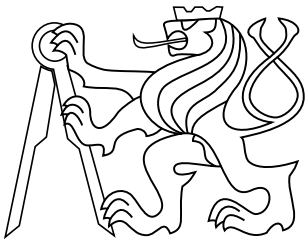
bakalářská práce

Rozšíření iterativního párového algoritmu pro řešení kolizí

Ladislav Vrbský



Květen 2014

Vedoucí práce: Mgr. Přemysl Volf, Ph.D.

České vysoké učení technické v Praze
Fakulta elektrotechnická, Katedra kybernetiky

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Ladislav V r b s k ý

Studijní program: Otevřená informatika (bakalářský)

Obor: Informatika a počítačové vědy

Název tématu: Rozšíření iterativního párového algoritmu pro řešení kolizí

Pokyny pro vypracování:

1. Seznamte se s algoritmy pro distribuované řešení kolizí v bezpilotních systémech, zvláště s iterativním párovým algoritmem IPPCA (Iterative Peer-to-peer collision avoidance algorithm).
2. Seznamte se se systémem AgentFly a jeho nedeterministickou implementací algoritmu IPPCA.
3. Navrhněte a implementujte deterministický algoritmus IPPCA do systému AgentFly.
4. Navrhněte a implementujte rozšíření algoritmu IPPCA zlepšující některé jeho vlastnosti.
5. Porovnejte a vyhodnoťte původní a rozšířenou verzi algoritmu IPPCA.

Seznam odborné literatury:

[1] Michael S. Nolan: Fundamentals of Air Traffic Control. 2010.
[2] Michal Pěchouček, David Šišlák: Agent-based approach to free-flight planning, control, and simulation. IEEE Intelligent Systems, 2009.
[3] Přemysl Volf, David Šišlák: Convergence of Peer-to-Peer Collision Avoidance among Unmanned Aerial Vehicles. IAT 2007.

Vedoucí bakalářské práce: Mgr. Přemysl Volf, Ph.D.

Platnost zadání: do konce letního semestru 2014/2015

L.S.

doc. Dr. Ing. Jan Kybic
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 10. 1. 2014

Poděkování

Děkuji vedoucímu této práce, panu Mgr. Přemyslu Volfovi, Ph.D., za poskytnutou pomoc a cenné rady, kterých si velmi vážím. Také děkuji mé rodině a všem ostatním, kteří mě v průběhu této práce podporovali.

Prohlášení

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

V Praze dne ....

Podpis autora práce

Abstrakt

Tato práce se zabývá kooperativním algoritmem pro řešení kolizí bezpilotních letadel. Analyzuje problém zabraňování kolizím a popisuje přístupy k jeho řešení. Je navržena deterministická verze algoritmu IPPCA. Následně je představeno vylepšení algoritmu, používající heuristický přístup pro generování letových plánů. Algoritmus IPPCA je spolu se svým rozšířením implementován. Nakonec je původní verze algoritmu porovnána se svojí vylepšenou verzí pomocí několika simulačních scénářů a konfigurací vylepšené varianty algoritmu. Výsledky ukazují, že heuristický přístup je schopen vyřešit hromadnou kolizi letadel pomocí průměrně dvoutřetinového množství generovaných letových plánů při zachování podobné kvality výsledných trajektorií letu. Navržený heuristický přístup má velice pozitivní dopad na množství generovaných letových plánů. Nicméně byl zaznamenán zvýšený počet přenesených letových plánů, čímž se toto rozšíření stává méně doporučitelným pro použití v letovém prostoru s vysokou hustotou komunikace.

Abstract

This thesis deals with a cooperative algorithm for collision resolution among unmanned aerial vehicles (UAVs). It analyses the collision avoidance problem and describes approaches to its resolution. A deterministic version of the IPPCA algorithm is then designed. Subsequently, an improvement of the algorithm using heuristic approach to flight plan generation is proposed. Both IPPCA algorithm and its improvement are implemented. Finally, the original version of the algorithm is compared with its upgraded version using several simulation scenarios and configurations of the upgraded version. Results show, that the heuristic approach is able to solve a multi-airplane collision by generating two thirds the amount of flight plans on average while maintaining a similar quality of the resulted flight trajectories. The proposed heuristic approach has a very positive impact on quantity of flight plan generation. However, an increased volume of flight plan transfer has been recored, making the improved version less recommendable in airspace with high communication density.

Obsah

Zkratky

UAV	unmanned aerial vehicle – bezpilotní letoun
IPPCA	algoritmus – Iterative Peer-to-peer Collison Avoidance
RBCA	algoritmus – Rule-Based Collision Avoidance
MPCA	algoritmus – Multi-Party Collision Avoidance

1. Úvod

Letecká doprava existuje již více než 100 let, přesto je nejmladším druhem dopravy osob a zboží. Letecké přepravní služby byly poprvé nabízeny veřejnosti v roce 1912 v USA. K zásadnějšímu rozvoji tohoto druhu přepravy došlo po první světové válce, kdy i v Evropě začaly vznikat první letecké společnosti. V počátcích letectví postačilo k řízení letového provozu používání jednoduchých signálů. Později si rozvoj letecké dopravy vyžádal zavedení prvních komplexnějších pravidel.

Postupem času se letecká doprava stávala stále více dostupnější pro širší masu cestujících i pro přepravu zboží. Její zvyšující se možnosti a výhody přinesly větší zájem o její využívání, čímž postupně vzrůstala hustota leteckého provozu. Tento rozvoj si vyžádal centralizované řízení leteckého provozu. Hustota letecké dopravy roste neustále, cože s sebou přináší některé nové problémy a požadavky, jejichž vyřešení se stalo velmi naléhavým úkolem. Mezi tyto problémy patří zejména omezená kapacita letišť, vzdušného prostoru a omezené kognitivní schopnosti osob řídících letový provoz. Dalším problémem je zastaralost systémů zajišťujících poskytování těchto služeb.

Od září roku 2015 navíc bude možné použití bezpilotních systémů v civilním letovém prostoru. Tím již nyní rapidně vzrůstají nároky na bezpečnost bezpilotních systémů. Je důležité bezpečnost takových systémů prověřovat za použití simulací letového provozu. Projekt AgentFly se zabývá simulací letového provozu a umožňuje testování různých nastavení systémů, jejichž dopady není jinak možné snadno předpovědět. Bezpilotní systémy jsou často založeny na konceptu volného letu, který umožňuje letadlům měnit trajektorii svého letu bez potřeby schválení změny od osob, řídící letový provoz.

Tato práce má za cíl dílčím způsobem napomoci ke zvýšení efektivity v oblasti leteckého provozu. Konkrétně je v práci analyzován problém předcházení kolizí bezpilotních letadel. Je představeno několik druhů antikolizních algoritmů a hlouběji tato práce rozebírá algoritmus IPPCA, který je používán v konceptu volného letu v projektu AgentFly. Pro původní nedeterministickou variantu algoritmu navrženo její převedení do deterministické verze. Následně je prezentováno rozšíření tohoto algoritmu, které cílí na efektivitu řešení kolizí bezpilotních letadel. Základní i rozšířená varianta algoritmu je implementována v projektu AgentFly, který umožňuje simulaci letového provozu. Použití deterministické varianty algoritmu umožňuje efektivnější testování jednotlivých komponent letového provozu.

2. Specifikace problému

Využívání letecké dopravy v posledních letech neustále roste. Podle studie [1] se hustota letového provozu v následujících dvaceti letech znásobí více než dvaapůlkrát. Proto je nutné přicházet s inovacemi, týkajících se zejména zefektivnění využití leteckého prostoru. Jednou z inovací je zavádění bezpilotních systémů, které nevyžadují neustálou kontrolu od řídících letového provozu. Velký důraz je ovšem kladen na bezepčnost použití bezpilotních letadel. Ta musí být schopna automaticky předejít kolizím s ostatním letadly. Pro zabránění kolizí bezpilotních letadel slouží v projektu AgentFly (viz 2.3) antikolizní algoritmy. Jedním z nich je algoritmus IPPCA (viz sekce 2.2), který řeší kolizi pro dvojici letadel. V projektu AgentFly existuje algoritmus IPPCA ve své nedeterministické verzi pro nasazení na bezpilotní letadla.

Pomocí simulací jsou prováděny studie různých součástí řízení letového provozu. V těchto případech není žádoucí, aby byly výsledky studií ovlivněny nedeterministickým chováním algoritmu pro řešení kolizí. Z tohoto důvodu je v této práci představena deterministická varianta algoritmu IPPCA. Dále se práce zabývá analýzou algoritmu, návrhem jeho vylepšení a porovnáním základní a vylepšené varianty.

2.1. Popis problému

Autonomní bezpilotní systémy musí být schopny předcházet vzájemným kolizím ve vzdušném prostoru. Proto musí umět detekovat a řešit kolize v blízké budoucnosti. Dále musí mít bezpilotní letadla, stejně jako pilotovaná letadla, naplánovaný *letový plán*, který budou provádět. Podle [2] je letový plán definován jako neomezený počet geografických bodů s případným časovým omezením. Prvotní plán je naplánován před startem letadla bez ohledu na možné kolize s ostatními letadly.

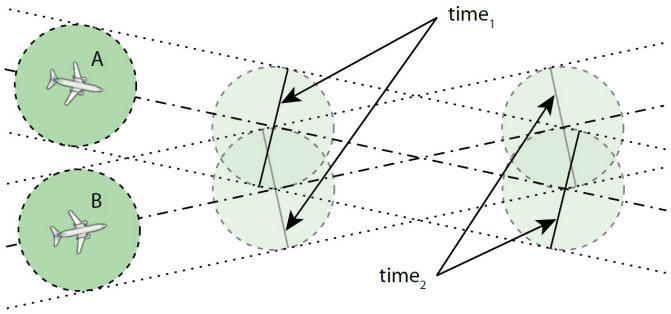
2.1.1. Detekce kolize

Důležitou součástí každého bezpilotního systému je detekce kolizí a jejich vyřešení. Detekce kolize je proces, při kterém se predikuje, že se někdy v budoucnu dostane letadlo do situace, kdy nebude dodržena bezpečná vzdálenost vůči jinému letadlu. Detekované kolizi se pak letadla snaží zabránit. Letadla mohou detekovat a řešit kolize kooperativně nebo nekooperativně. Kooperativita řešení kolize nezávisí na kooperativitě detekce, a naopak. Palubní systém bezpilotního letadla zpravidla zahrnuje více typů detekce kolizí i více typů jejich řešení. Jakýkoliv detektor proto musí definovat kolizi správně (viz další odstavec), aby mohl řešení kolize provést libovolný řešitel – tzv. *solver*.

Kolize nastává, pokud je jednomu letadlu jiným letadlem narušena jeho definovaná

bezpečnostní zóna. Bezpečnostní zóna je definována určitým prostorem kolem každého letadla. Podle [3] je bezpečnostní zóna sférický prostor s definovaným poloměrem. Definuje vzdušný prostor kolem letadla, kde se žádné jiné letadlo nesmí objevit. Tento prostor využívá letadlo pro zabránění turbulencím, způsobeným jiným letadlem, a nepřesnostem provedení letového plánu (malé odchylky od letového plánu jsou povoleny).

Narušení bezpečnostní zóny trvá po dobu od výskytu jiného letadla v této zóně až po dobu, kdy ji toto letadlo opustí. Kolize je proto definována počátečním a koncovým bodem v čase a prostoru, kdy je letadlu narušována bezpečnostní zóna (viz obrázek 1).



Obr. 1. Identifikace prvího a posledního bodu kolize. Obrázek byl převzat z [3].

