nám hovoria o type optimalizačného problému. V nasledujúcich kapitolách si jednotlivé typy opíšeme.

2.2 Lineárne programovanie

Lineárne programovanie (LP) je špecifickým prípadom matematického programovania, kedy je účelová funkcia optimalizačnej úlohy a jej ohraničenia lineárne (Berežný a Kravecová, 2012; Cechlárová a Semanišin, 1999; Plesník et al., 1990; Rosinová a Dúbravská, 2007).

Definícia 2.2.1 (Lineárna funkcia)**.** Funkciu n-premenných $f(x), x = [x_1, \ldots, x_n]^T, x \in \mathbb{R}^n$ nazývame lineárnou funkciou, ak spĺňa nasledujúce podmienky

- $f(x + y) = f(x) + f(y)$ aditívnosť
- $f(\theta x) = \theta f(x)$ proporcionalita

Poznámka 2.2.2. Podmienku linearity vieme pretransformovať aj do jedného vzťahu. V tom prípade, by sme hovorili, že spojitá funkcia $f(x)$ je lineárna, ak

$$f(\theta x + (1 - \theta)y) = \theta f(x) + (1 - \theta)f(y) \tag{2.4}$$

platí pre všetky $x, y \in \mathbb{R}^n$ a všetky $\theta \in \mathbb{R}$.

V prípade, že funkcie f_0 (2.1a) a f_i (2.1b) sú lineárne v zmysle všeobecného optimalizačného problému definovaného v (2.1), potom môžeme daný problém formulovať a riešiť ako problém, respektíve úlohu lineárneho programovania.

Definícia 2.2.3 (Úloha lineárneho programovania)**.** Všeobecnú úlohu lineárneho programovania nazývame úlohu extremalizácie (minimalizovania alebo maximalizovania) lineárnej účelovej funkcie viacerých premenných

$$\min_x \qquad \sum_{j=1}^{n} (c_j x_j) \tag{2.5a}$$

za podmienok v tvare

$$\sum_{j=1}^{n} (a_{ij} x_j) \left\{ \begin{matrix} \leq \\ = \\ \geq \end{matrix} \right\} b_i \qquad i = 1, \ldots, m, \tag{2.6a}$$

kde $x_1, \ldots, x_n$ nazývame rozhodovacie premenné, $c_1, \ldots, c_n$ sú koeficienty účelovej funkcie, $a_{11}, a_{12}, \ldots, a_{mn}$ sú koeficienty ohraničení a $b_1, \ldots, b_m$ koeficienty pravých strán ohraničení.

Účelovú funkciu a ohraničenia z Definície 2.2.3 vieme zapísať aj vo vektorovom zápise

$$\min_x \qquad c^T x \tag{2.7a}$$

$$\text{v.n.} \qquad Ax \left\{ \begin{matrix} \leq \\ = \\ \geq \end{matrix} \right\} b. \tag{2.7b}$$

Ohraničenie 2.7b rozdeľujeme na nerovnosti a rovnosti a zapisujeme ho v tvare

$$\min_x \quad c^T x \tag{2.8a}$$
$$\text{v.n.} \quad Ax \leq b, \tag{2.8b}$$
$$A_{\text{eq}} x = b_{\text{eq}}, \tag{2.8c}$$

kde $c \in \mathbb{R}^n, A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m, A_{\text{eq}} \in \mathbb{R}^{m_{\text{eq}} \times n}$ a $b_{\text{eq}} \in \mathbb{R}^{m_{\text{eq}}}$. V tomto prípade m_{eq} a m predstavuje počet ohraničení v tvare rovnosti a nerovnosti.

2.2.1 Štandardný tvar úlohy lineárneho programovania

Okrem zápisu (2.8), môžeme problém formulovať aj v tzv. štandardnom tvare

$$\min_x \quad c^T x \tag{2.9a}$$
$$\text{v.n.} \quad A_{\text{eq}} x = b_{\text{eq}}, \tag{2.9b}$$
$$x \geq 0. \tag{2.9c}$$

Obe tieto formulácie (2.8 a 2.9) sú si ekvivaletné a je možné previesť zápis reprezentácie z jednej do druhej a naopak. Na prevody úlohy lineárneho programovania z jedného tvaru do druhého používame niekoľko základných transformácií.

Zmenu extremalizácie účelovej funkcie realizujeme vynásobením pôvodnej účelovej funkcie číslom -1. Majme teda nasledovnú úlohu

$$\max_x \quad f(x). \tag{2.10}$$

Úpravami vzťahu (2.10), spomínanými vyššie, dostaneme

$$\min_x \quad -f(x). \tag{2.11}$$

Zmenu ohraničujúcej podmienky v tvare nerovnice typu $\leq$ na ohraničujúcu podmienku v tvare nerovnice typu $\geq$ alebo naopak, realizujeme opäť vynásobením pôvodného ohraničenia číslom -1

$$\sum_{j=1}^{n} (a_{ij} x_j) \geq b_i \tag{2.12}$$

Po úprave získame zo vzťahu (2.12) formuláciu

$$\sum_{j=1}^{n} (-a_{ij} x_j) \leq -b_i. \tag{2.13}$$

Následne môžeme $x \in \mathbb{R}^n$ zapísať ako rozdiel dvoch vektorov x^+ a x^-. Substitúciou tvaru $x = x^+ - x^-$ dostaneme

$$\begin{aligned} \min_x \quad & c^T x^+ - c^T x^- && \text{(2.14a)} \\ \text{v.n.} \quad & Ax^+ - Ax^- \leq b, && \text{(2.14b)} \\ & A_{\text{eq}} x^+ - A_{\text{eq}} x^- = b_{eq}, && \text{(2.14c)} \\ & x^+ \geq 0, \quad x^- \geq 0. && \text{(2.14d)} \end{aligned}$$

Nerovnosť v (2.14b) vieme konvertovať na rovnosť pridaním premennej $s \in \mathbb{R}^m$, pričom $s \geq 0$

$$\begin{aligned} \min_x \quad & c^T x^+ - c^T x^- && \text{(2.15a)} \\ \text{v.n.} \quad & Ax^+ - Ax^- + s = b, && \text{(2.15b)} \\ & A_{\text{eq}} x^+ - A_{\text{eq}} x^- = b_{\text{eq}}, && \text{(2.15c)} \\ & x^+ \geq 0, \quad x^- \geq 0, \quad s \geq 0. && \text{(2.15d)} \end{aligned}$$

V tejto chvíli je už vidieť, že problém definovaný v (2.15) je ekvivalentný s (2.9) s novou optimalizačnou premennou $z = [{x^+}^T, {x^-}^T, s^T]^T$

$$\begin{aligned} \min_x \quad & \tilde{c}^T z && \text{(2.16a)} \\ \text{v.n.} \quad & \tilde{A}_{\text{eq}} z = \tilde{b}_{\text{eq}}, && \text{(2.16b)} \\ & z \ \geq 0 && \text{(2.16c)} \end{aligned}$$

s

$$\tilde{c} = \begin{bmatrix} c \\ -c \\ \mathbb{O}_{m\times 1} \end{bmatrix}, \tilde{A}_{\text{eq}} = \begin{bmatrix} A & -A & \mathbb{I}_{m\times m} \\ A_{\text{eq}} & -A_{\text{eq}} & \mathbb{O}_{m_{\text{eq}}\times m} \end{bmatrix}, \tilde{b}_{\text{eq}} = \begin{bmatrix} b \\ b_{\text{eq}} \end{bmatrix}. \qquad (2.17)$$

Konvertované LP má $2n + m$ premenných a $m_{\text{eq}} + 2n + m$ ohraničení. Štandardná forma (2.9) môže byť ľahko prevedená do reprezentácie (2.8) výberom $A = \mathbb{I}_{n\times n}$ a $b = \mathbb{O}_{m\times 1}$.

Rovnostné ohraničenia v (2.8c) môžeme odstrániť parametrizovaním všetkými zlúčiteľnými riešeniami v $A_{\text{eq}}x = b_{\text{eq}}$ s $x = Fz + x_0$, kde $F \in \mathbb{R}^{n\times k}$ je matica nulových vektorov matice A_{eq}, x_0 je akékoľvek partikulárne riešenie v (2.8c) a $z \in \mathbb{R}^k$ je teraz nová optimalizovaná premenná. Povšimnime si, že platí $k = n - \text{rank}(A_{\text{eq}})$. S touto zmenou premenných vieme LP (2.8) prepísať na

$$\begin{aligned} \min_z \quad & c^T(Fz + x_0) && \text{(2.18a)} \\ \text{v.n.} \quad & A(Fz + x_0) \leq b. && \text{(2.18b)} \end{aligned}$$

Tento problém má $n - \text{rank}(A_{\text{eq}})$ optimalizačných premenných a m ohraničení.

V praxi platí, že ak riešime LP s n premennými a $m+m_{\text{eq}}$ ohraničeniami, potrebujeme v priemere približne $O((n^3 + n^2 s)\sqrt{m + m_{\text{eq}}})$ operácií (den Hertog, 1994)).

Definícia 2.2.4 (Zlúčiteľné riešenie)**.** Pre úlohy lineárneho programovania platia nasledovné tvrdenia (Berežný a Kravecová, 2012):

- Vektor $x \in \mathbb{R}^n$ vyhovujúci ohraničeniam (2.1b) nazývame zlúčiteľným riešením danej úlohy lineárneho programovania.

- Úlohu lineárneho programovania nazývame zlúčiteľnou, ak má aspoň jedno zlúčiteľné riešenie. V opačnom prípade ju nazývame nezlúčiteľnou.

- Zlúčiteľné riešenie $x \in \mathbb{R}^n$, v ktorom účelová funkcia (2.1a) danej úlohy LP nadobúda požadovanú extremálnu (minimálnu alebo maximálnu) hodnotu, nazývame optimálnym riešením x^*.

- Zlúčiteľnú úlohu lineárneho programovania nazývame neohraničenou, ak hodnoty zlúčiteľného riešenia môžu byť ľubovoľne veľké (v kladnom alebo zápornom smere). V opačnom prípade prípustnú úlohu LP nazývame ohraničenou.

Množina zlúčiteľných riešení podľa Definície 2.2.4 môže byť prázdna, neprázdna ohraničená a neprázdna neohraničená. Pre problém LP môžeme mať počet optimálnych riešení nula, jedno, alebo nekonečne veľa. Tabuľka 2.1 znázorňuje prehľadne jednotlivé kombinácie možností, ktoré môžu nastať.

Tabuľka 2.1: Vzťah medzi množinou prípustných riešení a počtom optimálnych riešení

	prázdna	neprázdna ohraničená	neprázdna neohraničená
žiadne	•	-	•
práve jedno	-	•	•
nekonečne veľa	-	•	•

2.2.2 Simplexová metóda

Majme danú úlohu LP v štandardnom tvare (2.9) s n-premennými a m-ohraničujúcimi podmienkami nasledovne

$$\min_x \quad \sum_{j=1}^{n}(c_j x_j) \tag{2.19a}$$

$$\text{v.n.} \quad \sum_{j=1}^{n}(a_{ij} x_j) = b_i \quad \text{pre } i = 1, \ldots, m, \tag{2.19b}$$

$$x_j \geq 0 \quad \text{pre } j = 1, \ldots, n. \tag{2.19c}$$

Nech existuje báza $\mathcal{B} = \{A_{B(1)}, \ldots, A_{B(m)}\}$ matice ohraničení A_{eq} zo vzťahu (2.9) a nech $x_{\mathcal{B}}$ je bázické zlúčiteľné riešenie prislúchajúce tejto báze, ktorého bázické zložky sú $x_{10}, x_{20}, \ldots, x_{m0}$, pričom nebázické zložky sú nulové, potom platí

$$\sum_{i=1}^{m}(x_{i0} A_{B(i)}) = b. \tag{2.20}$$

Keďže $\mathcal{B}$ je báza matice A, tak každý stĺpec A_j pre $j = 1, \ldots, n$, matice A sa dá vyjadriť ako lineárna kombinácia bázových stĺpcov

$$A_j = \sum_{i=1}^{m}(x_{ij} A_{B(i)}), \tag{2.21}$$

kde x_{ij} je i-ta súradnica stĺpca A_j v báze $\mathcal{B}$.

Na základe toho je možné odvodiť nasledujúcu vetu (Berežný a Kravecová, 2012):

Veta 2.2.5. Nech je dané nejaké bázické zlúčiteľné riešenie x sústavy $Ax = b$ prislúchajúce báze $\mathcal{B} = \{A_{B(1)}, \ldots, A_{B(m)}\}$. Nech A_j je taký stĺpec matice A, že $A_j \notin \mathcal{B}$. Potom riešenie x'_0 určené nasledovne

$$x'_{i0} = \begin{cases} x_{i0} - \theta x_{ij}; & \text{pre } i \neq k \\ \theta; & \text{pre } i = k, \end{cases} \tag{2.22}$$

kde

$$\theta = \frac{x_{k0}}{x_{kj}} = \min\{\frac{x_{i0}}{x_{ij}};\ \text{pre } i \text{ také, že } x_{ij} > 0\} \tag{2.23}$$

je zlúčiteľným bázickým riešením s bázickými zložkami x'_{i0} pre $i = 1, \ldots, m$, prislúchajúce báze $\mathcal{B}' = \mathcal{B} \cup \{A'_j\} \setminus \{A'_k\}$. Symbol A'_j označuje aktuálny j-tý stĺpec, ktorý do bázy vstupuje a A'_k označuje aktuálny k-tý stĺpec, ktorý z bázy vystupuje.

Definícia 2.2.6. Prvok x_{kj}, definovaný vo Vete 2.2.5, nazývame pivot a prechod medzi bázickými zlúčiteľnými riešeniami nazývame pivotovanie.

Nech je daná úloha LP v štandardnom tvare s n-premennými a m-ohraničujúcimi podmienkami v maticovom tvare nasledovne

$$\min_x \quad c^T x \tag{2.24a}$$
$$\text{v.n.} \quad Ax = b \tag{2.24b}$$
$$x \geq 0 \tag{2.24c}$$

a nech $x_0 = [x_1, \ldots, x_n]^T$ je nejakým bázickým zlúčiteľným riešením ohraničení tejto úlohy LP. Hodnotu účelovej funkcie v tomto bázickom zlúčiteľnom riešení vypočítame

$$u_0 = c^T x. \tag{2.25}$$

Definícia 2.2.7 (Relatívna cena)**.** Pre každé $j = 1, \ldots, n$ definujeme u_j nasledovne

$$u_j = \sum_{i=1}^{m} (c_j x_{ij}). \tag{2.26}$$

Číslo $\overline{c_j} = c_j - u_j$ nazývame relatívna cena stĺpca A_j.

Ak v bázickom zlúčiteľnom riešení x_0 urobíme pivotovanie, pri ktorom stĺpec A_j vstúpi do bázy, účelová funkcia sa zmení o hodnotu

$$\theta \overline{c_j} = \theta(c_j - u_j). \tag{2.27}$$

Ak pre každé $j = 1, \ldots, n$ je $\overline{c_j} = c_j - u_j$ nezáporné, tak dané bázické zlúčiteľné riešenie x_0 je optimálne.

Ak existuje j také, že platia súčasne nasledujúce podmienky

- $\overline{c_j} < 0$
- $x_{ij} \leq 0$ pre všetky $i = 1, \ldots, m$,

tak táto úloha LP je zlúčiteľná neohraničená.

Primárny algoritmus simplexovej metódy

Simplexový algoritmus je iteračný výpočtový postup na hľadanie optimálneho riešenia úlohy LP (Berežný a Kravecová, 2012). Majme danú úlohu LP v štandardnom tvare (2.9) s n-premennými a m-ohraničujúcimi podmienkami nasledovne

$$\min_x \quad \sum_{j=1}^{n} (c_j x_j) \tag{2.28a}$$
$$\text{v.n.} \quad Ax = b, \tag{2.28b}$$
$$x \geq 0. \tag{2.28c}$$

Nech existuje báza $\mathcal{B} = \{A_{B(1)}, \ldots, A_{B(m)}\}$ matice ohraničení A. Každý stĺpec A_j pre $j = 1, \ldots, n$ matice A sa dá vyjadriť ako lineárna kombinácia bázových stĺpcov a teda:

$$A_j = \sum_{i=1}^{m} (x_{ij} A_{B(i)}). \tag{2.29}$$

Nech $x_{\mathcal{B}} = [x_1, \ldots, x_n]^T$ je bázické riešenie prislúchajúce báze $\mathcal{B}$. Platí pe neho, že $x_j = 0$ pre všetky j také, že $A_j \notin \mathcal{B}$ a tiež:

$$\sum_{j=1}^{n} (x_j A_j) = \sum_{i=1}^{m} (x_{i0} A_{B(i)}) = b, \tag{2.30}$$

kde symbolom x_{i0} označujeme také x_j, že $A_j = A_{B(i)}$. Hodnotu účelovej funkcie pre dané bázické zlúčiteľné riešenie vypočítame takto:

$$u_0 = \sum_{j=1}^{n} (x_j c_j) = \sum_{i=1}^{m} (x_{i0} c_{B(i)}). \tag{2.31}$$

Pri riešení úlohy LP prostredníctvom simplexového algoritmu musíme sústavu rovníc úlohy LP prepísať do maticovej podoby, ktorá sa postupne mení, až kým sa nedospeje k riešeniu. Táto matica sa zapisuje do simplexovej tabuľky.

Tabuľka 2.2: Forma simplexovej tabuľky

$\mathcal{B}$	x_0	x_1	x_2	$\cdots$	x_n
-	$-f$	$\overline{c_1} = x_{01}$	$\overline{c_2} = x_{02}$	$\cdots$	$\overline{c_n} = x_{0n}$
$x_{B(1)}$	x_{10}	x_{11}	x_{12}	$\cdots$	x_{1n}
$x_{B(2)}$	x_{20}	x_{21}	x_{22}	$\cdots$	x_{2n}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$x_{B(n)}$	x_{m0}	x_{m1}	x_{m2}	$\cdots$	x_{mn}

Nech je báza $\mathcal{B}$ tvorená stĺpcami jednotkovej matice rozmeru $m \times x$, potom:

- stĺpec $\mathcal{B}$ obsahuje označenia premenných, ktorým prislúchajú bázické stĺpce,
- stĺpec x_0 obsahuje hodnoty príslušného bázického riešenia, teda príslušné prvky pravých strán ohraničení (s výnimkou $-f$),
- stĺpec x_j obsahuje súradnice stĺpca A_j v báze $\mathcal{B}$,
- v riadku označenom '-' sa zapíše do bunky $-f$ nula a do ostatných buniek koeficienty účelovej funkcie. Pred začatím riadkových úprav nastavíme tento riadok tak, aby nad bázickými stĺpcami boli nuly. Po jednotlivých riadkových operáciách dostaneme v bunke $-f$ hodnotu účelovej funkcie pre príslušné bázické riešenie a v ostatných bunkách tohto riadku hodnoty zodpovedajúcich relatívnych cien.

Ak pivotujeme simplexovú tabuľku podľa prvku x_{kj} určeného vzorcom (2.23), tak pre všetky $i = 1, \ldots, m$ a $l = 0, \ldots, n$ sa prvky simplexovej tabuľky zmenia nasledovne:

$$x'_{il} = \begin{cases} \frac{x_{il}}{x_{kj}} & \text{pre } i = k, \\ x_{il} - \frac{x_{il}}{x_{kj}} x_{kl} & \text{pre } i \neq k. \end{cases}$$

Simplexová tabuľka úlohy lineárneho programovania je primárne zlúčiteľná, ak $x_{i0} \geq 0$ pre každé $i = 1, \ldots, m$. Ak platí $x_{0j} \geq 0$ pre každé $j = 1, \ldots, n$, hovoríme, že táto tabuľka je duálne zlúčiteľná. Ak je tabuľka primárne aj duálne zlúčiteľná, označujeme simplexovú tabuľku úlohy lineárneho programovania ako optimálnu.

2.3 Kvadratické programovanie

Problémy, kde je účelová funkcia f_0 v (2.1) konvexná a kvadratická a ohraničenia f_i sú lineárne, definujeme ako problémy kvadratického programovania (v angličtine: quadratic programming – QP problem) (Hamala, 1972; Maňas, 1979)

$$\min_x \quad \frac{1}{2} x^T H x + c^T x \tag{2.32a}$$

$$\text{v.n.} \quad Ax \leq b, \tag{2.32b}$$

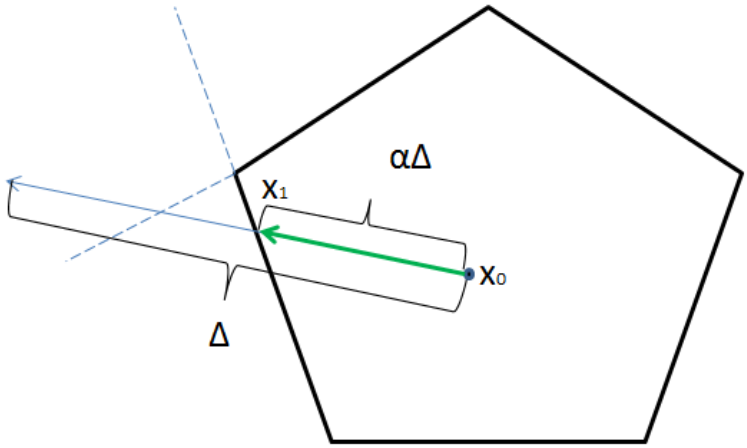
kde A je matica sústavy ohraničení ($A = [a_{ij}]$ pre $i = 1, \ldots, m$ a $j = 1, \ldots, n$), b je vektor pravej strany ohraničení ($b \in \mathbb{R}^m$), x je vektor rozhodovacích premenných ($x \in \mathbb{R}^n$), H je matica $n \times n$, c je vektor koeficientov účelovej funkcie ($c \in \mathbb{R}^n$) a n je počet rozhodovacích premenných úlohy. Aby bola účelová funkcia konvexná, musí byť H symetrická kladne semidefinitná matica ($H = H^T \succeq 0$).

Keďže sústava ohraničení úlohy kvadratického programovania je tvorená lineárnymi, a teda konvexnými funkciami a predpokladáme, že matica H je kladne semidefinitná, a teda aj účelová funkcia úlohy je konvexná, tak úloha kvadratického programovania je konvexným optimalizačným problémom.

Efektivita riešenia kvadratického programovania býva v hrubom merítku rovnaká ako v prípade lineárneho programovania, ale napriek tomu sú riešenia QP v priemere približne päťkrát pomalšie ako LP riešenia (Neumaier, 2004). Jednotlivé softvérové balíky si uvedieme v kapitole (2.7).

2.3.1 Metóda aktívnych množín

Na riešenie úloh QP je jedným z najčastejšie používaných prístupov metóda aktívnych množín (Lau et al., 2009). Táto metóda rieši kvadratický problém (2.32) prostredníctvom identifikovania aktívnej množiny jej riešenia x^* a jej postup zachytáva Algoritmus 1.



Obr. 2.2: Grafické znázornenenie metódy aktívnych ohraničení. Mnohouholník predstavuje množinu prípustných riešení. Šípka ukazuje posun bodu x_0 o hodnotu Δ redukovanú na hodnotu $\alpha\Delta$ kvôli ohraničeniu. Týmto spôsobom sme našli bod x_1, ktorý vstupuje do nasledujúcej iterácie cyklu.

Na začiatok si nájdeme počiatočné zlučiteľné riešenie x_0. Metóda začína s počiatočným odhadom W_0 množiny aktívnych ohraničení. V kroku 4 riešime problém

$$\min_{\Delta_k} \quad \frac{1}{2}(x_k + \Delta_k)^T H(x_k + \Delta_k) + c^T(x_k + \Delta_k) \tag{2.33a}$$

$$\text{v.n.} \quad A_{W_k}(x_k + \Delta_k) = b_{W_k}, \tag{2.33b}$$

kde A_{W_k} sú iba tie riadky matice A, ktoré sú indexované prvkami aktívnej množiny W_k. Tento problém riešime pomocou Lagrangeových násobičov. Získame zlepšenie Δ_k ako zlepšujúci smer pôvodného odhadu x_k. Ak $\Delta_k = 0$ a zároveň sú všetky λ nezáporné, potom x_k je optimálnym riešením problému definovaného v (2.32). Ak $\Delta_k = 0$, ale existuje λ, ktorá je záporná, potom odstránime z množiny W_k index, ktorý zodpovedá indexu najmenšej hodnoty λ a pokračujeme v cykle s hodnotou x_k aj do ďalšej iterácie. V prípade, že získané hodnota Δ_k je rôzna od nuly, existuje možnosť, že je potrebné pridať ďalšie ohraničenie. Ak bod $x_{k+1} = x_k + \Delta_k$ spĺňa ohraničenia (2.32b), aktívnu množinu nemeníme (teda $W_{k+1} = W_k$) a pokračujeme ďalšou iteráciou. Ak $x_k + \Delta_k$ poruší ohraničenia, nájdeme také skrátenie vektora Δ_k, aby bod $x_k + \alpha\Delta_k$ ležal v ohraničeniach. Zároveň pridáme jedno ohraničenie do aktívnej množiny. Graficky je nájdenie bodu x_{k+1} zobrazené na Obr. 2.2. Následne pokračujeme iteratívne ďalej až do chvíle, kým nenájdeme presnú množinu aktívnych ohraničení.

V Algoritme 1, A_i predstavuje i-ty riadok A a A_{W_k} je matica získaná ponechaním

Algoritmus 1 Metóda aktívnych množín

Vstupy: úloha kvadratického programovania (2.32)

Výstupy: optimálne riešenie x^*

1: výpočítaj počiatočné zlúčiteľné riešenie x_0
2: $W_0 = \emptyset$
3: $k = 0$
4: **while** true **do**
5: $k \leftarrow k + 1$
6: rieš $\begin{bmatrix} H & A_{W_k}^T \\ A_{W_k} & 0 \end{bmatrix} \begin{bmatrix} \Delta_k \\ \lambda \end{bmatrix} = \begin{bmatrix} -c - Hx_k \\ 0 \end{bmatrix}$ pre (Δ_k, λ)
7: **if** $\Delta_k = 0$ **then**
8: **if** $\lambda \geq 0$ **then**
9: $x^* = x_k$ a ukonči cyklus
10: **else**
11: odstráň z množiny W_k index, ktorý zodpovedá najmenšej hodnote λ
12: $x_{k+1} = x_k$
13: **end if**
14: **else**
15: $D_k = \{i \notin W_k : A_i\Delta_k > 0, \frac{b_i - A_i x_k}{A_i \Delta_k} < 1\}$
16: **if** $D_k = \emptyset$ **then**
17: $x_{k+1} = x_k + \Delta_k$
18: $W_{k+1} = W_k$
19: **else**
20: $\alpha = \min_{i \in D_k} \left\{ \frac{b_i - A_i x_k}{A_i \Delta_k} \right\}$
21: $x_{k+1} = x_k + \alpha \Delta_k$
22: vytvor W_{k+1} pridaním jedného elementu D_k do W_k
23: **end if**
24: **end if**
25: **end while**
26: **return** x^*

i-teho riadku z A pre všetky $i \in W_k$. Ak sú všetky hodnoty b vo vzťahu (2.32b) kladné, potom $x_0 = (0, 0, \ldots, 0)^T$ je zlúčiteľné počiatočné riešenie pre (2.32).

2.4 Celočíselné lineárne programovanie

Pod úlohou celočíselného lineárneho programovania (angl.: *Integer Linear Programming – ILP*) rozumieme maximalizáciu alebo minimalizáciu lineárnej účelovej funkcie pri lineárnych ohraničeniach, pričom na rozdiel od LP predpokladáme, že všetky optimalizované premenné sú celočíselné. Formulácia úlohy ILP má nasledujúci tvar

$$\min_x \quad c^T x \tag{2.34a}$$
$$\text{v.n.} \quad Ax \leq b, \tag{2.34b}$$
$$x \in \mathbb{Z}^n. \tag{2.34c}$$

Ak z úlohy celočíselného lineárneho programovania (2.34) odstránime podmienku celočíselnosti premenných (2.34c), dostaneme úlohu lineárneho programovania, ktorú nazývame relaxovanou úlohou ILP. Ak máme optimálne riešenie relaxovanej úlohy celočíselného lineárneho programovania celočíselné, tak je zároveň aj optimálnym riešením úlohy celočíselného lineárneho programovania.

2.4.1 Zmiešané celočíselné programovanie

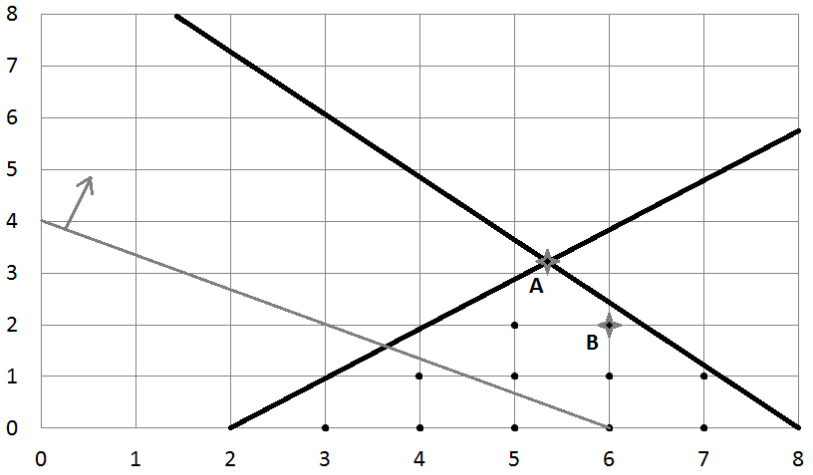
Ak je vektor x optimalizovaných premenných zložený z reálnej aj binárnej časti, čiže $x = [x_r^T, x_b^T]^T$ s $x_r \in \mathbb{R}^{n_r}$ a $x_n \in \{0, 1\}^{n_b}$ a účelová funkcia f_0 a ohraničenia f_i sú lineárne, tak problém definovaný v (2.34) sa rieši ako zmiešané celočíselné programovanie (angl.: *Mixed Integer Linear Programming – MILP*). Formálne ho môžeme zapísať

$$\min_x \quad c_r^T x_r + c_b^T x_b \tag{2.35a}$$
$$\text{v.n.} \quad A_r x_r + A_b x_b \leq b, \tag{2.35b}$$
$$x_b \in \{0, 1\}^{n_b}. \tag{2.35c}$$

MILP problém sa rozsiahlo študuje rovnako ako po teoretickej tak aj po praktickej stránke (Floudas (1995), Fletcher a Leyffer (1998)), pričom vzniklo aj mnoho softvérových implementácií (ILOG, Inc. (2009), Dash Associates (1999)).

2.4.2 Metódy riešenia celočíselného programovania

Pokiaľ by sme úlohu ILP riešili pomocou metód na riešenie LP, s tým, že by sme výsledok zaokrúhlili, nie je zaručené, že by sme dostali optimálne riešenie. Príklad je zobrazený



Obr. 2.3: Aproximácia celočíselného programovania: bod A – optimálne riešenie LP, bod B – optimálne riešenie ILP

na Obr. 2.3. Dokonca by sa mohlo stať, že riešenie by nebolo prípustné. Celočíselné optimálne riešenie totiž môže byť vnútorným bodom množiny prípustných riešení, ktorú tvoria izolované celočíselné body (angl.: *lattice points*).

Napriek tomu sa v praxi zaokrúhľovanie používa v úlohách, kedy je suboptimálne riešenie akceptovateľné. Ide hlavne o úlohy, ktoré obsahujú veľa premenných, kde je problémom časová a pamäťová náročnosť. Časová zložitosť je hlavnou nevýhodou ILP. Kým úloha LP je riešiteľná v polynomiálnom čase, úloha ILP má náročnosť NP-hard (Nemhauser a Wolsey, 1988), čiže všeobecne nie je známy polynomiálny algoritmus na riešenie. No napriek tomu, niektoré špeciálne prípady sú riešiteľné pomocou polynomiálnych algoritmov. Tie sú založené napríklad na tzv. *pažravom* prístupe (angl.: *greedy algorithm*) (Cormen et al., 2001), ktorý v každom svojom kroku vyberá lokálne optimálne riešenie v nádeji, že je toto riešenie globálne optimálne. Jeden z najznámejších prípadov, kde je možné použiť tento prístup je úloha minimálnej kostry – Kruskalov algoritmus (Jensen a Bard, 2003).