V případě kolize dvou letadel s různým rozměrem jejich bezpečnostní zóny může po určitou dobu dojít k narušování bezpečnostní zóny pouze jednoho z těchto letadel. K narušení bezpečnostní zóny dojde pouze u letadla, které má bezpečnostní zónu větší. U letadla s menší bezpečnostní zónou k jejímu narušení vůbec nemusí dojít. I tento typ kolizi musí rozpoznat obě zúčastněná letadla. Počáteční a koncové časy kolize jsou identické pro oba letové plány letadel. Kolize tedy trvá během narušení bezpečnostní zóny alespoň jednoho z dvojice letadel, nezávisle na tom, jestli je druhému letadlu narušena bezpečnostní zóna také.

Detekce kolizí probíhá v konceptu volného letu (viz sekce 2.3) decentralizovaně. Každé letadlo kontroluje letadla ve svém lokálním okolí. Provádět detekci kolizí lze kooperativně a nekooperativně. V závislosti na tom, zda jsou subjekty navzájem kooperativní, je zvolen určitý druh detekce.

Kooperativní detektor využívá komunikace letadel, která při vzájemné detekci druhého letadla na radaru provedou tzv. kooperativní pozdrav. Pomocí kooperativního pozdravu posílá detektor informace o svém letadlu a své letové plány. Tyto informace společně s letovými plány jsou nutné pro kooperativní detekci. Kooperativni detekce proběhne po přijetí letových plánů za předpokladu, že má detektor informace o typu druhého letadla. Pro vyřešení kolize je nutné znát místo a čas jejího prvního a posledního bodu. Tyto atributy zjistí detektor porovnáním svého letového plánu s letovým plánem druhého letadla. K detekci používá informace o obou letadlech, které definují rozměry jejich bezpečnostních zón, dosah radarů, aj.

Pokud není komunikace mezi letadly možná (např. z důvodu poruchy), je nutné vyřešit kolize nekooperativně. Jedinou dostupnou informací jsou v tomto případě data z naměřených pozic senzorů radaru. Blíže se nekooperativním řešením kolizí věnuje [4], kde jsou představeny dva prediktory, predikující pohyb druhého letadla pro účely

detekce kolize: *prediktor lineární* a *prediktor založený na Taylorově řadě*. Obě nekooperativní metody pro odhad potřebují řadu 3D pozic v čase.

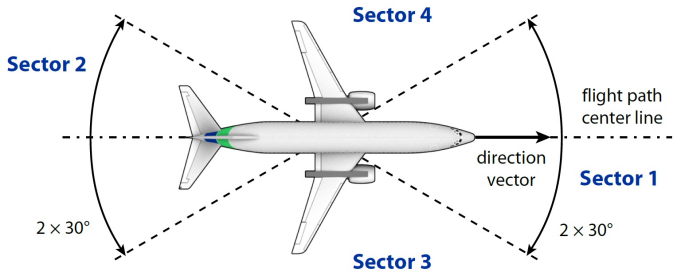
2.1.2. Algoritmy pro řešení kolizí v bezpilotních systémech

Existuje více způsobů, jak řešit vzniklou kolizi u bezpilotních letadel. Algoritmy se dělí na kooperativní a nekooperativní (tj. spolupracující a nespolupracující) a na doménově závislé a nezávislé. Kooperativní algoritmy jsou dále dělelny na aktivní a pasivní. Některé z těchto algoritmů jsou zde představeny. Iterativnímu párovému algoritmu je věnována zvláštní sekce 2.2.

Pasivní algoritmy se od aktivních liší tím, že po detekci kolize již nevyžadují žádnou další komunikaci. Podle konkrétní zdetekované kolizní situace se určí pevně daný postup, který obě letadla realizují (viz odstavec níže – Algoritmus RBCA).

Algoritmus RBCA

Rule-Based Collision Avoidance, nebo-li RBCA „je doménově závislý pasivní kooperativní algoritmus pro řešení kolizí, který je založen na *Vizuálních letových pravidlech* definovaných Úřadem pro civilní letectví. Každé letadlo provede jeden z předdefinovaných úhybných manévrů dle následujícího postupu. Nejdříve se definuje typ kolize mezi letadly. Typ kolize je určen na základě úhlu mezi směrovými vektory letadel, promítnutými na vodorovnou plochu. V závislosti na klasifikaci kolize (viz obrázek 2) každé letadlo aplikuje úhybný manévr z množiny definovaných pravidel. Manévry jsou parametrizovány tak, že používají informaci o kolizi a jejím úhlu, takže je řešení přizpůsobené identifikované budoucí kolizi.“ [3]



Obr. 2. Identifikace typu kolize, využívaná algoritmem RBCA. Obrázek byl převzat z [3]

Algoritmus MPCA

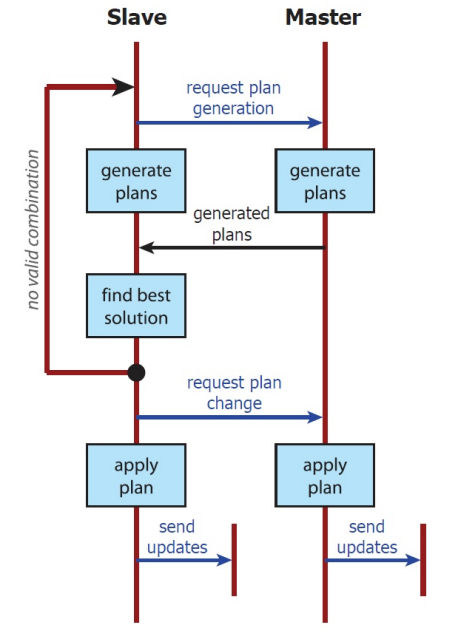
Algoritmus Multi-Party Collision Avoidance (MPCA) je dle [3] určen k řešení lokální kolize většího počtu letadel. Kolizi řeší všechna letadla pomocí *koordinátora*, zodpovědného za expanzi stavového prostoru letových plánů a hledání optimálního řešení multikolize. Multi-party koordinátor je agent, jehož rolí je najít množinu bezkolizních plánů pro skupinu letadel s možnou kolizí – *multi-party skupina*. Koordinátor uchovává

informaci o skupině, strom stavového prostoru letových plánů, vybírá, které letadlo bude pozváno do skupiny, žádá letadla ve skupině o generování letových plánů a posílá informace o nalezených nekolidujících plánech.

2.2. Algoritmus IPPCA

Předmětem této práce je Iterativní párový algoritmus pro řešení kolizí (Iterative Peer-to-peer Collison Avoidance algorithm), nebo-li IPPCA. IPPCA je aktivní kooperativní doménově nezávislý algoritmus. Tento algoritmus je vhodný i pro řešení hromadných kolizí. Každou kolizi řeší decentralizovaně – řídícím prvkem kolize je jedno z dvojice letadel, které spolu kolizi řeší. Rozšířením tohoto algoritmu pro hromadné kolize je algoritmus MPCA.

Algoritmus hledá řešení pomocí generovaných letových plánů oběma letadly (viz sekce 2.2.1). Nejprve provede kartézský součin generovaných plánů obou letadel. Nad vzniklou množinou párů letových plánů hledá podmnožinu párů, které kolizi řeší. Výběr partikulárního řešení z množiny řešení se řídí návrhem pro konkrétní nasazení algoritmu. Algoritmus generuje plány a hledá řešení tak dlouho, dokud není řešení nalezeno. Řešení kolize může být přerušeno. Například z důvodu nutnosti vyřešit kolizi s jiným letadlem. Dalším důvodem může být omezení počtu cyklů algoritmu, které mohou proběhnout. Toto omezení je dáno návrhem algoritmu. V případě přerušení řešení kolize záleží další kroky na konkrétním návrhu algoritmu. Může se například hledat řešení s nejmenším porušením bezpečnostních zón. Jiným postupem je řešení kolize zcela zrušit a neměnit letové plány.



Obr. 3. Vyjednávání během algoritmu IPPCA. Obrázek byl převzat z [3]

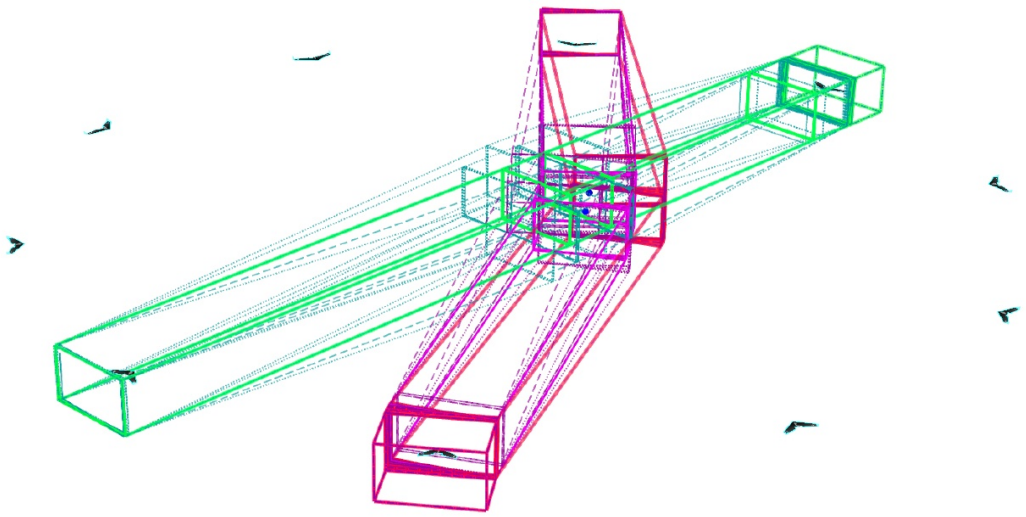
Průběh algoritmu popisuje obrázek 3. Na začátku řešení se jedno z dvojice letadel pravidlem určí jako vedoucí kolize, tzv. *master*. Druhé letadlo je masterovi podřízené,

tzv. *slave*, a pouze reaguje na jeho požadavky. Master vyžádá generované plány od slavea a vygeneruje i svoje plány. Master pak hledá řešení nad množinou párů letových plánů. Pokud řešení existuje, informuje o něm slavea a obě letadla příslušné letové plány použijí. Letadla pošlou kooperativním detektorům, se kterými mají navázané spojení, tyto nové plány. Neexistuje-li bezkolizní dvojice, vygenerují letadla další plány. Nové plány se přidají k předchozím a master znovu hledá nejlepší řešení. Tento cyklus se opakuje, dokud není řešení kolize přerušeno (z důvodů, uvedených v předchozím odstavci).

2.2.1. Generování letových plánů

Nový letový plán se generuje použitím jednoho nebo více manévrů ze sady manévrů, které má letadlo dle konfigurace k dispozici. Typicky jde o rychlostní (zrychlit, zpomalit), vertikální (stoupat, klesat) a horizontální (zatočit doprava, zatočit doleva) manévry. Navíc se vygeneruje i plán beze změn, protože i ten může být součástí řešení kolize. Závisí na návrhu, ale většinou se v jednom průchodu algoritmu generuje více než jeden plán pro řešení kolize. Příklad generování plánů je uveden na obrázku 4.

Pokud to návrh umožňuje, generované plány se do množiny plánů pro hledání řešení přidávají a není nutné generovat stejné plány opakovaně. Algoritmus může volit pro generování plánů heuristický přístup. Z důvodu maximalizace užitku (záleží na jeho definici) se často nejdříve hledá řešení blízko řešené kolize. Z vygenerovaných plánů se před jejich použitím vyřadí ty, jejichž trajektorie by vytvářely dřívější kolizi, než je ta, kterou letadlo řeší.