Podstatný rozdiel pri MILP oproti štandardnému ILP problému definovanom v (2.34) je jeho binárne obmedzenie (2.35c), ktoré nám vytvára nekonvexný problém, ktorý značne komplikuje riešenie globálneho optima. Zložitosť algoritmu je, v najhoršom prípade, exponenciálna v počte binárnych premenných n_b.

Medzi najrozšírenejšie metódy riešenia úloh celočíselného programovania a MILP pat-

ria:

- enumeračná metóda (angl.: *Enumerative method*) (Eiselt et al., 2000)
- metóda sečných nadrovín (angl.: *Cutting plane algorithm*) (Schrijver, 1986)
- metóda vetiev a hraníc (angl.: *Branch and bound method*) (Williams, 2009)

Enumeračná metóda

Výpočet riešenia je založený na prehľadávaní oblasti zahrňujúcej všetky prípustné riešenia. Vzhľadom k celočíselnému obmedzeniu premenných je počet týchto riešení konečný ale ich počet je neprakticky veľký. Preto je táto metóda vhodná iba pre malé problémy s obmedzeným počtom diskrétnych premenných.

Postup je možné zovšeobecniť na úlohu MILP tak, že ku každej kombinácii diskrétnych premenných je vyriešená úloha LP, kde sú diskrétne premenné brané ako konštanty. Potom je možné prehľadávanie realizovať ako binárny strom, čo umožňuje redukovať počet neprípustných stavov.

Metóda sečných nadrovín

Algoritmus začína výpočtom úlohy tak, že zanedbáva požiadavku na celočíselné hodnoty a úloha je vyriešená klasickými metódami LP. Výpočet sa realizuje iteratívne tak, že v každom kroku je pridaná ďalšia obmedzujúca podmienka zužujúca oblasť prípustných riešení. Hovorí sa aj o zavádzaní tzv. platných nerovností (angl.: *valid inequalities*). Každá nová podmienka musí spĺňať nasledujúce vlastnosti:

- optimálne riešenie nájdené pomocou LP sa stane neprípustným
- žiadne celočíselné riešenie prípustné v predchádzajúcom kroku sa nesmie stať neprípustným

Nové obmedzenie splňujúce tieto vlastnosti je pridané v každej iterácii. Vzniknutá úloha ILP je vždy znovu riešená ako LP. Proces sa opakuje až dovtedy kým sa nenájde prípustné celočíselné riešenie. Konvergencia takéhoto algoritmu potom závisí na spôsobe pridávania obmedzujúcich podmienok. Medzi najpoužívanejší algoritmus používajúci túto metódu patrí *Gomoryho algoritmus* (Floudas a Pardalos, 2009).

Najprv sa pomocou simplexovej metódy vyrieši relaxácia danej ILP. Potom sa pridávajú k úlohe LP ohraničujúce podmienky (Gomoryho rezy), ktoré zužujú množinu prípustných riešení. Na riešenie úlohy rozšírenej o takýto rez je výhodné použiť duálnu simplexovú metódu.

Nech je daná úloha celočíselného lineárneho programovania v štandardnom tvare

$$\min_x \quad c^T x \tag{2.36a}$$
$$\text{v.n.} \quad Ax = b, \tag{2.36b}$$
$$x \geq 0, x \in \mathbb{Z}^n. \tag{2.36c}$$

Majme optimálnu tabuľku pre relaxáciu úlohy ILP. Prvky danej tabuľky budeme označovať γ_{ij}.

Veta 2.4.1. Nech v optimálnej simplexovej tabuľke existuje také $i \in \{1, 2, ..., m\}$, pre ktoré platí $\gamma_{i0} \notin \mathbb{Z}$. Ak k tabuľke pridáme rovnicu:

$$\sum_{j=1}^{n} \{\gamma_{ij}\} x_j - g = \{\gamma_{i0}\}, \quad g \geq 0, \tag{2.37}$$

tak žiadne celočíselné prípustné riešenie ILP sa nevylúči a nová simplexová tabuľka bude primárne neprípustná, duálne prípustná a bázická (Berežný a Kravecová, 2012).

Algoritmus:

1. Vyriešime relaxáciu úlohy ILP. Ak získame optimálne riešenie, potom je zároveň aj riešením danej úlohy. V opačnom prípade pokračujeme ďalej.

2. Keďže relaxácia ILP nemá celočíselné riešenie, existuje nejaké $\gamma_{i0} \notin \mathbb{Z}$. Podľa toho riadku pridáme nové ohraničenie, tzv. Gomoryho rez:

$$\sum_{j=1}^{n} \{\gamma_{ij}\} x_j - g = \{\gamma_{i0}\}, \quad g \geq 0. \tag{2.38}$$

3. Do optimálnej tabuľky pre relaxáciu pridáme jeden stĺpec pre premennú g a riadok pre Gomoryho rez.

4. Pomocou simplexovej metódy nájdeme nové optimum. Ak je celočíselné, je zároveň optimom aj pôvodnej ILP. Ak nie je, vrátime sa k bodu 2.

Symbolom $\{a\}$ označujeme zlomkovú časť a. Platí, že $\{a\} = a - \lfloor a \rfloor$. Symbolom $\lfloor a \rfloor$ označujeme zaokrúhlenie čísla a nadol, čiže najbližšie menšie alebo rovné celé číslo k číslu a.

Metóda vetiev a hraníc

Princíp tejto iteračnej metódy spočíva v tom, že sa postupne menia ohraničenia daných premenných a vytvárajú sa nové úlohy LP (Chattová, 2011). Jednotlivé úlohy riešime každú zvlášť a dosiahnuté riešenia posudzujeme vzhľadom na podmienku celočíselnosti.

Schému algoritmu môžeme zhrnúť do nasledujúcich dvoch krokov:

- **vetvenie (angl.: *branching*)** - predstavuje rozklad množiny prípustných riešení na navzájom disjunktné podmnožiny. K danej úlohe, ktorej premenná x_j má neceločíselné hodnotu $\overline{b_j}$, vytvoríme dve čiastkové úlohy a k jednej z nich pridáme obmedzenie

$$x_j \leq \lfloor \overline{b_j} \rfloor, \tag{2.39}$$

kým k druhej zase obmedzenie

$$x_j \geq \lfloor \overline{b_j} \rfloor + 1. \tag{2.40}$$

Ostatné ohraničenia rovnako ako účelovú funkciu ponechávame v obidvoch úlohách nezmenené.

V nasledujúcich prípadoch máme splnenú podmienku na ukončenie vetvenia v danom uzle:

  - relaxovaná LP úloha nemá prípustné riešenie,
  - optimálna hodnota účelovej funkcie relaxovanej úlohy nie je lepšia ako hodnota účelovej funkcie v doteraz v najlepšom nájdenom prípustnom riešení,
  - optimálne riešenie relaxovanej úlohy LP je celočíselné,

- **ohraničovanie (angl.: *bounding*)** - určíme pre každú z podmnožín dolnú (ak máme minimalizačnú úlohu), resp. hornú (ak máme maximalizačnú úlohu) hranicu hodnôt účelovej funkcie na danej podmnožine. Našim cieľom je teda získať odhad, ktorý poskytuje informáciu nakoľko dobré (v zmysle hodnoty účelovej funkcie) môže byť najlepšie prípustné riešenie. Preto sa aj v prípade tejto metódy rieši relaxovaná úloha k danej ILP úlohe.

Pri riešení niektorých typov úloh ILP je metóda veľmi efektívna. Týka sa to predovšetkým praktických úloh s menším počtom premenných, keďže nám počet riešených parciálnych úloh rastie exponenciálne. V niektorých prípadoch je možné určiť dolnú a hornú hranicu pre jednotlivé premenné, pričom optimálne riešenie leží niekde medzi týmito hranicami. Čím máme užší inicializačný interval, tým sa musí spracovať menší priestor prehľadávania.

2.4.3 Problém obchodného cestujúceho

Problém obchodného cestujúceho (angl.: *Travelling Salesman Problem – TSP*) je jedna z najznámejších a najviac študovaných optimalizačných úloh (Crowder a Padberg, 1980; Miller et al., 1960; Reinelt, 1994). Je to aj vďaka veľkému množstvu jej praktických aplikácií - logistika dopravy, distribúcia tovaru, apod. TSP je však aj oporným bodom pri riešení mnohých iných optimalizačných úloh. Definujme si jeho znenie.

Problém 2.4.2. Daný je zoznam miest a vzdialeností medzi nimi. Úlohou je navrhnúť najkratšiu možnú trasu tak, aby sa navštívilo každé z miest práve raz a potom sa navrátilo do pôvodného mesta.

Problém 2.4.2 sa z hľadiska kombinatorickej optimalizácie považuje za NP-hard problém. Z toho vyplýva, že s rastúcim počtom miest nám výpočtová zložitosť pre najhorší možný scenár narastá superpolynomiálne, iným slovom, exponenciálne. Z tohoto dôvodu sa zavádzajú pri riešení rôzne heuristické metódy. Jednou z nich je napríklad heuristika algoritmu najbližšieho suseda, kde ako už názov napovedá si hľadáme cestu tak ,že vyberieme nasledujúce mesto také, ktoré je k nám aktuálne najbližšie. V priemere táto metóda vráti o 25% dlhšiu trasu ako je optimálna (Johnson a McGeoch, 1997).

TSP vieme formulovať ako úlohu celočíselného lineárneho programovania. Majme zoznam miest označených $0, \ldots, n$. Definujme si premenné x_{ij} nasledovne

$$x_{ij} = \begin{cases} 1 & \text{ak cesta vedie z mesta } i \text{ do mesta } j, \\ 0 & \text{inak.} \end{cases} \tag{2.41}$$

Pre $i = 0, \ldots, n$ si definujeme pomocnú premennú u_i a c_{ij} predstavuje vzdialenosť z mesta i do mesta j. Samotnú formulácia úlohy TSP je

$$\min_{x} \quad \sum_{i=0}^{n} \sum_{j=1, j\neq i}^{n} (c_{ij} x_{ij}) \tag{2.42a}$$

$$\text{v.n.} \quad x_{ij} \in \{0, 1\} \quad i, j = 1, \ldots, n, \tag{2.42b}$$

$$u_i \in \mathbb{Z} \quad i = 1, \ldots, n, \tag{2.42c}$$

$$\sum_{i=0, i\neq j}^{n} x_{ij} = 1 \quad j = 1, \ldots, n, \tag{2.42d}$$

$$\sum_{j=0, j\neq i}^{n} x_{ij} = 1 \quad i = 1, \ldots, n, \tag{2.42e}$$

$$u_i - u_j + n x_{ij} \leq n - 1 \quad 1 \leq i \neq j \leq n, \tag{2.42f}$$

kde ohraničenie (2.42d) vyjadruje, že každé mesto môže byť navštívené práve z jedného mesta, ohraničenie (2.42e) značí, že z každého mesta môže byť iba jedna cesta a posledné ohraničenie (2.42f) nám zabezpečuje, že existuje iba jedna uzavretá trasa, na ktorej ležia všetky mestá. Inými slovami nemôže existovať viacero trás, ktoré by pokrývali dokopy všetky mestá, ale navzájom by neboli prepojené.

2.5 Semidefinitiné programovanie

Na úlohu semidefinitného programovania (SDP) sa dá pozerať ako na rozšírenie úlohy lineárneho programovania (Wolkowicz et al., 2012). Vezmime si teda úlohu lineárneho

programovania v štandardnom tvare (2.9). Podmienka nezápornosti $x \geq 0$ z LP je v prípade SDP nahradená zovšeobecnenou nerovnosťou vzhľadom na priestor kladne semidefinitných matíc. Semidefinitné programovanie môžeme považovať za všeobecnejšie ako LP práve vďaka tejto nelineárnej podmienke. Preto veľké množstvo konvexných úloh možno formulovať a efektívne riešiť ako úlohu SDP (Vandenberghe a Boyd, 1994, 1999).

V úlohe SDP optimalizujeme lineárnu funkciu symetrickej matice X, vzhľadom na m lineárnych maticových nerovností za podmienky kladne semidefinitnej matice X. Takáto podmienka je síce nelineárna, ale konvexná. Úlohu SDP možno definovať v nasledujúcom tvare

$$\min_{X} \quad C \bullet X \tag{2.43a}$$

$$\text{v.n.} \quad A_i \bullet X = b_i \quad \text{pre } i = 1, \ldots, m, \tag{2.43b}$$

$$X \succeq 0, \tag{2.43c}$$

kde $X \in \mathbb{S}^n$, čo je priestor symetrických $n \times n$ matíc, $b \in \mathbb{R}^m$, $A_i \in \mathbb{R}^{n \times n}$ sú koeficienty ohraničení a $C \in \mathbb{R}^{n \times n}$ je koeficient účelovej funkcie. Nerovnosť $X \succeq 0$ značí obmedzenie matice X na kladne semidefinitnú maticu.

Zápis $C \bullet X$ predstavuje skalárny súčin symetrických matíc C a X:

$$C \bullet X = \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} X_{ij}. \tag{2.44}$$

V úlohe (2.43) predpokladáme symetrickosť matíc C a A_i. Ak by C nebola symetrická, bolo by možné maticu C nahradiť maticou $\frac{1}{2}(C + C^T)$, ktorá už symetrická je.

2.6 Programovanie nad kužeľmi druhého rádu

Programovanie nad kužeľmi druhého rádu (angl.: *Second Order Cone Programming*) – SOCP) je trieda úloh konvexnej optimalizácie, v ktorých minimalizujeme lineárnu funkciu cez prienik polyédrickej množiny a kužeľov druhého rádu (Alizadeh a Goldfarb, 2003; Lobo et al., 1998). Úlohu SOCP môžeme formulovať v tvare

$$\min_{x} \quad c^T x \tag{2.45a}$$

$$\text{v.n.} \quad Ax = b, \tag{2.45b}$$

$$\|\overline{x}\| \leq x_0, \tag{2.45c}$$

kde $c \in \mathbb{R}^n, b \in \mathbb{R}^m, A \in \mathbb{R}^{m \times n}, x = (x_0; \overline{x}) \in \mathbb{R}^n$, pričom $x_0 \in \mathbb{R}, \overline{x} \in \mathbb{R}^{n-1}$, a ohraničenie $\|\overline{x}\| \leq x_0$ je ohraničením kužeľa druhého rádu.

Kužeľ druhého rádu s dimenziou n definujeme nasledovne

$$\mathcal{Q}_n = \{x = (x_0; \overline{x}) \in \mathbb{R}^n : x_0 \geq \|\overline{x}\|\}, \tag{2.46}$$

kde $\|.\|$ je štandardná Euklidovská norma, $x_0 \in \mathbb{R}$, $\overline{x} = (x_1, x_2, ..., x_{n-1})^T \in \mathbb{R}^{n-1}$ a n je dimenzia kužeľa $\mathcal{Q}_n$. V prípade $n = 1$ nám ostane z kužeľa druhého rádu iba polpriamka ležiaca na osi x_0 a začínajúca v bode $x_0 = 0$.

2.7 Softvérové riešenie optimalizačných problémov

V súčasnosti je vyvinutých množstvo špecializovaných programových balíkov (angl.: *solver*) na riešenie optimalizačných problémov a to aj s veľkým počtom rozhodovacích premenných a ohraničení. Programové balíky máme voľne dostupné, alebo komerčné a medzi najznámejšie patria:

- voľné
  - **GLPK** - (GNU Linear Programming Kit) je jeden z najznámejších softvérových balíkov určený predovšetkým na riešenie lineárneho programovania, ale aj na zmiešaného celočíselného programovania. Je to množina procedúr organizovaná do používateľnej knižnice a napisaná v jazyku ANSI C. Na riešenie lineárnych úloh používa simplexovú metódu a metódu vnútorného bodu. Metódu vetvenia a hraníc spolu s metódou Gomoryho rezy používa na riešenie úloh zmiešaného celočíselného programovania (Makhorin, 2008).
  - **PENLAB** - je softvérový balík určený na riešenie nelineárnych semidefinitných úloh. Pozornosť pri tomto balíku bola venovaná skôr prehľadnosti kódu ako jeho efektívnosti. Cieľom bolo, aby bol balík ľahko pochopiteľný pre používateľov, ktorých má zároveň motivovať k jeho neustálemu vylepšovaniu. Ja napísaný v MATLABE a jeho učel je nielen vedecký ale aj výučbový (Fiala et al., 2013).
  - **SDPT3** - zameriava sa na riešenie úloh semidefinitného programovania a programovania nad kužeľom druhého rádu. Naprogramovaný je v MATLABE, ale niektoré časti sú uložené v C alebo vo FORTRANE. Dôvodom bola neefektívnosť niektorých operácii v MATLABE ako napríklad extrahovanie vybraných elementov z matice. Numerické výsledky ukazujú, že používané algoritmy sú pomerne efektívne a robustné pri problémoch s rozmermi rádovo sto premenných (Toh et al., 1999).
  - **SEDUMI** - je takisto určený na riešenie úloh semidefinitného programovania a programovania nad kužeľom druhého rádu. Na riešenie používa metódu vnútorného bodu (Sturm, 1999).
- komerčné

– **GUROBI** - je komerčný softvér na riešenie úloh lineárneho programovania, kvadratického programovania ale aj zmiešaného celočíselného programovania a programovania nad kužeľom druhého rádu. Podporuje množstvo programovacích ale aj modelovacích jazykov. Na riešenie používa simplexovú metódu (LP), bariérové algoritmy (QP), metódu vetvenia a rezov (MIP). Okrem štandardných metód používa na riešenie LP a MIP aj tzv. súbežný optimalizovač (angl.: *Concurrent Optimizer*), ktorý využíva efektívne viac jadrový procesor. Výpočtové metódy používa rovnaké ako GLPK, avšak paralelne spúšťa niekoľko rôznych stratégii na výpočet naraz, čím dosahuje výpočet vo výrazne kratšom čase (Gurobi Optimization, 2013).

– **CPLEX** - je ďalší z rady komerčných softvérových balíkov určený predovšetkým na riešenie lineárnych optimalizačných problémov. Okrem štandardných metód, ktoré využíva GUROBI, používa CPLEX aj preosievací algoritmus (ILOG, 2007).

– **MOSEK** - sa vyznačuje najmä používaním metódy vnútorného bodu, ktorá dokáže efektívne riešiť predovšetkým úlohy lineárneho programovania, programovania nad kužeľom druhého rádu a semidefinitného programovania. Osobitnou črtou optimalizovača je jeho založenie na tzv. homogénnom modeli, čo znamená, že dokáže spoľahlivo detekovať primárny alebo duálny neriešiteľný stav (Mosek, 2010).

Tabuľka 2.3: Zameranie softvérových balíkov na jednotlivé optimalizačné úlohy

	LP	QP	MILP	SDP	SOCP
GLPK	•	-	•	-	-
PENLAB	-	-	-	•	-
SDPT3	-	-	-	•	•
SEDUMI	-	-	-	•	•
GUROBI	•	•	•	-	•
CPLEX	•	•	•	-	•
MOSEK	•	•	•	•	•

Tabuľka 2.3 zobrazuje prehľadne zameranie jednotlivých spomínaných programových balíkov. Predovšetkým voľne dostupné balíky sú väčšinou úzko zamerané a preto je dobré zvoliť správny výber podľa typu optimalizačného problému. V prípade, že máme k dispozícii komerčný balík, postačí nám na riešenie väčšiny optimalizačných úloh.

2.8 Zhrnutie

V kapitole 2 sme sa zaoberali prehľadom základných optimalizačných úloh. Vyjadrili sme si všeobecný optimalizačný problém a povedali sme si kedy hovoríme o konvexnom optimalizačnom probléme.

Do prehľadu optimalizačných úloh sme si zaradili v prvom rade lineárne programovanie, kde sme si ukázali niekoľko spôsobov ako zapísať úlohu LP. Okrem štandardného tvaru v kapitole 2.2.2 sme si ukázali ako riešiť úlohu LP prostredníctvom simplexovej metódy. Obdobným spôsobom sme si v kapitole 2.3 uviedli metódu aktívnych množín, prostredníctvom ktorej riešime úlohu kvadratického programovania. V rámci kapitoly 2.4 sme rozobrali aj úlohou zmiešaného celočíselného programovania. Z metód riešenia sme sa zaoberali najmä metódou vetvenia a rezov. V rámci tejto kapitoly sme sa zaoberali aj s jednou z najviac študovaných optimalizačných úloh - problém obchodného cestujúceho. V kapitole 2.5 sme si zadefinovali úlohu semidefinitného programovania a v kapitole 2.6 sme dôkladne rozobrali vlastnosti kužeľov druhého rádu ako aj definovali úlohu programovania nad kužeľmi druhého rádu.

LP a QP využijeme v kapitole 3 v prediktívnom riadení, MILP a SOCP využijeme v kapitole 4 na získanie optimálneho poradia stanovíšť pri hľadaní trasy pre heterogénne vozidlo.

Kapitola 3

Prediktívne riadenie systémov s ohraničeniami

V tejto kapitole si preberieme základné princípy prediktívneho riadenia (angl.: *Model Predictive Control - MPC*) zariadenia, ktoré je potrebné riadiť tak, aby boli dodržané všetky procesné ohraničenia. Základná myšlienka v MPC je využitie modelu procesu na predpovedanie jeho budúceho správania a súčasnú optimalizáciu akčných zásahov. Veľkou výhodou optimálneho riadenia je, že ekonomické kritéria a obmedzenia procesu sú zapúzdrené priamo vo formulácii optimalizačného problému riadenia vedúceho k bezpečnému a efektívnemu riadeniu procesu.

3.1 Pozadie MPC

O MPC by sme mohli uvažovať aj ako o metóde, ktorá odráža do určitej miery ľudské správanie. V každom momente sa rozhodneme akú akciu vykonáme, pričom sa snažíme činnosť viesť k najlepším predpokladaným výkonom. Či už hovoríme o človeku alebo o procese riadenia, v každom takomto momente sa rozhodujeme len na základe dostupných informácií, ktoré máme. Preto ak chceme vykonať tento výber, potrebujeme vedieť, čo očakávame od procesu riadenia a podľa toho zvoliť interný model, ktorý nám pomôže spraviť správne rozhodnutie. Po každej riadiacej akcii aktualizujeme naše poznatky, sledujeme zmeny okolia na riadiacu činnosť a robíme ná základe toho ďalšie rozhodnutia. V bodoch by sme vedeli povedať, že zákon prediktívneho riadenia má nasledujúce zložky (Rossiter, 2003):

- zákon riadenia, ktorý závisí na predpokladanom správaní
- výstupné predpovede, ktoré sú počítané pomocou procesného modelu

- optimálne akčné zásahy, ktoré sú určené optimalizáciou určitej miery predpokladanej výkonnosti

- posuvný horizont, ktorý hovorí o tom, že riadiaci vstup je aktualizovaný v každej perióde vzorkovania

Aj keď sme vyššie dali do popredia riadenie s predikciou budúcnosti, nie všetky riadiace zákony ju musia brať do úvahy. Napríklad PID (proporcionálne-integračne-derivačný) regulátor berie do úvahy predikciu len vo veľmi obmedzenej miere. Napriek tomu je široko používaný v priemyselných riadiacich systémoch, stal sa základným prvkom skorých riadiacich systémov a štandardným nástrojom pri vzniku procesného riadenia v roku 1940 (Åström a Hägglund, 2006; Kozák a Huba, 2003). MPC má oproti nemu ale dve veľké výhody. Počíta s predikciou a aj s ohraničeniami. Tie sú v praxi nevyhnutné na dodržanie bezpečnosti a výkonnosti. To znamená, že bez splnenia ohraničení, predikcie a optimalizácie nemôžeme hovoriť o bezpečnosti, výkonnosti a stabilite v procese riadenia.

Model v MPC, ako riadiacej stratégii, je nevyhnutný na predpovedanie budúceho správania procesu. Model nám definuje ako sa proces chová a ukazuje závislosť výstupu od aktuálne nameraného stavu a súčasných/budúcich vstupoch. V tejto práci sa budeme zaoberať predovšetkým lineárnymi modelmi, ale v skutočnosti tento model môže byť aj nelineárny (Fontes a Vinter, 2000; Magni et al., 2009). V praxi väčšina MPC algoritmov používa lineárne modely, takže závislosť predpovedí na budúcom riadení je potom takisto lineárna. Nelineárny model je užitočný v prípadoch, kedy lineárne aproximácie nie sú dostatočne presné. Dodať však treba, že pri nelineárnych modeloch musíme rátať aj s podstatne vyššími výpočtovými nárokmi a preto ho reálne vieme využiť iba v prípadoch keď nie sme v tomto smere nijako obmedzení. V prediktívnom riadení je použitý model iba na výpočet predikcie výstupu systému. Podľa potreby by sme mali preto zvážiť zložitosť modelu, tak aby bol čo najjednoduchší a zároveň dával dostatočne presné predpovede.

Vo fáze rozhodovania sa, ktorá akcia bude pre nás v danej chvíli najlepšia je dôležité stanoviť kritériá, podľa ktorých to budeme vedieť zistiť. Pre výber je nevyhnutné mať číselnú definíciu, ktorá oceňuje hodnotu vstupu tak, aby boli čo najnižšie *náklady*. Na toto nám slúži účelová funkcia. Jej výber a definovanie je netriviálna záležitosť, ktorá je dodnes výskumnou oblasťou inžinierstva a teoretického posúdenia (Rodriguez a Cortes, 2012). Voľba účelovej funkcie má často nemalý vplyv na kvalitu riadenia. Hlavnou požiadavkou je, aby pri sme čo najmenšej cene dosiahli správnu kvalitu riadenia. Predpokladané vstupy sú vybrané tak, aby minimalizovali danú funkciu. Účelová funkcia musí byť pritom natoľko jednoduchá, aby plnila požadovanú účinnosť.

Každé rozhodnutie a teda výber ďalšej akcie do procesu vychádza z dostupných informácii, ktoré máme z aktuálneho stavu zariadenia alebo prostredia. Tento náš limit sa

každým krokom rozširuje o ďalšie poznatky a mohli by sme o ňom hovoriť ako o limite nášho videnia do budúcnosti. Ako sa pohybujeme vpred, naša hranica videnia sa posúva s nami a tento jav sa nazýva posuvný horizont. Neustále vyzdvihujeme nové informácie z obzoru a táto informácia sa používa pre aktualizáciu riadiacich akcií. Tento jav nám predstavuje predikované správanie s ohraničeným výhľadom do budúcnosti. S príchodom nových informácií sa vstup automaticky upravuje tak, aby zohľadnil nové prijaté informácie. Pre nás by bolo ideálne, keby sme toto dokázali robiť od súčasnosti až do nekonečna. To však v praxi nie je možné a preto sa snažíme nájsť čo najvhodnejší dosah nášho videnia, tak aby sme vedeli urobiť čo najlepšie rozhodnutie a aby to predstavovalo pre nás čo najmenšie náklady. Toto je tiež oblasť teoretického posúdenia a výskumu v MPC. Horizont pre predikciu musí byť totiž vybraný tak, aby boli dodržané všetky potrebné podmienky dynamického riadenia.

Keď hovoríme o bezpečnej prevádzke riadenia, je nutné hovoriť aj o jeho stabilite. Už v minulosti bola venovaná veľká pozornosť prediktívnym zákonom riadenia na zaistenie stability, pričom je dôležité určiť špecifikáciu požadovaného výkonu. Ak zvolíme správnu konfiguráciu, MPC bude vždy garantovať stabilitu uzavretej slučky. Preto nastáva otázka na zamyslenie, ako dostať rovnováhu medzi výkonom, dobrou citlivosťou a tiež rovnováhou medzi vstupnou aktivitou a rýchlosťou reakcie. Toto všetko nám rieši použitie váhových matíc. To znamená, že dávame rôzny dôraz na výkon v rôznych časových krokoch riadenia podľa významu. Zovšeobecnenie hodnôt pre určenie váh je náročné, pretože každý proces má iné priority. Existuje ale aj možnosť, že sa nám podarí definovať relatívne jednotnú dôležitosť výkonu a vtedy je ladenie oveľa jednoduchšie.

Dôležitou výhodou v MPC je jeho schopnosť systematickým spôsobom ošetriť ohraničenia. Ohraničenia stability a výkonu sú zahrnuté vo formulácii optimalizačného problému a na základe nich algoritmus MPC zariadi optimalizáciu predikovaného výkonu. Podrobnosti o tom, ako sú zahrnuté obmedzenia do systému sú veľmi závislé na nasadenom algoritme. V každom prípade sa ale MPC snaží pri všetkých obmedzeniach o dosiahnutie čo najväčšieho výkonu.

MPC systematicky narába s prijímanými informáciami, čo umožňuje zavčasu zareagovať na očakávanú situáciu. Napríklad pri PID sa v prípade nepriaznivej situácie snaží systém čo najlepšie zareagovať v danej chvíli, ale pri MPC sa vieme pripraviť na situáciu skôr. Takto vieme s väčším úspechom zabezpečiť bezpečnú prevádzku.

3.2 Predikčné modely a ohraničenia

Uvažujeme systém, ktorého dynamické správanie sa je dané aktualizačnou rovnicou stavu v diskrétnom čase:

$$x(t+1) = f(x(t), u(t)) \tag{3.1}$$

kde $x(t) \in \mathbb{R}^n$ je stavový vektor v čase $t \in \mathbb{N}_+$, pričom $\mathbb{N}_+$ je množina celých kladných čísel, $x(t+1)$ predstavuje stav v nasledujúcom kroku, $u(t) \in \mathbb{R}^m$ je vektor akčných zásahov a $f(\cdot)$ je funkcia aktualizovania stavu. Predpokladá sa, že stavy a vstupy podliehajú nasledovným obmedzeniam:

$$x(t) \in \mathbb{X} \subseteq \mathbb{R}^n, \tag{3.2}$$

$$u(t) \in \mathbb{U} \subseteq \mathbb{R}^m. \tag{3.3}$$

Pre obe obmedzenia platí, že chceme aby boli splnené $\forall t \geq 0$. Predpokladá sa, že hodnota $x(t)$ je dostupná v akomkoľvek čase t. V najbežnejších prípadoch môže byť funkcia aktualizujúca stav $f(x(t), u(t))$ ľubovoľného typu – lineárna, po častiach afínna, nelineárna, nespojitá, atď. My sa ale budeme zaoberať iba prípadom, kedy je stavová funkcia lineárna.

3.2.1 Stavové modely

Lineárne systémy sú definované v oblasti diskrétneho času ich stavovou reprezentáciou, takže môžeme funkciu aktualizácie stavu prepísať nasledovne:

$$f(x(t), u(t)) = Ax(t) + Bu(t), \tag{3.4}$$

kde $A \in \mathbb{R}^{n \times n}$ a $B \in \mathbb{R}^{n \times m}$ predstavujú aktualizačné matice stavu, ktorých hodnoty nezávisia od času t. Optimalizačný problém sa v lineárnych systémoch rieši relatívne jednoduchšie ako v prípade nelineárnych a z tohto dôvodu sa zvyknú nelineárne funkcie $f(x, u)$ linearizovať (Rau a Schroder, 2002)).

Stavové modely môžeme definovať nasledovne:

$$\underbrace{\begin{bmatrix} x_1(t+1) \\ x_2(t+1) \\ \vdots \\ x_n(t+1) \end{bmatrix}}_{x_{t+1}} = \underbrace{\begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n,1} & a_{n,2} & \dots & a_{n,n} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix}}_{x_t} + \underbrace{\begin{bmatrix} b_{1,1} & b_{1,2} & \dots & b_{1,m} \\ b_{2,1} & b_{2,2} & \dots & b_{2,m} \\ \vdots & \vdots & \vdots & \vdots \\ b_{n,1} & b_{n,2} & \dots & b_{n,m} \end{bmatrix}}_{B} \underbrace{\begin{bmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_m(t) \end{bmatrix}}_{u_t} \tag{3.5}$$

$$\underbrace{\begin{bmatrix} y_1(t) \\ y_2(t) \\ \vdots \\ y_n(t) \end{bmatrix}}_{y_t} = \underbrace{\begin{bmatrix} c_{1,1} & c_{1,2} & \dots & c_{1,n} \\ c_{2,1} & c_{2,2} & \dots & c_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ c_{l,1} & c_{l,2} & \dots & c_{l,n} \end{bmatrix}}_{C} \underbrace{\begin{bmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_n(t) \end{bmatrix}}_{x_t} + \underbrace{\begin{bmatrix} d_{1,1} & d_{1,2} & \dots & d_{1,m} \\ d_{2,1} & d_{2,2} & \dots & d_{2,m} \\ \vdots & \vdots & \vdots & \vdots \\ d_{l,1} & d_{l,2} & \dots & d_{l,m} \end{bmatrix}}_{D} \underbrace{\begin{bmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_m(t) \end{bmatrix}}_{u_t} \tag{3.6}$$

V skrátenej forme by sme mohli napísať:

$$x_{t+1} = Ax_t + Bu_t, \tag{3.7a}$$

$$y_t = Cx_t + Du_t \tag{3.7b}$$

kde $y \in \mathbb{R}^l$ je vektor výstupov z procesu a matice A, B, C, D definujú samotný stavový model.

Veľkou výhodou MPC je, že vie pracovať bežne aj so systémami, ktoré majú veľa vstupov a výstupov (angl.: *Multiple Input Multiple Output – MIMO*).

3.3 Ohraničené optimálne riadenie na konečnom horizonte

V tejto kapitole si opíšeme základy ohraničeného optimálneho riadenia na konečnom horizonte. Ďalej o ňom budeme hovoriť len ako o MPC probléme. Predpokladáme, že máme model riadeného procesu v tvare systému (3.1), kde je hodnota nameraného stavu $x_0 = x(t)$ v čase t. Predikovaná hodnota stavu x_k v čase $x(t+k)$ a riadiacej akcie u_k v čase $u(t+k)$ je daná počiatočným stavom $x(t)$. Predikcia je realizovaná cez $k = 0, \dots N$, kde N nazývame predikčný horizont, pre ktorý platí $N \in \mathbb{N}_+ < \infty$. Potom problém ohraničeného riadenia v konečnom čase je definovaný ako:

$$J_N^*(x_0) = \min_{U_N} \quad \ell_N(x_N) + \sum_{k=0}^{N-1} \ell(x_k, u_k) \tag{3.8a}$$

$$\text{v.n.} \quad x_{k+1} = f(x_k, u_k), \tag{3.8b}$$

$$x_k \in \mathbb{X}, \quad k = 0, \dots, N-1, \tag{3.8c}$$

$$u_k \in \mathbb{U}, \quad k = 0, \dots, N-1, \tag{3.8d}$$

$$x_N \in \tau_{\text{set}}, \tag{3.8e}$$

kde $J_N^*(x_0)$ predstavuje optimálnu hodnotu účelovej funkcie s počiatočným stavom x_0,

$$U_N = [u_0^T, \dots, u_{N-1}^T]^T \in \mathbb{R}^{Nm} \tag{3.9}$$

je celá sekvencia vstupov optimálneho riadenia, x_N je konečný predikovaný stav, $\ell_N(x_N, u_N)$ je terminálna penalizácia, $\ell(x_k, u_k)$ je priebežná penalizácia v kroku k a τ_{set} reprezentuje množinu konečných ohraničení, ktorá býva často použitá pre pridanie určitých vlastností ako napríklad stabilita a ohraničenia potrebné behom celého priebehu riadenia.

Definícia 3.3.1 (Prípustná množina $\mathbb{X}_N$)**.** Množina počiatočných stavov x_0, pre ktoré je možné riešiť problém optimálneho riadenia definujeme ako N-krokovú množinu $\mathbb{X}_N \subseteq R^n$:

$$\mathbb{X}_N = \{x_0 \in \mathbb{R}^n \mid \exists U_N \text{ také, že platí (3.8b)–(3.8e)}\}, \tag{3.10}$$

kde U_N je definované v (3.9).

Cieľom je vyriešiť MPC problém definovaný v (3.8) ako optimalizačný problém pre akékoľvek $x_0 \in \mathbb{X}_N$, čiže nájsť optimálnu sekvenciu vstupov U_N^* takú, aby boli splnené ohraničenia a zminimalizované kritéria výkonnosti funkcie (3.8)). Jednotlivé časti optimalizovanej účelovej funkcie závisia od konkrétneho cieľa riadenia. Zvyčajne sa v praxi zvažujú nasledovné nastavenia (Mikleš a Fikar, 2007):

- *Regulačné problémy* – cieľom je riadenie systému tak, aby vzdialenosť x_k od počiatku súradnicovej sústavy bola minimálna a popritom sme boli schopní zminimalizovať aj náklady na riadenie. Naviac tiež penalizujeme aj vzialenosť u_k od počiatku (teda od nulovej hodnoty). Tieto požiadavky vieme formálne zapísať nasledovne:

$$\ell_N(x_N) = \|Q_N x_N\|_r, \tag{3.11a}$$
$$\ell(x_k, u_k) = \|Q_{\text{x}} x_k\|_r + \|Q_{\text{u}} u_k\|_r, \tag{3.11b}$$

  kde $Q_N \in \mathbb{R}^{n\times n}, Q_{\text{x}} \in \mathbb{R}^{n\times n}$, a $Q_{\text{u}} \in \mathbb{R}^{m\times m}$ sú váhové matice používané na ladenie výkonnosti. Pritom $\|\cdot\|_r$ reprezentuje štadardnú penalizáciu v tvare r-normy, čiže $\|P_z\|_1 = \sum_i |P_i z_i|$, pod $\|z\|_2$ s miernym zneužitím notácie rozumieme $z^T P z$ a $\|P_z\|_\infty = \max_i |P_i z_i|$ pre ľubovoľný vektor z a maticu P.

- *Sledovacie problémy* – cieľom je riadenie systému tak, aby rozdiel medzi predikovaným stavom x_k a predpísaným referenčným bodom x_{ref} bol minimálny a popritom sa snažíme zminimalizovať aj zvýšené náklady na riadenie.

$$\ell_N(x_N) = \|Q_N(x_N - x_{\text{ref}})\|_r, \tag{3.12a}$$
$$\ell(x_k, u_k) = \|Q_{\text{x}}(x_k - x_{\text{ref}})\|_r + \|Q_{\text{u}} \Delta u_k\|_r, \tag{3.12b}$$

  kde $\Delta u_k = u_k - u_{k-1}$. Minimalizácia inkrementu riadenia je v tomto prípade použitá na dosiahnutie integračných vlastností, a tým pádom k odstráneniu trvalej regulačnej odchýlky.

- *Minimálny čas riadenia* – chceme riadiť stavy systému z x_0 do cieľovej množiny za najmenší počet krokov.

Predtým, ako si povieme, ako sa MPC problém rieši priamo pre systémy v tvare (3.1), si vyjadríme potrebný predpoklad.

Predpoklad 3.3.2. Množiny $\mathbb{X}, \mathbb{U}$ a τ_{set} sú všetky kompaktné (uzavreté a ohraničené), konvexné v tvare polytopov a účelová funkcia je zložená z členov v tvare r-normy. Naviac uvažujeme, že systém $x_{k+1} = f(x_k, u_k)$ je riaditeľný a že v každom časovom kroku poznáme stav systému $x(t)$.

Predpoklad 3.3.2 hovorí aj o tom, že $\mathbb{X}$ môže byť prepísaný do tvaru $\{x \mid H_{\text{x}}x \leq K_{\text{x}}\}$. Podobne máme $\mathbb{U} = \{u \mid H_{\text{u}}u \leq K_{\text{u}}\}$ a $\tau_{\text{set}} = \{x \mid Lx \leq M\}$.

3.3.1 Ohraničené optimálne riadenie v konečnom čase pre lineárne systémy

Ak je predikčný model (3.8) daný lineárnou stavovou reprezentáciou ako $x_{k+1} = Ax_k + Bu_k$ a sú splnené všetky podmienky Predpokladu 3.3.2, potom môže byť MPC problém formulovaný buď ako lineárny alebo ako kvadratický program, závislý od r-normy v (3.11).

Lineárna účelová funkcia

Uvažujme predikčný model (3.8) závislý od r-normy, kde index p je z množiny $\{1, \infty\}$. Potom máme MPC problém v nasledujúcom tvare:

$$J_N^*(x_0) = \min_{U_N} \quad \|Q_N x_N\| + \sum_{k=0}^{N-1} (\|Q_{\text{x}} x_k\|_r + \|Q_{\text{u}} u_k\|_r) \tag{3.13a}$$

$$\text{v.n.} \quad x_0 = x(t), \tag{3.13b}$$

$$x_{k+1} = Ax_k + Bu_k, \tag{3.13c}$$

$$x_k \in \mathbb{X}, \quad k = 0, \ldots, N-1, \tag{3.13d}$$

$$u_k \in \mathbb{U}, \quad k = 0, \ldots, N-1, \tag{3.13e}$$

$$x_N \in \tau_{\text{set}}. \tag{3.13f}$$

Keďže funkcie $\|\cdot\|_{\{1,\infty\}}$ nie sú lineárne, musíme spraviť niekoľko úprav, aby sme dosiahli lineárnu účelovú funkciu. Vzťah $\min \|z\|_1$ so $z = [z_1, \ldots, z_n]^T$ vieme prepísať do ekvivalentného tvaru $\min \sum_i \varepsilon_i$ vzhľadom na $\varepsilon_i \geq z_1, \varepsilon_1 \geq -z_1, \ldots, \varepsilon_n \geq z_n$ a $\varepsilon_n \geq -z_n$ s použitím doplnkových premenných $\varepsilon_i \in \mathbb{R}$, kde $i = 1, \ldots, n$.

Obdobným spôsobom vieme vzťah $\min \|z\|_\infty$ prepísať takisto do tvaru $\min \varepsilon_i$, za podmienok $\varepsilon \geq z_1, \varepsilon \geq -z_1, \ldots, \varepsilon \geq z_n$ a $\varepsilon \geq -z_n$. V tomto prípade nám postačí jedna doplnková premenná ε.

Substitúciou $x_k = A^k x_0 + \sum_{j=0}^{k-1} A^k B u_{k-1-j}$ a pri uvažovaní $r = \infty$ v (3.13a) môžeme naformulovať MPC problém ako lineárny program v tvare:

$$J_N^*(x_0) = \min_{\substack{U_N, \varepsilon_0, \ldots, \varepsilon_{N-1}, \\ \delta_0, \ldots, \delta_{N-1}, \gamma}} \sum_{k=0}^{N-1} (\varepsilon_k + \delta_k) + \gamma \tag{3.14a}$$

$$\text{v.n.} \quad x_{k+1} = Ax_k + Bu_k, \tag{3.14b}$$

$$H_{\mathrm{x}} x_k \leq K_{\mathrm{x}}, \quad k = 0, \ldots, N-1, \tag{3.14c}$$

$$H_{\mathrm{u}} u_k \leq K_{\mathrm{u}}, \quad k = 0, \ldots, N-1, \tag{3.14d}$$

$$Lx_N \leq M. \tag{3.14e}$$

$$Q_{\mathrm{u}} u_k \leq \mathbf{1}\varepsilon_k, -Q_{\mathrm{u}} u_k \leq \mathbf{1}\varepsilon_k, \quad k = 0, \ldots, N-1, \tag{3.14f}$$

$$Q_{\mathrm{x}} x_k \leq \mathbf{1}\delta_k, -Q_{\mathrm{x}} x_k \leq \mathbf{1}\delta_k, \quad k = 0, \ldots, N-1, \tag{3.14g}$$

$$Q_N x_N \leq \mathbf{1}\gamma, -Q_N x_N \leq \mathbf{1}\gamma, \tag{3.14h}$$

kde ohraničenia (3.14f) až (3.14h) sú použité pre opis účelovej funkcie s tým, že $\mathbf{1}$ je vektor jednotiek príslušných rozmerov, ε_k modeluje $\|Q_{\mathrm{u}} u_k\|_\infty$, δ_k vyjadruje $\|Q_{\mathrm{x}} x_k\|_\infty$ a γ definuje $\|Q_N x_N\|_\infty$.

Podobným spôsobom vieme vyjadriť MPC problém v prípade použitia 1-normy.

$$J_N^*(x_0) = \min_{\substack{U_N, \varepsilon_0, \ldots, \varepsilon_{N-1}, \\ \delta_0, \ldots, \delta_{N-1}, \gamma}} \sum_{k=0}^{N-1} (\mathbf{1}^T \varepsilon_k + \mathbf{1}^T \delta_k) + \mathbf{1}^T \gamma \tag{3.15a}$$

$$\text{v.n.} \quad x_{k+1} = Ax_k + Bu_k, \tag{3.15b}$$

$$H_{\mathrm{x}} x_k \leq K_{\mathrm{x}}, \quad k = 0, \ldots, N-1, \tag{3.15c}$$

$$H_{\mathrm{u}} u_k \leq K_{\mathrm{u}}, \quad k = 0, \ldots, N-1, \tag{3.15d}$$

$$Lx_N \leq M. \tag{3.15e}$$

$$Q_{\mathrm{u}} u_k \leq \varepsilon_k, -Q_{\mathrm{u}} u_k \leq \varepsilon_k, \quad k = 0, \ldots, N-1, \tag{3.15f}$$

$$Q_{\mathrm{x}} x_k \leq \delta_k, -Q_{\mathrm{x}} x_k \leq \delta_k, \quad k = 0, \ldots, N-1, \tag{3.15g}$$

$$Q_N x_N \leq \gamma, -Q_N x_N \leq \gamma, \tag{3.15h}$$

kde ohraničenia (3.15f) až (3.15h) sú použité pre opis účelovej funkcie s tým, že ε_k, δ_k, γ sú vektory príslušných rozmerov, pričom ε_k modeluje $\|Q_{\mathrm{u}} u_k\|_1$, δ_k vyjadruje $\|Q_{\mathrm{x}} x_k\|_1$ a γ definuje $\|Q_N x_N\|_1$. Treba si uvedomiť, že ε_k a γ sú vo vzťahu (3.14) skalárne premenné, kým v (3.15) sú vektory.

Kvadratická účelová funkcia

Ak predpokladáme, že $r = 2$, čiže $l_N(x_N) = x_N^T Q_N x_N$ a $l_N(x_k, u_k) = x_k^T Q_x x_k + u_k^T Q_u u_k$, MPC problém (3.8) bude mať tvar:

$$J_N^*(x_0) = \min_{U_N} x_N^T Q_N x_N + \sum_{k=0}^{N-1} x_k^T Q_x x_k + x_k^T Q_u u_k, \tag{3.16a}$$

$$\text{v.n.} \quad x_{k+1} = Ax_k + Bu_k, \tag{3.16b}$$

$$H_{\mathrm{x}} x_k \leq K_{\mathrm{x}}, \quad k = 0, \ldots, N-1, \tag{3.16c}$$

$$H_{\mathrm{u}} u_k \leq K_{\mathrm{u}}, \quad k = 0, \ldots, N-1, \tag{3.16d}$$

$$Lx_N \leq M. \tag{3.16e}$$

Túto formuláciu vieme prepísať do štandardnej QP formy (2.32) dvomi spôsobmi.

V prvom prípade môžeme uvažovať o premenných $u_0, \ldots u_{N-1}$ rovnako ako o $x_1, \ldots x_N$ ako o optimalizačných premenných, čiže:

$$V_N = \left[U_N^T, X_N^T\right]^T \in \mathbb{R}^{N(m+n)}, \tag{3.17}$$

kde

$$U_N = \left[u_0^T, \ldots, u_{N-1}^T\right]^T \in \mathbb{R}^{Nm}, \tag{3.18a}$$

$$X_N = \left[x_1^T, \ldots, x_N^T\right]^T \in \mathbb{R}^{Nn}. \tag{3.18b}$$

Prepojenie medzi x_k, u_k a x_{k+1} reprezentované vzťahom (3.16b) je potom zabezpečené rovnostným ohraničením v tvare $G_{\mathrm{eq}} V_N = W_{\mathrm{eq}} + E_{\mathrm{eq}} x_0$. Ohraničenia (3.16c) – (3.16e) potom prevedieme do tvaru $GV_N \leq W + Ex_0$. Nakoniec môže byť účelová funkcia (3.16a) reprezentovaná ako $V_N^T \tilde{Q} V_N$ s $\tilde{Q} = \mathrm{diag}(Q_u, \ldots, Q_u, Q_x, \ldots, Q_x, Q_N)$.

Takže problém definovaný v (3.16) vieme riešiť ako QP problém:

$$J_N^*(x_0) = \min_{V_N} V_N^T \tilde{Q} V_N, \tag{3.19a}$$

$$\text{v.n.} \quad GV_N \leq W + Ex_0, \tag{3.19b}$$

$$G_{\mathrm{eq}} V_N = W_{\mathrm{eq}} + E_{\mathrm{eq}} x_0, \tag{3.19c}$$

Pre ilustráciu si uvedieme, aký tvar by mohli mať matice $G, W, E, G_{\mathrm{eq}}, W_{\mathrm{eq}}, E_{\mathrm{eq}}$ v prípade, ak $N = 3$. Potom budeme mať:

$$\underbrace{\begin{bmatrix} H_{\mathrm{u}} & 0 & 0 & 0 & 0 & 0 \\ 0 & H_{\mathrm{u}} & 0 & 0 & 0 & 0 \\ 0 & 0 & H_{\mathrm{u}} & 0 & 0 & 0 \\ 0 & 0 & 0 & H_{\mathrm{u}} & 0 & 0 \\ 0 & 0 & 0 & 0 & H_{\mathrm{u}} & 0 \\ 0 & 0 & 0 & 0 & 0 & L \end{bmatrix}}_{G} \underbrace{\begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{V_N} \leq \underbrace{\begin{bmatrix} K_{\mathrm{u}} \\ K_{\mathrm{u}} \\ K_{\mathrm{u}} \\ K_{\mathrm{x}} \\ K_{\mathrm{x}} \\ M \end{bmatrix}}_{W} + \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}}_{E} x_0, \tag{3.20}$$

$$\underbrace{\begin{bmatrix} -B & 0 & 0 & I & 0 & 0 \\ 0 & -B & 0 & -A & I & 0 \\ 0 & 0 & -B & 0 & -A & I \end{bmatrix}}_{G_{\text{eq}}} \underbrace{\begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{V_N} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}}_{W_{\text{eq}}} + \underbrace{\begin{bmatrix} A \\ 0 \\ 0 \end{bmatrix}}_{E_{\text{eq}}} x_0. \tag{3.21}$$

Druhý spôsob prevedenia je zámena rovnostného ohraničenia $x_{k+1} = Ax_k + Bu_k$ v (3.16b) s implicitnou reprezentáciou $x_k = A^k x_0 + \sum_{j=0}^{k-1} A^j B u_{k-1-j}$. Potom:

$$X_N = \tilde{A}x_0 + \tilde{B}U_N, \tag{3.22}$$

s

$$\tilde{A} = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}, \tilde{B} = \begin{bmatrix} B & 0 & \cdots & 0 \\ AB & B & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ A^{N-1}B & A^{N-2}B & \cdots & B \end{bmatrix}, \tag{3.23}$$

a problém vieme prepísať do tvaru:

$$J_N^*(x_0) = \quad x_0^T Y x_0 + \min_{U_N} \{U_N^T H U_N + x_0^T F U_N\}, \tag{3.24a}$$

$$\text{v.n.} \quad GU_N \leq W + Ex_0, \tag{3.24b}$$

s $H = \tilde{B}^T \tilde{Q} \tilde{B} + \tilde{Q}_u$, $F = \tilde{A}^T \tilde{Q} \tilde{A}$, kde $\tilde{Q} = \text{diag}(Q_x, \ldots Q_x, Q_n)$ a $\tilde{Q}_u = \text{diag}(Q_u, \ldots Q_u)$.

Ak je známa hodnota x_0, môžeme získať implicitnú reprezentáciu optimalizovača

$$U_N^* = \left[u_0^{*T}, u_1^{*T}, \ldots, u_{N-1}^{*T} \right]^T, \tag{3.25}$$

s riešením korešpondujúcim QP problému použitím príslušného softvéru ako bolo opísané v Kapitole 2.3.

3.4 Posuvný horizont

Na dosiahnutie najlepšej možnej miery výkonnosti a na zabezpečenie splnenia ohraničení v každom časovom kroku by bolo ideálne, ak by predikčný horizont N bol nekonečno. To však (až na parciálne výnimky, viď. Grieder et al. (2004)) vedie na optimalizačný problém s nekonečným počtom premenných, čo nie je realizovateľné. Preto v MPC uvažujeme konečný predikčný horizont. Našou úlohou je nájsť postupnosť vstupov optimálneho riadenia

U_N^* definovanú v (3.25), ktorá minimalizuje dané výkonnostné kritérium (3.8a), pričom berie do úvahy obmedzenia stavov systému a riadiacich vstupov (Kvasnica, 2009). MPC je zvyčajne implementované v uzavretej slučke a posuvnom horizonte. V tomto prípade je MPC problém formulovaný s počiatočnou podmienkou závislou od aktuálne dostupného stavu:

$$x_0 = x(t). \tag{3.26}$$

Ak by sme riešili MPC problém ako optimalizačný problém, optimálne riadenie U_N^* by sme získali ako postupnosť číselných hodnôt $u_0^*, \ldots, u_{N-1}^*$ ako je znázornené vo vzťahu (3.8). Následne, iba prvý prvok vektora U_N^*, teda u_0^* sa v tej chvíli implementuje ako riadiaci signál. V ďalšom kroku $t+1$ riešime problém (3.8) znovu, ale tento raz pre:

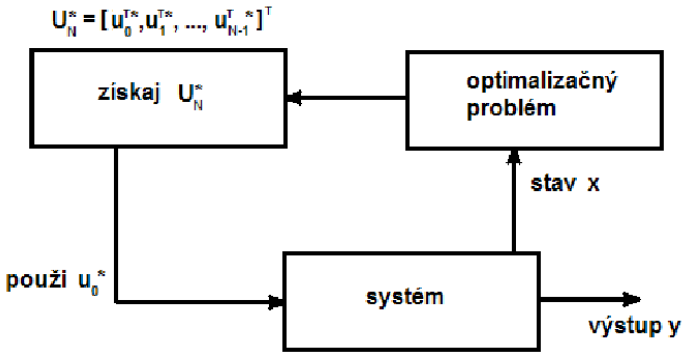
$$x_0 = x(t+1), \tag{3.27}$$

ku ktorému prislúcha nová sekvencia riadiach signálov a tento proces sa opakuje do nekonečna. Prípady, kde sa tento postup opakuje v každom kroku, nazývame MPC s posuvným horizontom.

Optimálne riešenie pre MPC problém definovaný v (3.8) môže byť určené v reálnom čase pomocou metód LP, alebo QP pre počiatočnú podmienku $x_0 = x(t)$. Pri všetkých prístupoch hrá dominantnú úlohu vektor optimálneho riadenia U_N^*, ktorý môže byť aplikovaný do systému v podobe otvorenej slučky na pohyb systémových stavov z $x(t)$ do $x(t+N)$. Ak $x(t+N)$ nie je rovný žiadanej hodnote, problém musí byť riešený znovu, pre novú hodnotu počiatočnej podmienky $x_0 = x(t+N+1)$, aby sme získali novú sekvenciu akcií riadenia. Avšak nie je isté, že po týchto krokoch sa stane nové riešenie automaticky uskutočniteľné. Na prekonanie tohto problému je možné rozšíriť predikčný horizont N na nekonečno, čo vedie k tzv. optimálnemu riadeniu v nekonečnom čase (Grieder et al., 2004). Hoci je tento spôsob veľmi nápomocný, jeho riešenie je často výpočtovo veľmi náročné. Preto sa sa stalo bežným javom priblížiť sa použitiu nekonečného horizontu s riešením MPC problému, používaným opakovane v každom kroku. Táto stratégia sa nazýva riadenie s posuvným horizontom (angl.: *Receding Horizon Control – RHC*). Ďalším dôvodom pre používanie RHC stratégie je fakt, že predikované správanie systému reprezentované predikčným modelom na základe vypočítanej optimálnej vstupnej trajektórie U_N^* sa zvyčajne líši od aktuálneho správania riadeného objektu. Takýmto spôsobom by sme v lepšom prípade nezískali optimálnu možnosť riadenia, avšak v horšom by mohlo dôjsť v praxi k poškodeniu kvôli porušeniu daných ohraničení. Pri RHC k takémuto problému nedochádza, nakoľko ponúka spätnú väzbu do riadiaceho systému.

Politika RHC je založená na riešení problému optimálneho riadenia v konečnom čase v každom časovom kroku. V ňom vždy získame novú vstupnú sekvenciu riadenia U_N^*. Toto môžeme riešiť pomocou príslušných metód LP, QP. Následne vezmeme ale iba prvý

element zo získanej sekvencie a ten aplikujeme do riadeného procesu. V ďalšom kroku je stav meraný znovu a procedúra sa opakuje od začiatku pre aktualizovanú hodnotu počiatočnej podmienky x_0, pozri Obr. 3.1. Žiaľ, práve nutnosť optimalizácie v každom kroku robí MPC ťažko použiteľným na riadenie rýchlych systémov, pripadne systémov, kde musíme použiť na implementáciu jednoduchý hardware. Z tohto dôvodu je dodnes predmetom výskumu vytváranie metód so zníženými nárokmi na spotrebu pamäte či času.



Obr. 3.1: RHC implementácia v reálnom čase

3.5 Zhrnutie

V kapitole 3 sme opísali základy prediktívneho riadenia (MPC) ako riadenia, ktorého základná myšlienka je využitie modelu procesu na predpovedanie jeho budúceho správania a súčasnú optimalizáciu akčných zásahov. Vyzdvihli sme jeho pozitíva a i slabšie stránky v porovnaní s napríklad PID regulátorom. V kapitole 3.2 sme si skrátene opísali lineárne modely systémov, pričom sme bližšiu pozornosť venovali stavovým modelom, ktoré budú pre nás zaujimavé ešte neskôr v práci. V kapitole 3.3 sme si opísali základy ohraničeného optimálneho riadenia v konečnom čase. Vzhľadom na cieľ riadenia sme sa zaoberali rozdelením problémov na regulačné, sledovacie, či problémy s minimálnym časom riadenia. Uviedli sme si problém prediktívneho riadenia s lineárnou a kvadratickou účelovou funkciou, ktorej použitie závisí napríklad od použitia normy pri určovaní vzdialenosti. V záverečnej časti sme sa venovali posuvnému horizontu prediktívneho riadenia, ktorý by bol síce ideálny, keby bol nekonečný, no prakticky na nájdenie optimálnej postupnosti krokov riadenia sme schopní pracovať iba s konečným predikčným horizontom. Politika RHC je vhodná pre adaptáciu riadenia na zmeny, no práve hľadanie optimálneho riešenia v každom časovom kroku predstavuje vysoký nárok na hardvérové časti, a tak je MPC neustálym predmetom výskumu na znižovanie náročnosti na výkon či pamäť. MPC a RHC

využijeme v kapitole 4 pri sledovaní trasy heterogénnym vozidlom.

Kapitola 4

Smerovanie heterogénneho vozidla

Hneď v úvode práce sme spomínali hlavný cieľ dizertačnej práce. Ním je navrhnúť formuláciu optimalizácie problému plánovania trasy pre heterogénne vozidlo, pričom chceme zabezpečiť, aby sme riešenie našli bez použitia heuristických metód a teda, aby nájdené riešenie bolo skutočne optimálne. V nasledujúcich kapitolách sa budeme zaoberať konkrétnym detailným riešením jednotlivých cieľov a podcieľov stanovených v kapitole 1.4.

Ako prvé si v tejto kapitole formálne definujeme tri hlavné problémy. Potom si v kapitole 4.2 ukážeme problém plánovania trasy pre dané poradie návštev jednotlivých stanovíšť. Najprv si uvedieme MI-NLP formuláciu z publikácie Garone et al. (2012) a potom prejdeme k stanoveniu výpočtovo užitočnejšej MI-SOCP formulácie. Následne sa budeme zaoberať aj diskutovaním o možných jednoduchých rozšíreniach vhodných pre simuláciu scenárov z reality. V kapitole 4.3 rozoberieme zložitejšiu situáciu, kedy budeme potrebovať optimalizovať aj poradie bodov, pričom ukážeme dve možné stratégie riešenia. Prvá prináša síce suboptimálne riešenie, ale je výpočtovo jednoduchšia, kým druhá prináša napriek veľkej výpočtovej zložitosti skutočne optimálne riešenie. V kapitole 4.4 sa opäť priblížime k scenáru bližšiemu k realite, kde uvažujeme situáciu, kedy by sa cieľové stanovištia pohybovali v reálnom čase. Pre uplatnenie formulácie si vytvoríme jednoduchú situáciu, kde budeme riadiť heterogénne vozidlo prostredníctvom prediktívneho riadenia podľa získanej trajektórie. V rámci prípadových štúdií zhrnieme získané výsledky v číslach, rozoberieme výpočtové zložitosti jednotlivých prípadov a budeme sa zaoberať aj mierou suboptimality navrhnutých riešení.

4.1 Stanovenie problému

Hlavnou prekážkou pri riešení problémov hľadania optimálnej cesty je poradie lokalít, ktoré má vozidlo navštíviť. V práci sa budeme venovať nielen problému nájdenia optimálnej cesty pre heterogénne vozidlo za predpokladu, že je poradie návštevy jednotlivých bodov dané, ale budeme sa zaoberať aj návrhom optimálne zvoleného poradia.

Uvažujeme systém heterogénneho vozidla, ktoré pozostáva z dvoch vozidiel. Prvé má nízku maximálnu rýchlosť, ale je schopné dlhej cesty, pričom nesie druhé vozidlo. Preto ho nazývame aj *nosič*. Druhé, tzv. *agilné vozidlo*, je schopné rýchleho presunu avšak iba na krátku vzdialenosť. Pre lepšiu ilustráciu budeme pod týmto heterogénnym systémom rozumieť loď (ako nosič), ktorá nesie helikoptéru (agilné vozidlo). Predpokladáme, že loď sa pohybuje konštantnou rýchlosťou v_{c} a jej dosah nie je nijako obmedzený. Index c je označenie z anglického slova *carrier* a budeme ho používať pre všetky premenné týkajúce sa lode. Helikoptéra je buď odstavená na lodi, alebo je vo vzduchu a v takom prípade sa pohybuje konštantnou rýchlosťou v_{h}. Index h budeme používať pre všetky premenné týkajúce sa helikoptéry. Jej limit pre maximálnu prejdenú vzdialenosť bude ohraničovať premenná $t_{\mathrm{h,max}}$, čo je čas, ktorý vydrží helikoptéra vo vzduchu bez doplnenia paliva. Kedykoľvek helikoptéra pristane na lodi predpokladáme, že okamžite natankuje plnú nádrž, pričom čas čerpania zanedbávame.

Heterogénne vozidlo začína v bode q_{s} a jeho poslaním je navštíviť každý z bodov (stanovíšť) $q_1, \ldots, q_n$ presne raz, kým dorazí do konečnej destinácie q_{f}. Kým počiatočný a cieľový bod musíme navštíviť s loďou, priebežné body môžu byť navštívené iba helikoptérou, pričom helikoptéra môže navštíviť viacero bodov na jedno vzlietnutie.

Poznámka 4.1.1. Priebežné stanovištia nie je povolené navštíviť loďou, nakoľko v praxi vo väčšine prípadov nie je nosič uspôsobený na to, aby plnil úlohu agilného vozidla.

Našim cieľom je nájsť optimálnu trasu, ktorá minimalizuje čas na splnenie celej misie vzhľadom na dané limity. Formálne by sme mohli stanoviť problém nasledovne:

Problém 4.1.2. Máme dané: zoradenú množinu bodov $q_i \in \mathbb{R}^2$ pre $i = 1, \ldots, n$, štartovací bod $q_{\mathrm{s}} \in \mathbb{R}^2$, finálny bod $q_{\mathrm{f}} \in \mathbb{R}^2$ (body q_{s} a q_{f} môžu byť zhodné), rýchlosť lode v_{c}, rýchlosť helikoptéry v_{h} a dosah helikoptéry $t_{\mathrm{h,max}}$. Je potrebné určiť:

- množinu indexov $\mathcal{I}_1, \ldots, \mathcal{I}_m$, kde $\mathcal{I}_i \subseteq \{1, 2, \ldots, n\}$, označujúcu, ktoré body q_i helikoptéra navštívi behom jedného vzletu,
- množinu vzlietajúcich a pristávajúcich bodov $\{\tau_i, \ell_i\}$ takých, že τ_i predstavuje bod, kedy helikoptéra opustí loď a navštívi body $q_i, \ldots, q_j$ (indexované $\mathcal{I}_i$), kým pristane opäť na loď v bode ℓ_j.