Obr. 4. Generování manévrů během algoritmu IPPCA. Obrázek byl převzat z [3]

2.3. Projekt AgentFly

Koncept volného letu, tedy *free-flight concept*, podle [5], navrhuje opuštění centralizovaných, předdefinovaných, předem zamluvených letových koridorů. Doporučuje decentralizaci řízení letového provozu mezi několik (pilotovaných či bezpilotních) letadel. „AgentFly je softwarový prototyp pro řízení letového provozu, který podporuje koncept volného letu – free flight. Každé letadlo v systému je modelováno jako nezávislá entita, která může hostit několik inteligentních agentů. Každá entita je zodpovědná za naplanování a provedení letového plánu. Během letu detekují agenti případné kolize a zahájí vzájené peer-to-peer vyjednávání, při kterém inteligentním způsobem přeplánují aktuální letové plány na bezkolizní. Cílem tohoto prototypu je ukázat schopnost řešit distribuovaně a flexibilně pomocí multi-agentní technologie problém řízení letového provozu heterogenních autonomních letadel s důrazem na (i) vysoký počet letadel, (ii) snížení nároků na řízení mimo palubu letadla a (iii) umožnění kombinace kooperativních a nekooperativních letadel v jednom letovém prostoru.“ [2]

2.3.1. Práce s časem a komunikace

Pro vzájemnou komunikaci bezpilotních letadel se v projektu AgentFly využívají *zprávy*. Zprávy nesou následující informace:

- Adresa odesílatele
- Adresa příjemce
- Označení typu zprávy
- Obsah zprávy – může být libovolný

Odeslané zprávy od jednoho letadla jsou doručeny příjemci ve stejném pořadí, jako byly odeslány. Není ale možné předem určit čas, ve který bude zpráva doručena. Problematická je situace, kdy mezi sebou komunikují více než dvě letadla. V případě příjmu zpráv od více příjemců nelze určit pořadí, v jakém budou zprávy doručeny. I přesto, že zprávy od každého z odesílatelů zvlášť mají správné pořadí. To má za následek nedeterminičnost komunikace a s tím spojený nedeterministický průběh simulace letového provozu.

Nově lze v simulaci využít i komunikaci pomocí *event*. Chování event je deterministické a ve většině případů nahrazují posílání zpráv, které do algoritmu nedeterminičnost zanáší. Čas se v simulaci dělí na jednotlivé diskrétní části – milisekundy. Na začátku každé z nich, se simulační čas zastaví a po postupném zpracování všech event přejde k času nejbližší eventy. Eventy nesou stejné druhy informací, jako zprávy. Použití event nese následující pravidla a s nimi spojené výhody a nevýhody:

- Eventy jsou, stejně jako zprávy, označeny podle druhu informace, kterou nesou (letový plán, informace z radaru atp.). Tato označení mají definovanou prioritu a určují pořadí, ve kterém jsou příjemci v danou milisekundu doručeny. Řazení event je dané konkrétním návrhem algoritmu a nezávisí na pořadí, v němž byly eventy odeslány.
- Každá eventa má definovaný čas doručení, minimálně však 1 ms od času odeslání, nezávisle na tom, zda je odeslána přímo odesílateli nebo jinému letadlu. Je zajištěno, že v určený čas bude doručena, pokud bude příjemce stále přítomen v simulaci (tedy pokud např. nepřistane).

- Příjemce nemůže předem zjistit kolik event kterého typu mu bude danou milisekundu doručeno ani kolik bude event celkově.
- Systémové eventy slouží k informování jednotlivých letadel a mají určité výjimky. Typicky se jedná o údaje ze senzorů. Tyto eventy jsou doručeny okamžitě v danou milisekundu, kdy jsou vytvořeny – jedná se totiž o komunikaci v rámci jednoho letadla. Vznik systémových event nemusí proběhnout během zpracování jiné eventy. Řídí se označením, stejně jako ostatní eventy, jsou pro ně ale rezervovány nejprioritnější typy event. Z toho plyne, že systémové eventy jsou doručeny příjemci jako první v dané milisekundě.
- Eventy se dají rozeslat hromadně všem letadlům najednou jednou eventou. Každé letadlo může vytvořit libovolné množství skupin pro hromadné eventy. Jako příjemce hromadné eventy pak určí tuto skupinu. Jednotlivá letadla se přihlašují do skupin ostatních letadel, aby jim přicházely informace, odeslané do skupiny. Nemá-li příjemce už o odběr hromadných zpráv z nějaké skupiny zájem (např. z důvodu ukončení kontaktu s letadlem, jemuž skupina patří), z příslušné skupiny se odhlásí.
- Eventu je možné poslat pouze v průběhu zpracování jiné příchozí eventy (s výjimkou systémových event). Jedna vždy navazuje na nějakou předchozí. Algoritmus nemůže například vždy reagovat na přijatou zprávu odesláním eventy (ať už sobě, nebo jinému letadlu).

Pokud musí letadlo například počkat s odesláním eventy do doby, kdy jsou zpracovány všechny příchozí eventy jednoho typu, je nutné, aby si naplánovalo upozorňovací eventu za 1 ms. Při jejím zpracování může eventu odeslat. To ale přináší další rizika, jelikož některé informace už nemusí být po 1 ms platné. Použití více vláken v implementaci v tomto případě nepomůže, protože je potřeba zaručit, aby byla eventa odeslána při zpracování jiné eventy.

2.3.2. Nedeterministická varianta algoritmu IPPCA

Nedeterministická verze algoritmu IPPCA je již v projektu AgentFly implementována. Je navržena zejména pro nasazení na simulovaná nebo reálná bezpilotní letadla. Tato varianta se uplatní zejména v případech, kdy čas pozastavit nelze a algoritmus musí přesto korektně fungovat. Hlavním problémem je nedeterminičnost komunikace mezi letadly. Tato varianta nevyužívá ke komunikaci ani ke krokování času eventy. Ke komunikaci využívá pouze zprávy. Mimo nedeterminičnost zpráv také nelze zajistit, aby výpočetní operace trvaly určitý předem známý čas. Z tohoto důvodu nelze zajistit ani přesný čas, kdy bude která zpráva odeslána. To přináší zejména problémy, co se týká synchronizace vzájemné komunikace.

Tyto problémy jsou řešeny použitím stavů, odlišných pro jednotilvé části algoritmu. Mezi těmito stavy přechází jednotlivé části algoritmu v závislosti na komunikaci s ostatními letadly. To zajišťuje jednodušší zacházení s příchozími zprávami. Některé mohou být v daném stavu irelevantní a algoritmus se jimi zabývat nebude, naopak může příjemce čekat pouze na určitou zprávu, která ho přesune do dalšího stavu.

Protože čas v tomto případě pozastavit nelze, letadla se v průběhu řešení kolize stále pohybují v prostoru. Při řešení kolize se předem určí čas, do kterého se musí kolize vyřešit. Úsek letových plánů obou letadel je neměnný až do místa, kde se v tomto určeném čase budou letadla nacházet. Až od tohoto místa uvažují letadla při řešení

kolize změny v plánech. Pokud letadla kolizi nevyřeší do určeného času, kolize se zruší. Po zrušení se provede detekce kolizí a pokud budou kolizi řešit stejná letadla, interval neměnných částí plánů bude pozměněn. Letadla tedy budou moct řešit kolizi na jiném úseku letových plánů.

Řešení kolize navíc může být zcela zrušeno během svého průběhu – změna plánu třetího letadla může vést ke kolizi s jedním (nebo i s oběma) letadly, řešící kolizi. V tomto případě musí řešící pár řešení přerušit a vyřešit vzniklou dřívější kolizi.

V simulačním prostředí tato implementace tedy představuje velkou nevýhodu v tom, že při dvou různých průbězích jednoho scénáře se program chová jinak. Je-li potřeba zaznamenat chyby, nedostatky, nebo pouze zkoumat, proč a jak se algoritmus chová, často to z tohoto důvodu není možné.

2.4. Zadání problému

Cílem práce je deterministická implementace algoritmu IPPCA a její rozšíření. Základní deterministický algoritmus bude poté porovnán se svojí rozšířenou verzí a obě verze algoritmu budou vyhodnoceny.

Celkem budou navrženy a implementovány následující části:

- Kooperativní detektor kolizí
- Správce kolizí
- Část pro kooperativní řešení kolizí pomocí algoritmu IPPCA
- Rozšíření algoritmu IPPCA
- Simulační scénáře a jejich vyhodnocení

2.4.1. Deterministická varianta algoritmu IPPCA

Determinismus algoritmu IPPCA je důležitý pro simulační využití. Testuje-li simulace ostatní komponenty projektu AgentFly, je potřeba, aby se algoritmus IPPCA choval deterministicky. Nedeterministické chování algoritmu způsobuje potíže při opakovaném spouštění simulace. Algoritmus se při každém spuštění chová jinak a není proto jednoduché testovat jiné části projektu, které na chování algoritmu IPPCA závisí. Bude navržena deterministická varianta tohoto algoritmu.

Algoritmus IPPCA vyžaduje pro řešení kolizí také jejich detekci. Proto budou navrženy následující podpůrné komponenty:

- Deterministický kooperativní detektor, sestávající z komunikační části a části pro detekci kolizí.
- Správce kolizí, přijímající detekované kolize. Tyto kolize předává k řešení.

2.4.2. Rozšíření algoritmu

V první části práce je algoritmus navržen pouze ve své základní formě. Pro lepší a efektivnější funkčnost algoritmu je pak možné algoritmus nějakým způsobem vylepšit. To

bude předmětem druhé části práce, kde bude pro algoritmus bude navrženo rozšíření. Toto rozšíření se týká heuristiky generování letových plánů při řešení kolize pomocí algoritmu IPPCA.

2.4.3. Implementace

Po celkovém návrhu budou v projektu AgentFly implementovány všechny části navrženého algoritmu. Vedle základní verze bude implementováno její rozšíření. Rozšíření nebude nahrazovat původní verzi algoritmu a obě řešení bude možné navzájem porovnat. Výsledek programovací části je přiložen na CD.

3. Návrh deterministické varianty algoritmu IPPCA

V této kapitole je navržena deterministická verze algoritmu IPPCA. Simulační prostředí projektu AgentFly umožňuje použití event (viz sekce 2.3.1), které tento algoritmus využívá. Použití event výrazně pomáhá k determiničnosti algoritmu. Podle zpracování event se řídí i průběh simulačního času, který je možno kontrolovat.

3.1. Návrh kooperativního detektoru kolizí

Ke kooperativní detekci kolizí a výměně a uchování informací, které letadla k detekci potřebují, slouží kooperativní detektor. Ke komunikaci využívají detektory pouze eventy. První informace, kterou si letadla musí vyměnit, je typ jejich letadel. Eventa s tímto účelem slouží zároveň jako kooperativní pozdrav. Zároveň s kooperativním pozdravem posílá část letového plánu. Typ letadla a letový plán jsou nutné k provedení kooperativní detekce.

Při každé změně letového plánu se posílá nový plán všem letadlům, se kterým je ve spojení. Letové plány se neposílají celé, ale pouze po částech. Proto musí letadla průběžně posílat části svého letového plánu ostatním letadlům. K tomu využívají detektory hromadné odesílání event. Při navázání kontaktu s prvním letadlem začne detektor části svého letového plánu pravidelně rozesílat. Po každém rozeslání si naplánuje eventu za určitý čas. Tato eventa pak detektor upozorní, aby znovu rozeslal část letového plánu.

Plány se posílají pouze po částech z několika důvodů:

- Odeslaný úsek ve většině případů stačí k tomu, aby byla detekována kolize. Díky periodickému odesílání částí plánů mají letadla dostatečnou část plánu pro včasnou detekci a vyřešení kolize.
- Plán se může brzy z důvodu vyřešení kolize změnit a předchozí plán již není platný. Proto není tak důležité detekovat vzdálené kolize.
- Kvůli minimalizaci objemu přenesených dat.

Detektor je schopen zpracovat následující eventy:

- Kooperativní pozdrav detektorů – obsahem této zprávy je typ letadla odesílatele, důležitý pro řešení kolizí.
- Letový plán – část letového plánu jiného letadla, určená k detekci kolizí.
- Žádost o delší letový plán – z důvodu nalezení kolize, jejíž koncový bod nebylo možné z dříve poslaného plánu určit.
- Údaje z vlastního radaru – údaje z detekce ostatních letadel na radaru pomocí senzorů letadla. Rozlišuje se pouze, jestli letadlo *je/není* na radaru.

- Informace o stavu radaru jiného letadla. Tato informace je potřeba pro ukončení kontaktu mezi dvojicí letadel. Dokud senzory alespoň jednoho z letadel detekují druhé letadlo, kontakt být ukončen nesmí. Relativně komplexním prolémem, nicméně velmi důležitým, je korektně zajistit, aby letadla správně kontakt ukončila. Řešení tohoto problému je popsáno v sekci 3.1.2.
- Eventy, které detektor pošle sám sobě. Tou první je upomínka pro opětovné rozeslání plánů ostatním letadlům. Jiná eventa upozorňuje, aby letadlo prohlásilo spojení za ukončené a informace o druhém letadlu smazalo. Další eventy slouží pro testovací účely.

3.1.1. Detekce kolizí

Detektor porovnává svůj letový plán s úsekem letového plánu, poskytnutého druhým letadlem. Najde-li počáteční bod kolize, hledá i její koncový bod, tedy první následující stav, kdy spolu letové plány nekolidují.

Pokud takové místo nelze najít, alespoň jeden z letových plánů, na nichž je detekce prováděna, je pro detekci krátký. Proto detektor zažádá o delší úsek letového plánu. Delší úsek má dvakrát větší časový interval, než standartně odesílaný úsek. Po přijetí delšího plánu se detekce provede znovu.

Pokud již detektor poslal svůj letový plán, obsahující celý zbylý úsek až do jeho úplného konce, odešle eventu s prázdným obsahem. Druhému letadlu takto indikuje, že byl celý letový plán již odeslán v předchozí eventě s letovým plánem. Konec kolize musí být přesto určen. Detektory definují čas konce kolize jako čas konce kratšího z plánů, na kterých je kolize řešena. K nastavenému času se přiřadí i příslušná místa v letových plánech obou letadel.

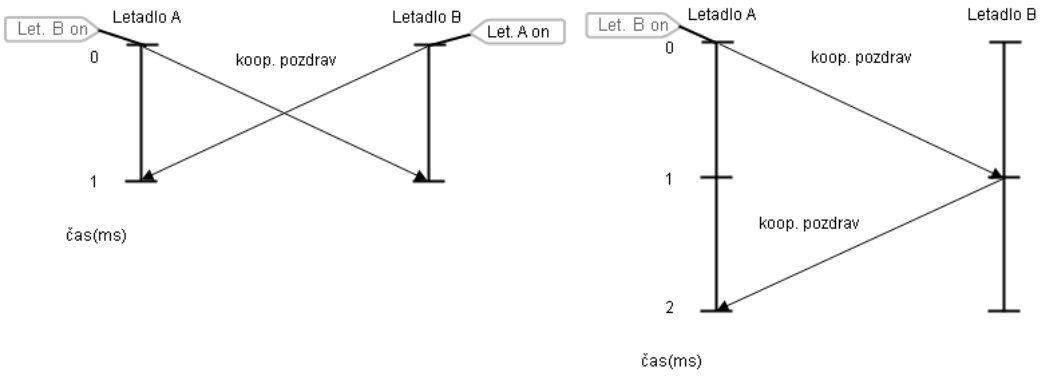
Nalezené kolize se registrují u správce kolizí (viz sekce 3.2). Ostatním částem algoritmu IPPCA slouží detekce v následujících případech:

- Po změně vlastního letového plánu (např. z důvodu aplikace úhybného manévru) je potřeba znovu zkontrolovat kolize s ostatními letadly.
- V případě, že je solver kolizí vyzván k řešení kolize letadlem, které není registrované jako nejdříve kolidující letadlo. Detekce kolizí se použije pro ujištění, že je seznam kolizí aktuální.
- Kontroluje-li solver, jestli generované plány nevytváří bližší kolizi s ostatními letadly, provádí tuto kontrolu také použitím tohoto detektoru. Takto detekované kolize se u správce kolizí neregistrují.

3.1.2. Navázání a ukončení komunikace

Pro kooperativní detekci je nutné, aby mělo letadlo k dispozici letové plány ostatních letadel a informace o typu ostatních letadlel. Tyto informace posílá algoritmus při kooperativním pozdravu a je důležité, aby vzájemnou kolizi (pokud ji lze detekovat) mohla zdetekovat obě letadla. Navázání komunikace kooperativních detektorů může být symetrické i asymetrické. K symetrickému navázání komunikace dochází, pokud mají obě letadla stejný rozsah radarů. Detekce druhého letadla na radaru proběhne u obou letadel ve stejnou milisekundu. Oba detektory odešlou kooperativní pozdrav a letové plány

ve stejný čas (viz levá část obrázku 5). Asymetrické navázání komunikace nastává, pokud se letadla navzájem detekují na radarech v jiný čas. Na kooperativní pozdrav musí kooperativní detektor odpovědět také kooperativním pozdravem a odesláním letového plánu, aby mohl detektor druhého letadla zdetekovat kolize (viz pravá část obrázku 5). Kooperativní detektor takto musí odpovědět i v případě, že jeho senzory druhé letadlo nedetekují.



Obr. 5. Navázání komunikace kooperativních detektorů. Orámované popisky značí systémové informace z radaru. Vnitřní popisek on/off v tomto rámečku značí, že nově je/není druhé letadlo detekováno na radaru. Odesílání letových plánů a informací z radaru je pro udržení přehlednosti nezobrazeno.

K ukončení komunikace dochází, když se obě letadla navzájem nedetekují na svých radarech. Dosah senzorů radaru není na všech letadlech stejný. Proto nemůže letadlo předpokládat, že když nedetekuje druhé letadlo, že jím zároveň není detekováno. Z důvodu komunikace eventami, jejichž minimální doba přenosu je 1 ms, byl navržen speciální způsob ukončení komunikace mezi letadly, (viz obrázek 6):

1. Detektor reaguje na změny aktualizace vlastního radaru okamžitě. Odešle eventu druhému letadlu s binární informací, že již druhé letadlo na radaru *detekuje/nedetekuje*.
2. Ukončení komunikace nemusí být vždy symetrické a každé letadlo ho řídí individuálně – každé z letadel může ukončit komunikaci v jinou milisekundu (viz Situace 1 a 2 na obrázku 6).
3. K ukončení komunikace dojde nejdříve za 2 ms od odeslání negativní informace. Po uplynutí této doby se ihned po přijetí negativní informace od druhého letadla komunikace ukončí, za předpokladu, že vlastní senzory stále nedetekují druhé letadlo na radaru (viz Situace 2 na obrázku 6 z pohledu letadla A).
4. Pokud (viz Situace 4 na obrázku 6) u *letadla B* dojde během dvou milisekund od odeslání negativního stavu radaru k opětovné detekci *letadla A* na radaru a detektor letadla B do této doby obdržel i odeslal negativní informaci, nepošle letadlu A pouze pozitivní informaci z radaru, ale i kooperativní pozdrav a letový plán.

   V ostatních případech opětovné detekce druhého letadla na radaru stačí poslat pouze pozitivní informaci (viz Situace 3 na obrázku 6). Tento postup slouží ke správnému opětovnému navázání komunikace. Druhé letadlo by v čase přijetí pozitivní informace již nejspíš (ne vždy, viz bod 5.) provedlo ukončení komunikace a samotná informace o stavu radaru od již neznámého letadla by pro něj byla bezcennou informací. Pokud letadlo B komunikaci neukončí, jako pozitivní informaci

z radaru prvního letadla nyní očekává kooperativní pozdrav, nebo jen pozitivní informaci od prvního letadla (viz Situace 5 na obrázku 6).
Je pouze jeden případ, kdy se při opětovné detekci druhého letadla odesílají pouze pozitivní informace z radaru. Tou je detekce pozitivního stavu 1 ms poté, kdy byla odeslána negativní informace a zároveň jen tehdy, pokud detektor ještě neobdržel negativní informaci od druhého letadla. Tato situace nastává pouze v symetrickém odeslání negativní informace (viz Situace 3 na obrázku 6).

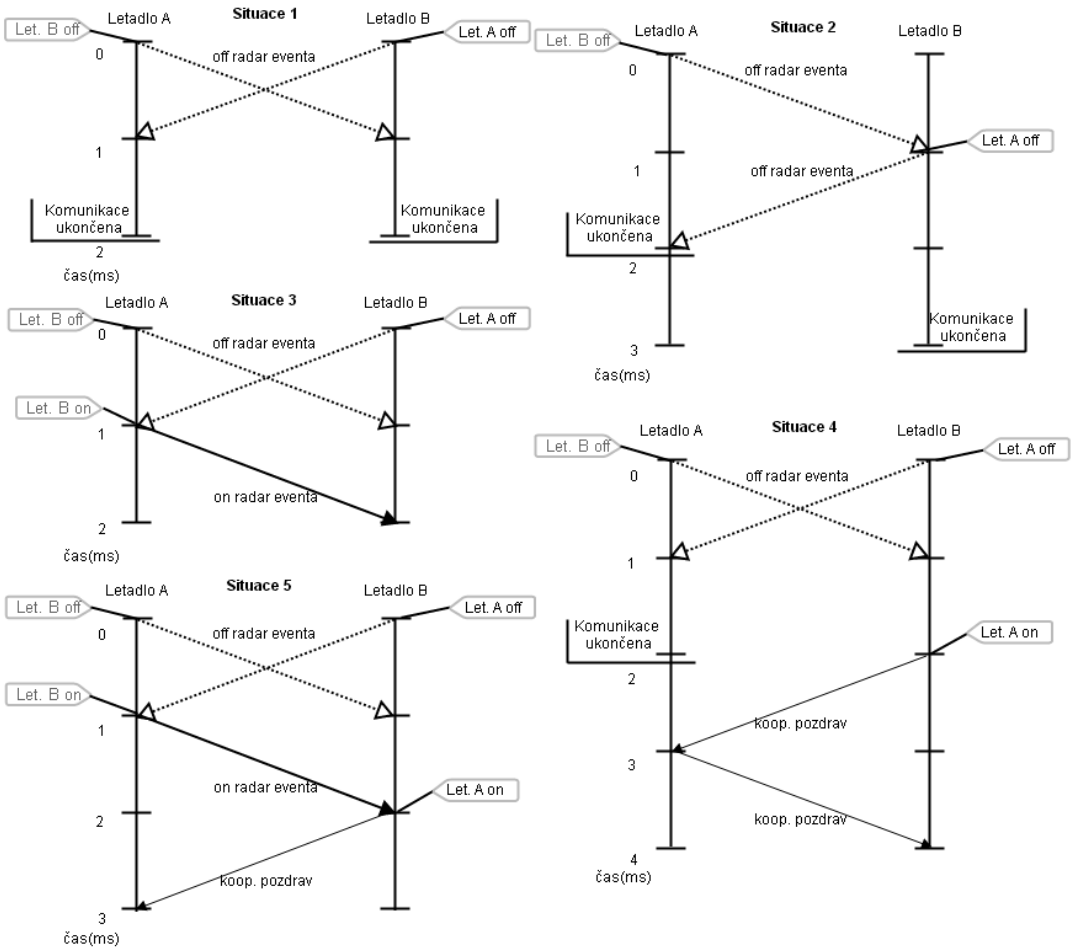
5. V Situaci 5 na obrázku 6 nastává podobná situace jako Situace 4. Rozdílné jsou informace z radaru u letadla A, jehož senzory detekují v čase 1 ms letadlo B. Detektor reaguje odesláním pozitivní informace letadlu B. Letadlo B se při detekci letadla A chová stejně, jako v Situaci 4, protože je o detekci letadla A na radaru informován ještě před přijetím negativní informace od letadla A. V této situaci tedy musí poslat s pozitivní informací i kooperativní pozdrav a letový plán. Důvodem je pořadí priorit detekční a informační eventy. Letadlo A kooperativní pozdrav ignoruje (letový plán naopak použije pro detekci kolizí). Informace, obsažené v kooperativním pozdravu, již obdrželo dříve. Kooperativní pozdrav a letový plán letadlo A posílat nemusí, protože letadlo B má tyto informace také.
6. Předchozími postupy jsou ošetřeny všechny kombinace údajů z radarů u dvou letadel (situace se vždy bere v úvahu po párech letadel) pro správné ukončení komunikace nebo pro její opětovné navázání. Pozitivní detekce druhého letadla po ukončení komunikace je případem navázání komunikace, protože žádné informace o druhém letadlu již neexistují. Proto se takový případ řídí postupem, znázorněným na obrázku 5.