Ciele sú nasledovné:

- kompletný čas misie je minimálny,
- pre každú lietaciu fázu obsahuje príslušná množina indexov $\mathcal{I}_i$ iba indexy q_i, ktoré su v dosahu doletu helikoptéry,
- každý prostredný bod q_i je navštívený presne raz *v poradí* $i = 1, \ldots, n$, čiže množiny indexov $\mathcal{I}_i$ sú vo vzájomnej exklúzii $\mathcal{I}_j \cap \mathcal{I}_k = \emptyset$ pre všetky $j \neq k$, kým ich zjednotenie spĺňa $\bigcup_i \mathcal{I}_i = \{1, \ldots, n\}$.

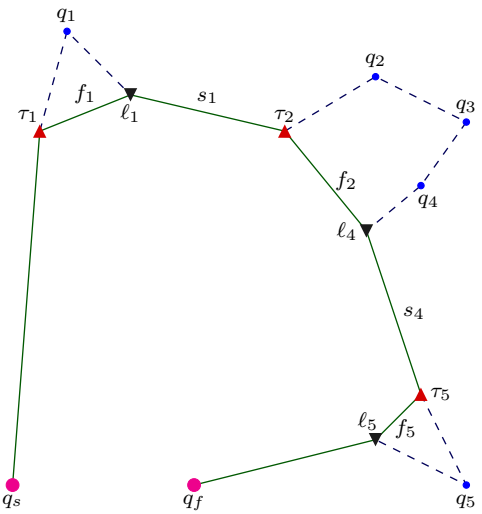
Všimnime si, že kým je helikoptéra vo vzduchu a obieha body $q_i, \ldots, q_j$, loď ide priamou cestou z bodu τ_i do ℓ_j. Preto minimálne množstvo vzletov je $m = 1$ (kedy dosah helikoptéry umožní obletieť všetky dané body $q_1, \ldots, q_n$ na jeden vzlet), kým maximum je $m = n$.

Pre lepšie pochopenie premenných predstavených v Probléme 4.1.2 sa pozrime na obrázok 4.1, ktorý ilustruje úlohou, v ktorej chceme navštíviť 5 bodov $q_1, \ldots, q_5$, začínajúcich v q_s and končiacich v q_f. V scenári zobrazenom na obrázku 4.1 loď ide nasledujúcou cestou: $q_\mathrm{s} \to \tau_1 \to \ell_1 \to \tau_2 \to \ell_4 \to \tau_5 \to \ell_5 \to q_\mathrm{f}$. Keď dorazí na pozíciu τ_1, helikoptéra vzlietne a navštívi jediný bod q_1 (teda $\mathcal{I}_1 = \{1\}$) predtým, ako sa vráti opäť na loď na pozícii ℓ_1 kvôli natankovaniu. Odtiaľto idú obe vozidlá spoločne, až kým nedosiahnu bod τ_2. Tu helikoptéra vzlietne opäť. Tentoraz navštívi body q_2, q_3, q_4, ktoré korešpondujú s $\mathcal{I}_2 = \{2, 3, 4\}$. Medzitým loď postupuje priamo do ℓ_4, kde sa stretne s helikoptérou. Dvojica postupuje ďalej spoločne do τ_5 kde sa helikoptéra opäť oddelí, aby navštívila q_5 (s $\mathcal{I}_5 = \{5\}$) a potom sa opäť vrátila na loď, ktorá medzitým doplávala do bodu ℓ_5. Odtiaľto sa naše heterogénne vozidlo presunie späť do prístavu v cieľovom bode q_f.

Povšimnime si, že v Probléme 4.1.2 predpokladáme, že poradie bodov je dané. Podobná situácia často nastáva pri záchranných operáciách, kde prioritu určujú rôzne faktory podľa naliehavosti situácie ako napríklad zdravotný stav zachraňovaných osôb. V iných prípadoch môže byť najvýhodnejšie poradie také, ktoré zabezpečí, aby bol celkový čas misie čo najkratší. Napríklad, ak by helikoptéra mala za úlohu skontrolovať stav jednotlivých veží veterných elektrární, alebo vrtných veží. Preto si teraz zadefinujeme rozšírenie k Problému 4.1.2.

Problém 4.1.3. Máme dané: štartovací bod q_s, konečný bod q_f a prostredné body q_i, $i = 1, \ldots, n$. Je potrebné určiť optimálnu trasu minimálnej časovej dĺžky a optimálne poradie, v ktorom majú byť body q_i navštívené tak, aby heterogénne vozidlo navštívilo všetky prostredné body práve raz.

Kým v Probléme 4.1.2 index i v q_i odkazuje na pevne dané poradie bodov navštívených helikoptérou, v Probléme 4.1.3 sú tieto indexy rozhodovacími premennými a sú



Obr. 4.1: Ilustrácia optimálnej dráhy pre heterogénne vozidlo. τ_i a ℓ_j predstavujú vzletové a pristávacie body helikoptéry. Plné čiary znázorňujú trajektóriu lode, prerušované čiary zobrazujú trasu helikoptéry, ktorá potrebuje navštíviť body $q_1, \ldots, q_5$ v stanovenom poradí. Kvôli časovému obmedzeniu helikoptéry pre dĺžku letu však musí helikoptéra spraviť niekoľko zastávok na lodi a natankovať.

optimalizované.

V oboch spomínaných problémoch predpokladáme, že navštevované body sú stacionárne a ich poloha sa nebude meniť. V bežnom živote však môžeme mať aj situáciu, kedy potrebujeme obísť ciele, ktoré sa pohybujú. Napríklad v prípade záchrany topiacich môžeme predpokladať, že jednotlivé osoby plávajú alebo, že ich unáša prúd. V takom prípade získaná naplánovaná trasa vyriešením Problému 4.1.3 nám nepomôže, nakoľko v čase pohybu heterogénneho vozidla bude už situácia úplne iná ako v čase, keď sme trajektóriu plánovali. V práci sa budeme zaoberať aj riešením takéhoto problému.

Problém 4.1.4. Máme dané: štartovací bod q_s, konečný bod q_f a prostredné body $q_i(t)$, pre $i = 1, \ldots, n$, pričom pozície týchto bodov sú časovo premenlivé. Je potrebné určiť optimálnu trasu minimálnej dĺžky, aby boli body q_i navštívené heterogénnym vozidlom práve raz.

Problém 4.1.4 budeme uvažovať najprv pre prípad, kde je poradie návštev bodov pevne dané a potom aj pre prípad, kde je optimalizované. Poznamenajme, že v každom čase t poznáme len polohu bodov q_{i_t}. Nepredpokladáme teda, že by sme vedeli pohyb navštevovaných bodov predvídať.

4.2 Riešenie s pevne daným poradím

4.2.1 Nelineárna formulácia

V tejto kapitole popíšeme riešenie Problému 4.1.2 pomocou zmiešanej celočíselnej nelineárnej formulácie (MI-NLP) tak ako bolo navrhnuté v (Garone et al., 2012). Majme danú binárnu maticu α s m riadkami zodpovedajúcimi jednotlivým vzletovým fázam helikoptéry a ku ktorým prislúchajú množiny $\mathcal{I}_1, \ldots, \mathcal{I}_m$. Potom ak platí, že $\alpha_{i,j} = 1$, interpretujeme situáciu nasledovne: počas k-tej vzletovej fázy helikoptéra postupne oblietava body $q_i, \ldots, q_j$ bez jediného medzipristátia. V príklade ukázanom na Obr. 4.1 má matica nasledovný tvar

$$\alpha = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.1}$$

Aby sme mohli zabezpečiť, že každý bod q_i je navštívený práve raz, musíme uplatniť nasledujúce ohraničenia

$$\sum_{i=1}^{k} \sum_{j=k}^{n} \alpha_{i,j} = 1, \quad k = 1, \ldots, n. \tag{4.2}$$

Jednoducho si vieme overiť, že α z (4.1) spĺňa podmienku (4.2), a zahŕňa tri vzletové fázy:

- v prvej navštívi iba bod q_1 (čiže $\mathcal{I}_1 = \{1\}$),
- druhým vzlietnutím pokryje body $q_2, \ldots, q_4$ (čo zodpovedá $\mathcal{I}_2 = \{2, 3, 4\}$),
- v poslednej obletí opäť jediný bod q_5 s $\mathcal{I}_5 = \{5\}$.

Výhoda tejto navrhnutej sémantiky matice α, uplatnenej vzťahom (4.2) je, že najviac n prvkov z matice α môže byť rovných hodnote jedna. Táto vlastnosť do určitej miery znižuje exponenciálnu zložitosť celočíselného programovania.

Ku každej vzletovej fáze priraďujeme jej celkový letový čas $f_{i,j} \geq 0$, ktorý predstavuje dobu od vzlietnutia helikoptéry z bodu τ_i, obletenie cez potrebných návštevných miest $q_i, \ldots, q_j$, až po pristátie opäť na lodi v bode ℓ_j. Čas, ktorý môže stráviť helikoptéra vo vzduchu máme ohraničený nasledovným vzťahom

$$\alpha_{i,j} f_{i,j} \leq t_{\mathrm{h,max}}. \tag{4.3}$$

Násobenie $\alpha_{i,j}$ s $\alpha_{i,j} \in \{0,1\}$ zaisťuje, že ohraničenie bude mať platnosť iba v prípade ak $\alpha_{i,j} = 1$, čo zodpovedá výberu bodov $q_i, \ldots, q_j$ na obletenie. Ak platí $\alpha_{i,j} = 0$, ohraničenie je nie aktívne. K tomuto musíme zohľadniť ešte fakt, že za čas, ktorý je vrtuľník vo vzduchu, musí stihnúť jeho nosič prejsť z τ_i do ℓ_j aby na loď mohla helikoptéra opäť bezpečne pristáť. Ak predpokladáme, že loď sa hýbe priamočiaro konštantnou rýchlosťou v_c, musí byť splnené nasledovné ohraničenie

$$\alpha_{i,j}\|\tau_i - \ell_j\| \leq v_\mathrm{c} f_{i,j}. \tag{4.4}$$

V opačnom prípade, by helikoptéra priletela k bodu stretu ℓ_j skôr ako loď a počas čakania by sa jej mohlo minúť palivo.

Ako posledné musíme zohľadniť celkovú dĺžku trasy, ktorú helikoptéra plánuje preletieť a teda vzdialenosť z bodu τ_i cez $q_i, \ldots, q_j$ až po ℓ_j konštantnou rýchlosťou v_h. Formálne zapíšeme podmienku nasledovne

$$\alpha_{i,j}\left(\|\tau_i - q_i\| + d_{i,j} + \|q_j - \ell_j\|\right) \leq v_\mathrm{h} f_{i,j}, \tag{4.5}$$

kde $d_{i,j}$ predstavuje celkovú dĺžku po častiach lineárnej dráhy, ktorá spája jednotlivé navštívené body $q_i, \ldots, q_j$, tak aby bola jej dĺžka minimálna. Matematicky vyjadrené

$$d_{i,j} = \sum_{k=i}^{j-1} \|q_k - q_{k+1}\|. \tag{4.6}$$

Všimnime si, že matica $d \in \mathbb{R}^{n\times n}$ s hodnotami $d_{i,j}$ tak ako vystupuje vo vzťahu (4.6) môže byť vypočítaná ešte pred samotným riešením problému, a je teda považovaná za maticu konštánt, pretože pozície bodov q_i sú pevne dané. Čas $s_{i,j}$, ktorý sa plaví loď spolu s helikoptérou od doby kedy helikoptéra pristála ℓ_j, až po ďalšie vzlietnutie τ_{j+1}, sa nám viaže na α cez ohraničenie

$$\alpha_{i,j}\|\ell_j - \tau_{j+1}\| \leq v_\mathrm{c} s_{i,j}. \tag{4.7}$$

Celkový čas na vykonanie misie t_m, ktorý chceme minimalizovať je zložený zo štyroch častí

1. čas, ktorý prechádza loď spolu s helikoptérou zo začínajúceho bodu q_s až po prvé vzlietnutie τ_1, vyjadrený ako

$$1/v_\mathrm{c}\|q_\mathrm{s} - \tau_1\|, \tag{4.8}$$

2. čas, ktorý potrebuje loď, kým sa presunie z bodu vzlietnutia do bodu, kde helikoptéra opäť pristane definovaný ako

$$\sum_{i=1}^{n}\sum_{j=i}^{n} f_{i,j}, \tag{4.9}$$

3. čas, ktorý sa plaví loď spolu s helikoptérou od doby kedy helikoptéra pristála, až po ďalšie vzlietnutie

$$\sum_{i=1}^{n}\sum_{j=i}^{n-1} s_{i,j}, \tag{4.10}$$

4. čas, ktorý cestuje dvojica opäť spolu od posledného pristátia až do záverečného stanovišťa q_{f}

$$1/v_{\mathrm{c}}\|\ell_n - q_{\mathrm{f}}\|. \tag{4.11}$$

Keď sčítame všetky časy dohromady, vyjadríme celkový čas trvania misie

$$t_{\mathrm{m}} = 1/v_{\mathrm{c}}\left(\|q_{\mathrm{s}} - \tau_1\| + \|\ell_n - q_{\mathrm{f}}\|\right) + \sum_{i=1}^{n}\sum_{j=i}^{n} f_{i,j} + \sum_{i=1}^{n}\sum_{j=i}^{n-1} s_{i,j}. \tag{4.12}$$

Potom riešenie Problému 4.1.2 získame riešením optimalizačného problému v tvare

$$\min_{\alpha,f,s,\tau,\ell}\ t_{\mathrm{m}} \tag{4.13a}$$

$$\text{v.n.}\ \sum_{i=1}^{k}\sum_{j=k}^{n}\alpha_{i,j} = 1, \quad k = 1,\ldots,n, \tag{4.13b}$$

$$\alpha_{i,j} f_{i,j} \le t_{\mathrm{h,max}}, \tag{4.13c}$$

$$\alpha_{i,j}\|\tau_i - \ell_j\| \le v_{\mathrm{c}} f_{i,j}, \tag{4.13d}$$

$$\alpha_{i,j}\left(\|\tau_i - q_i\| + d_{i,j} + \|q_j - \ell_j\|\right) \le v_{\mathrm{h}} f_{i,j}, \tag{4.13e}$$

$$d_{i,j} = \sum_{k=i}^{j-1}\|q_k - q_{k+1}\|, \tag{4.13f}$$

$$\alpha_{i,j}\|\ell_j - \tau_{j+1}\| \le v_{\mathrm{c}} s_{i,j}, \tag{4.13g}$$

s rozhodovacími premennými $\alpha \in \{0,1\}^{n\times n}$, $f \in \mathbb{R}^{n\times n}$, $f_{i,j} \ge 0$, $s \in \mathbb{R}^{n\times n}$, $s_{i,j} \ge 0$, $\tau \in \mathbb{R}^{2\times n}$, a $\ell \in \mathbb{R}^{2\times n}$, pričom (4.13f) je iba vzorec pre výpočet konštánt. Všimnime si, že každý stĺpec z τ a z ℓ predstavuje súradnice vzlietnutia pristátia 2-dimenzionálnom Euklidovom priestore. Keďže $f_{i,j}$ a $s_{i,j}$ sú minimalizované s (4.12), ak $\alpha_{i,j} = 0$ je optimálne riešenie k (4.13), potom $f_{i,j} = 0$ a $s_{i,j} = 0$ sú zlúčiteľné optimálne riešenia. Toto vyplýva z (4.4), (4.5), a (4.7) čo má za následok, že $f_{i,j} \ge 0$ (a $s_{i,j} \ge 0$) pre $\alpha_{i,j} = 0$.

Problém vyjadrený vzťahom (4.13) je problém zmiešaného celočíselného programovania s nelineárnymi ohraničeniami. Celočíselná zložka je predstavená binárnou rozhodovacou premennou α. Nelinearita vyplýva zo vzťahu medzi $\alpha_{i,j}$ a reálnymi rozhodovacími premennými v (4.3), (4.4), (4.5), a (4.7).

Poznámka 4.2.1. Akonáhle získame optimálne riešenie k (4.13) ekvivalenciu medzi α a indexovými množinami $\mathcal{I}$ z Problému 4.1.2 vieme zostaviť nasledovne:

nech i je index riadku α, ktorý obsahuje aspoň jednu nenulovú hodnotu. Potom $\mathcal{I}_i = \{i, \ldots, j\}$, pre j také, že $\alpha_{i,j} = 1$.

Poznámka 4.2.2. Problém vyjadrený vzťahom (4.13) je vždy riešiteľný. V najhoršom prípade je každý prostredný bod q_i navštívený individuálne, čo znamená, že α bude jednotková matica. Zlúčiteľnosť riešenia vyplýva z faktu, že dosah lode je neobmedzený a celkový čas trvania misie nemá žiadne ohraničenia.

4.2.2 Formulácia zmiešaného celočíselného programovania nad kužeľom druhého rádu

Hlavné obmedzenie formulácie zmiešaného celočíselného nelineárneho programovania (MI-NLP) optimalizačného problému vyjadrenom vo vzťahu (4.13) pramení z jej výpočtovej zložitosti. Konkrétne, autori v Garone et al. (2012) demonštrovali, že problém je riešiteľný, v primeranom čase, avšak iba pre malé množstvo bodov q_i, čiže pre nízku hodnotu n. Presnejšie, najzložitejší prípad akým sa v danom zdroji zaoberali bol $n = 7$. V tejto kapitole si ukážeme ako je možné *ekvivalentne* preformulovať MI-NLP (4.13) na problém zmiešaného celočíselného programovania nad kužeľom druhého rádu (MI-SOCP, rozoberaným podrobne v kapitole 2.6), ktorý je možné efektívne riešiť pre *stovky* bodov. Hoci MI-SOCP problémy sú nekonvexné kvôli prítomnosti celočíselných premenných, vo chvíli kedy sa tieto premenné zafixujú prostredníctvom algoritmu vetvenia a hraníc (angl.: *branch-and-bound*) popísaného v kapitole 2.4.2, problém sa stáva konvexným SOCP. Na druhej strane, v MI-NLP problémoch sú dokonca aj *vnútorné* problémy nekonvexné.

Začneme tým, že si pripomenieme, že netriviálna časť z (4.13) sú nelineárne ohraničenia, kde sa rôzne rozhodovacie premenné navzájom násobia. Hoci, bližší pohľad na (4.3)−(4.5) a (4.7) odhaľuje, že takéto nelineárne vzťahy zahŕňajú násobenie medzi binárnymi premennými $\alpha_{i,j}$ aa konvexnou funkciou. Vezmime si ako príklad (4.3). Ohraničenie môže byť ekvivalentne zapísané ako logický výraz v tvare

$$(\alpha_{i,j} = 1) \Rightarrow f_{i,j} \leq t_{\mathrm{h,max}}. \tag{4.14}$$

Všimnime si, že nezávisle od hodnoty $\alpha_{i,j}$, je letový čas $f_{i,j}$ ohraničený zospodu s $f_{i,j} \geq 0$ pre akúkoľvek kombináciu i a j. Podobne, (4.4) môže byť zapísaný ako:

$$(\alpha_{i,j} = 1) \Rightarrow \|\tau_i - \ell_j\| \leq v_{\mathrm{c}} f_{i,j}, \tag{4.15}$$

čo predstavuje striktne pozitívnu spodnú hranicu $f_{i,j}$ ak $\alpha_{i,j} = 1$. Poznamenajme, že pre $\alpha_{i,j} = 0$ ohraničenie (4.4) zahŕňa $0 \leq v_{\mathrm{c}} f_{i,j}$, čo je opäť ekvivalentné spodnej hranici $f_{i,j} \geq 0$. Rovnakým postupom môžeme postupovať ďalej aj pri (4.5), čo je ekvivaletné s

$$(\alpha_{i,j} = 1) \Rightarrow (\|\tau_i - q_i\| + d_{i,j} + \|q_j - \ell_j\|) \leq v_{\mathrm{h}} f_{i,j} \tag{4.16}$$

a (4.7) môžeme prepísať do

$$(\alpha_{i,j} = 1) \Rightarrow \|\ell_j - \tau_{j+1}\| \leq v_{\mathrm{c}} s_{i,j} \tag{4.17}$$

s časom plavenia ohraničeným zospodu vzťahom $s_{i,j} \geq 0$.

Výhoda prepísania (4.3)−(4.5) a (4.7) na skupinu implikačných pravidiel v (4.14)−(4.17) je, že ich môžeme ďalej zjednodušiť a previesť na množinu ohraničení, ktoré sú konvexné aj v rozhodovacích premenných $\alpha_{i,j}$, $f_{i,j}$, $s_{i,j}$, τ_i a ℓ_j použitím základných pravidiel výrokovej logiky (Williams, 1993).

Veta 4.2.3. Uvažujme binárnu premennú $\delta \in \{0, 1\}$, reálne premenné $x \in \mathbb{R}^m$, a účelovú funkciu $g : \mathbb{R}^m \to \mathbb{R}$. Potom

$$(\delta = 1) \Rightarrow g(x) \leq 0 \tag{4.18}$$

vtedy a len vtedy ak

$$g(x) \leq M(1 - \delta), \tag{4.19}$$

platí pre nejakú konštantu M.

Dôkaz. Máme dané dva logické výroky Y_1 a Y_2,

$$(Y_1 \Rightarrow Y_2) \Leftrightarrow (\overline{Y}_1 \vee Y_2), \tag{4.20}$$

kde $\overline{Y}_1$ je negácia Y_1 a $\vee$ je logický operátor "or". Následne jednoducho zistíme, že

$$([\delta = 1] \vee [g(x) \leq 0]) \Leftrightarrow (g(x) \leq M\delta)\,. \tag{4.21}$$

Potom (4.19) priamo vyplýva z (4.20) a (4.21) berúc v úvahu negáciu δ ako $\overline{\delta} = 1 - \delta$ (pripomeňme si, že δ je binárna premenná). □

Aplikovanie Vety 4.2.3 na (4.14) nám umožní prepísať logickú implikáciu na

$$f_{i,j} - t_{\mathrm{h,max}} \leq M(1 - \alpha_{i,j}), \tag{4.22}$$

so spodným ohraničením $f_{i,j} \geq 0$. Podotknime, že vzťah (4.22) je lineárny v reálnych rozhodovacích premenných $f_{i,j}$ a v binárnych premenných $\alpha_{i,j}$. Podobne, (4.15)−(4.17) môžeme prekonvertovať na

$$\|\tau_i - \ell_j\| - v_{\mathrm{c}} f_{i,j} \leq M(1 - \alpha_{i,j}), \tag{4.23a}$$

$$(\|\tau_i - q_i\| + d_{i,j} + \|q_j - \ell_j\|) - v_{\mathrm{h}} f_{i,j} \leq M(1 - \alpha_{i,j}), \tag{4.23b}$$

$$\|\ell_j - \tau_{j+1}\| - v_{\mathrm{c}} s_{i,j} \leq M(1 - \alpha_{i,j}). \tag{4.23c}$$

Všimnime si, že všetky premenné v (4.23) sú konvexné v prislúchajúcich rozhodovacích premenných. Ba čo viac, vďaka Euklidovým normám, (4.23) môžeme prepísať ako množinu ohraničení *kužeľa druhého rádu*, viac v Boyd a Vandenberghe (2009).

Hľadanie optimálneho poradia vzletových a pristávacích bodov z (4.13) môžeme teda formulovať ekvivalentným zápisom nasledovne

$$\min_{\alpha,f,s,\tau,\ell} t_{\mathrm{m}} \tag{4.24a}$$

$$\text{v.n. } f_{i,j} - t_{\mathrm{h,max}} \leq M(1-\alpha_{i,j}), \tag{4.24b}$$

$$\|\tau_i - \ell_j\| - v_{\mathrm{c}} f_{i,j} \leq M(1-\alpha_{i,j}), \tag{4.24c}$$

$$(\|\tau_i - q_i\| + d_{i,j} + \|q_j - \ell_j\|) - v_{\mathrm{h}} f_{i,j} \leq M(1-\alpha_{i,j}), \tag{4.24d}$$

$$\|\ell_j - \tau_{j+1}\| - v_{\mathrm{c}} s_{i,j} \leq M(1-\alpha_{i,j}), \tag{4.24e}$$

$$f_{i,j} \geq 0,\ s_{i,j} \geq 0,\ \alpha_{i,j} \in \{0,1\}, \tag{4.24f}$$

s t_{m} ako v (4.12) a ohraničeniami stanovenými pre všetky $i,j \in \{0,\ldots,n\}$. Problém definovaný v (4.24) je problém zmiešaného celočíselného programovania nad kužeľom druhého rádu, ktorý sme schopný riešiť s GUROBI solverom (Gurobi Optimization, 2013), ktorý využíva metódu vetvenia a rezov na efektívne eliminovanie neriešiteľných kombinácií binárnych premenných, aby sme sa vyhli prehľadávaniu exponenciálneho množstva prípadov.

Poznámka 4.2.4. Je dôležité poznamenať, že (4.24) je *nekonzervatívne* preformulovanie (4.13). Ak $\alpha_{i,j} = 1$, potom ohraničenia z (4.24) sú rovnaké ako v (4.13), čo je zreteľné z (4.22)−(4.23). Ak $\alpha_{i,j} = 0$, potom prislúchajúce optimálne hodnoty $f_{i,j}$ a $s_{i,j}$ budú nulové, pretože sú minimalizované v (4.12) a pretože sú ohraničené zospodu nulou (4.24f).

Na vyriešenie (4.24) tak efektívne ako je možné, hodnota M v (4.22) a (4.23) musí byť vybraná čo najnižšia možná. Ako je poznamenané v Kvasnica (2008), ak by sme nezvolili pevnú hodnotu pre konštantu M mohol by sa nám rádovo zdvihnúť výpočtový čas pre riešenie (4.24). Preto je dôležité odvodiť čo najpresnejšie možné hodnoty M používané v (4.22)− (4.23). Ako je uvedené v Williams (1993), najpresnejšia hodnota M, ktorá môže byť využitá v (4.19) je daná nasledovne

$$M = \max_{x \in \Omega} g(x), \tag{4.25}$$

kde Ω je (ohraničená) doména funkcie $g(\cdot)$. V prípade (4.22) je M jednoducho určené ako $M = t_{\mathrm{h,max}}$. V (4.23a) je najnižšia hodnota M:

$$M = \max_{\tau_i,\ell_j,f_{i,j}} (\|\tau_i - \ell_j\| - v_{\mathrm{c}} f_{i,j}). \tag{4.26}$$

Pretože funkcia $\|\tau_i - \ell_j\| - v_{\mathrm{c}} f_{i,j}$ je konvexná v τ_i, ℓ_j, a v $f_{i,j}$, maximálna hodnota sa nadobúda v jednom z vrcholov prislúchajúcej domény

$$\Omega = \Omega_\tau \times \Omega_\ell \times \Omega_f. \tag{4.27}$$

Tu sú Ω_τ a Ω_ℓ podmnožiny $\mathbb{R}^2$, ktorá vymedzuje prehľadávací priestor pre vzletové a pristávacie body. V praxi môžu byť tieto množiny získané určením najmenšieho možného štvorca, v ktorom sa body návštev q_i, $i = 1, \ldots, n$ môžu nachádzať. Takéto štvorcové ohraničenie môžeme ľahko vypočítať ako:

$$\Omega_\tau = \Omega_\ell = \{x \mid \min_i q_i \leq x \leq \max_i q_i\}, \tag{4.28}$$

kde nám minimá a maximá určujú súradnice bodov q_i. Nakoniec nám vyplýva z (4.3)

$$\Omega_f = \{f \mid 0 \leq f \leq t_{\mathrm{h,max}}\}. \tag{4.29}$$

Preto najtesnejšie M v (4.23a) môžeme vypočítať z (4.26) vyhodnotením $\|\tau_i - \ell_j\| - v_{\mathrm{c}} f_{i,j}$ v každom rohu $\Omega_\tau \times \Omega_\ell \times \Omega_f$, berúc v úvahu maximálnu hodnotu. Tesné hodnoty M v (4.23b) a v (4.23c) vieme získať obdobným spôsobom.

Poznámka 4.2.5. Priamy dôsledok Poznámky 4.2.2 a 4.2.4 je, že MI-SOCP formulácia (4.24) je vždy riešiteľná, za predpokladu, že hodnota M je vybraná podľa vzťahu (4.25).

Rozličné štartovacie pozície častí heterogénneho vozidla

Za istých okolností môže byť potreba riešiť optimalizáciu trasy, keď je agilné vozidlo oddelené od nosiča. Takáto situácia môže nastať v prípade, keď sa zmenia počiatočné podmienky v čase, keď už je heterogénne vozidlo v pohybe. Môže ísť napríklad o nahlásenie nového stanovišťa na obletenie. Pôvodný navigačný plán tak už nebude platiť. Podrobne sa budeme tejto situácii venovať pri riešení Problému 4.1.4 v kapitole 4.4. Teraz si povieme, ako sa nám v takom prípade zmení formulácia problému definovaného v (4.24).

Uvažujeme teda, že v čase optimalizácie je helikoptéra vo vzduchu. Z toho nám vyplýva, že dolet bude pre prvý let kratší o dobu, ktorú je helikoptéra mimo lode. Pre ostatné lety ostáva čas nezmenený. Môže nastať tiež situácia, že hlasené stanovište, ktoré bolo treba navštíviť už obletieť netreba hoci helikoptéra už k nemu smerovala. Alebo ak by bolo stanovište pohyblivé a išlo by smerom od helikoptéry, mohlo by sa stať, že by helikoptéra už nemala dostatok paliva na bezpečný návrat na loď. Preto treba pri novej formulácii rátať aj so situáciou, že helikoptéra pri svojom prvom lete nenavštívi ani jeden bod. Poďme sa teraz pozrieť ako spomínané zmeny zahrnieme do matematickej formulácie problému.

Keďže helikoptéra je v čase hľadania optimálnej trajektórie vo vzduchu, môžeme považovať jej počiatočný bod q_{sh} za bod vzlietnutia, čo zaznamenáme pridaním nasledujúceho ohraničenia

$$\tau_1 = q_{sh}. \tag{4.30}$$

Keďže je čas, ktorý ma helikoptéra k dispozícii pri prvom lete kratší ako pri nasledujúcich, musíme ohraničenie (4.22) prepísať na tvar

$$f_{1,j} - t_{\mathrm{h,rest}}, \leq M(1 - \alpha_{1,j}), \tag{4.31a}$$

$$f_{i,j} - t_{\mathrm{h,max}} \leq M(1 - \alpha_{i,j}), \qquad \text{pre } i = 2, \ldots, n. \tag{4.31b}$$

Nesmieme zabudnúť, že loď nám zo štartovacieho bodu q_{s} nejde do bodu prvého vzlietnutia, ale do bodu prvého pristátia, preto sa ohraničenie (4.23a) musí upraviť nasledovne

$$\|q_{\mathrm{s}} - \ell_j\| - v_{\mathrm{c}} f_{1,j} \leq M(1 - \alpha_{1,j}), \tag{4.32a}$$

$$\|\tau_i - \ell_j\| - v_{\mathrm{c}} f_{i,j} \leq M(1 - \alpha_{i,j}), \qquad \text{pre } i = 2, \ldots, n. \tag{4.32b}$$

Doteraz sme uvažovali, že ak helikoptéra vzlietne, obíde 1 až n bodov. Teraz ale musíme rátať aj so situáciou, že helikoptéra, žiaden bod nenavštívi. Tento fakt nám komplikuje situáciu pri definovaní α ohraničením (4.2), ktoré hovorí, že každý bod má byť navštívený práve raz a zároveň zahŕňa, že počas vzletu musí byť navštívený aspoň jeden bod. Na maticu α sú napojené aj ďalšie ohraničenia, ktoré pracujú s indexmi, ktoré závisia na jej obsahu. Aby sme predišli kompletnému prerobeniu ohraničení, pomôžeme si malým trikom. Pridáme si na začiatok poľa navštevovaných bodov ešte jeden *falošný* bod, ktorého súradnice sú zhodné so súradnicami počiatočnej polohy helikoptéry q_{sh}. Tým dosiahneme rozšírenie matice α o jedna, čo predstavuje počiatočnú letovú fázu.

Mohli by sme predpokladať, že pridaním falošného bodu si zhoršíme časovú efektivitu, nakoľko výpočtová zložitosť oboch algoritmov závisí od veľkosti n. Treba si však uvedomiť, že výpočtová zložitosť nie je daná priamo počtom bodov, ale maximálnym možným počtom vzletových fáz, pričom platí, že ich počet je ohraničený práve počtom bodov n. V našom prípade sme nútení uvažovať o jednu vzletovú fázu viac, kedy helikoptéra nemusí navštíviť ani jeden bod, ale rovnako môže navštíviť všetky. Preto pridanie falošného bodu prináša výhodu ponechania pôvodnej definície α spolu s ohraničeniami na nej závislými. Keďže falošný bod má rovnaké súradnice ako helikoptéra v čase optimalizácie, je považovaný okamžite za navštívený a proces pokračuje ďalej akoby neexistoval.

Účelová funkcia (4.12) sa takisto nevyhne zmene, nakoľko tá zarátava aj čas zo štartovacieho bodu po prvý vzlet. Niekomu by mohlo napadnúť vymeniť vzletový bod za bod prvého pristátia, treba si však uvedomiť, že tento čas je zarátaný v rámci letového času $f_{i,j}$, preto stačí z účelovej funkcie čas zo štartovacieho bodu po prvý vzlet iba vynechať.

$$t_{\mathrm{m}} = {}^{1}\!/_{v_{\mathrm{c}}} \|\ell_n - q_{\mathrm{f}}\| + \sum_{i=1}^{n} \sum_{j=i}^{n} f_{i,j} + \sum_{i=1}^{n} \sum_{j=i}^{n-1} s_{i,j}. \tag{4.33}$$

Hľadanie optimálneho poradia vzletových a pristávacích bodov pre heterogénne vozidlo s rozličnými štartovacími pozíciami jeho častí, môžeme teda formulovať nasledovne:

$$\min_{\alpha,f,s,\tau,\ell} t_{\mathrm{m}} \tag{4.34a}$$

$$\text{v.n. } \tau_1 = q_{sh}, \tag{4.34b}$$

$$f_{1,j} - t_{\mathrm{h,rest}} \leq M(1-\alpha_{1,j}), \tag{4.34c}$$

$$f_{i,j} - t_{\mathrm{h,max}} \leq M(1-\alpha_{i,j}), \quad \text{pre } i = 2,\ldots,n, \tag{4.34d}$$

$$\|q_{\mathrm{s}} - \ell_j\| - v_{\mathrm{c}} f_{1,j} \leq M(1-\alpha_{1,j}), \tag{4.34e}$$

$$\|\tau_i - \ell_j\| - v_{\mathrm{c}} f_{i,j} \leq M(1-\alpha_{i,j}), \quad \text{pre } i = 2,\ldots,n, \tag{4.34f}$$

$$(\|\tau_i - q_i\| + d_{i,j} + \|q_j - \ell_j\|) - v_{\mathrm{h}} f_{i,j} \leq M(1-\alpha_{i,j}), \tag{4.34g}$$

$$\|\ell_j - \tau_{j+1}\| - v_{\mathrm{c}} s_{i,j} \leq M(1-\alpha_{i,j}), \tag{4.34h}$$

$$f_{i,j} \geq 0,\ s_{i,j} \geq 0,\ \alpha_{i,j} \in \{0,1\}. \tag{4.34i}$$

Účelová funkcia (4.12) je užšou formou (4.34a), ohraničenia (4.34c) až (4.34f) sú výpočtovo identické s ohraničeniami (4.24b) a (4.24c). Ide len o zmenu parametra pre $i = 1$, teda pre prvý let. Problém definovaný v (4.34) je teda aj naďalej problémom zmiešaného celočíselného programovania nad kužeľom druhého rádu ako v prípade problému (4.24).

Rozšírenia riešeného problému

Ukázali sme ako sa riešenie Problému 4.1.2 dá získať riešením (4.24) ako MI-SOCP problém. V tejto kapitole si predstavíme niekoľko modifikácií (4.24), ktoré nám navrhnutý scenár približujú viac k skutočnosti.

Ako prvé, môžeme účelovú funkciu (4.24a) rozšíriť o vzatie do úvahy minimalizáciu času letu helikoptéry, ktorý je priamo úmerný spotrebe paliva. Toto môžeme vyriešiť pridaním vzťahu (4.35) do (4.24a)

$$\sum_{i=1}^{n}\sum_{j=1}^{n} f_{i,j}. \tag{4.35}$$

Ako druhé môže byť požadované zminimalizovať počet vzletov helikoptéry. Takáto požiadavka by mohla nastať z rôznych technických príčin alebo aj z bezpečnostných. Napríklad v prípade zlých poveternostných podmienok môže každé vzlietnutie a pristátie predstavovať zvyšené riziko nehody. Toto môžeme dosiahnuť zahrnutím vzťahu (4.36) do (4.24a)

$$\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_{i,j}. \tag{4.36}$$

Aby bolo možné priradiť jednotlivým cieľom priority, oba vzťahy sú prepojené s pokutovými premennými $\gamma_{\mathrm{f}} \geq 0$ a $\gamma_{\mathrm{a}} \geq 0$ pre (4.35), respektíve pre (4.36). Celková hodnota

funkcie (4.24a) potom bude

$$\min_{\alpha,f,s,\tau,\ell} \; t_{\mathrm{m}} + \gamma_{\mathrm{f}} \sum_{i=1}^{n}\sum_{j=1}^{n} f_{i,j} + \gamma_{\mathrm{a}} \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_{i,j}. \tag{4.37}$$

Ohraničenia (4.24) môžu byť takisto rozšírené. Jednou z možností je určiť maximálne množstvo vzlietnutí helikoptéry. V takom prípade doplníme do ohraničenia

$$\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_{i,j} \leq m_{\max}, \tag{4.38}$$

kde $m_{\max}$ je maximálne želané množstvo vzlietnutí. Treba však vziať do úvahy, že vybratie príliš nízkej hodnoty $m_{\max}$ môže viesť k neriešiteľnosti problému (4.24) ak dosah helikoptéry, reprezentovaný hodnotou $t_{\mathrm{h,max}}$ v (4.3) a (4.22), nedovolí helikoptére navštíviť všetky ciele počas $m_{\max}$ vzlietnutí. Aby sme sa vyhli takémuto obmedzeniu, navrhujeme použitie jemnejšej verzie (4.38)

$$\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_{i,j} \leq m_{\max} + z. \tag{4.39}$$

Tu je $z \geq 0$ nová reálna optimalizovaná premenná, ktorá ruší pevné ohraničenie. Za účelom zamedzenia tomu, aby sme pevné pravidlo neporušili inak ako v nutnom prípade, premenná z musí byť penalizovaná s $\gamma_{\mathrm{z}} z$ v účelovej funkcii s premennou γ_{z}, ktorá musí byť dostatočne vysoká (Kerrigan a Maciejowski, 2000).

4.2.3 Prípadová štúdia

V kapitole 4.2.2 sme si naformulovali problém zmiešaného celočíselné problému nad kužeľom druhého rádu (4.24) pre hľadanie optimálnych bodov vzletov a pristátí τ_i, ℓ_j (ktoré predstavujú zároveň trajektóriu nosiča) a matice α, ktorá určuje navštívené body helikoptérou v jednotlivých vzletových fázach. V tejto kapitole si najprv ukážeme príklad hľadania optimálneho navigačného plánu pre heterogénne vozidlo s použitím navrhnutej formulácie (4.24) a v ďalšej časti spravíme analýzu zložitosti, pričom porovnáme nami navrhnutú formuláciu MI-SOCP (4.24) s formuláciou MI-NLP (4.13) uvedenej v (Garone et al., 2012).

Problémy boli naformulované s použitím YALMIP (Löfberg, 2004), ktorý automaticky určuje optimálne hodnoty M v (4.22)−(4.23). MI-NLP problém (4.13) bol riešený YALMIPovým globálnym solverom vetvenia a medzí s `fmincon` ako podsolverom, ktorý bol konfigurovaný na uskutočnenie najviac $1 \cdot 10^4$ iterácií s najviac $5 \cdot 10^4$ vyhodnoteniami funkcií. Výsledný MI-SOCP problém bol riešený s GUROBI. Všetky výpočty boli uskutočnené na počítači s 3.2 GHz procesorom a 4 GB pamäťou.

Tabuľka 4.1: Umiestnenie bodov $q_1, \ldots, q_{10}$. Všetky hodnoty sú v uvedené v kilometroch.

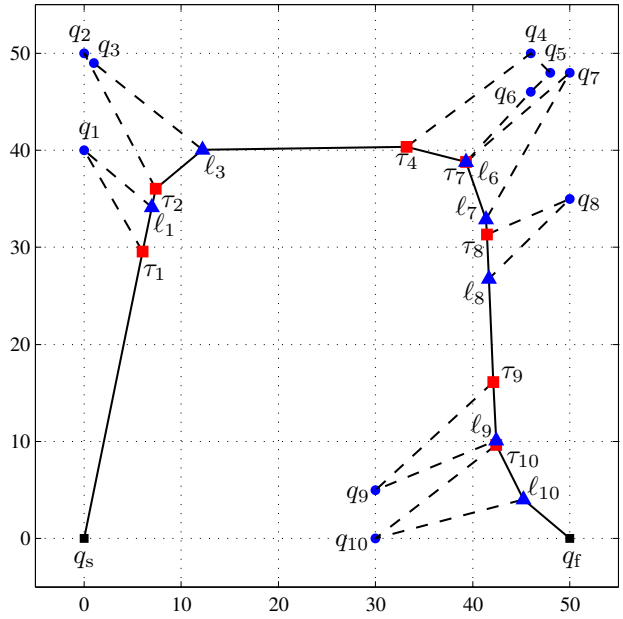
i	1	2	3	4	5	6	7	8	9	10
x_i	0	0	1	46	48	46	50	50	20	30
y_i	40	50	49	50	48	46	48	35	5	0

Príklad

Uvažujeme prípad s 10 bodmi q_i, $i = 1, \ldots, 10$, ktorých súradnice v 2-dimenzionálnom Euklidovom priestore sú uvedené v Tabuľke 4.1. Heterogénne vozidlo pozostáva z nosiča - lode, ktorá cestuje konštantnou rýchlosťou $v_{\mathrm{c}} = 18$ km/h a má neobmedzený dosah. Nosič nesie agilné vozidlo - helikoptéru, ktorej rýchlosť je $v_{\mathrm{h}} = 90$ km/h, ale má obmedzený čas pobudnutia vo vzduchu $t_{\mathrm{h,max}} = 21$ min. Loď začína v bode $q_{\mathrm{s}} = [0,\ 0]^T$ s helikoptérou na palube a jej cieľom je skončiť, po obletení všetkých desiatich stanovíšť helikoptérou, v bode $q_{\mathrm{f}} = [50,\ 0]^T$.

Optimálne riešenie k (4.24), ktoré podľa popisu v kapitole 4.2.2 je ekvivalentné s riešením (4.13), bolo získané približne za 2.2 sekundy. Optimálna trasa lode a helikoptéry spolu s optimálnymi bodmi vzletov a pristátí sú ukázané na Obrázku 4.2.

Berúc do úvahy dáta o polohe všetkých stanovíšť z Tabuľky 4.1, optimálna cesta pozostáva zo 7 letových fáz. V prvej, helikoptéra opustila loď na pozícii τ_1, navštívila jedno stanovište q_1 a navrátila sa späť pre doplnenie paliva na pozícii ℓ_1. Druhá letová fáza zahŕňala stanovištia q_2 a q_3. Ďalej pokračovali spoločne do τ_4, kde sa helikoptéra oddelila, aby navštívila body q_4, q_5, a q_6, pred pristátím na palubu v bode ℓ_6. V nasledujúcej časti je navštívené stanovište q_7. Všimnime si, že hoci stanovište q_7 bolo v tesnej blízkosti ďalších troch stanovíšť q_4, q_5, a q_6, tak helikoptéra nemohla navštíviť všetky štyri stanovištia behom jedného letu, nakoľko by jej došlo palivo. Musela sa preto vrátiť opäť na loď a stanovište q_7 navštíviť osamote v ďalšej letovej fáze. V nasledujúcich troch letových fázach boli po jednom obletené aj stanovištia q_8, q_9, q_{10}, kde sa po každej návšteve musela vrátiť helikoptéra na opätovné doplnenie paliva. V závere sa loď spolu s helikoptérou presunuli do cieľového bodu q_{f}. Najnižší čas trvanie misie bol vypočítaný riešením (4.24) na $t^{\star}_{\mathrm{m}} = 6.248$ hodín, čo zodpovedá urazenej celkovej vzdialenosti 112.464 kilometrov. Binárna matica $\alpha \in \{0,1\}^{10\times 10}$ obsahovala 7 nenulových hodnôt(čo zodpovedá 7 letovým fázam). Konkrétne nenulové údaje boli $\alpha_{1,1}$, $\alpha_{2,3}$, $\alpha_{4,6}$, $\alpha_{7,7}$, $\alpha_{8,8}$, $\alpha_{9,9}$ a $\alpha_{10,10}$. Tieto hodnoty korešpondujú indexovým množinám bodov navštíveným počas jedného letu, daným ako $\mathcal{I}_1 = \{1\}$, $\mathcal{I}_2 = \{2,3\}$, $\mathcal{I}_3 = \{4,5,6\}$, $\mathcal{I}_4 = \{7\}$, $\mathcal{I}_5 = \{8\}$, $\mathcal{I}_6 = \{9\}$, $\mathcal{I}_7 = \{10\}$. Tieto množiny spolu s príslušnými bodmi vzletov a pristátí τ_i, ℓ_j, predstavujú optimálny kompletný navigačný plán pre heterogénne vozidlo.



Obr. 4.2: Optimálna trajektória heterogénneho vozidla. Plné čiary predstavujú trasu nosiča, prerušované čiary reprezentujú trasu pre agilné vozidlo. q_i sú navštevované stanovištia, τ_i su body vzletov a ℓ_j reprezentuje body pristátí helikoptéry na palube lode. Obe osy sú uvedené v kilometroch.

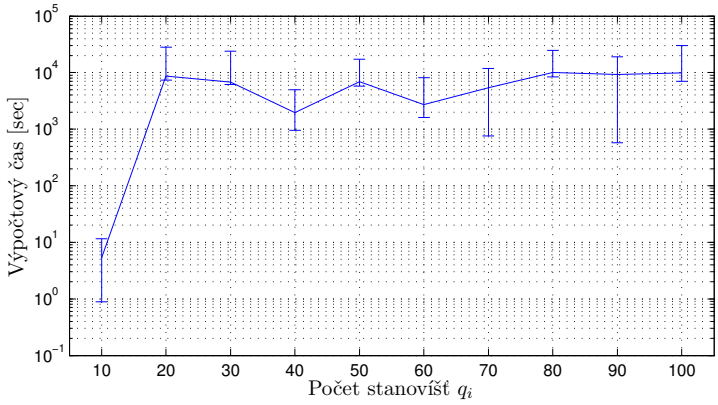
Analýza zložitosti

Ukázali sme si príklad optimálnej trasy pre heterogénne vozidlo. Teraz sa pozrieme, ako sa nám bude meniť čas pre hľadanie riešenia MI-SOCP problému definovaného v (4.24) s rastúcim počtom stanovíšť (bodov) na obletenie, q_i. Na vykonanie analýzy sme náhodne rozložili n bodov pre $n \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$ v štvorcovom ohraničenom priestore s dĺžkou hrany $[0, \min\{50, 5n\}]$ km. Pre každé n sme vygenerovali tri množiny bodov náhodne rozložených v danom priestore. Rýchlosť lode bola nastavená na hodnotu $v_c = 18\,\mathrm{km\,h^{-1}}$, rýchlosť helikoptéry bola nastavená na hodnotu $v_h = 90\,\mathrm{km\,h^{-1}}$ s maximálnou dobou doletu $t_{h,max} = 25\,\mathrm{min}$. Následne sme pre každý scenár našli optimálny navigačný plán riešením (4.24) a zmerali celkový výpočtový čas.

Získané výsledky sú ukázané graficky na Obr. 4.3. Plná čiara ukazuje priemerný výpočtový čas pre každé n (počet stanovíšť q_i). Úsečky reprezentujú rozpätie časov pre jednotlivé scenáre. Ako vidíme, formulácia MI-SOCP (4.24) si vie poradiť aj s rastúcim počtom stanovíšť, napriek teoreticky možnej najvyššej zložitosti problému. Nájsť optimálnu trasu pre $n \leq 50$, nám trvá od niekoľkých sekúnd po pár minút. Pre $50 \leq n \leq 100$, výpočtový čas dosahuje hodinu a viac. Rôznorodé správanie zobrazené na Obrázku 4.3 je spôsobené náhodným rozložením bodov. Preto môže byť výpočtový čas pre hľadanie cesty pri $n = 70$ kratší ako pri riešení MI-SOCP, kde uvažujeme návštevu iba 50 bodov. Výkyvy výpočtového času dokonca nezávisia len od rozloženia bodov, ale vyplývajú priamo z orezávacej stratégie nelineárneho zmiešaného celočíselného programovania. Solveru sa skrátka niekedy podarí využiť vhodnejšiu voľbu riešenia. Avšak toto závisí vždy od konkrétneho riešeného problému. Celkovo vzaté, aj keď sme potrebovali obletieť 100 stanovíšť, stále sme boli schopní získať optimálne riešenie Problému 4.1.2 za necelé 3 hodiny výpočtového času.

Tabuľka 4.2 porovnáva zložitosti MI-NLP problému v (4.13) s navrhnutou MI-SOCP formuláciou v (4.24). Konkrétne sme riešili oba problémy pre $n \in \{4, 5, 6, 7\}$ náhodne generovaných bodov. Poznamenajme, že všetky časy boli získané pri rovnakom hardvérovom i softvérovom nastavení pre obe simulácie. Za zmienku stojí fakt, že výpočtová zložitosť formulácie MI-NLP v (Garone et al., 2012) presahuje jednu hodinu už pre $n = 7$ bodov. Ako môžeme vidieť, zvolený prístup MI-SOCP navrhnutý v tejto práci je výrazne efektívnejší ako MI-NLP formulácia v (Garone et al., 2012) a umožňuje riešiť oveľa rozsiahlejšie problémy hľadania optimálnej trasy pre heterogénne vozidlá.

Tabuľka 4.3 porovnáva zložitosti MI-SOCP problému definovaného v (4.24) s navrhnutou MI-SOCP formuláciou pre heterogénne vozidlo s rozličnými štartovacími pozíciami jeho častí (4.34). Konkrétne sme riešili oba problémy pre $n \in \{4, 5, 6, 7, 8, 9\}$ náhodne generovaných bodov. Pre získanie presnejších výsledkov sme každý scenár zopakovali s tými istými bodmi 100-krát. Tak ako pri porovnávaní časov s MI-NLP formuláciou, boli aj teraz



Obr. 4.3: Čas potrebný na získanie optimálneho riešenia (4.24) ako funkcie n, počtu stanovíšť q_i. Úsečky predstavujú minimálny a maximálny výpočtový čas pre každé z 3 náhodných scenárov pre každé n. Os y je v logaritmickom meradle

Tabuľka 4.2: Časy potrebné pre solver na riešenie navrhnutej MI-SOCP formulácie (4.24) a formulácie MI-NLP (4.13).

n	MI-SOCP čas [s]	MI-NLP čas [s]
4	0.06	314.9
5	0.10	1145.8
6	0.25	1931.8
7	0.71	3684.5

všetky časy získané pri rovnakom hardvérovom i softvérovom nastavení pre obe simulácie. Keďže vo formulácii (4.34) máme pri rovnakom počte bodov maximálny počet letových fáz o jedna vyšší ako pri formulácii (4.24), mohli by sme očakávať časové zhoršenie ako keby sme mali o jedno stanovište na obletenie viac. Experimentálne sme ale ukázali, že časové zhoršenie pri heterogénnom vozidle s rozličnými štartovacími pozíciami jeho častí, je nižšie. Predstavuje približne 24%, kým časové zhoršenie pri pridaní jedného stanovištia predstavuje asi 84%. Treba podotknúť, že z grafu na Obr. (4.3) vyplýva, že časová zložitosť pri väčšom počte bodov kolíše natoľko, že rozdiel jedného stanovišťa je z časového hľadiska zanedbateľný.

Tabuľka 4.3: Časy potrebné pre solver na riešenie navrhnutej MI-SOCP formulácie (4.24) a MI-SOCP formulácie pre heterogénne vozidlo s rozličnými štartovacími pozíciami jeho častí (hvrp) (4.34).

n	MI-SOCP čas [s]	MI-SOCP (hvrp) čas [s]	zhoršenie [%]
4	0.06	0.07	16%
5	0.10	0.11	10%
6	0.25	0.29	16%
7	0.48	0.71	47%
8	1.02	1.11	8%
9	1.07	1.62	51%

4.3 Riešenie s optimalizovaným poradím

Určenie optimálnej trasy minimálnej dĺžky riešením MI-NLP (4.13) alebo jeho MI-SOCP verzie (4.24) vyžaduje, aby sme poradie, v ktorom budeme cieľové body q_i, $i = 1, \ldots, n$ navštevovať, poznali (Klaučo et al., 2014). V tejto kapitole si ukážeme ako nájsť optimálnu cestu za predpokladu, že poradie bodov nie je dané, ale môže byť optimalizované za účelom znížiť čas trvania misie. Na realizáciu použijeme dva prístupy.

Jedným z nich je opomenúť heterogenitu vozidla a uvažovať, že sa pohybujeme iba ako jedno vozidlo. Potom môžeme využiť známy problém cestujúceho obchodníka – TSP (Miller et al., 1960), prostredníctvom ktorého nájdeme optimálne poradie návštev cieľových bodov. Avšak takéto riešenie nám ponúkne iba suboptimálne riešenie pre heterogénne vozidlo. Preto predstavíme aj druhý spôsob riešenia, ktorý nájde skutočne optimálne riešenie, ktoré berie do úvahy aj špecifikum nášho vozidla. Pri optimalizovaní poradia, v prvom prípade prebieha optimalizácia cez indexy navštevovaných bodov v (4.5) a (4.6). Alternatívna stratégia predstavuje novú množinu reprezentujúcu trasu medzi bodmi, ktorá bude

zodpovedať optimálnej permutácii navštívených cieľov.

Teraz si popíšeme druhý spôsob, schopný nájsť optimálne riešenie. V kapitole 4.3.2 predstavíme verziu s použitím TSP. Neskôr porovnáme obe riešenia.

4.3.1 Hľadanie optimálneho riešenia

Najprv si zlúčme všetky prostredné body q_i (čiže všetky okrem q_s a q_f) do matice $Q \in \mathbb{R}^{2\times n}$

$$Q = \begin{bmatrix} q_1 & q_2 & \cdots & q_n \end{bmatrix}. \tag{4.40}$$

Následne si predstavíme binárnu permutačnú maticu $P \in \{0,1\}^{n\times n}$, kde suma hodnôt v jednotlivých riadkoch a stĺpcoch je rovná jednej

$$\sum_{i=1}^{n} P_{i,j} = 1, \quad j = 1,\ldots,n, \tag{4.41a}$$

$$\sum_{j=1}^{n} P_{i,j} = 1, \quad i = 1,\ldots,n. \tag{4.41b}$$

Potom

$$\widetilde{Q} = QP \tag{4.42}$$

značí maticu permutovaných prostredných bodov. Nech $\tilde{q}_j$ predstavuje j-tý stĺpec z

$$\widetilde{Q} = \begin{bmatrix} \tilde{q}_1 & \tilde{q}_2 & \cdots & \tilde{q}_n \end{bmatrix}. \tag{4.43}$$

Potom ak $P_{i,j} = 1$, potom i-tý pôvodný prostredný bod q_i bude v skutočnosti navštívený ako j-tý. Výhoda tejto formulácie je, že poradie permutovaných bodov, čiže $\tilde{q}_j$ pre $j = 1,\ldots,n$ je teraz zafixované a optimalizácia poradia bude presunutá na binárnu permutačnú maticu.

S touto zmenou sa ohraničenie (4.5) stáva

$$\alpha_{i,j}\left(\|\tau_i - \tilde{q}_i\| + \tilde{d}_{i,j} + \|\tilde{q}_j - \ell_j\|\right) \le v_\mathrm{h} f_{i,j}, \tag{4.44}$$

ktoré musí platiť pre všetky $i = 1,\ldots,n$ a $j = 1,\ldots,n$. Tu predstavuje $\tilde{d}_{i,j}$ Euklidovu vzdialenosť medzi $\tilde{q}_i$ a $\tilde{q}_j$. Všimnime si, že v kapitole 4.2, boli tieto vzdialenosti vypočítané predom a považované za konštanty, viď. (4.6). Avšak v tomto prípade, body $\tilde{q}_j$ závisia od výberu binárnej permutačnej matice P, ktorá je teraz optimalizačnou premennou. Preto vzdialenosti musia byť vypočítané vo vnútri optimalizačného problému cez

$$\tilde{d}_{i,j} = \sum_{k=i}^{j-1} \|\tilde{q}_k - \tilde{q}_{k+1}\|. \tag{4.45}$$

Optimalizačné premenné v novom probléme sú $\tilde{q}_i$, $i = 1, \ldots, n$ ako permutované body návštev, P ako binárna permutačná matica, α ako binárna matica sekvencie vzlietnutí/pristátí, τ_i a ℓ_j ako súradnice vzletov a pristátí, respektíve aj premenné f a s, ktoré predstavujú čas letu a plavenia. Problém vieme vyriešiť nasledovne

$$\min_{\widetilde{Q},P,\alpha,f,s,\tau,\ell} \quad t_{\mathrm{m}} \tag{4.46a}$$

$$\text{v.n.} \quad f_{i,j} - t_{\mathrm{h,max}} \leq M(1-\alpha_{i,j}), \tag{4.46b}$$

$$\|\tau_i - \ell_j\| - v_{\mathrm{c}} f_{i,j} \leq M(1-\alpha_{i,j}), \tag{4.46c}$$

$$\tilde{d}_{i,j} = \textstyle\sum_{k=i}^{j-1} \|\tilde{q}_k - \tilde{q}_{k+1}\|, \tag{4.46d}$$

$$\big(\|\tau_i - \tilde{q}_i\| + \tilde{d}_{i,j} + \|\tilde{q}_j - \ell_j\|\big) - v_{\mathrm{h}} f_{i,j} \leq M(1-\alpha_{i,j}), \tag{4.46e}$$

$$\|\ell_j - \tau_{j+1}\| - v_{\mathrm{c}} s_{i,j} \leq M(1-\alpha_{i,j}), \tag{4.46f}$$

$$\textstyle\sum_{i=1}^{n} P_{i,j} = 1, \tag{4.46g}$$

$$\textstyle\sum_{j=1}^{n} P_{i,j} = 1, \tag{4.46h}$$

$$\widetilde{Q} = QP, \tag{4.46i}$$

$$f_{i,j} \geq 0,\ s_{i,j} \geq 0, \tag{4.46j}$$

$$P_{i,j} \in \{0,1\},\ \alpha_{i,j} \in \{0,1\}. \tag{4.46k}$$

V účelovej funkcii (4.46a), je čas t_{m} definovaný ako (4.12), ktorý berie do úvahy permutované body návštev nepriamo cez $f_{i,j}$, ktoré reprezentuje čas letu z $\tilde{q}_i$ do $\tilde{q}_j$. V ohraničeniach, (4.46b) predstavuje ohraničenie času letu, ktoré je identické s (4.22), čo je vlastne ekvivalencia vzťahu (4.3) s použitím veľkého M. Ďalej, (4.46c) a (4.46f) sú rovnaké ako v (4.23a) a v (4.23c) a reprezentujú opäť ekvivalentnú verziu s použitím veľkého M k (4.4), respektíve k (4.7). Tieto dve ohraničenia zabezpečujú, aby sa agilná časť heterogénneho vozidla (teda helikoptéra) stretla s loďou v správnom bode. Permutované návštevné body $\tilde{q}_j$ (zobraté ako príslušný riadok matice $\widetilde{Q}$) vstupujú do optimalizačného problému cez (4.46e), čo je M verzia k (4.44), spolu s (4.42) a (4.45) vložená ako (4.46i), respektíve (4.46d).

Optimalizačný problém v (4.46) je stále MI-SOCP pretože (4.46b), (4.46c), a (4.46f) sú ohraničenia nerovnosti nad kužeľom druhého rádu, (4.46d) je iba substitúcia pre $\tilde{d}_{i,j}$ v (4.46e), a (4.46g)−(4.46j) sú lineárne ohraničenia. Avšak, v porovnaní s (4.24), problém v (4.46) má n^2 dodatočných binárnych premenných kvôli permutačnej matici P. Na druhej strane, (4.46) optimalizuje poradie bodov, kým (4.24) funguje iba za predpokladu, že je poradie bodov dané.

Akonáhle vypočítame optimálne riešenie (4.46), optimálne poradie môžeme priamočiaro extrahovať z binárnej permutačnej matice P. Poznamenajme, že optimálne body

vzletov helikoptéry a jej pristátí τ_i, respektíve ℓ_j, sú už nastavené podľa optimálneho poradia.

Heterogénne vozidlo s rozličnými štartovacími pozíciami jeho častí

V kapitole 4.2.2 sme si popísali ako sa nám zmení formulácia (4.24) v prípade, že v čase optimalizácie je helikoptéra vo vzduchu. Teraz si povieme, ako sa nám v takom prípade zmení formulácia problému s optimalizovaným poradím bodov (4.46).

Podmienky ostávajú nezmenené a teda pre ohraničenia (4.30), (4.31) a (4.32) platia rovnaké pravidlá ako pri formulácii s pevne daným poradím. Pri optimalizovanom poradí musíme ale vziať do úvahy, že pridaný fiktívny bod je braný ako všetky ostatné a teda, že jeho poradie musíme takisto optimalizovať. My však vieme, že fiktívne stanovište má byť obletené vždy ako prvé a preto rozšírime ohraničenie (4.41) o vzťah

$$P_{1,1} = 1, \tag{4.47}$$

čím pevne stanovíme v permutačnej matici P jeho prvenstvo. Účelová funkcia (4.33) nám ostáva nezmenená. Výsledný problém hľadania optimálnej cesty pre heterogénne vozidlo s rozličnými štartovacími pozíciami jeho častí, môžeme teda naformulovať v tvare

$$\min_{\alpha,f,s,\tau,\ell} \; t_{\mathrm{m}} \tag{4.48a}$$

$$\text{v.n.} \; \tau_1 = q_{sh}, \tag{4.48b}$$

$$f_{1,j} - t_{\mathrm{h,rest}} \leq M(1-\alpha_{1,j}), \tag{4.48c}$$

$$f_{i,j} - t_{\mathrm{h,max}} \leq M(1-\alpha_{i,j}), \quad \text{pre } i = 2,\ldots,n, \tag{4.48d}$$

$$\|q_{\mathrm{s}} - \ell_j\| - v_{\mathrm{c}} f_{1,j} \leq M(1-\alpha_{1,j}), \tag{4.48e}$$

$$\|\tau_i - \ell_j\| - v_{\mathrm{c}} f_{i,j} \leq M(1-\alpha_{i,j}), \quad \text{pre } i = 2,\ldots,n, \tag{4.48f}$$

$$\tilde{d}_{i,j} = \textstyle\sum_{k=i}^{j-1} \|\tilde{q}_k - \tilde{q}_{k+1}\|, \tag{4.48g}$$

$$\big(\|\tau_i - \tilde{q}_i\| + \tilde{d}_{i,j} + \|\tilde{q}_j - \ell_j\|\big) - v_{\mathrm{h}} f_{i,j} \leq M(1-\alpha_{i,j}), \tag{4.48h}$$

$$\|\ell_j - \tau_{j+1}\| - v_{\mathrm{c}} s_{i,j} \leq M(1-\alpha_{i,j}), \tag{4.48i}$$

$$\textstyle\sum_{i=1}^{n} P_{i,j} = 1, \tag{4.48j}$$

$$\textstyle\sum_{j=1}^{n} P_{i,j} = 1, \tag{4.48k}$$

$$P_{1,1} = 1, \tag{4.48l}$$

$$\widetilde{Q} = QP, \tag{4.48m}$$

$$f_{i,j} \geq 0, \; s_{i,j} \geq 0, \tag{4.48n}$$

$$P_{i,j} \in \{0,1\}, \; \alpha_{i,j} \in \{0,1\}. \tag{4.48o}$$

Účelová funkcia (4.48a) je zhodná s (4.34a). Ohraničenia (4.48b) až (4.48f) sú identické s ohraničeniami (4.34b) až (4.34f) z problému (4.34) a ohraničenia (4.48g) až (4.48o) sú zhodné s ohraničeniami (4.46d) až (4.46k) z problému (4.46). Problém (4.48) je rovnako ako problém (4.34) a (4.46) aj naďalej problémom zmiešaného celočíselného programovania nad kužeľom druhého rádu, hoci v porovnaní s (4.46) je výpočtovo citeľne náročnejší. Dopadu na výpočtovú zložitosť sa budeme venovať v kapitole 4.3.3.