3.2. Návrh správce kolizí

Správce kolizí je nutný k rozhodování, která kolize má být vyřešena. Je to spojující prvek mezi detektory a solvery. Detektor registruje nalezené kolize u správce kolizí, který je spravuje, řadí a vybírá kolizi kterou bude solver řešit. Koliduje-li letadlo s více letadly, řeší postupně kolize od té nejbližší v čase. Po vyřešení jedné kolize se znovu provede detekce všech kolizí a nalezení nejbližší kolize. Výzvy pro řešení kolize, které nejsou nejbližší, jsou odmítány. Proto se v případě hromadné kolize vždy vyřeší partikulární kolize podle jejich pořadí v čase od nejbližšího k nejvzdálenějšímu.

Pro vybrání nejprioritnější kolize, kterou bude solver řešit je nutné zajistit, aby byly zdetekovány opravdu všechny kolize, které v daném čase zdetekovat lze. Letadlo nemůže předpovědět, od kolika letadel v tomto čase (v dané milisekundě, viz sekce 2.3.1) obdrží letové plány. Všechny letové plány dorazí bezprostředně po sobě a žádný jiný typ eventy mezi nimi nedorazí. Protože ale nelze předem určit, kolik bude těchto event celkem, situace je řešena dvěma způsoby:

- Manažer sám sobě pošle eventu na další milisekundu (dřívější doručení systém neumožňuje). Tato eventa má nižší prioritu, než eventa s letovými plány. Proto je v další milisekundě při přijetí této eventy zaručena kompletnost všech detekovaných kolizí.
- Solver reaguje na eventu s nižší prioritou, než má eventa s letovými plány. Pokud je solver vyzván jiným solverem k řešení kolize, příslušná eventa má nižší prioritu než letové plány. Seznam kolizí je proto aktuální a kompletní a může být zkontrolováno, zdali vytváří prioritní kolizi vyzívající letadlo.



Obr. 6. Ukončení komunikace kooperativních detektorů. Orámované popisky značí systémové informace z radaru. Vnitřní popisek on/off v tomto rámečku značí, že nově je/není druhé letadlo detekováno na radaru. On/off radar eventy značí informaci o tom, že příjemce eventy je/není detekován na radaru. Tečkované čáry značí eventu s negativní informací z radaru, tučné plné čáry značí eventu s pozitivní informací z radaru a tenké plné čáry značí kooperativní pozdrav. Pozn.: Pro udržení přehlednosti není v případě odesílání kooperativního pozdravu zobrazeno odesílání letových plánů pozitivních informací z radaru.

Správce kolizí třídí kolize vzestupně, podle času, kdy nastanou. Při výskytu dvou a více kolizí ve stejný čas jsou kolize řazeny podle identifikátoru letu ostatních letadel. Při rovnosti časů primárně záleží na determičnosti řazení, spíše než na upřednostnění kolize s letadlem s opravdovou vyšší prioritou letu.

Při vyřešení kolize solverem se daná kolize ze seznamu vyřadí. Pokud došlo ke změně letového plánu, vyřadí se i všechny ostatní detekované kolize a detektor provede detekci kolizí u všech letadel, se kterými je ve spojení.

3.3. Návrh kooperativního řešitele kolizí

Pro řešení kooperativní kolize je třeba navrhnout a implementovat kooperativního řešitele, nebo-li solver. Jeho hlavní funkcí je generování letových plánů a hledání řešení

s nejoptimálnějším společným užitkem – tzv. *společnou utilitou*. Druhou součástí je komunikační čast, sloužící k výměně informací mezi solvery jednotlivých letadel. Solver musí být schopen řešení kolize řídit i být řízen solverem druhého letadla, tj. fungovat jako master i jako slave. „Jedinými doménově závislými komponentami v algoritmu jsou definice funkce utility a vyhýbacích manévrů.“ [3]

3.3.1. Komunikační část

Pro vyřešení kolize je nutné, aby obě letadla souhlasila, že budou kolizi v daný moment řešit. Někdy s řešením kolize jedno z letadel nesouhlasí, protože jeho prioritní kolize je jiná. Toto dorozumívání se děje za pomocí event, které musí obsahovat správnou verzi letového plánu příjemce. Tím se zajistí, že mají obě letadla aktuální letové plány letadla, se kterým budou řešit kolizi. Proto je jisté, že obě letadla detekují a řeší totožnou kolizi. Neobsahuje-li tato eventa správnou verzi letového plánu, odesílatel je o tom informován příslušnou eventou. Algoritmus je navržen tak, že je připraven i na situaci, kdy s jedním letadlem požaduje řešení kolize více letadel ve stejný čas. Letadlům, jejichž kolize nejsou pro toto letadlo prioritní, odešle solver odmítnutí pomocí eventy.

Dohodnou-li se dvě kolidující letadla, že spolu budou řešit kolizi, celé další vyjednávání algoritmu IPPCA probíhá v rámci zpracovávání jedné eventy u obou letadel. Obě letadla musí zajistit, že zahájí toto vnitřní vyjednávání ve stejnou simulační milisekundu. Pouze pro řešení kolize se místo event uplatní posílání zpráv, které jsou nezávislé na simulačním čase, který je v tomto případě pozastaven. Samotné vyřešení kolize proto trvá 0 ms simulačního času. Po skončení řešení kolize se zpracování rozpracované eventy ukončí. Ukončení proběhne nezávisle na tom, zda byla kolize vyřešena nebo byl průběh řešení přerušen. Pokud by jedno z letadel zahájilo vnitřní vyjednávání o řešení kolize dříve, než druhé, nebylo by možné kolizi vyřešit v rámci zpracování jedné eventy u obou letadel. Jedno z letadel by v takovém případě nemělo v úmyslu kolizi řešit a nastaly by problémy s průběhem simulace z důvodu pozastavení rozpracování eventy u jednoho z letadel.

3.3.2. Řešící část

Generování letových plánů pro vyřešení kolize je řízeno parametrizací. S ohledem na optimalizaci společné utility se v prvním kole řešení používají manévry s malou odchylkou vzhledem k původnímu vektoru letu. Tzn. například pouze mírné změny rychlosti a vertikálního a horizontálního směru letu. Do první sady generovaných plánů se navíc přidá původní plán beze změn. Při každém generování plánů se zvýší parametr, určující velikost odchylky generovaných plánů od původního směru letu. Postupným generováním se zvětšuje stavový prostor letových plánů. Další způsob generování letových plánů je navržen v sekci 4.1.

„Plán, upravený použitím manévru, obsahuje utilitu, což je vážený součet několika částí, za použití následující rovnice:

$$u = \frac{\sum_i \alpha_i u_i}{\sum \alpha_i},$$

kde α_i je váha pro i-tou komponentu funkce utility. Funkce utility se používá pro zahrnutí záměru letadla v navrženém řešení problému. V závislosti na konfiguraci může obsahovat různé komponenty, které je třeba vzít v úvahu, jako například délka plánu, priorita letů, stav paliva a další faktory.“ [3]

V implementaci slouží pro určení utility plánů tzv. *evaluátor*, který hodnotí letový plán pomocí funkce utility. Použitý evaluátor v implementaci hodnotí hodnotí plán podle množství spáleného paliva při jeho provedení. Další evaluátor, hodnotící délku trvání letu, byl implementován zejména pro účely testování v kapitole 6.

K vyřešení kolize nesmí být použity plány, které vytvářejí dřívější kolizi s ostatními letadly. Proto vyřešením kolize zpravidla nevznikne jiná dřívější kolize. Pouze při simultánním vyřešení více kolizí může ke vzniku dřívější kolize dojít. Důvodem je simultánní aplikace nových letových plánů. Letadla o změnách letových plánů ostatních letadel v tomto čase nejsou informována. Proto může ke dřívější kolizi dojít.

4. Návrh rozšíření algoritmu IPPCA

Pro implementovaný algoritmus IPPCA je navrženo jeho rozšíření. Rozšíření se týká efektivity generování letových plánů pro kooperativní řešení kolize mezi dvěma letadly. V kapitole 6 je rozšíření porovnáno se základní variantou algoritmu IPPCA.

4.1. Heuristický generátor letových plánů

Řozšiřující algoritmus používá parametrizovaný přístup pro prohledávání stavového prostoru generovaných plánů s nejoptimálnější odhadovanou hodnotou utility. Snaží se minimalizovat počet generovaných letových plánů s ohledem na jejich utilitu. V základní verzi algoritmu při řešení kolize generují letadla letové plány vždy stejně. V každém kole algoritmu je vygenerován jeden letový plán pro každý manévr, které má letadlo definovaný. Parametr pro generování plánu se v každém kole zvýší o konstantní krok. V situaci, kdy se ve vzdušném prostoru vyskytuje malé množství letadel, je tento přístup dostačující. Se zvyšujícím se zahuštěním letového prostoru je vhodné navrhnout algoritmus, který generuje letové plány efektivněji. Navržené rozšíření se týká způsobu parametrizace manévrů a výběru jednotlivých manévrů. Rozšiřující algoritmus pracuje se stejnými informacemi od detektoru o vygenerovaných plánech, jako původní verze algoritmu.

Cílem algoritmu je schopnost efektivnějšího generování validních letových plánů. V závislosti na nasazení algoritmu se však liší požadavky a náhled na efektivitu generování plánů. Algoritmus byl navržen tak, aby byl schopen se částečně přizpůsobit požadavkům pro nasazení. Různá nastavení algoritmu ovlivňují:

- množství generovaných plánů – výpočetní a paměťovou náročnost. Důležité minimalizovat pro nasazení na hardware, jehož výpočetní nebo paměťové prostředky jsou velmi omezené;
- množství poslaných plánů – objem přenesených dat při komunikaci. Důležité minimalizovat ve vzdušném prostoru s vysokou hustotou komunikace nebo pro jiné důvody omezení komunikace;
- počet iterací algoritmu – počet zpráv, které musí být přeneseny, než se kolize vyřeší;
- trajektorii a hodnotu utility výsledného řešení.