4.3.2 Riešenie pomocou TSP

Suboptimálne riešenie Problému 4.1.3 môžeme získať pri odmyslení si, že máme heterogénne vozidlo, čiže predpokladáme, že vozidlo je homogénne a pohybuje sa konštantou rýchlosťou. V takomto prípade môžeme použiť štandardný TSP algoritmus (Miller et al., 1960) na určenie optimálneho poradia návštev jednotlivých bodov pre homogénne vozidlo. Podrobne sme sa všeobecnému TSP problému venovali v kapitole 2.4.3. Ako sme poznamenali už predtým, takto určené poradie nemusí byť optimálne pre prípad heterogénneho vozidla. Prípadom, do akej veľkej miery sa nám podarí získať optimálne riešenie, sa budeme zaoberať v kapitole 4.3.3.

Majme maticu $\widetilde{Q} = QP$ s permutovanými bodmi návštev ako v (4.42) a (4.43), s Q definovaným ako v (4.40). Ďalej uvažujme binárnu permutačnú maticu $P \in \{0,1\}^{n\times n}$, ktorá optimálne preskupí návštevné zastávky q_i, $i = 1,\dots,n$. Celková vzdialenosť, ktorú precestuje homogénne vozidlo vypočítame nasledovne

$$l = \|q_\mathrm{s} - \tilde{q}_1\| + \left(\sum_{i=1}^{n-1} \|\tilde{q}_i - \tilde{q}_{i+1}\|\right) + \|\tilde{q}_n - q_\mathrm{f}\|. \tag{4.49}$$

Cieľom je nájsť optimálny výber P tak, aby sme minimalizovali celkovú vzdialenosť vzhľadom na ohraničenie, že každý zo želaných bodov q_i pre $i = 1,\dots,n$ sa navštívi práve raz. Čo je vlastne ekvivalentné s P za podmienky (4.41). Hľadanie optimálnej permutačnej matice (a teda optimálnej sekvencii návštev prostredných bodov) môžeme uskutočniť ako

$$\min_{\widetilde{Q},P} l = \|q_\mathrm{s} - \tilde{q}_1\| + \left(\sum_{i=1}^{n-1} \|\tilde{q}_i - \tilde{q}_{i+1}\|\right) + \|\tilde{q}_n - q_\mathrm{f}\| \tag{4.50a}$$

$$\text{v.n.} \quad \sum_{i=1}^{n} P_{i,j} = 1, \quad j = 1,\dots,n, \tag{4.50b}$$

$$\sum_{j=1}^{n} P_{i,j} = 1, \quad i = 1,\dots,n, \tag{4.50c}$$

$$\widetilde{Q} = QP, \tag{4.50d}$$

čo je zmiešané celočíselné programovanie (MILP). Vo chvíli, keď problém vyriešime i-tý stĺpec z $\widetilde{Q}$, definovanom v (4.42) a (4.43), je bod, ktorý bude navštívený ako i-tý na ceste

$q_s \to \tilde{q}_1 \to \cdots \to \tilde{q}_n \to q_f$. Keď získame optimálne TSP usporiadanie z (4.50), hľadanie suboptimálneho riešenia Problému 4.1.3, môžeme uskutočniť riešením (4.24) s pevne daným poradím bodov, ktoré sme práve získali. Potrebujeme teda riešiť dva problémy zmiešaného celočíselného programovania. Avšak MILP tvaru (4.50) je zvyčajne jednoduchšie riešiť v porovnaní s MI-SOCP, pretože sú všetky podmienky lineárne. V nasledujúcej kapitole spravíme analýzu suboptimality tohto prístupu na prípadovej štúdii.

4.3.3 Prípadová štúdia

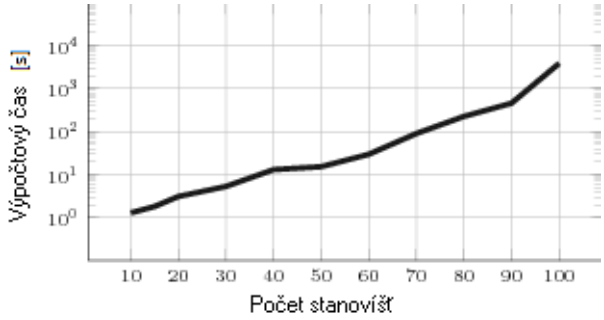
V kapitole 4.2.3 sme si zanalyzovali, ako sa bude mení výpočtová zložitosť MI-SOCP (4.24), vzhľadom na rastúce množstvo bodov, ktoré sa plánuje s heterogénnym vozidlom navštíviť. Pritom sme na príkladoch demonštrovali, že formulácia MI-SOCP (4.24) oveľa lepšia v porovnaní s formuláciou MI-NLP v Garone et al. (2012). Teraz si ukážeme ako na ako sa v podobných situáciach zachová formulácia (4.46). Treba si však uvedomiť, že keďže tá optimalizuje aj poradie bodov, je výrazne náročnejšia v porovnaní s (4.24), kde je poradie bodov pevne dané. V takom prípade môže byť subotimálne riešenie, založené na TSP prístupe z kapitoly 4.3.2, oveľa vhodnejšie vzhľadom na výpočtový čas. Z tohto hľadiska sa v kapitole 4.3.3 zaoberáme tiež mierou suboptimality riešenia.

Všetky výpočty boli uskutočnené na počítači s 4-core 3.2 GHz procesorom a so 4 GB RAM. Všetky optimalizačné problémy boli formulované s použitím YALMIP (Löfberg, 2004). Problémy (4.46) a (4.50) boli riešené s GUROBI.

Analýza výpočtovej zložitosti

Teraz sa budeme venovať analýze, ako sa bude meniť formulácia (4.46), optimalizujúca poradie, s rastúcim množstvom návštevných bodov. Táto formulácia rieši Problém 4.1.3. V tomto prípade uvažujeme $n = \{5, 7, 9, 10\}$ náhodne situovaných bodov. V porovnaní s (4.24), poradie optimalizujúca formulácia (4.46) má n^2 dodatočných binárnych optimalizovaných premenných kvôli permutačnej matici P, ktorá je zahrnutá vo vzťahu (4.42). Jednotlivé časy sú uvedené v Tabuľke 4.4. Ako môžeme vidieť, výpočtový čas pre $n = 9$ bodov je už takmer hodina, čo je výrazné navýšenie oproti niekoľkým sekundám potrebným v prípade riešenia (4.24), kde sme predpokladali dané poradie návštev.

Tabuľka 4.5 porovnáva zložitosti MI-SOCP problému (4.24) s navrhnutou MI-SOCP formuláciou pre heterogénne s rozličnými štartovacími pozíciami jeho častí (4.34). Konkrétne sme riešili oba problémy pre $n \in \{4, 5, 6, 7\}$ náhodne generovaných bodov. Pre získanie presnejších výsledkov sme každý scenár zopakovali s tými istými bodmi 10-krát. V analýze zložitosti pri pevne danom poradí bodov (4.2.3) sme experimentálne ukázali, že časové zhoršenie pri heterogénnom vozidle s rozličnými štartovacími pozíciami jeho častí, je nižšie ako zhoršenie pri pridaní jedného stanovišťa. Pri optimalizovanom poradí



Obr. 4.4: Výpočtový čas potrebný pre vyriešenie (4.50) ako funkcie zvyšujúceho sa počtu prostredných bodov.

Tabuľka 4.4: Časy potrebné pre solver na riešenie problému s optimalizovaným poradím (4.46).

n	time [s]
5	2
7	62
9	3485

sme si tento fakt potvrdili. Priemerné zhoršenie na testovaných scenároch predstavuje približne 89%, kým časové zhoršenie pri pridaní jedného stanovišťia predstavuje asi 243%. Z Tabuľky 4.5 vyplýva, že zhoršenie narastá geometrickým radom, avšak z testov predpokladáme, že pomer 89 ku 243 by mal byť približne zachovaný. Na rozdiel od riešenia pri pevne danom poradí, v prípade riešenia s väčším počtom bodov na obletenie môže časové zhoršenie zavážiť.

Tabuľka 4.5: Časy potrebné pre solver na riešenie navrhnutej MI-SOCP formulácie s optimalizovaným poradím (4.46) a formulácie pre heterogénne vozidlo s rozličnými štartovacími pozíciami jeho častí (hvrp) (4.34).

n	MI-SOCP čas [s]	MI-SOCP (hvrp) čas [s]	zhoršenie [%]
4	0.51	0.87	70%
5	2.20	4.11	86%
6	6.13	17.13	179%
7	61.96	199.46	221%

Druhou možnosťou je optimalizovať poradie návštev bodov použitím suboptimálnej procedúry popísanej v kapitole 4.3.2. Tu je najprv riešený TSP problém (4.50) s opo-

menutím heterogénnych vlastností vozidla. Následne riešime problém definovaný v (4.24) s pevne daným poradím podľa TSP riešenia. Výpočtová zložitosť vzťahu (4.50), ktorý je riešený ako problém lineárneho zmiešaného celočíselného programovania, je ukázaná na Obr. 4.4. Ako vidíme, suboptimálne určovanie poradia bodov môžeme získať za menej ako hodinu, dokonca až pre $n = 100$ bodov. Za výborný výsledok sme ale museli zaplatiť stratou optimality oproti riešeniu (4.46), ktoré ponúka zaručene optimálne riešenie. Hodnotu suboptimality si detailne rozoberieme v nasledujúcej kapitole.

Analýza suboptimality

Ako bolo zdôraznené v kapitole 4.3.2, TSP poradie získané riešením (4.50) prehliada vlastnosti heterogénneho vozidla a teda je suboptimálne. V tejto kapitole si zanalyzujeme stratu optimality oproti riešeniu (4.46), ktoré optimalizuje poradie pričom zahŕňa aj vlastnosti heterogenity.

Tabuľka 4.6: Súradnice nezoradeného listu návštevných bodov.

súradnica x	súranica y
0.00	50.00
27.00	25.00
35.00	15.00
15.00	10.00
22.50	25.00
50.00	50.00
48.00	48.00

Uvažovali sme 7 prostredných bodov vo vyhradenej oblasti $50 \times 50\,\mathrm{km}$. Súradnice týchto bodov sú uvedené v Tabuľke 4.6. V testovacom scenári sú súradnice štartovacieho a koncového bodu pevne dané ako $q_\mathrm{s} = [0\ 0]^T$ a $q_\mathrm{f} = [50\ 0]^T$. Nastavili sme maximálnu hodnotu doletu helikoptéry na $t_{\mathrm{h,max}} = 25\,\mathrm{min}$, jej rýchlosť na $v_\mathrm{h} = 90\,\mathrm{km\,h^{-1}}$ a rýchlosť lode je $v_\mathrm{c} = 18\,\mathrm{km\,h^{-1}}$.

Ako prvé sme hľadali optimálnu trasu vozidla prostredníctvom formulácie (4.46) ako MI-SOCP v (Löfberg, 2004), čo sme riešili pomocou CPLEX. Čas potrebný na získanie optimálneho riešenia bol 37 s. Kompletná trasa je zobrazená na Obrázku 4.5(a). Minimálny celkový čas misie bol 5.8319 h.

Potom, sme zanedbali heterogenitu vozidla a optimalizovali sme najprv poradie bodov riešením TSP problému (4.50). Získané TSP poradie sme potom použili v (4.24) na získanie trasy pre heterogénne vozidlo. Čas potrebný pre riešenie TSP problému bol 3.5 s. Následne, riešenie MI-SOCP (4.24) zabralo 8.1 s. Kompletná trasa získaná týmto spôso-

bom je vyobrazená na Obrázku 4.5(b). Celkový čas misie v tomto prípade bol 5.8502 h čo je o 65 s (alebo tiež o 0.3%) viac ako v prípade skutočne optimálneho prípadu.

Rozdiel medzi globálnym optimálnym poradím bodov a TSP riešením môžeme vidieť na Obrázku 4.5. V optimálnom prípade je každý z bodov navštívený samostatne, okrem bodov q_5 a q_6 z Obr. 4.5(a), ktoré boli pokryté v rámci jedného vzletu. Na druhej strane, v prípade suboptimálneho prípadu, založenom na TSP, body q_2 a q_3 na Obr. 4.5(b) sú navštívené spoločne, kým ostatné v rámci samostatného vzlietnutia po jednom.

Aby sme mohli suoptimalitu prístupu založenou na TSP lepšie posúdiť, vyšetrovali sme 50 náhodne vybraných scenárov, každý so 7 náhodne rozloženými bodmi. V každom testovacom prípade sme našli aj optimálne riešenie, ktoré bralo v úvahu heterogenitu vozidla a bolo riešené ako MI-SOCP problém definovaného v (4.46). Následne bolo globálne optimálne riešenie porovnané s dráhou získanou pomocou TSP poradia. Podotýkame, že táto dráha je najprv získaná riešením MI-LP (4.50), ked sme získali poradie, a potom riešením MI-SOCP (4.24), kde sme s využitím získaného poradia získali kompletnú trasu. Suboptimalita TSP poradia je vyjadrená ako:

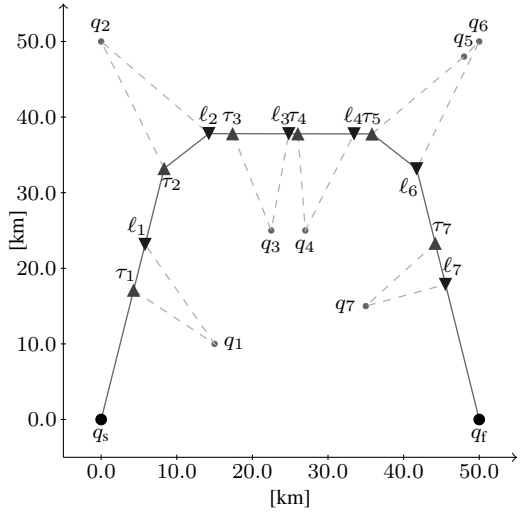
$$\sigma = \frac{t_{\text{TSP}} - t_{\min}}{t_{\min}} \cdot 100\%, \tag{4.51}$$

kde t_{TSP} je čas misie určený riešením MI-SOCP problému definovaného v (4.24) s daným poradím prostredníctvom TSP algoritmu. Čas misie $t_{\min}$ je najkratší čas potrebný pre obletenie cieľových bodov, získaný riešešním MI-SOCP problému definovaného v (4.46). Výsledky tejto analýzy sú zobrazené na Obr. 4.6.

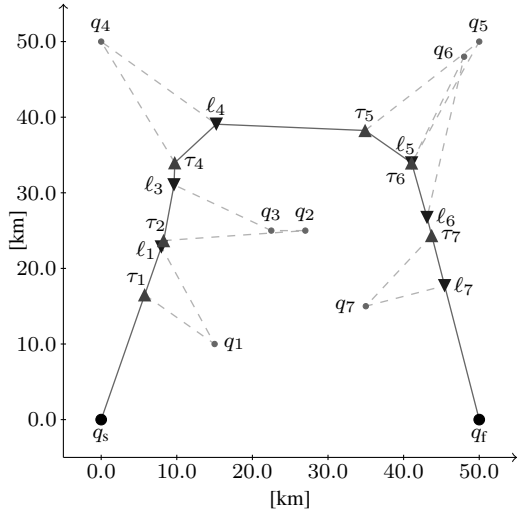
Ako môžeme vidieť, tak v 31 z 50 prípadov, čo predstavuje 62%, je suboptimalita TSP poradia menšia ako 1% vzhľadom na globálne optimum. V 15 prípadoch (alebo v 30%), je suboptimalita medzi 1% a 20%. Avšak, v 4 náhodných prípadoch dosahuje suboptimalita dokonca až 60%. Tieto výsledky nám ukazujú akú suboptimalitu prístupu, založenom na TSP algoritme, môžeme očakávať pre prípady, ak výpočtová zložitosť skutočne optimálneho riešenia MI-SOCP (4.46) je nadmerne vysoká, kvôli veľkému množstvu cieľových bodov.

4.4 Plánovanie s pohybujúcimi sa bodmi

V kapitole 4.3 sme si popísali spôsob ako nájsť optimálnu trajektóriu pre heterogénne vozidlo s cieľom navštíviť všetky body – stanovištia a optimalizovať poradie týchto návštev. Pri týchto bodoch sme uvažovali, že ich poloha je počas celého pohybu nemenná. V tejto kapitole si popíšeme riešenie Problému 4.1.4 a teda ako nájsť optimálnu trasu v prípade, že sa nám body pohybujú a ako optimalizovať trajektóriu pohybu heterogénneho vozidla v reálnom čase podľa aktuálnej situácie. Kým doteraz sme optimalizovali trasu predom,

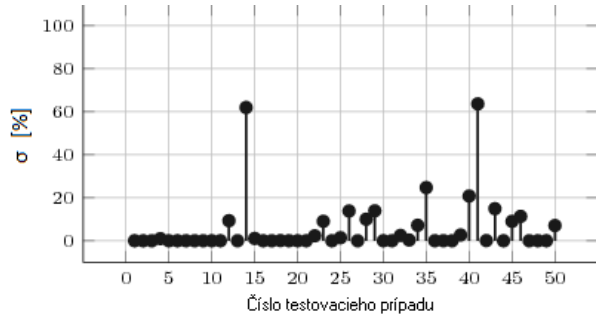


(a) Optimálna trasa

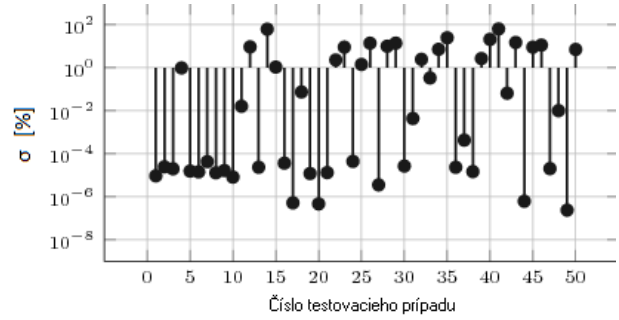


(b) Suboptimálna trasa

Obr. 4.5: Optimálna trasa porovnávaná so suboptimálnou, danou TSP poradím. Plná čiara znázorňuje dráhu lode a čiarkovaná predstavuje trasu helikoptéry. Miesta vzlietnutia sú dané ako τ_i a miesta pristátia späť na lodi ako ℓ_i.



(a) Suboptimalita vyjadrená na lineárnej stupnici



(b) Suboptimalita vyjadrená na logaritmickej stupnici

Obr. 4.6: Tento obrázok ukazuje suboptimalitu TSP trasy vzhľadom na optimálne poradie bodov. Porovnanie je zobrazené lineárne aj logaritmicky pre lepšie porovnanie rozdielov blízkych nule.

teraz budeme musieť získavať trajektóriu aj behom pohybu vozidla v pravidelných časových intervaloch ΔT, nakoľko poloha stanovíšť sa bude v priebehu času meniť. Uvažujeme, že na začiatku je helikoptéra na lodi. Po získaní optimálnej trasy sa loď s helikoptérou budú podľa nej pohybovať až do uplynutia času T, kedy sa vykoná opätovné hľadanie optimálnej trajektórie pre heterogénne vozidlo vzhľadom na jeho novú polohu a aktuálnu pozíciu stanovíšť na obletenie. Na zrealizovanie simulácie pohybu budeme uvažovať prediktívne riadenie (MPC) pevného bodu, ktorý bude reprezentovať heterogénne vozidlo. Keďže riešime úlohu pre heterogénne vozidlo všeobecne, opomíname pri riadení akékoľvek špecifické vlastnosti nosiča a agilného vozidla. Uvažujeme, že stanoviština, ktoré je potreba navštíviť agilným vozidlom sa budú pohybovať stálou rýchlosťou qv_i v jednom smere do cieľového bodu qg_i. Rýchlosti a ciele pohybu jednotlivých stanovíšť sú pritom rôzne.

Celý proces plánovania je zachytený v Algoritme 2. Na vyriešenie potrebujeme poznať pozíciu nosiča (lode) q_{s}, pozíciu agilnej časti (helikoptéry) q_{sh}, pozíciu stanovíšť q_i pre $i = 1 \ldots n$, cieľovú pozíciu q_{f} a periódu vzorkovania ΔT. Na začiatku je pozícia nosiča a agilného vozidla zhodná $q_{\mathrm{s}} = q_{\mathrm{sh}}$. V prvom kroku inicialiyujeme časový krok T. Celý proces prebieha v cykle až do momentu, kedy sú navštívené všetky stanoviština q_i agilným vozidlom a nosič skončí v cieľovom bode q_{f}. V kroku 2 zmeráme aktuálnu pozíciu heterogénneho vozidla. V kroku 3 si overujeme, či je helikoptéra na lodi a podľa toho riešime problém definovaný v (4.46) alebo (4.48), na ktorý potrebujeme pozíciu oboch častí heterogénneho vozidla a zoznam nenavštívených stanovíšť. Jeho riešením získame optimálnu trajektóriu pre heterogénne vozidlo, ktorú vyextrahujeme v kroku 8. Pre nosič ju získame spojením množiny bodov $\{q_{\mathrm{s}}, \tau_{1,i}, \ell_{1,j}, \ldots, \tau_{n,k}, \ell_{n,l}, q_{\mathrm{f}}\}$, kde q_{s} je aktuálna poloha nosiča, τ je bod oddelenia agilnej časti, ℓ je bod pristátia na nosiči, q_{f} je cieľová pozícia a indexy i, j a k, l predstavujú indexy obletených bodov q_i až q_j, respektíve q_k až q_l od vzletu až po pristátie. Pre agilné vozidlo získame trajektóriu obdobným spôsobom, spojením množín bodov $\{\tau_{1,i}, q_i, \ldots, q_j, \ell_{1,j}\}$ až $\{\tau_{n,k}, q_k, \ldots, q_l, \ell_{n,l}\}$. Následne v kroku 9 sledujeme vyextrahovanú trajektóriu heterogénnym vozidlom po dobu ΔT. Potom si prejdenú trasu uložíme. V ďalšom kroku prechádzame cez všetky stanoviština a v prípade, že sa nachádzali na prejdenej dráhe helikoptéry, označíme ich za navštívené. V opačnom prípade stanovištiam priradíme novú pozíciu. Posunieme časový krok o dobu ΔT a proces opakujeme kým nosič spolu s agilným vozidlom neskončia v cieľovom bode q_{f}. Získaná trajektória pre nosič $d_{\mathrm{r,c}}$ a agilné vozidlo $d_{\mathrm{r,h}}$ je výstupom procesu.

Na vykonanie simulácie budeme potrebovať riešiť dva rôzne typy úloh:

- nájdenie optimálneho plánu pre navigáciu heterogénneho vozidla
- riadenie heterogénneho vozidla tak, aby sledovalo získanú trajektóriu

Úlohu hľadania optimálnej trasy sme si podrobne rozobrali v kapitolách 4.2 a 4.3. Teraz si

Algoritmus 2 Proces plánovania trasy heterogénneho vozidla pri pohybujúcich sa stanovištiach

Vstupy: pozícia nosiča q_{s}, pozícia agilnej časti q_{sh}, pozície stanovíšť q_i pre $i = 1 \ldots n$, cieľová pozícia q_{f}, perióda vzorkovania ΔT

Výstupy: získanie optimálnej trajektórie pre nosič $d_{\mathrm{r,c}}$ a pre agilné vozidlo $d_{\mathrm{r,h}}$ s podmienkou navštívenia všetkých stanovíšť q_i agilnou časťou heterogénneho vozidla a dosiahnutia cieľovej pozície q_{f} v optimálnom čase

$T = 0$
while nebude nosič v cieľovej pozícii q_{f} **do**
  zmeraj aktuálnu polohu nosiča q_{s} a agilnej časti q_{sh}
  if agilné vozidlo je súčasťou nosiča ($q_{\mathrm{s}} = q_{\mathrm{sh}}$) **then**
    nájdi optimálnu trajektóriu pre heterogénne vozidlo vyriešením problému (4.46)
  else
    nájdi optimálnu trajektóriu pre heterogénne vozidlo s rozličnými štartovacími pozíciami jeho častí vyriešením problému definovanom v (4.48)
  end if
  vyextrahuj trajektóriu pre nosič a agilnú časť
  sleduj vyextrahovanú trajektóriu pre nosič a agilnú časť po dobu ΔT
  ulož prejdenú trasu nosiča do $d_{\mathrm{r,c}}$ a agilného vozidla do $d_{\mathrm{r,h}}$
  for $i = 1, \ldots, n$ **do**
    if agilné vozidlo malo na prejdenej trajektórii stanovíšte q_i **then**
      označ stanovište q_i ako navštívené
    else
      priraď novú pozíciu stanovišťu q_i
    end if
  end for
  $T = T + \Delta T$
end while
return $d_{\mathrm{r,c}}$, $d_{\mathrm{r,h}}$

popíšeme ako vyriešime problém sledovania nájdenej trajektórie heterogénnym vozidlom.

4.5 Sledovanie optimálnej trajektórie

Máme optimálnu trajektóriu, ktorá je plánovanou trasou pre heterogénne vozidlo. Našim cieľom je sledovanie tejto trajektórie, pričom chceme zohľadniť dynamiku vozidla a akčné členy, ktoré naň počas pohybu pôsobia. Keďže sa naše heterogénne vozidlo skladá z dvoch častí, potrebujeme opísať matematický model pre každú z nich, čiže aj pre nosič, aj pre agilné vozidlo.

V praxi musíme pri riadení heterogénneho vozidla počítať aj s nepredvídateľnými situáciami, ktoré ovplyvnia riadenie a konečný výsledok nemusí zodpovedať želanému stavu. V našom príklade s loďou a helikoptérou sa môže stať, že pri silnom vetre v prípade helikoptéry, alebo prúde či veľkých vlnách pri lodi, sa nám vozidlo vychýli z dráhy a skončí na inej pozícii ako sme si želali. V takom prípade by helikoptéra nenavštívila želaný bod alebo by sa loď s helikoptérou nemuseli vôbec stretnúť. Aby sme mohli prípad nepredvídateľnosti nasimulovať, musíme ho zapracovať do matematického modelu.

Inou záležitosťou je, ak nám nastane zmena vo vnútri modelu dôsledkom zmeny v riadenom objekte, o ktorej hovoríme ako o neurčitosti. Keď sa pozrieme opäť na náš príklad s loďou a helikoptérou, náš matematický model predpokladá pri riadení s konštatnou hmotnosťou vozidla. Avšak za predpokladu, že helikoptéra opustí loď, hmotnosť lode sa zmení. Ak predpokladáme, že motory lode posúvajú loď konštatnou silou, náhla zmena hmotnosti môže spôsobiť vychýlenie lode zo želanej dráhy.

Je viacero možností, aký predikčný model zvoliť na formuláciu prediktívneho riadenia. Je možné použiť napríklad lineárne, nelineárne, či fuzzy modely. My sme sa rozhodli použiť lineárny model z dôvodu konvexnej formulácie prediktívneho riadenia a dostatočnej všeobecnosti. Uvažujeme teda dva lineárne systémy v diskrétnej časovej oblasti, opísané stavovými modelmi v nasledujúcom tvare:

$$x_{\mathrm{c}}(t+\Delta t) = A_{\mathrm{c}}(m_{\mathrm{c}})x_{\mathrm{c}}(t) + B_{\mathrm{c}}(m_{\mathrm{c}})u_{\mathrm{c}}(t) + Ew_{\mathrm{c}}(t), \tag{4.52a}$$

$$x_{\mathrm{h}}(t+\Delta t) = A_{\mathrm{h}}(m_{\mathrm{h}})x_{\mathrm{h}}(t) + B_{\mathrm{h}}(m_{\mathrm{h}})u_{\mathrm{h}}(t) + Ew_{\mathrm{h}}(t), \tag{4.52b}$$

kde $x_{\mathrm{c}}(t), x_{\mathrm{h}}(t) \in \mathbb{R}^{n_x}$ je stavový vektor v čase t a $u_{\mathrm{c}}(t), u_{\mathrm{h}}(t) \in \mathbb{R}^{n_u}$ je vektor riadiacich príkazov, m_c je pomerná hmotnosť lode a m_h je pomerná hmotnosť helikoptéry. Matica E určuje, ktorý zo stavov nám porucha ovplyvňuje a $w_{\mathrm{c}}(t), w_{\mathrm{h}}(t)$ je veľkosť poruchy v čase t. Stavy a vstupy podliehajú nasledovným ohraničeniam

$$x_{\mathrm{c}}(t) \in \mathbb{X}_{\mathrm{c}},\ u_{\mathrm{c}}(t) \in \mathbb{U}_{\mathrm{c}},\ \forall t \geq 0, \tag{4.53a}$$

$$x_{\mathrm{h}}(t) \in \mathbb{X}_{\mathrm{h}},\ u_{\mathrm{h}}(t) \in \mathbb{U}_{\mathrm{h}},\ \forall t \geq 0, \tag{4.53b}$$

kde $\mathbb{X}_\mathrm{c}, \mathbb{X}_\mathrm{h}, \mathbb{U}_\mathrm{c}$ a $\mathbb{U}_\mathrm{h}$ sú polytopy príslušnej dimenzie. Ďalej predpokladáme, že páry $(A_\mathrm{c}, B_\mathrm{c})$ a $(A_\mathrm{h}, B_\mathrm{h})$ sú riaditeľné a hodnoty stavov $x_\mathrm{c}(t), x_\mathrm{h}(t)$ sú dostupné v každej perióde vzorkovania.

Pre systémy (4.52) a ohraničenia (4.53) uvažujeme jednoduchšiu verziu MPC problému bez poruchy:

$$U_N^* = \arg\min_{U_N^*} \sum_{k=0}^{N-1} \|Q_\mathrm{p}(p_k - p_{\mathrm{ref},k})\|_r + \|Q_\mathrm{u} u_k\|_r \tag{4.54a}$$

$$\text{v.n. } x_0 = x(t), \tag{4.54b}$$

$$x_k = Ax_k + Bu_k, \quad k = 0, \ldots, N-1, \tag{4.54c}$$

$$p_k = Cx_k, \quad k = 1, \ldots, N-1, \tag{4.54d}$$

$$x_k \in \mathbb{X}, \quad k = 0, \ldots, N, \tag{4.54e}$$

$$u_k \in \mathbb{U}, \quad k = 0, \ldots, N-1, \tag{4.54f}$$

kde $U_N^* = (u_0^T, \ldots, u_{N-1}^T)^T$ je postupnosť optimálnych akčných zásahov, $p_{\mathrm{ref},k}$ je referencia na polohu vozidla v k-tom kroku predikčného horizontu, x_k a u_k predstavujú predikovaný stav a vstup v čase $t + k$, určený na základe nameraného aktuálneho stavu $x(t)$, p_k je polohový vektor, kde $k = 0, \ldots, N-1$. Matica C je konštanta, prostredníctvom ktorej získavame zo stavového vektoru x_k, polohový vektor p_k. Ďalej, Q_p a Q_u sú penalizačné matice príslušných rozmerov s $Q_\mathrm{p} \succeq 0$, $Q_\mathrm{u} \succ 0$, a index r predstavuje normu, v ktorej korešpondujúce hodnoty by mali byť minimalizované (ak $r = 2$, potom môžeme ekvivalentne minimalizovať $\|Qz\|_2^2 = z^T Q z$ namiesto $\|Qz\|_2 = \sqrt{z^T Q z}$). V prípade použitia 2-normy sa problém definovaný v (4.54) elementárnymi úpravami prepíše do tvaru (2.32) a môžeme ho riešiť s niektorým zo softvérových balíkov popísaných v sekcii 2.7, pričom softvér na riešenie použije metódu aktívnych množín popísanú v kapitole 2.3.1. V prípade použitia 1-normy alebo ∞-normy sa problém definovaný v (4.54) elementárnymi úpravami prepíše do tvaru (2.9), pričom ak použijeme softvérový balík, ten na riešenie použije simplexovú metódu popísanú v kapitole 2.2.2. Spôsob prevedenia vzťahu (4.54) do príslušného tvaru v závislosti od normy, je popísaný v kapitole 3.3.

Problém definovaný v (4.54) je formulovaný všeobecne pre obe časti heterogénneho vozidla. Preto pre formuláciu problému pre nosič musíme zameniť premenné x,u,A,Q_p, Q_u, $\mathbb{X}$, $\mathbb{U}$ za x_c, u_c, A_c, Q_pc, Q_uc, $\mathbb{X}_\mathrm{c}$, $\mathbb{U}_\mathrm{c}$ a pre formuláciu problému pre agilné vozidlo za x_h,u_h, A_h, Q_pc, Q_uc, $\mathbb{X}_\mathrm{h}$, $\mathbb{U}_\mathrm{h}$.

Referenciu $p_{\mathrm{ref},k}$ vypočítame pomocou Algoritmu 3. Na výpočet potrebujeme štartovaciu pozíciu $[p_\mathrm{sx}, p_\mathrm{sy}]^T$, cieľovu pozíciu $[p_\mathrm{fx}, p_\mathrm{fy}]^T$, rýchlosť vozidla v, periódu vzorkovania Δt. Proces výpočtu realizujeme na celom predikčnom horizonte N. Ako prvé vypočítame dráhu d, ktoré je vozidlo schopné prejsť za čas t pri rýchlosti v. Ak je dráha d, ktorú je vozidlo schopné prejsť dlhšia ako vzdialenosť cieľového bodu $[p_\mathrm{fx}, p_\mathrm{fy}]^T$ od aktuálnej

pozície $[p_{\mathrm{sx}}, p_{\mathrm{sy}}]^T$, priradíme referencii $[p_{\mathrm{rx}}, p_{\mathrm{ry}}]^T$ cieľovú pozíciu $[p_{\mathrm{fx}}, p_{\mathrm{fy}}]^T$. Ak je kratšia, vypočítame v kroku 6 a 7 referenciu na polohu, ktorú v kroku 9 uložíme referenciu na polohu pre k-ty krok predikčného horizontu a v kroku 10 vypočítanú referenciu určíme ako novú štartovaciu polohu pre ďalšiu iteráciu. Na záver vrátime referenčné hodnoty na polohu pre celý predikčný horizont $p_{\mathrm{ref},0}, \ldots, p_{\mathrm{ref},N-1}$.

Algoritmus 3 Výpočet referencie pri prediktívnom riadení

Vstupy: štartovacia pozícia $[p_{\mathrm{sx}}, p_{\mathrm{sy}}]^T$, cieľová pozícia $[p_{\mathrm{fx}}, p_{\mathrm{fy}}]^T$, rýchlosť v, perióda vzorkovania Δt, predikčný horizont N

Výstupy: referencia na polohu p_{ref}

1: **for** $k = 1, \ldots, N$ **do**
2: $d = \Delta t v$
3: **if** $d > \|[p_{\mathrm{sx}}, p_{\mathrm{sy}}]^T - [p_{\mathrm{fx}}, p_{\mathrm{fy}}]^T\|$ **then**
4: $[p_{\mathrm{rx}}, p_{\mathrm{ry}}]^T = [p_{\mathrm{fx}}, p_{\mathrm{fy}}]^T$
5: **else**
6: $a = [p_{\mathrm{fx}} - p_{\mathrm{sx}}, p_{\mathrm{fy}} - p_{\mathrm{sy}}]^T$
7: $[p_{\mathrm{rx}}, p_{\mathrm{ry}}]^T = [p_{\mathrm{sx}}, p_{\mathrm{sy}}]^T + \frac{d}{\|a\|} A$
8: **end if**
9: $p_{\mathrm{ref},k} = [p_{\mathrm{rx}}, p_{\mathrm{ry}}]^T$
10: $[p_{\mathrm{sx}}, p_{\mathrm{sy}}]^T = [p_{\mathrm{rx}}, p_{\mathrm{ry}}]^T$
11: **end for**
12: **return** $p_{\mathrm{ref},0}, \ldots, p_{\mathrm{ref},N-1}$

4.5.1 Prípadová štúdia

V kapitole 4.3 sme si naformulovali problém zmiešaného celočíselné problému nad kužeľom druhého rádu (4.46) pre hľadanie optimálnych bodov vzletov a pristátí τ_i, ℓ_j (ktoré predstavujú zároveň trajektóriu nosiča) a optimálneho poradia návštev tak, aby bola trajektória heterogénneho vozidla minimálna. Ukázali sme si formuláciu, kedy predpokladáme, že agilné vozidlo je na nosiči, ale aj formuláciu, kedy je heterogénne vozidlo v separátnom stave (4.48). V kapitole 4.5 sme naformulovali problém sledovania optimálnej trajektórie za použitia prediktívneho riadenia (4.54). V tejto kapitole si uvedieme príklad hľadania optimálneho navigačného plánu pre heterogénne vozidlo, kde si budeme musieť poradiť s pohybujúcimi sa stanovišťami na obletenie, pričom využijeme všetky spomínané formulácie (4.46, 4.48, 4.54).

Problémy boli naformulované s použitím YALMIP (Löfberg, 2004), ktorý automaticky určuje optimálne hodnoty M v (4.22)−(4.23). Výsledný MI-SOCP i MPC problém boli

riešené s GUROBI. Všetky výpočty boli uskutočnené na počítači s 3.2 GHz procesorom a 4 GB pamäťou.

Príklad

Uvažujeme prípad s 5 bodmi q_i, $i = 1, \ldots, 5$, ktorých súradnice v 2-dimenzionálnom Euklidovom priestore sú uvedené v Tabuľke 4.7. Heterogénne vozidlo tak ako v ostatných doteraz uvedených príkladoch pozostáva z nosiča – lode, ktorá cestuje konštantnou rýchlosťou $v_{\mathrm{c}} = 18$ km/h a má neobmedzený dosah. Nosič nesie agilné vozidlo – helikoptéru, ktorej rýchlosť je $v_{\mathrm{h}} = 90$ km/h, ale má obmedzený čas pobudnutia vo vzduchu $t_{\mathrm{h,max}} = 21$ min. Loď začína v bode $q_{\mathrm{s}} = [0,\ 0]^T$ s helikoptérou na palube a jej cieľom je skončiť, po obletení všetkých piatich stanovíšť helikoptérou, v bode $q_{\mathrm{f}} = [50,\ 0]^T$.

Pri probléme prediktívneho riadenia (4.54) máme matematický model pre systémy (4.52a, 4.52b), ktoré prevedieme na kvadratický alebo lineárny problém podľa toho ako nastavíme normu p a vyriešime ich GUROBI solverom. Systémy predstavujú planárny pohyb objektu. Pre nosič aj helikoptéru uvažujeme model so 4 stavmi v tvare dvoch dvojitých integrátorov, ktoré korešpondujú s pozíciou objektu v x-ovej a y-ovej osi, spoločne s príslušnou rýchlosťou. Riadený vstup predstavuje vektor akcelerácie v dvoch osiach. Náš stavový model pre loď (4.52a) teda bude vyzerať nasledovne:

$$\begin{bmatrix} \dot{p}_{\mathrm{cx}} \\ \dot{p}_{\mathrm{cy}} \\ \dot{v}_{\mathrm{cx}} \\ \dot{v}_{\mathrm{cy}} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_{\mathrm{cx}} \\ p_{\mathrm{cy}} \\ v_{\mathrm{cx}} \\ v_{\mathrm{cy}} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{1}{m_{\mathrm{c}}} & 0 \\ 0 & \frac{1}{m_{\mathrm{c}}} \end{bmatrix} \begin{bmatrix} u_{\mathrm{cx}} \\ u_{\mathrm{cy}} \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} w_x \\ w_y \end{bmatrix}, \tag{4.55}$$

kde p_{cx} a p_{cy} sú pozície objektu v x-ovej a y-ovej osi, v_{cx} a v_{cy} sú príslušné rýchlosti. Premenné u_{cx}, u_{cy} predstavujú akceleráciu, ktoré sú považované za ovládacie vstupy a m_c je pomerná hmotnosť lode. Tá sa mení vzhľadom na to, či je helikoptéra na lodi alebo vo vzduchu.

Stavový model pre helikoptéru (4.52b) je v obdobnom tvare ako model pre nosič. Pre úplnosť si ho zobrazíme.

$$\begin{bmatrix} \dot{p}_{\mathrm{hx}} \\ \dot{p}_{\mathrm{hy}} \\ \dot{v}_{\mathrm{hx}} \\ \dot{v}_{\mathrm{hy}} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_{\mathrm{hx}} \\ p_{\mathrm{hy}} \\ v_{\mathrm{hx}} \\ v_{\mathrm{hy}} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \frac{1}{m_{\mathrm{h}}} & 0 \\ 0 & \frac{1}{m_{\mathrm{h}}} \end{bmatrix} \begin{bmatrix} u_{\mathrm{hx}} \\ u_{\mathrm{hy}} \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} w_x \\ w_y \end{bmatrix}. \tag{4.56}$$

Všimnime si, že systémy (4.55) a (4.56) sú v spojitom čase, a preto je ich potreba sdiskretizovať s vhodným výberom vzorkovacej periódy.

Predikčný horizont sme nastavili na $N = 20$, normu $p = 2$, penalizačné matice pre akčný zásah $Q_{\mathrm{cu}} = Q_{\mathrm{hu}} = I_{2\times 2}$ a penalizačné matice pre stav $Q_{\mathrm{cp}} = Q_{\mathrm{hp}} = I_{2\times 2}$. Nechali

sme teda rovnakú prioritu sledovaniu trasy aj výkyvu akčných zásahov. Periódu vzorkovania Δt v prediktívnom riadení sme nastavili na 1 sekundu a perióda opätovného optimalizovania trasy ΔT je 10 minút. Pri zapracovaní neurčitosti sme v prípade helikoptéry na palube nastavili $m_c = 1.1$ a teda sme predpokladali, že helikoptéra zväčší hmotnosť lode o jednu desatinu pôvodnej hmotnosti. V prípade ak je helikoptéra vo vzduchu platí $m_c = 1$ a $m_h = 1$. Poruchu w_x a w_y sme v každom kroku t určili náhodne v rozpätí od -0.005 do 0.005, čo môžeme brať ako vychýlenie pozície heterogénneho vozidla v kilometroch.

Stanovištia majú okrem polohy určenej súradnicami x a y ďalšie dva atribúty – rýchlosť a cieľový bod, ktorý určuje smer pohybu bodu. V prípade, že stanovište dosiahne cieľový bod, zastaví sa. Táto vlastnosť predstavuje určitú nepredvídateľnosť ich pohybu. Body sa presúvajú konštantnou rýchlosťou až do doby, kým dosiahnu svoj cieľový bod, alebo do konca simulácie. Rýchlosť každého bodu sa pohybuje v rozpätí od 2 do 5 km/h. Všetky parametre sme určili pred simuláciou náhodne, pričom stavový priestor bodov pre pohyb bol obmedzený na štvorec o veľkosti hrany 200 km. Počiatočné nastavenia stanovíšť obsahuje Tabuľka 4.7.

Tabuľka 4.7: Umiestnenie bodov $q_1, \ldots, q_5$ a ich konfigurácie. Všetky hodnoty sú v uvedené v kilometroch.

počiatočná poloha [km]	rýchlosť [km/h^2]	cieľová pozícia [km]
$[5, 22]^T$	2	$[-13, 92]^T$
$[26, 45]^T$	4	$[36, -31]^T$
$[29, 12]^T$	2	$[92, -44]^T$
$[36, 18]^T$	2	$[73, 92]^T$
$[48, 45]^T$	4	$[85, 21]^T$

Podobné ohraničenia na priestor platia aj pre heterogénne vozidlo. Ich stavový vektor sa skladá z polohy a rýchlosti v oboch osiach. Množiny ohraničení pre stav majú nasledovný tvar:

$$\mathbb{X}_c = \{x \mid \begin{pmatrix} -100 \\ -100 \\ -10 \\ -10 \end{pmatrix} \leq x \leq \begin{pmatrix} 100 \\ 100 \\ 45 \\ 45 \end{pmatrix}\}, \tag{4.57a}$$

$$\mathbb{X}_h = \{x \mid \begin{pmatrix} -100 \\ -100 \\ -35 \\ -35 \end{pmatrix} \leq x \leq \begin{pmatrix} 100 \\ 100 \\ 200 \\ 200 \end{pmatrix}\}, \tag{4.57b}$$

kde $-100, 100$ sú hodnoty pre polohu uvedené v kilometroch a $-10, 45, -35, 200$ sú hodnoty pre rýchlosť v kilometroch za hodinu. Akčný člen obsahuje zrýchlenie v oboch osiach. Množiny ohraničení pre akčné zásahy sú v tvare:

$$\mathbb{U}_{\mathrm{c}} = \{u \,|\, \begin{pmatrix} -0.005 \\ -0.005 \end{pmatrix} \leq u \leq \begin{pmatrix} 0.005 \\ 0.005 \end{pmatrix}\}, \tag{4.58a}$$

$$\mathbb{U}_{\mathrm{h}} = \{u \,|\, \begin{pmatrix} -0.025 \\ -0.025 \end{pmatrix} \leq u \leq \begin{pmatrix} 0.025 \\ 0.025 \end{pmatrix}\}, \tag{4.58b}$$

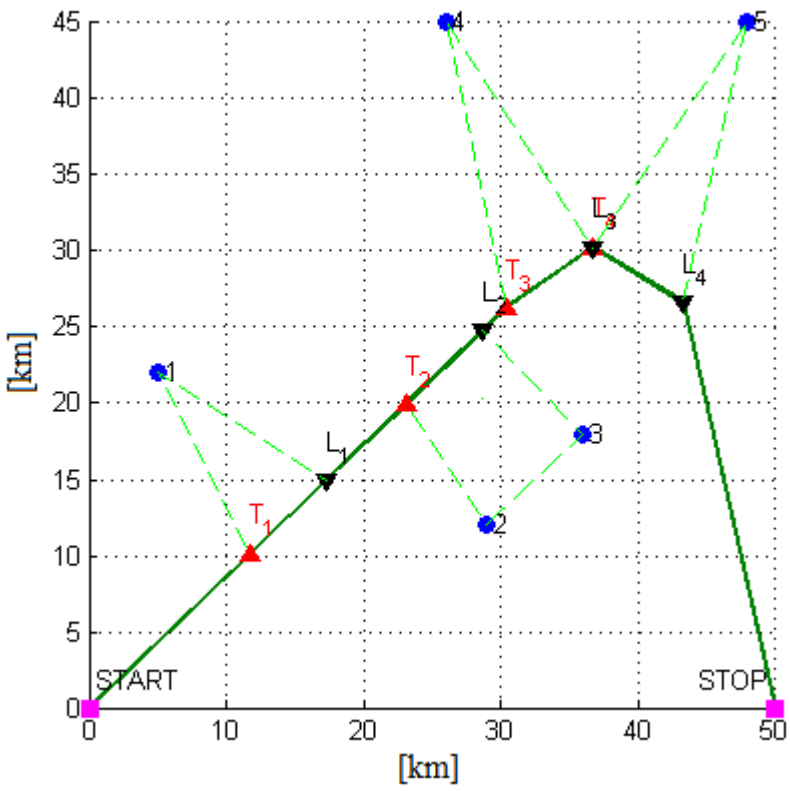
kde sú hodnoty uvedené v kilometroch za hodinu na druhú a predstavujú maximálne a minimálne možné zrýchlenie za čas Δt.

Na Obr. 4.7 vidíme počiatočnú optimálnu trasu, ktorú by sme sledovali heterogénnym vozidlom za predpokladu, že by boli stanovištia statické. Plnou čiarou je zobrazená trajektória lode a prerušovaná čiara určuje trasu pre helikoptéru. Body τ_1 až τ_4 predstavujú body vzletov, body ℓ_1 až ℓ_4 sú miesta pristání a plné krúžky sú stanovištia na obletenie, označené indexom, v akom poradí budú navštívené. Podľa plánu mal nosič prejsť $82,63$ km a celkové trvanie cesty by bolo 4 hodiny a 35 minút. Loď, na palube s helikoptérou, však nasledovala trasu iba po dobu $\Delta T = 600$s, po ktorých sa hľadala optimálna trasa opäť, s aktuálnou polohou nosiča. Keďže perióda vzorkovania pri prediktívnom riadení $\Delta t = 1$s, proces optimalizácie riadenia prebehol za túto dobu 600-krát, medzitým sa zmenila aj poloha jednotlivých stanovíšť a trajektória sa behom procesu neustále mení.

Obr. 4.8 zobrazuje situáciu, kedy máme uprostred optimalizácie helikoptéru vo vzduchu a nová optimálna trasa sa hľadá pomocou formulácie (4.48). Poloha helikoptéry je v tomto prípade označená ako τ_1 a ak si všimneme indexy stanovíšť, postrehneme, že pod helikoptérou je na rovnakej súradnici aj naše prvé fiktívne stanovište, ktoré sa vyžaduje pre potrebu optimalizácie.

Splnenie celej misie pre heterogénne vozidlo sa nakoniec podarilo podľa optimálneho plánu za kratší čas, ako bolo pôvodne naplánované. Nosič mal optimálnu trasu dlhú $80,63$ km, čiže o 2 km menej ako mal prejsť podľa prvého optimálneho plánu. V skutočnosti však kvôli zapracovanej poruche a neurčitosti prešiel presne $92,81$ km, čo je približne o 15% dlhšia trasa ako naplánovaná. Celkové trvanie misie bolo 4 hodiny a 28 minút. Obletenie piatich stanovíšť vyžadovalo 4 letové fázy, pričom stanovište 2 a 3 bolo obletené v jednej fáze rovnako ako v prvom optimálnom pláne. Kompletná finálna trajektória heterogénneho vozidla je na Obr. 4.9. Jej prejdenie vyžadovalo celkovo 20 riešení problémov pre nájde optimálnej trasy, z čoho pri 11 bola helikoptéra na lodi a pri 9 vo vzduchu, čo je takmer polovica z celkového počtu optimalizácii trasy. Graf na Obr. 4.10 prehľadne zobrazuje s koľkými stanovišťami prebehli jednotlivé optimalizácie behom misie.

Na Obr. 4.11 je zobrazená časť trajektórie heterogénneho vozidla, kde vidíme odchýlku skutočného pohybu od naplánovanej trasy. Hodnoty na osiach sú uvedené v kilometroch.

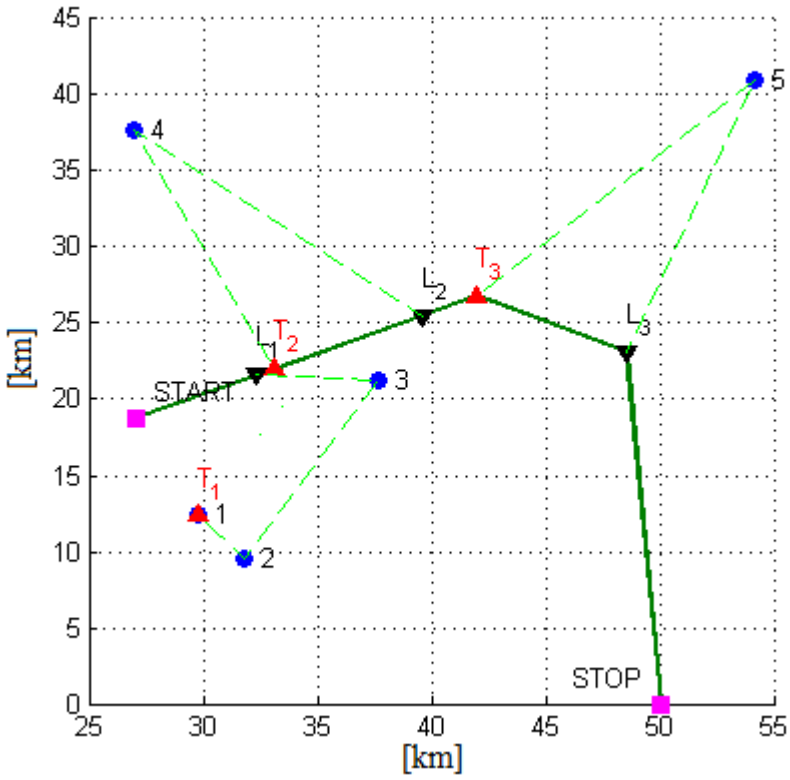


Obr. 4.7: Optimálna trajektória heterogénneho vozidla v počiatočnom stave. Plné čiary predstavujú trasu nosiča, prerušované čiary reprezentujú trasu pre agilné vozidlo. q_i sú navštevované stanovištia, τ_i su body vzletov a ℓ_j reprezentuje body pristátí helikoptéry na palube lode. Obe osy sú uvedené v kilometroch.

Priemerná odchýlka na celej trase od plánovanej trajektórie bola $7,9$ m. V prípade, že by sme chceli dosiahnuť menšie výkyvy akčných zásahov, alebo menšiu priemernú odchýlku, museli by sme zmeniť parametre penalizačných matíc Q_{u} a Q_{p}. Treba však pripomenúť, že zlepšenie jedného parametra sa odrazí na zhoršení druhého a tak znížením odchýlky trasy, dosiahneme väčšie výkyvy akčných zásahov a naopak.

Analýza vplyvu normy v prediktívnom riadení

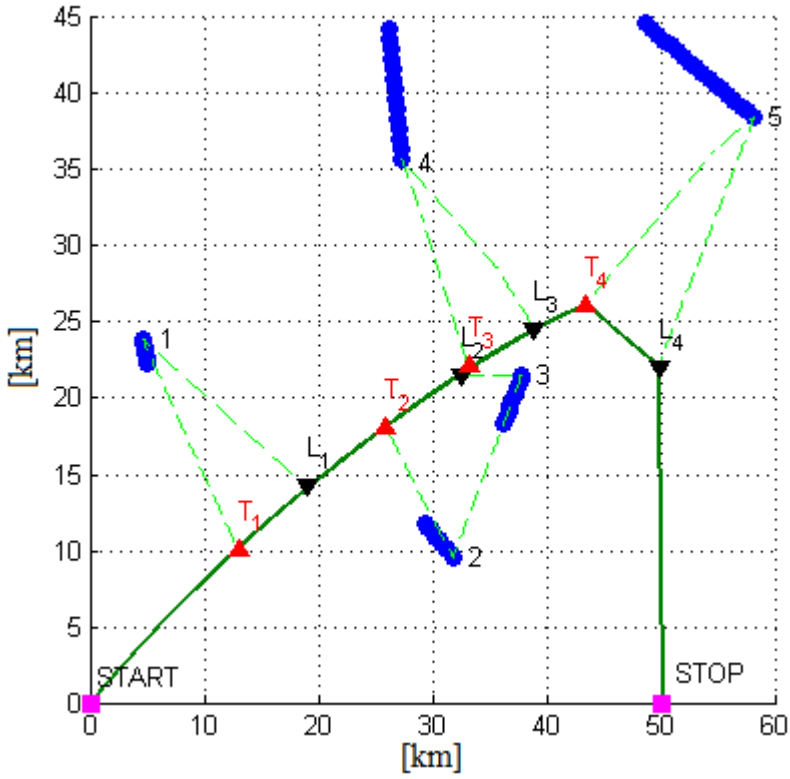
V príklade uvedenom vyššie v kapitole 4.5.1 sme použili pri prediktívnom riadení nastavenie normy $r = 2$. V tejto kapitole si ukážeme ako sa nám mení priebeh riadenia v prípade použitia aj noriem $r = 1$ a $r = \infty$. V závislosti od zvolenej normy riešime rôzne MPC problémy. Tieto rozdiely spolu s určením účelovej funkcie sme si uviedli v kapitole 3.3.



Obr. 4.8: Optimálna trajektória pre heterogénne vozidlo s rozličnými štartovacími pozíciami jeho častí v priebehu simulácie. Plné čiary predstavujú trasu nosiča, prerušované čiary reprezentujú trasu pre agilné vozidlo. q_i sú navštevované stanovištia, τ_i su body vzletov a ℓ_j reprezentuje body pristátí helikoptéry na palube lode. Obe osy sú uvedené v kilometroch.

Na analýzu sme si vzali príklad z kapitoly (4.5.1) s identickými parametrami a modelom pre loď.

Tabuľka 4.8 nám porovnáva celkovú odchýlku polohy v kilometroch a spotrebu energie $\sum_{u=1} T\|u_i - u_{i-1}\|$. Suverénne najnižšie výkyvy akčných zásahov nám poskytuje 2-norma, pričom má však aj najväčšie vychýlenie z plánovanej dráhy. Toto vychýlenie je však porovnateľné s ostatnými normami, kým hodnota výkyvu akčného zásahu je výrazne lepšia. Časovo je najvýhodnejšia 1-norma.

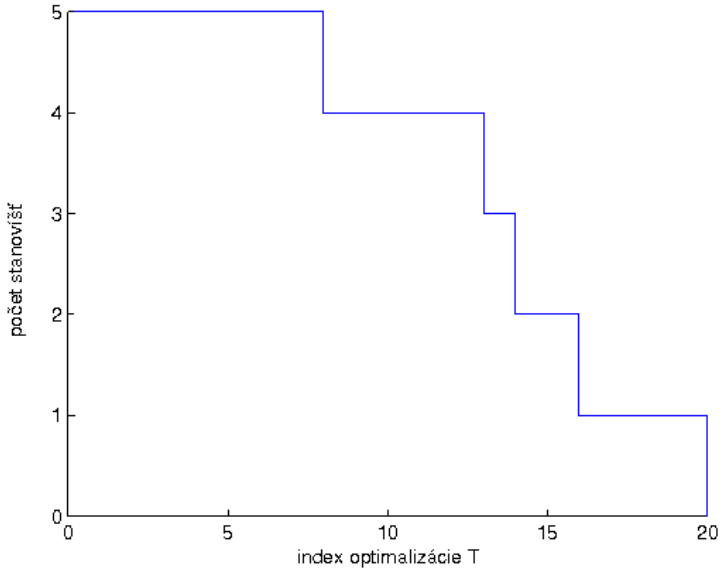


Obr. 4.9: Trajektória heterogénneho vozidla za celý priebeh misie. Plné čiary predstavujú trasu nosiča, prerušované čiary reprezentujú trasu pre agilné vozidlo. Modrou farbou sú zobrazené dráhy pohybu stanovíšť q_i, τ_i su body vzletov a ℓ_j reprezentuje body pristátí helikoptéry na palube lode. Obe osy sú uvedené v kilometroch.

4.6 Zhrnutie

V kapitole 4 sme sa zaoberali problémom plánovania trasy pre heterogénne vozidlo. Pod heterogénnym vozidlom rozumieme systém, ktorý pozostáva z dvoch vozidiel. Prvé – nosič, má nízku maximálnu rýchlosť, ale je schopné ďalekej cesty, pričom nesie druhé vozidlo. Kým druhé - agilné vozidlo, je schopné rýchleho presunu, avšak iba na krátku vzdialenosť. Naším cieľom bolo obletieť agilným vozidlom pole stanovíšť a následne sa dostať do cieľovej destinácie tak, aby trvanie tejto misie trvalo minimálny čas. Pri navrhovaní ich riešenia sme sa chceli vyhnúť heuristickým metódam ponúkajúcim suboptimálne riešenie. Stanovený problém sme podľa počiatočných podmienok rozdelili na tri rôzne problémy.

V kapitole 4.2 sme sa zoberali scenárom, kedy máme poradie návštev jednotlivých stanovíšť pevne dané. Najprv sme uviedli riešenie Problému (4.1.2) pomocou zmiešanej celo-



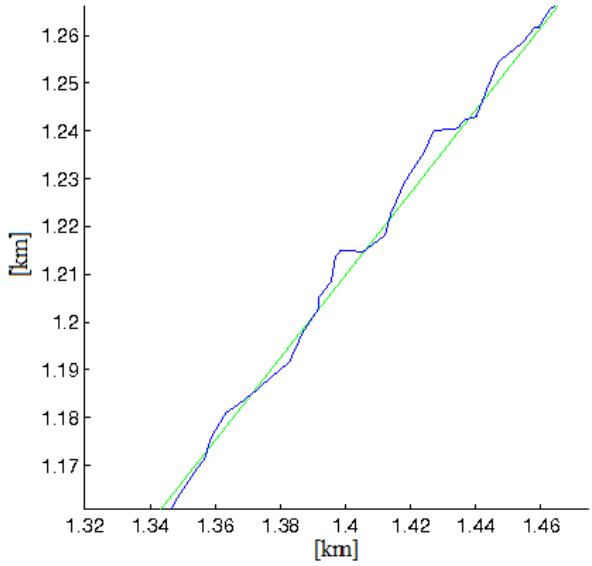
Obr. 4.10: Počet stanovíšť v jednotlivých krokoch simulácie. Na vertikálne osi máme počet stanovíšť, horizontálna os predstavuje časový krok T.

Tabuľka 4.8: Porovnanie prediktívneho riadenia pri norme 1,2 a ∞.

norma	odchýlka [km]	spotreba energie	výpočtový čas [s]
norma 1	3.21	4.70	$4,77$
norma 2	4.95	0.94	$5,23$
norma ∞	3.25	4.94	$6,31$

číselnej nelineárnej formulácie (MI-NLP) tak ako bolo navrhnuté v (Garone et al., 2012). Hlavné obmedzenie tejto formulácie optimalizačného problému definovaného v (4.13) pramení z jej výpočtovej zložitosti. Konkrétne, autori v (Garone et al., 2012) demonštrovali, že problém je riešiteľný, v primeranom čase, avšak iba pre malé množstvo stanovíšť. Následne sme si v kapitole 4.2.2 ukázali ako je možné *ekvivalentne* preformulovať MI-NLP (4.13) na problém zmiešaného celočíselného programovania nad kužeľom druhého rádu (MI-SOCP), ktorý je možné efektívne riešiť aj pre *stovky* bodov. Riešenie sme uviedli aj pre heterogénne vozidlo s odlišnými štartovacími pozíciami jeho častí vo formulácii 4.34. V kapitole 4.2.2 sme sa zaoberali aj rozšírenými scenármi, kedy by nás mohlo zaujímať napríklad obmedzenie počtú pristátí a vzletov, prípadne minimalizovanie spotreby paliva helikoptéry.

Výsledky, ktoré sme dosiahli navrhnutou formuláciou MI-SOCP sme využili pre rieše-

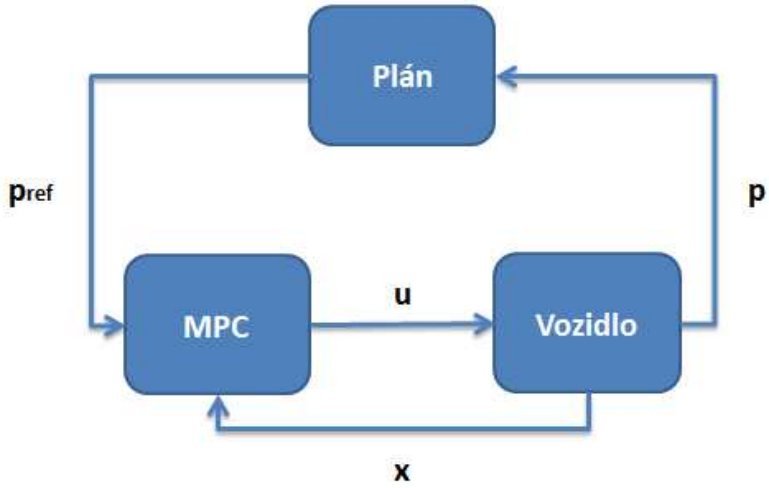


Obr. 4.11: Vychýlenie heterogénneho vozidla oproti plánovanej trase. Zelená čiara určuje plánovanú trasu. Modrá predstavuje skutočnú dráhu vozidla. Hodnoty oboch osí sú v kilometroch.

nie oveľa komplexnejšieho Problému 4.1.3. V kapitole 4.3 sme si ukázali ako nájsť optimálnu cestu za predpokladu, že poradie bodov nie je dané, ale môže byť optimalizované za účelom znížiť čas trvania misie. Formulácia (4.46) nám dokázala optimalizovať poradie návštev a pritom aj minimalizovať trvanie misie pre malý počet stanovíšť za rozumný čas. Avšak s rastúcim počtom stanovíšť sme nám trvanie výpočtu exponenciálne narastalo a pre 9 stanovíšť sme dosiahli výsledok až za takmer hodinu. Preto sme zvolili aj druhý prístup pre riešenie Problému 4.1.3 s využitím TSP algoritmu (Miller et al., 1960), kedy sme zvládli riešenie problému s rovnakým počtom stanovíšť za niekoľko sekúnd. Zaplatili sme za to však mierou suboptimality, ktorá sa pohybovala vo väčšine prípadov pod 1%, ale v najhoršom prípade mohla dosiahnuť až 60%.