Důraz je kladen na univerzálnost pro nasazení algoritmu. Algoritmus zachovává doménovou nezávislost algoritmu IPPCA. Je nezávislý na typu letadla, na definovaných manévrech i na jejich počtu. Nerozlišuje se, jestli algoritmus pracuje na letadle mastera nebo slavea kolize. Způsob hledání plánů s optimální utilitou je nezávislý na použitém evaluátoru.

4.2. Popis fungování algoritmu

Záměrem algoritmu je, aby každý směr v prohledávání stavového prostoru individuálně minimalizoval počet iterací mezi nalezenými validními letovými plány. Zohledněna musí být i kvalita řešení, tedy utilita letového plánu. Jednotlivé manévry poskytují predikci utility generovaného plánu v případě, že bude stavový prostor rozvinut ve směru tohoto manévru. Řídící prvek algoritmu rozhodne podle predikovaných utilit a koeficientu pro každý manévr, kterým směrem stavový prostor rozvine (který manévr bude vybrán).

Validita letových plánů je omezena trajektoriemi ostatních letů a jinými statickými nebo dynamickými bezletovými zónami, které jsou v prostoru různě rozmístěny. Algoritmus proto používá pro jednotlivé manévry zvláštní parametr pro generování letových plánů. Každý manévr se s ohledem na kvalitu snaží najít parametr takový, aby byl generovaný plán validní, tj. aby nevytvářel dřívější kolizi s ostatními letadly. Zvyšování parametru o jeden krok při každém vygenerování je zachováno. Může se ale lišit velikost kroku – pokud poslední generovaný plán nebyl validní, délka kroku se zvýší. Algoritmus se takto snaží rychleji zabránit v dalším generování nevalidních letových plánů. Po vygenerování validního plánu se velikost kroku vrátí zpět na původní hodnotu.

U každého manévru algoritmus predikuje utilitu dalšího generovaného letového plánu. Druh predikce se může lišit v závislosti na použitém evaluátoru a okolních podmínkách. Byly použity dva prediktory utility – *lineární extrapolační prediktor* a *kvadratický extrapolační prediktor*. Predikci provádí metodou lineární, resp. kvadratické extrapolace. Výpočet závisí pouze na posledních dvou, resp. třech validních plánech pro daný manévr (nevalidní plány se neukládají).

Koeficient pro i-tý manévr se aktualizuje na počátku generování každé sady letových plánů pomocí rovnice

$$k_i = \begin{cases} \dfrac{1}{|V_i|}, & |V_i| > 0, \\ \\ 1, & \text{jinak}, \end{cases}$$

kde $|V_i|$ je počet validních generovaných plánů pro i-tý manévr. Algoritmus pomocí koeficientů pracuje se znalostí, že řešení kolize nebylo v předchozích iteracích nalezeno. To znamená, že byl stavový prostor v řešení nejspíše rozvinut špatným směrem. Aktualizací koeficientů se proto upřednostní méně použité manévry.

Pro výběr zvoleného manévru se jednotlivé manévry seřadí. Při porovnání i-tého a j-tého manévru je priorita p určena následujícím způsobem:

$$p(i,j) = \begin{cases} s_i < s_j, & sign(i,j) = -1, \\ s_j < s_i, & sign(i,j) = 1, \\ s_i = s_j, & sign(i,j) = 0, \end{cases}$$

kde symbol $<$ znamená, že prvek nalevo má vyšší prioritu a symbol $=$ značí rovnost obou prvků. Funkce $sign$ nabývá následujících hodnot:

$$sign(i,j) = \begin{cases} -1, & \Big((|p_i - p_j| < R) \wedge (u_i \cdot c_i < u_j \cdot c_j)\Big) \vee \Big((|p_i - p_j| \geq R) \wedge (p_i < p_j)\Big), \\ 1, & \Big((|p_i - p_j| < R) \wedge (u_i \cdot c_i > u_j \cdot c_j)\Big) \vee \Big((|p_i - p_j| \geq R) \wedge (p_i > p_j)\Big), \\ 0, & \text{jinak}, \end{cases}$$

kde p_i a u_i značí parametr, respektive odhadovanou utilitu pro i-tý manévr. R je konstanta, daná konfigurací – definuje maximální rozdíl velikostí parametrů dvou manévrů, který je limitem pro porovnávání podle predikce utility a koeficientu. R slouží pro ošetření uvíznutí při parametrizovaném hladovém postupu algoritmu. Jedinou výjimkou při porovnání je manévr, generující plán beze změn. Manévr beze změn je po vygenerování jednoho plánu vždy považován za méně prioritní.

Při generování první sady vygeneruje algoritmus jeden letový plán pro každý z definovaných manévrů. Pokud je maximální velikost sady vyšší, než počet manévrů, další manévry se již vyberou v pořadí podle jejich seřazení. Ve všech dalších kolech algoritmu IPPCA se vždy vybírá první prvek po seřazení. Predikce utility se provádí po generování každého validního plánu. Koeficienty manévrů se aktualizují až po generování celé sady.

Stanoveným cílem je zjistit konfiguraci algoritmu, která minimalizuje počet vygenerovaných plánů pro nalezení řešení. Zároveň by měl algoritmus, vzhledem k původní variantě, zachovávat hodnotu utilit na přibližně stejných nebo lepších hodnotách. Minimalizace počtu generovaných plánů je dána zejména zvyšováním kroku v jednotlivých směrech. Pro nalezení optimální hodnoty velikosti generované sady letových plánů bude proto testováno několik nastavení této velikosti. Postupným zvýšováním velikosti generované sady dostane algoritmus lepší podmínky pro optimalizaci utility – odesláním větší sady se zvyšuje počet validních kombinací letových plánů obou letadel, které řeší kolizi. S větším počtem validních řešení roste pravděpodobnost výskytu lepších hodnot výsledných utilit. Počet generovaných plánů v jedné sadě musí být shora omezen. Toto omezení samo o sobě omezuje celkový počet vygenerovaných i přenesených plánů. Hlavním důvodem je ale funkčnost koeficientů k_i (viz první rovnice v této sekci). Tento koeficient brání algoritmu v pokračování rozvíjení stavového prostoru špatným směrem. Jelikož se koeficienty k_i aktualizují mezi generovanými sadami, je neúčinné volit větší velikost sady plánů.

5. Implementace

Projekt AgentFly i nově implementovaný algoritmus a jeho rozšíření je napsán v programovacím jazyce Java. Veškerá implementace byla v projektu AgentFly provedena v balíčku `agentfly.atm.freeflight`. Implementace využívá různé části projektu AgentFly. Potřebné části byly za tímto účelem importovány do implementace.

5.1. Implementace deterministické varianty algoritmu IPPCA

Nejdůležitějšími nově implementovanými třídami jsou:

- `CooperativeCollisionDetector` – kooperativní detektor kolizí. Obsahuje všechny potřebné funkce pro kooperativní detekci a komunikaci detektorů. Informace o letadlech uchovává v objektech třídy `AirplaneRecord`.
- `CdrManager` – Správce kolizí. Vybírá kolizi, která má být vyřešena. Informace o kolizích uchovává ve třídě `RegisteredCooperativeCollision`.
- `EomUavPilotIppcaCollisionSolver` – kooperativní solver kolizí. Informace o ostatních letadlech uchovává ve třídě `DeconflictionEntityIppca`.

Další nově implementované třídy, důležité pro chod algoritmu, jsou:

- `CooperativeCollisionData` – uchovává informace o kooperativních letadlech – identifikátor letu, typ letadla, letový plán a pozice případné kolize. Je to univerzální třída pro uchování informací pro libovolný kooperativní detektor.
- `AirplaneRecord` – potomek třídy `CooperativeCollisionData`. Nadtřídu rozšiřuje o senzorické informace z radarů letadel.
- `DeconflictionEntityIppca` – shromažďuje informace pro solver – časy a verze poslaných žádostí o řešení kolize, generované plány, objekt třídy `AirplaneRecord` atd.
- `InitialIppcaData` – obsahuje informace pro výzvu k řešení kolize – adresu pro posílání zpráv a verzi plánu příjemce, na němž chce druhé letadlo řešit kolizi.
- `PlanWrapperPair` – uchovává informace o generovaném plánu letu při řešení kolize.
- `CdrConstants` – definuje hodnoty konstant, použitých v nově implementovaných třídách.
- `RegisteredCooperativeCollision` – uchovává informace o detekované kolizi od jakéhokoliv detektoru. Objekty této třídy používá třída `CdrManager`.
- Další třídy pro vytvoření scénářů pro testováaní algoritmu.

Eventy jsou v projektu definovány v adresáři `aglobex.simulation.timemanager.events`. Pořadí event odeslaných systémy letadla definuje původní návrh projektu. Systémové eventy mají nejvyšší a pevně dané priority. Jejich pořadí určuje třída `SystemEventsConstants`. Pro určení pořadí ostatních event pro implementované třídy byla vytvořena třída `CdrConstants`. Všechny třídy, které přijímají eventy se musí pro příjem event registrovat u třídy `EventOrganizer` pomocí metody

registerEventReceiver. Všechny tři hlavní imlementované třídy registraci provádí ve svých iniciujících funkcích. U detektoru a správce kolizí je to metoda register a pro solver metoda initAndConfigurationPreDeregister.

Hromadné rozesílání event se v algoritmu využívá pouze pro odeslání letových plánů. Ze strany odesílatele funguje odesílání hromadných event podobně, jako odeslání eventy pouze jednomu letadlu. Jako příjemce se určí složení dvou řetězců: konstantní předpony pro určitou hromadnou skupinu a identifikátoru letu odesílatele. Příklad odeslání hromadné eventy pro kooperativní detektory: CdrConstants.COOPERATIVE_COLLISION_DETECTION_PREFIX+flightId, kde flightId je identifikátor letu odesílatele. Odebírání určitých hromadných zpráv řídí sami příjemci. O odběr do se příjemce přihlásí pomocí metody joinGroup a odběr zruší metodou disjoinGroup.

5.1.1. Detektor kolizí

Detektor zahrnuje komunikační a detekční část v jedné třídě. Implementuje funkcionalitu přijímače event – EventReceiver a zároveň i detekční části – CollisionDetector. Pro potřeby dalších tříd algoritmu byly implementovány i podpůrné detekční metody. Objekty třídy Airplane Record, uchovávající data o kontaktovaných letadlech a jejich koliziích jsou v detektoru uloženy v poli HashMap, kde je jako klíč použit identifikátor letu kolidujícího letadla. Po ukončení spojení s tímto letadlem je příslušný objekt smazán.

Detekce kolizí

K určení prvního a posledního bodu kolize byly použity systémové nástroje projektu AgentFly – metody findFirstCollision a findFirstNonCollisionPosition. Tyto metody pracují s letovými plány dvojice letadel. Detekovaná kolize se u správce kolizí zaregistruje pomocí metody registerCollison. Solver využívá při generování plánů detekční funkci detektoru – checkWithOtherAirplanes. Tuto funkci musí implementovat každý kooperativní detektor. Funkce checkWithOtherAirplanes kontroluje generovaný plán s ostatními letadly (kromě toho, se kterým se kolize řeší – je předán parametrem). Výstupní hodnotou je minimum z naměřených vzdáleností pomocí vestavěné metody findCollisionFreeSafetyZone. Správce kolizí využívá detekční funkci recheckCollisions, která detekuje všechny aktuální kolize a zaregistruje je ve třídě CdrManager.