Problém 4.1.4 sme riešili v kapitole 4.4, kde sme si popísali riešenie ako nájsť optimálnu trasu v prípade, že sa nám body pohybujú a teda ako optimalizovať trajektóriu pohybu heterogénneho vozidla v reálnom čase podľa aktuálnej situácie. Samotné sledovanie trajektórie prostredníctvom prediktívneho riadenia sme vysvetlili v kapitole 4.5. Diagram systému sledovania trasy podľa optimálneho plánu zobrazuje Obr. 4.12, kde pod vozidlom rozumieme nosič, alebo agilné vozidlo.

Na príklade v kapitole 4.5.1 sme ukázali, že vďaka dosiahnutým výsledkom pri výpočte



Obr. 4.12: Diagram systému sledovania plánovanej trasy prostredníctvom prediktívneho riadenia. Vozidlo predstavuje nosič alebo agilnú časť.

optimálnej trajektórie pre heterogénne vozidlo môžeme použiť plánovanie aj priamo v reálnom čase počas sledovania optimálnej trasy a to aj v prípade, že by išlo o riadenie robotického vozidla. Použitie v praxi je však čiastočne obmedzené vzhľadom na exponenciálne rastúci výpočtový čas pri zvyšovaní množstva stanovíšť na obletenie. V našom príklade by výpočtový čas nesmel presiahnuť 10 minút a teda s hardvérom, ktorý sme mali k dispozícii by sme úspešne zvládli misiu maximálne s ôsmymi stanovišťami. Poradiť by sme si mohli s predĺžením periódy vzorkovania pre hľadanie plánovanej trajektórie, alebo by sme sa mohli uspokojiť so suboptimálnym riešením popísaným v kapitole 4.3.2, vďaka čomu by sme zvládli optimalizovať trasu aj pre desiatky stanovíšť, a to stále v reálnom čase.

Záver a prínosy dizertačnej práce

V dizertačnej práci sme sa zaoberali problémom smerovania dopravných trás. Keďže je v poslednej dobe riešenie problému plánovania trasy pre jedno vozidlo v praxi nedostačujúce, venovali sme sa takzvanému heterogénnemu multi-vozidlovému systému, kde vozidlo pozostáva z viacerých častí. Nájdenie minimálnej dĺžky trasy prehľadaním všetkých možností je úloha, ktorej zložitosť narastá s pribúdajúcim počtom vrcholov superpolynomiálne, čo označujeme v kombinatorickej optimalizácii ako nedeterministický polynomiálne ťažký problém.

Na začiatku práce v kapitole 1 sme rozobrali dôkladne problematiku distribučných systémov a predovšetkým problém smerovania dopravných trás, ktorý vychádza z problému obchodného cestujúceho (TSP). Spomenuli sme niekoľko autorov, ktorí navrhli prvotné algoritmy na riešenie, pričom základom na postúpenie vo výpočtovom čase na riešenie problému, zabezpečovali rôzne heuristické metódy. Rozobrali sme aj význam neustáleho štúdia tejto problematiky a jej prínos v praxi. Pritom sme vyzdvihli, že práve v dnešnej dobe má zmysel orientovať sa na multi-vozidlový systém. Keďže v dnešnej dobe sa čoraz viac systémy automatizujú, prichádzajú požiadavky použitia robotických vozidiel. V tomto zmysle sa venujeme aj prediktívnemu riadeniu. Následne sme podrobne opísali aktuálne výskumy vo svete v danej oblasti a popísali sme si ciele, ktoré sme chceli v práci dosiahnuť.

V kapitole 2 sme sa zaoberali prehľadom základných optimalizačných úloh, pričom sme si vyjadrili všeobecný a konvexný optimalizačný problém. Do prehľadu optimalizačných úloh sme si zaradili v prvom rade lineárne programovanie, kde sme si ukázali ako riešiť ÚLP prostredníctvom simplexovej metódy. Obdobným spôsobom sme si uviedli metódu aktívnych množín, prostredníctvom ktorej riešime úlohu kvadratického programovania. V rámci kapitoly sme rozobrali aj úlohou zmiešaného celočíselného programovania. Z metód riešenia sme sa zaoberali najmä metódou vetvenia a rezov. V rámci tejto kapitoly sme

sa zaoberali aj s jednou z najviac študovaných optimalizačných úloh – problém obchodného cestujúceho. Venovali sme sa aj úlohe semidefinitného programovania a rozobrali sme tiež vlastnosti kužeľov druhého rádu ako aj definovali úlohu programovania nad kužeľmi druhého rádu.

V kapitole 3 sme opísali základy prediktívneho riadenia (MPC) ako riadenia, ktorého základná myšlienka je využitie modelu procesu na predpovedanie jeho budúceho správania a súčasnú optimalizáciu akčných zásahov. Skrátene sme si opísali lineárne modely systémov, pričom sme bližšiu pozornosť venovali stavovým modelom, ktoré sme využili neskôr pri smerovaní heterogénneho vozidla. Opísali sme si základy ohraničeného optimálneho riadenia v konečnom čase. Uviedli sme si problém prediktívneho riadenia s lineárnou a kvadratickou účelovou funkciou, ktorej použitie u nás záviselo od použitia normy pri určovaní vzdialenosti. V záverečnej časti kapitoly sme sa venovali posuvnému horizontu prediktívneho riadenia.

V kapitole 4 sme sa zaoberali problémom plánovania trasy pre heterogénne vozidlo, kde sme podrobne rozobrali riešenie problémov, ktoré predstavovali hlavný prínos tejto práce. Zaoberali sme sa prípadom multi-vozidlového systému, kde pod heterogénnym vozidlom rozumieme systém, ktorý pozostáva z dvoch vozidiel. Prvé – nosič, má nízku maximálnu rýchlosť, ale je schopné ďalekej cesty, pričom nesie druhé vozidlo a druhé – agilné vozidlo, je schopné rýchleho presunu, avšak iba na krátku vzdialenosť. Naším cieľom bolo obletieť agilným vozidlom pole stanovíšť a následne sa dostať do cieľovej destinácie tak, aby trvanie tejto misie trvalo minimálny čas. Pri navrhovaní ich riešenia sme sa chceli vyhnúť heuristickým metódam ponúkajúcim suboptimálne riešenie. Stanovený problém sme podľa počiatočných podmienok rozdelili na tri rôzne problémy. V prvom sme mali poradie návštev jednotlivých stanovíšť dané. Vychádzali sme z práce Garone et al. (2012), kde bol rozobratý podobný problém avšak riešený ako problém zmiešanej celočíselnej nelineárnej formulácie (MI-NLP). Riešenie bolo obmedzené na nízky počet stanovíšť. Pri obsiahnutí väčšieho počtu stanovíšť bola potreba uspokojiť sa so suboptimálnym riešením. Formuláciu MI-NLP sme preformulovali na problém zmiešaného celočíselného programovania nad kužeľom druhého rádu (MI-SOCP), ktorý je možné efektívne riešiť aj pre veľký počet bodov. Dosiahnuté výsledky sme využili na riešenie druhého problému, kde sme poradie návštev optimalizovali. Nová formulácia nám dokázala optimalizovať poradie návštev a pritom aj minimalizovať trvanie misie pre malý počet stanovíšť za rozumný čas. Išlo však o výpočtovo veľmi náročný proces, ktorý nás v tomto prípade takisto donútil uchýliť sa k suboptimálnemu riešeniu dosiahnutému vďaka použitiu TSP. V oboch prípadoch sme podrobne rozobrali časovú náročnosť i suboptimalitu riešenia. Na vyriešene tretieho problému sme museli nájsť optimálnu trasu v prípade, že sa nám body pohybujú a teda optimalizovať trajektóriu pohybu heterogénneho vozidla v reálnom čase podľa aktuálnej

situácie. Samotné sledovanie trajektórie bolo realizované prostredníctvom prediktívneho riadenia. Na príklade sme ukázali, že vďaka dosiahnutým výsledkom pri výpočte optimálnej trajektórie pre heterogénne vozidlo môžeme použiť plánovanie aj priamo v reálnom čase počas sledovania optimálnej trasy a to aj v prípade, že by išlo o riadenie robotického vozidla.

Hlavný prínos tejto práce z hľadiska návrhu riešenia problému plánovania trasy pre heterogénny multi-vozidlový systém je trojaký.

V prípade, ak máme pevne dané poradie návštev jednotlivých stanovíšť, sme ukázali ako nájsť optimálnu trajektóriu heterogénneho vozidla prostredníctvom riešenia optimalizačného problému zmiešaného celočíselného programovania nad kužeľom druhého rádu (4.24). Tento problém je výrazne jednoduchší ako riešenie formulácie celočíselného nelineárneho programovania, navrhnutej v Garone et al. (2012).

Druhým prínosom je rozšírená formulácia MI-SOCP procedúry (4.46), ktorá nám simultánne dokáže optimalizovať aj poradie návštev jednotlivých stanovíšť. Tento nový problém je však výpočtovo náročnejší, nakoľko sme boli nútení pridať ďalšie binárne premenné. Pre zmiernenie výpočtového zaťaženia sme rozdelili riešenie problému do dvoch fáz. Ako prvé vyriešime poradie návštev riešením klasického TSP problému, ktorý opomína heterogenitu vozidla a následne riešime optimalizačný problém pre pevne dané poradie bodov (4.24). Porovnanie v prípadovej štúdii ukázalo, že suboptimalita takého riešenia môže byť v rozsahu od 0% do 63%.

Ako posledné sme sa venovali prípadu, kedy sa nám navštevované body pohybujú. MI-SOCP formulácia bola rozšírená o ďalšie prípady, aby sme mohli optimalizovaný problém riešiť v uzavretej slučke a prostredníctvom jednoduchého prediktívneho riadenia pohybovať objektom po určenej trajektórii. Simuláciou sme tak ukázali použiteľnosť riešenia v praxi aj v prípade, že poloha cieľových stanovíšť by sa v priebehu pohybu heterogénneho vozidla menila.

Niektoré výsledky už boli publikované v recenzovaných v publikáciách:

- Klaučo, M. – Blažek, S. – Kvasnica, M.: An Optimal Path Planning Problem for Heterogeneous Multi-Vehicle Systems. International Journal of Applied Mathematics and Computer Science, 26(2), 297–308, 2016,
- Klaučo, M. – Blažek, S. – Kvasnica, M. – Fikar, M.: Mixed-Integer SOCP Formulation of the Path Planning Problem for Heterogeneous Multi-Vehicle Systems. In European Control Conference, Strasbourg, France, 1474–1479, 2014.

Dosiahnutými výsledkami sme ale v oblasti heterogénneho multi-vozidlového systému stále nič neuzavreli. V práci sme sa zaoberali prípadom, kedy dané stanovištia môže navštíviť iba agilné vozidlo. Novú formuláciu problému by sme získali v prípade, že by sme

stanovištia mohli navštevovať oboma časťami heterogénneho vozidla, teda aj nosičom. Ak by sme ďalej uvažovali heterogénne vozidlo, ktoré by obsahovalo viac ako jednu agilnú časť, výpočet pre zosúladenie viacerých dráh by určite predstavoval nový, náročný a komplexný problém. V praxi nám môže nastať aj situácia, kedy nebudeme musieť navštíviť konkrétny bod, ale postačujúce bude prísť iba do blízkej vzdialenosti k nemu, čo by nám opäť zmenilo formuláciu problému. Otvára sa teda ešte stále dostatok výziev na naviazanie výskumu obsiahnutého v tejto práci.

Literatúra

ACKERMAN, E. 2013. Amazon promises package delivery by drone: Is it for real? URL `http://spectrum.ieee.org/automaton/robotics/drones/amazon-prime-air-package-drone-delivery`, [Online; accessed October-2016].

ALIZADEH, F.; GOLDFARB, D. 2003. Second-order cone programming. *Mathematical Programming*, 95, 1, 3–51. ISSN 0025-5610.

APPLEGATE, D. 2006. *The Traveling Salesman Problem: A Computational Study*. Princeton Series in Applied Mathematics. Princeton University Press. ISBN 9780691129938.

ÅSTRÖM, K.; HÄGGLUND, T. 2006. *Advanced PID control*. ISA-The Instrumentation, Systems, and Automation Society.

BEREŽNÝ, V.; KRAVECOVÁ, D. 2012. *Lineárne programovanie*. Košice : FEI TU. ISBN 978-80-553-0910-1.

BORRELLI, F.; FALCONE, P.; KEVICZKY, T.; ASGARI, J.; HROVAT, D. 2005. Mpc-based approach to active steering for autonomous vehicle systems. *International Journal of Vehicle Autonomous Systems*, 3, 2–4, 265 – 291. ISSN 1741-5306.

BOYD, S.; VANDENBERGHE, L. 2009. *Convex Optimization*. New York, USA: Cambridge University Press, 7th edition.

CECHLÁROVÁ, K.; SEMANIŠIN, G. 1999. *Lineárna optimalizácia*. UPJŠ Košice. ISBN 80-7079-385-4.

CHATTOVÁ, S. 2011. Riešenie problémov celočíselného programovania využitím techník lineárneho programovania. Fakulta matematiky, fyziky a informatiky, Univerzita Komenského v Bratislave. Bakalárska práca.

CITYNETMOBIL, E. R. P. 2013. Driverless cars take to the road. URL `http://cordis.europa.eu/result/rcn/90263_en.html`.

CLARKE, G.; WRIGHT, J. W. 1964. Scheduling of vehicles from a central depot to a number of delivery points. *Oper. Res.*, 12, 4, 568–581. ISSN 0030-364X.

CORMEN, T. H.; LEISERSON, C. E.; RIVEST, R. L.; STEIN, C. 2001. *Introduction to Algorithms*. Cambridge, MA, USA: MIT Press, 2nd edition. ISBN 0-262-03293-7, 9780262032933.

CORONA, D.; DE SCHUTTER, B. 2008. Adaptive cruise control for a SMART car: A comparison benchmark for MPC-PWA control methods. *IEEE Transactions on Control Systems Technology*, 16, 2, 365–372.

CROWDER, H.; PADBERG, M. 1980. Solving large-scale symmetric travelling salesman problems to optimality. *Management Science*, 26, 5, 495–509.

CUTLER, C. R.; RAMAKER, B. L. 1979. *Dynamic Matrix Control - A Computer Control Algorithm*. 86th National Convention of American Institute of Chemical Engineers.

DANTZIG, G. B.; RAMSER, J. H. 1959. The truck dispatching problem. *Management Science*, 6, 80–91.

DASH ASSOCIATES 1999. *XPRESS-MP User Guide*. `http://www.dashopt.com`.

DEN HERTOG, D. 1994. *Interior point approach to linear, quadratic, and convex programming: algorithms and complexity*. Mathematics and its applications. Kluwer Academic Publishers.

EISELT, H.; SANDBLOM, C.-L.; BOFFEY, B.; LAPORTE, G.; SMITH, B.; RICHARDS, E.; SPIELBERG, K. 2000. *Integer programming and network models*. Berlin, New York, Paris: Springer. ISBN 3-540-67191-9.

FAGERHOLT, K. 1999. Optimal fleet design in a ship routing problem. *International Transactions in Operational Research*, 6, 5, 453–464.

FIALA, J.; KOČVARA, M.; STINGL, M. 2013. Penlab: A matlab solver for nonlinear semidefinite optimization. *arXiv preprint arXiv:1311.5240*.

FLETCHER, R.; LEYFFER, S. 1998. Numerical experience with lower bounds for MIQP branch–and–bound. *SIAM J. Optim.*, 8, 2, 604–616.

FLOUDAS, C. A. 1995. *Nonlinear and Mixed-Integer Optimization*. Oxford University Press.

FLOUDAS, C. A.; PARDALOS, P. M., editors 2009. *Encyclopedia of Optimization, Second Edition*. Springer. ISBN 978-0-387-74758-3.

FONTES, F. A. C. C.; VINTER, R. B. 2000. Nonlinear model predictive control: Specifying rates of exponential stability without terminal state constraints.

GARONE, E.; DETERME, J.-F.; NALDI, R. 2012. A travelling salesman problem for a class of heterogeneous multi-vehicle systems. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, 1166–1171. ISSN 0743-1546. doi:10.1109/CDC.2012.6426488.

GRIEDER, P.; BORRELLI, F.; TORRISI, F.; MORARI, M. 2004. Computation of the constrained infinite time linear quadratic regulator. *Automatica*, 40, 4, 701 – 708.

GUROBI OPTIMIZATION, I. 2013. Gurobi optimizer reference manual. URL `http://www.gurobi.com`.

HAMALA, M. 1972. *Nelineárne programovanie 2*. Matematické metódy v ekonomike. ALFA - vydavateľstvo technickej a ekonomickej literatúry.

HASLE, G.; LIE, K.; QUAK, E. 2007. *Geometric Modelling, Numerical Simulation, and Optimization:: Applied Mathematics at SINTEF*. Springer Berlin Heidelberg. ISBN 9783540687832.

HOFF, A.; ANDERSSON, H.; CHRISTIANSEN, M.; HASLE, G.; LØKKETANGEN, A. 2010. Industrial aspects and literature survey: Fleet composition and routing. *Computers & Operations Research*, 37, 12, 2041 – 2061. ISSN 0305-0548. doi: http://dx.doi.org/10.1016/j.cor.2010.03.015.

ILOG 2007. 11.0 user's manual. *ILOG CPLEX Division. Incline Village, NV*.

ILOG, INC. 2009. *CPLEX User Manual*. Gentilly Cedex, France. `http://www.ilog.fr/products/cplex/`.

JENSEN, P.; BARD, J. 2003. *Operations Research Models and Methods*. Operations Research: Models and Methods. Wiley. ISBN 9780471380047.

JOHNSON, D. S.; MCGEOCH, L. A. 1997. The traveling salesman problem: A case study in local optimization. *J. K. Local Search in Combinatorial Optimisation*, 215–310.

KERRIGAN, E.; MACIEJOWSKI, J. 2000. Soft constraints and exact penalty functions in model predictive control. In *Control 2000 Conference, Cambridge*.

KLAUČO, M.; BLAŽEK, S.; KVASNICA, M.; FIKAR, M. 2014. Mixed-integer socp formulation of the path planning problem for heterogeneous multi-vehicle systems. In *European Control Conference 2014*, 1474–1479. Strasbourg, France.

KOZÁK, Š.; HUBA, M. 2003. Control systems design. In *A Proceedings volume from the 2nd IFAC Conference Bratislava, Slovak Republic, 7-10 september 2003*.

KVASNICA, M. 2008. *Efficient Software Tools for Control and Analysis of Hybrid Systems*. Ph.D. thesis, ETH Zurich, ETH Zurich, Physikstrasse 3, 8092 Zurich, Switzerland.

KVASNICA, M. 2009. *Real-Time Model Predictive Control via Multi-Parametric Programming: Theory and Tools*. Saarbruecken: VDM Verlag.

LAU, M. S. K.; S. P. YUE, K. V. L.; MACIEJOWSKI, J. M. 2009. A comparison of interior point and active set methods for fpga implementation of model predictive control. In *Proceedings of the European Control Conference Budapest, Hungary 23-26 August 2009*.

LOBO, M. S.; VANDENBERGHE, L.; BOYD, S.; LEBRET, H. 1998. Applications of second-order cone programming. *Linear Algebra and its Applications*, 284, 1–3, 193 – 228. ISSN 0024-3795. International Linear Algebra Society (ILAS) Symposium on Fast Algorithms for Control, Signals and Image Processing.

LÖFBERG, J. 2004. YALMIP. Available from `http://users.isy.liu.se/johanl/yalmip/`.

MAGNI, L.; RAIMONDO, D.; ALLGÖWER, F. 2009. *Nonlinear Model Predictive Control: Towards New Challenging Applications*. Lecture Notes in Control and Information Sciences. Springer.

MAKHORIN, A. 2008. Glpk (gnu linear programming kit).

MATHEW, N.; SMITH, S.; WASLANDER, S. 2014. Optimal path planning in cooperative heterogeneous multi-robot delivery systems (accpeted). In *The Eleventh International Workshop on the Algorithmic Foundations of Robotics*. Istanbul, Turkey.

MAŇAS, M. 1979. *Optimalizační metody - 1.vydání.* Praha: Státní nakladatelství technické literatury.

MIAH, A. 2016. Amazon delivery drones are just the first step to a highway in the sky. URL `http://phys.org/news/2016-07-amazon-delivery-drones-highway-sky.html`, [Online; accessed October-2016].

MIKLEŠ, J.; FIKAR, M. 2007. *Process Modelling, Identification, and Control*. Springer London, Limited.

MILLER, C. E.; TUCKER, A. W.; ZEMLIN, R. A. 1960. Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, 7, 4, 326–329.

MOSEK, A. 2010. The mosek optimization software. *Online at http://www. mosek. com*, 54, 2–1.

NEMHAUSER, G. L.; WOLSEY, L. A. 1988. *Integer and Combinatorial Optimization*. New York, NY, USA: Wiley-Interscience. ISBN 0-471-82819-X.

NEUMAIER, A. 2004. *Complete Search in Continuous Global Optimiyation and Constraints satisfaction*. Acta Numerica, Lecture Notes in Control and Information Sciences. Cambridge University.

NEWS, B. 2014. Bbc news - uk to allow driverless cars on public roads in january. URL `http://www.bbc.com/news/technology-28551069`.

PLESNÍK, J.; DUPAČOVÁ, J.; VLACH, M. 1990. *Lineárna programovanie*. Alfa. ISBN 80-05-00679-9.

QIN, S.; BADGEWELL, T. 1997. An overview of industrial model predictive control technology. In *Chemical Process Control - V*, volume 93, no. 316, 232–256. AIChe Symposium Series - American Institute of Chemical Engineers.

RAU, M.; SCHRODER, D. 2002. Model predictive control with nonlinear state space models. In *Advanced Motion Control, 2002. 7th International Workshop on*, 136 – 141. doi:10.1109/AMC.2002.1026905.

REINELT, G. 1994. *The Traveling Salesman: Computational Solutions for TSP Applications*. Berlin, Heidelberg: Springer-Verlag. ISBN 3-540-58334-3.

RICHALET, J.; RAULT, A.; TESTUD, J.; PAPON, J. 1978. Model predictive heuristic control: Applications to industrial processes. *Automatica*, 14, 5, 413 – 428.

RODRIGUE, J.; COMTOIS, C.; SLACK, B. 2009. *The Geography of Transport Systems*. Taylor & Francis. ISBN 9780203884157.

RODRIGUEZ, J.; CORTES, P. 2012. *Cost Function Selection*, 145–161. John Wiley and Sons, Ltd.

ROSINOVÁ, D.; DÚBRAVSKÁ, M. 2007. *Optimalizácia*. STU Bratislava. ISBN 978-80-227-2795-2.

ROSSITER, J. A. 2003. *Model-Based Predictive Control: A Practical Approach.* CRC Press Inc.

SCHRIJVER, A. 1986. *Theory of Linear and Integer Programming.* New York, NY, USA: John Wiley & Sons, Inc. ISBN 0-471-90854-1.

SENTINEL, T. M. 1926. Phantom auto will tour city.

STURM, J. F. 1999. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization methods and software*, 11, 1-4, 625–653.

TOH, K.-C.; TODD, M. J.; TÜTÜNCÜ, R. H. 1999. Sdpt3—a matlab software package for semidefinite programming, version 1.3. *Optimization methods and software*, 11, 1-4, 545–581.

TOTH, P.; VIGO, D., editors 2001. *The Vehicle Routing Problem.* Philadelphia, PA, USA: Society for Industrial and Applied Mathematics. ISBN 0-89871-498-2.

TUNG, D. V.; PINNOI, A. 2000. Vehicle routing–scheduling for waste collection in hanoi. *European Journal of Operational Research*, 125, 3, 449 – 468. ISSN 0377-2217. doi: http://dx.doi.org/10.1016/S0377-2217(99)00408-7.

VANDENBERGHE, L.; BOYD, S. 1994. Semidefinite programming. *SIAM REVIEW*, 38, 49–95.

VANDENBERGHE, L.; BOYD, S. 1999. Application of semidefinite programming. *Applied Numerical Mathematics*, 29, 283–299.

WILLIAMS, H. 1993. *Model Building in Mathematical Programming.* Third Edition: John Wiley & Sons.

WILLIAMS, H. 2009. The problem with integer programming. *Working paper LSEOR.*

WOLKOWICZ, H.; SAIGAL, R.; VANDENBERGHE, L. 2012. *Handbook of Semidefinite Programming: Theory, Algorithms, and Applications.* International Series in Operations Research & Management Science. Springer US. ISBN 9781461543817.

Publikačná činnosť autora

- **Kapitola v knihe**

  1. Blažek, S. – Kvasnica, M. – Fikar, M.: Java-based Platform for Testing and Validation of Explicit Model Predictive Control, In Selected Topics in Modelling and Control, Editor(s): Mikleš, J., Veselý, V., Slovak University of Technology Press, no. 8, pp. 110–114, 2012.

- **Článok v časopise v zozname TML**

  1. Klaučo, M. – Blažek, S. – Kvasnica, M.: An Optimal Path Planning Problem for Heterogeneous Multi-Vehicle Systems. International Journal of Applied Mathematics and Computer Science, no. 2, vol. 26, pp. 297–308, 2016, ISSN: 1641-876X, IF: 1.037, indexované v Scopus a WoS.

- **Článok v recenzovanom zborníku z konferencie**

  – kategória AFC

  1. Blažek, S. – Kvasnica, M.: Polygonic Representation of Explicit Model Predictive Control in Two Dimensions. Editor(s): Ivan Taufer, Daniel Honc, Milan Javurek, In Proceedings of the 10th International Scientific - Technical Conference Process Control 2012, University of Pardubice, Kouty nad Desnou, Czech Republic, 2012.
  2. Klaučo, M. – Blažek, S. – Kvasnica, M. – Fikar, M.: Mixed-Integer SOCP Formulation of the Path Planning Problem for Heterogeneous Multi-Vehicle Systems. In European Control Conference 2014, Strasbourg, France, pp. 1474–1479, 2014, indexované v Scopus a WoS.

3. Oravec, J. – Blažek, S. – Kvasnica, M. – Di Cairano, S.: Polygonic Representation of Explicit Model Predictive Control. In IEEE Conference on Decision and Control, Florence, Italy, pp. 6422–6427, 2013, indexované v Scopus a WoS.

– kategória AFD

1. Kalúz, M. – Holaza, J. – Janeček, F. – Blažek, S. – Kvasnica, M.: A Robotic Traffic Simulator for Teaching of Advanced Control Methods. In Preprints of the 11th IFAC Symposium on Advances in Control Education, vol. 11, pp. 338–343, 2016.
2. Oravec, J. – Blažek, S. – Kvasnica, M.: Simplification of Explicit MPC Solutions via Inner and Outer Approximations. Editor(s): Fikar, M., Kvasnica, M., In Proceedings of the 19th International Conference on Process Control, Slovak University of Technology in Bratislava, Štrbské Pleso, Slovakia, pp. 389–394, 2013, indexované v Scopus a WoS.

Curriculum Vitae

Slavomír Blažek

dátum narodenia: 24. júl 1987
národnosť: slovenská
e-mail: xblazeks@gmail.com

Vzdelanie

- doktorandské štúdium: od 2011
  - Inštitúcia: Fakulta chemickej a potravinárskej technológie, STU v Bratislave
  - Študijný program: Riadenie procesov
- inžinierske štúdium: 2008–2010
  - Inštitúcia: Fakulta informatiky a informačných technológií, STU v Bratislave
  - Študijný program: Informačné systémy
- bakalárske štúdium: 2005–2008
  - Inštitúcia: Fakulta informatiky a informačných technológií, STU v Bratislave
  - Študijný program: Informatika

Záujmové vzdelanie

- umelecké štúdium: 1993–2010

  - Inštitúcia: Základná umelecká škola v Šamoríne (I.stupeň, II.stupeň, ŠPD)
  - Študijný program: klavír
- umelecké štúdium: 1994–2004
  - Inštitúcia: Základná umelecká škola v Šamoríne (I.stupeň, II.stupeň)
  - Študijný program: tanec

Jazykové znalosti

- anglicky – pokročilý
- nemecky – začiatočník
- španielsky – začiatočník
- rusky – základy

Pracovné skúsenosti

- založenie projektu projektu Dobrodruhovia (www.dobrodruhovia.eu): od 2012
  - vedenie 10-členného tímu
  - prezentovanie projektu
  - marketing
  - komunikácia s partnermi
  - zháňanie sponzorov
  - vypracovávanie grantových programov
  - práca na web stránke
- programátor analytik senior vo VUB a.s.: od 2011
  - Multichannel — internet a biznis banking, mobilné bankovníctvo
    * navrhovanie biznis riešenia
    * vytváranie technického dizajnu
    * implementácia softvérového riešenia
    * integrácia backendových systémov
    * plánovanie časových odhadov na úlohy

    - vedenie menšieho tímu (do 5 osôb)
    - podpora aplikácie
    - komunikácia so zákazníkom
    - práca pod stresom v náročných podmienkach

- programátor v Slovenská sporiteľňa a.s.: 2011
  - NIS (New Information System) – Printing
- softvérový vývojár vo VÚB a.s: 2010–2011
  - Multichannel — internet a biznis banking, mobilné bankovníctvo
  - vytvorenie testovacieho nástroja pre projekt JUM (Joint User Model)
- programátor v spoločnosti ASSECO Slovakia a.s.: 2006–2009
  - správa dokumentového systému
  - správa registratúry
  - účtovná kniha
- administrátor v internetovej miestnosti vo Výpočtovom centre UK: 2004

Absolvované školenia

- 2016 – Negociácia - efektívne postupy vyjednávania
- 2016 – Projektové riadenie PMI
- 2015 – Nové technológie v JAVA SE V7, V8
- 2014 -– Scrum mastership
- 2014 -– Agile a scrum
- 2014 – JQuery – HTML5/CSS3 a novinky vo webových technológiách
- 2013 – Workshop PWS a ESB
- 2012 – Java servlety a JSP
- 2012 – Manažment softvérových projektov
- 2011 – Programming in Spring
- 2006 -– Enterprise arichitect

Organizačné aktivity

- jún 2010: Inštrumentálny koncert ABBA v Šamoríne pod záštitou Základnej umeleckej školy (aj účinkujúci ako klavirista)
- apríl 2010: Objavný pohľad na svet a Slovensko – prednáška bývalého slovenského veľvyslanca v Argentíne Jána Juristu v Šamoríne
- február 2010: Deň kubánskej kultúry – koncert kubánskej kapely Team Cuba s výstavou fotografií Andreja Palacku v Šamoríne s návštevou veľvyslanca Kuby na Slovensku Davida Pavlovicha
- jún 2009: Koncert melódií filmov a muzikálov v Šamoríne pod záštitou Základnej umeleckej školy (aj účinkujúci ako klavirista)

Konferencie

- 2013 - 19th International Conference on Process Control - Simplification of Explicit MPC Solutions via Inner and Outer Approximations
- 2010 – Kognice a umělý život X. - Binárna tomografia 2D a 3D objektov pomocou evolučných algoritmov

Počítačové zručnosti

- JAVA expert
- XML pokročilý
- HTML pokročilý
- UML pokročilý
- C pokročilý
- JavaScript pokročilý
- Xquery pokročilý
- Matlab pokročilý
- J2EE mierne pokročilý (JSP, Servlets, EJB, JDBC)
- Tomcat mierne pokročilý

- Spring mierne pokročilý
- Maven mierne pokročilý
- Node.js mierne pokročilý
- SQL mierne pokročilý
- Hibernate základy
- Lisp základy
- Prolog základy
- Assembler základy

Osobitné oprávnenia

- vodičský preukaz sk. B – od 8.8.2005
- člen Vodnej záchrannej služby SČK, kval. stupeň STRIEBRO od 26.5.2006
- PADI certifikát

Záujmy

- výpočtová technika
- veda
- politika
- štúdium náboženstiev
- hra na klavíri
- športy (cyklistika, plávanie, ľadový hokej - brankár, kanoistika, potápanie)