5.1.2. Správce kolizí

Správce kolizí je implementován ve třídě CdrManager. Pro uložení jednotlivých kolizí využívá manažer seznam typu ArrayList. Pro řazení kolizí byl implementován komparátor, řadící kolize vzestupně podle času počátku kolize. Při rovnosti tohoto času se kolize řadí alfabeticky podle identifikátoru letu druhého letadla. Pro registraci kolizí byla implementována metoda registerCollision. Implementace třídy CdrManager je připravena na použití více druhů detektorů i více druhů solverů. V současné době je použit pouze implementovaný kooperativní detektor a implementovaný solver IPPCA.

Vyřešená kolize se odregistruje pomocí metody `deregisterCollision`. Metoda `deregisterCollision` vyřadí kolizi ze seznamu kolizí. Pokud se změnil letový plán letadla správce kolizí, kolize jsou vyřazeny všechny a je zavolána detekční metoda `recheckCollisions`. Pro výběr kolize k řešení slouží metoda `manageSolving`, která kolize seřadí a vybere tu prioritní. Pro kolizi vybere solver, který ji bude řešit a vyzve ho k řešení zavolám metody `solve`.

Další funkce třídy `CdrManager` byly implementovány pro potřeby solverů. Jsou to následující funkce:

- `boolean isCollidingWith` vrací *true/false* pokud letadlo předané parametrem je/není v seznamu kolizí.
- `boolean isClosestCollision` vrací *true/false* pokud letadlo předané parametrem vytváří/nevytváří nejprioritnější kolizi.
- `String getClosestCollision` vrací identifikátor letu letadla, vytvářejícího nejprioritnější kolizi.

5.1.3. Řešitel kolizí

Při implementaci kooperativního solveru byly využity některé metody z původní nedeterministické varianty algoritmu. Metody byly spíše modifikovány pro deterministickou variantu. Jsou to metody

- `generatePlans` – generuje jednotlivé sady letových plánů.
- `plansGenerated` – po generování, v případě že je letadlo určeno jako solver, odešle generované letové plány masterovi. Pro mastera tato metoda slouží k započetí hledání kolize voláním metody `putPlansTogether`.
- `putPlansTogether` – pomocí metod `fillPairs` a `findPairs` hledá validní řešení z vygenerovaných letových plánů obou letadel.
- `fillPairs` – provedení kartézského součinu generovaných letových plánů.
- `findPairs` – nalezení všech validních párů s nejoptimálnější utilitou.
- `plansPutTogether` – nalezené řešení pošle slaveovi, v případě nenalezení řešení posílá zprávu slaveovi s žádostí o další letové plány.

Komunikace

Pro příjem zpráv byla implementována metoda `handleIncomingMessage` a pro zpracování event metoda `processEvent`.

V průběhu řešení kolize probíhá komunikace pomocí zpráv – objektů třídy `Message`. Oba solvery, které řeší kolizi, musí provést komunikaci pomocí zpráv ve stejný simulační čas. Umožňuje to použití event, jejichž okamžité zpracování značí návratové hodnoty `EventProcessingState.processedAndStop` a `EventProcessingState.processedAndPass`. Pro označení eventy jako rozpracované se využijí návratové hodnoty `EventProcessingState.processingAndStop`, popřípadě `EventProcessingState.processingAndPass`. Při použití těchto návratových hodnot je simulační čas pozastaven do doby ukončení zpracování této eventy. Po ukončení řešení kolize se ukončí zpracování eventy zavoláním metody `eventProcessed`.

Řešení kolizí

Každý solver, nezávisle na jeho kooperativitě, musí implementovat metodu `solve`. Tato metoda přijímá prioritní kolize, vybrané správcem kolizí a zahajuje vyjednávání pro řešení kolize. V případě neočekávané výzvy k řešení kolize od jiného letadla volá solver metodu třídy `CdrManager` – `manageSolving`. Tato situace nastává v případě, kdy správce kolizí čeká na registraci všech kolizí a kolidující letadlo pošle zároveň s letovými plány i žádost o řešení kolize.

Pro uchování informací o letadlech, které chtějí kolizi řešit a těch, které požaduje k vyřešení správce kolizí, byla implementována třída `DeconflictionEntityIppca`. Objekty s těmito informacemi jsou uloženy v poli `HashMap`, kde je jako klíč zvolen identifikátor letu druhého letadla. Po vyřešení kolize je z důvodu nižší paměťové náročnosti objekt z pole `HashMap` smazán. Každý objekt `DeconflictionEntityIppca` obsahuje informace:

1. od solveru – verze a čas odeslané kooperativní výzvy k řešení, vlastní generované plány a parametr pro jejich generování;
2. od správce kolizí – identifikátor letadla, čas prvního bodu kolize, další informace o letadlu v objektu třídy `CooperativeCollisionData` – v případě implementovaného kooperativního dektektoru kolizí je to objekt třídy `AirplaneRecord`;
3. od letadla, reprezentovaného objektem – adresa pro posílání zpráv, verze letového plánu příjemce, generované plány druhého letada a poslední přijatá zpráva.

Pro generování plánů slouží metoda `generatePlans`. Nový plán se generuje na kopii současného letového plánu použitím definovaných manévrů. Vybraný manévr se aplikuje integrovanou funkcí `applyOperator`, která zároveň určí utilitu letového plánu podle předaného evaluátoru. V implementaci byl použit evaluátor spotřeby paliva `GpsFuelEvaluator`. Funkcí detektoru `checkWithOtherAirplanes` se zjistí validita generovaného plánu. Je-li návratová hodnota funkce ostře větší než 0, tento plán je validní a může být použit pro hledání řešení kolize. Validní letové plány jsou uloženy v objektu `DeconflictionEntityIppca`. Pro odeslání letových plánů se plány vloží do objektu třídy `GeneratedPlans` spolu s jejich identifikátorem, utilitou, označením, zda je kolize vyřešena a dříve zjištěnou maximální velikostí prostoru.

Funkce `findPairs` vybere validní páry s nejlepší utilitou. Validitu párů určuje pomocí funkce `findCollisionFreeSafetyZone`. Pokud je více validních párů s nejlepší utilitou, řešení se z nich vybere náhodně. Generování náhodných čísel musí být z důvodu determiničnosti algoritmu řízeno. Systém AgentFly řízení umožňuje pomocí registrace vlastního generátoru pseudonáhodných čísel.

Metoda `putPlansTogether` v případě nenalezení optimálního řešení v posledním kole algoritmu hledá co nejméně kolizní řešení postupným snižováním minimální povolené vzdálenosti letadel až na minimální definovanou hodnotu.

Metoda `plansPutTogether` v případě nevyřešení kolize posílá žádost slaveovi o další plány. Je-li kolize vyřešena pošle slaveovi identifikátor plánu, který má použít. Pro zrušení řešení kolize se používá zpráva typu `CANCEL_DECONFLICTION_MSG`. Metoda `finishDeconfliction` vyřešenou kolizi u správce kolizí odregistruje, odstraní objekt pro řešení kolize s druhým letadlem a zavolá metodu `manageSolving`. Po jejím zpracování se celá eventa uzavře zavoláním metody `eventProcessed`.

5.2. Implementace heuristického generátoru letových plánů

Heuristický generátor letových plánů byl implementován ve třídě `IppcaPlanGenerator`. Každý objekt této třídy slouží pro vyřešení jedné kolize. Po vyřešení kolize je z důvodu nižší paměťové náročnosti objekt smazán. Pro generování plánů využívá generátor následující proměnné, které mu jsou předány při volání konstruktoru:

- `ScenarioInformation info` – objekt, shromažďující informace o řešených kolizí – slouží pro analýzu efektivity algoritmu.
- `AirplaneRecord airplaneRecord` – objekt s informacemi o druhém letadle a o kolizi, která je řešena.
- `PlanWrapper clonedPlanWrapper` – kopie současného letového plánu, na které se budou aplikovat manévry.
- `UtilityValueEvaluator evaluator` – evaluátor generovaných plánů.
- `CollisionDetector collisionDetector` – detektor, který bude použit pro posouzení validity vygenerovaného plánu.
- `long unchangeablePointInMillis` – bod v čase, do kterého je letový plán neměnný.
- `double operatorStep` – nepoužívá se pro typ letadla, na kterém je simulace prováděna, pro generování plánů je ale jako parametr nutný. Je nastaven na hodnotu 0.
- `double szCheckingStep` – procentuální velikost kroku pro bezpečnostní zóny letadel při kontrole validity vygenerovaného plánu (viz také proměnná `szLimit`).
- `double szLimit` – minimální tolerovaná procentuální velikost nenarušené bezpečnostní zóny letadel. Používá se pro posouzení validity vygenerovaného plánu.
- `long paramLimit` – udává maximální počet vygenerovaných sad letových plánů pro vyřešení kolize. Tato proměnná je v současné implementaci pro strategii heuristického generátoru nevyužita, nicméně stále omezuje počet vygenerovaných sad letových plánů. V pozdější implementaci může sloužit například ke změně strategie algoritmu v posledních cyklech řešení kolize.
- `LinkedList<UtilityBasedOperator> operators)` – spojový seznam jednotlivých definovaných manévrů.

Konfigurační konstanty pro generování plánů mohou být upraveny podle požadavků pro nasazení. Generátor využívá následující konstanty:

- `int PLANS_PER_ROUND` – definuje, kolik plánů má být v vygenerováno pro jednu sadu plánů. Započítávají se i nevalidní generované plány.
- `int ESTIMATION` – definuje, která metoda má být použita pro odhad utility.
- `int MAX_PARAM_DIFF` – definuje maximální rozdíl parametrů při porovnání manévrů. V sekci 4.2 je reprezentována konstantou R.
- `boolean USE_COEF` – definuje, zda má být pro posouzení priorit manévrů použit koeficient. Při nastavení na hodnotu *false* se pro všechny manévry použije koeficient R s hodnotou 1.

Pro generování celé sady plánů slouží funkce `generatePlans`, která vrací spojový seznam vygenerovaných plánů. Tato funkce vybírá, který manévr bude použit pro generování plánu. Pro generování jednotlivých plánů využívá funkci `generateOnePlan`. Generování jednotlivých letových plánů funguje stejně, jako v původní variantě algoritmu, liší se pouze parametry pro generování.

Pro uložení informací o jednotlivých manévrech byla implementována vnitřní třída `OperatorProgress`. Objekty této třídy ukládá nadtřída `IppcaPlanGenerator` pro účely řazení v seznamu `ArrayList`. Třída `OperatorProgress` pracuje s následujícími proměnnými:

- `long generatedValidCount` – současný celkový počet validních generovaných plánů.
- `long skipped` – současný celkový počet vynechaných hodnot parametru pro generování plánů.
- `long nextParam` – značí, jaký parametr bude použit při dalším generování plánu.
- `long currentStepSize` – současná délka kroku parametru.
- `long lastParam` – poslední použitý parametr pro generování plánu.
- `double operatorCoef` – koeficient manévru. Neovlivňuje generování letového plánu – slouží pouze pro řazení manévrů.
- `UtilityBasedOperator operator` – objekt konkrétního manévru, který je při generování aplikován.
- `double estimatedValue` – odhadovaná hodnota utility v případě generování dalšího plánu. Odhad je proveden pro hodnotu parametru `nextParam`.
- `LinkedList<GeneratedPlanWrapper> validPlanWrappers` – spojový seznam validních generovaných plánů.
- `LinkedList<Long> generationParameters` – jednotlivé parametry, které byly použity pro generování validních plánů, uložených v seznamu `validPlanWrappers`.

Výpočet koeficientu (viz sekce 4.1) provádí funkce `recomputeNextCoef`. Pro porovnání objektů třídy `OperatorProgress` při jejich řazení byla implementována metoda `compareTo`. Odhad utility před vygenerováním plánu řídí metoda `recomputeEstimation`. V závislosti na konfiguraci používá pro odhad utility některou z extrapolačních funkcí. Implementované extrapolační funkce jsou `extrapolateLinear` a `extrapolateQuadratic`. Ty v závislosti na parametru `nextParam` odhadují utilitu pro daný manévr. Extrapolace se provádí pomocí proložení utilit posledních validních letových plánů lineární, resp. kvadratickou funkcí. Pro výpočet parametrů kvadratické funkce byla použita Gaussova eliminační metoda. Kód pro implementaci Gaussovy eliminační metody byl převzat z [6].

6. Srovnání a vyhodnocení

V této kapitole bude porovnána deterministická varianta algoritmu IPPCA s její rozšířenou variantou (viz kapitoly 3 a 4). Obě varianty jsou porovnány pomocí dvou evaluátorů (viz 3.3.2) ve dvou situacích. Celkově tedy čtyři různé testovací scénáře. Tyto scénáře byly spuštěny v simulačním prostředí projektu AgentFly. Pro testování jsou navrženy čtyři konfigurace rozšířené varianty algoritmu.

6.1. Nastavení testovacích scénářů a konfigurací algoritmu

Hodnota konstanty R (viz sekce 4.1) omezuje algoritmus při volení manévru s nejlepší odhadovanou utilitou. Jeho hodnota je u všech konfigurací generátoru nastavena na poloviční hodnotu velikosti generované sady letových plánů, které algoritmus vygeneruje při jedné iteraci hledání řešení. Při předchozím testování konfigurací vykazovaly ostatní hodnoty parametru R neoptimální výsledky pro většinu měřených hodnot. Testovaná letadla mají 6 typů předdefinovaných manévrů a k tomu jeden manévr beze změn. Původní algoritmus generuje první sadu, zahrnujících všech 7 manévrů. Další sady letových plánů generuje pro 6 manévrů (plán beze změn znovu negeneruje). Pro hledání manévru s nejlepší utilitou byly použity oba prediktory, představené v sekci 4.1.

První konfigurace (v tabulce 1 pod názvem Heuristika 1) slouží k minimalizaci počtu generovaných plánů. Používá stejnou velikost sady plánů, jako základní varianta algoritmu.

- Velikost generované sady plánů: 6
- Velikost parametru R: 3
- Predikce utility: lineární

Druhá konfigurace (v tabulce 1 pod názvem Heuristika 2) slouží k optimalizaci utility s ohledem na počet generovaných plánů. Tato heuristika je určitým kompromisem mezi optimalizováním utility a počtem vygenerovaných plánů.

- Velikost generované sady plánů: 12
- Velikost parametru R: 6
- Predikce utility: lineární

Třetí konfigurace (v tabulce 1 pod názvem Heuristika 3) je modifikací druhé konfigurace. Pro predikci utility používá extrapolaci pomocí kvadratické funkce.

- Velikost generované sady plánů: 12
- Velikost parametru R: 6
- Predikce utility: kvadratická

Čtvrtá konfigurace (v tabulce 1 pod názvem Heuristika 4) se snaží optimalizovat utilitu i na úkor většího objemu komunikace. Posíláním větších sad plánů také snižuje počet odeslaných zpráv s letovými plány.

- Velikost generované sady plánů: 24
- Velikost parametru R: 12
- Predikce utility: lineární

Všechny čtyři scénáře rozestaví všechna letadla ve vzdušném prostoru po kružnici s poloměrem 25 km. Při spuštění scénářů směřují trajektorie všech letů do středu kružnice. Každému letadlu je určen jako cíl letu bod na protilehlé straně kružnice tak, že jeho trajektoie letu při spuštění scénáře prochází středem kružnice. Každé letadlo proto detekuje kolizi se všemi ostatními letadly. Konfigurací letadel jsou definovány rozměry jejich bezpečnostních zón. Horizontální poloměr bezpečnostní zóny byl nastaven na 2 km a výška bezpečnostní zóny byla nastavena na 100m.

Ve scénářích jsou použity dva druhy evaluátorů utilit letových plánů (viz 3.3.2). Prvním je *palivový* evaluátor, hodnotící letový plán podle množství spotřebovaného paliva. Druhým použitým evaluátorem je *časový* evaluátor. Časový evaluátor hodnotí letový plán podle délky jeho trvání.

První scénář
- Počet letadel: 10
- Použitý evaluátor: palivový

Druhý scénář
- Počet letadel: 16
- Použitý evaluátor: palivový

Třetí scénář
- Počet letadel: 10
- Použitý evaluátor: časový

Čtvrtý scénář
- Počet letadel: 16
- Použitý evaluátor: časový

6.2. Výsledky

Výsledná utilita letových plánů jednotlivých letadel je nastavena na hodnotu utility posledního aplikovaného letového plánu. Vzhledem k tomu, že jsou všechny kolize v testovaných scénářích vyřešeny během prvních 40 ms simulace a trvání letů převyšuje 500 s, tato prvotní část letů je při vyhodnocení zanedbána.

Naměřené hodnoty jsou zobrazeny v tabulce 1. Pozn.: U druhého i třetího scénáře jedno z letadel svůj letový plán nezměnilo. S ohledem na způsob měření utilit proto nebylo možné zjistit utilitu jejich letových plánů a hodnoty těchto utilit nejsou ve výsledcích zahrnuty. Nejdůležitější tři hodnoty jsou v tabulce uvedeny v prvních řádcích:

- Počet vygenerovaných letových plánů – celkový počet vygenerovaných letových plánů bez ohledu na to, zda byly při kontrole označeny jako validní, nebo ne. Je důležité počet generovaných plánů pro nalezení řešení minimalizovat. Vyšší hodnoty znamenají neoptimální rozvíjení stavového prostoru generovaných letových plánů.

- Počet odeslaných letových plánů – počet odeslaných letových plánů od solverů, které v rámci řešení dané kolize figurují jako slave. Tato hodnota je hlavním indikátorem objemu komunikace a je vhodné ji minimalizovat. S vyšším procentem generovaných validních plánů (viz tabulka 1) roste počet odeslaných plánů v rámci jedné iterace algoritmu a zvyšuje se objem komunikace. Naproti tomu je ale vyšší pravděpodobnost nalezení řešení kolize v menším počtu iterací algoritmu, což naopak objem komunikace snižuje.
- Průměrná hodnota utility – nižší hodnoty znamenají lepší hodnocení evaluátorem. Hodnotu utility se snaží heuristický přístup minimalizovat a proto někdy může generovat větší množství nevalidních plánů. Pro zabránění uvíznutí při generování nevalidních plánů slouží konstanta R, která lépe zamezje uvíznutím pokud je její hodnota nižší.

Počet vyřešených kolizí není klíčovou hodnotou pro hodnocení optimálnosti heuristiky, ale je podle něj možné sledovat chování algoritmu při řešení hromadné kolize. Počet iterací algoritmu indikuje zejména množství odeslanách zpráv. Důležitější hodnotou, vzhledem k optimalizaci objemu komunikace, je ovšem počet odeslaných letových plánů. Procentuální poměr validních a nevalidních plánů znázorňuje poslední řádek jednotlivých částí. V těchto hodnotách se projevuje schopnost algoritmu měnit velikost kroku a rychleji opustit prostor generovaných plánů, které nejsou validní. Vyšší hodnoty poměru validních plánů znamenají vyšší pravděpodobnost nalezení řešení při menším počtu vygenerovaných plánů. Pokud ale algoritmus generuje plány, z nichž nelze řešení sestavit, objem komunikace díky tomu výrazně stoupá.

Heuristika 1 generuje letové plány s téměř dvakrát větším poměrem validity, než původní varianta algoritmu. Řešení nalezla ve dvou případech při velmi podobném množství generovaných plánů a ve dvou případech v polovičním množství. Při zvýšeném počtu odeslaných plánů mírném a zhoršení utility není tato heuristika zcela ideální.

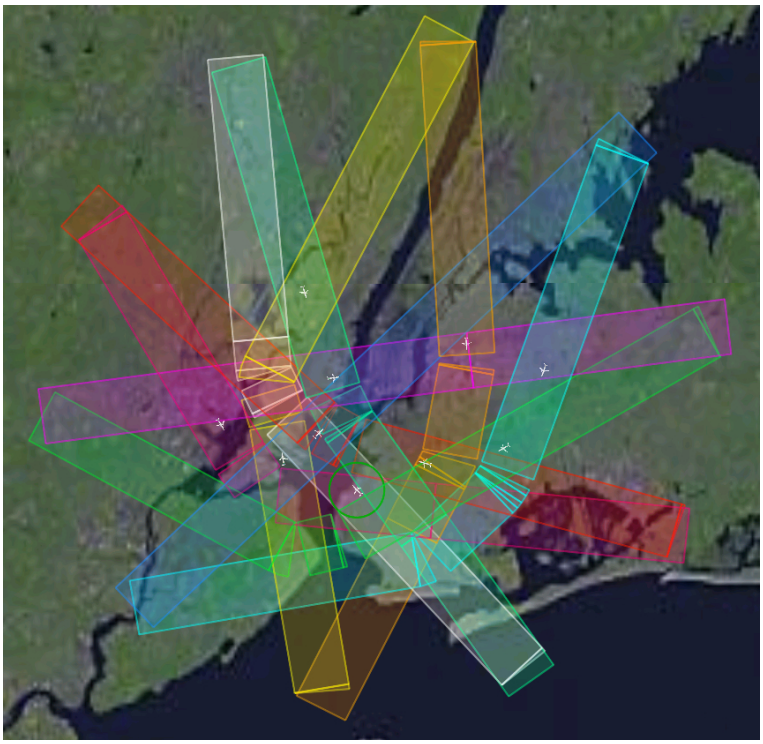
Heuristika 2 (viz obrázek 7) a heuristika 3 vykazují ve třech testovaných scénářích identické hodnoty. Mírně rozdílné je jejich chování ve čtvrtém scénáři, kde má heuristika 3 lepší výsledky všech měřených hodnot. Pouze v některých situacích se tedy projeví predikce utility pomocí extrapolace kvadratickou funkcí. Vzhledem k tomu, že odhad pomocí kvadratické křivky je výpočetně náročnější, je vhodné tento odhad použít spíše pro nasazení na výkonnější hardware. Výsledné hodnoty jednotlivých parametrů jsou u obou heuristik velice vyvážené a jedinou výrazně horší hodnotou oproti základní variantě algoritmu je počet odeslaných letových plánů. Heuristika 2 i 3 se jeví jako velmi dobré a při další optimalizaci pro jednotlivá nasazení algoritmu by bylo nejspíš vhodné vycházet z jejich konfigurací.

Počet iterací je u 4. heuristiky podle očekávání výrazně nižší než u původní varianty. Oproti ostatním metodám pro generování letových plánů ovšem tato heuristika není optimální z hlediska objemu komunikace. Ve srovnání s původní variantou vzrostl objem komunikace na trojnásobek. Tato heuristika má dobré výsledky hodnot utility. Hlavním důvodem je větší velikost generované sady plánů. Díky její velikosti mohl algoritmus více využít zvyšování kroku po generování nevalidních plánů. Proto vzrostl procentuální poměr validních generovaných plánů oproti základní variantě na trojnásobek. I přes používanou zvyšovanou délku kroku však heuristika 4 udržela výslednou hodnotu utility na velmi dobrých hodnotách. Oproti základní variantě algoritmu zlepšuje tato heuristika celkovou hodnotu utility u obou použitých evaluátorů.

#Letadel	Palivový evaluátor	Původní alg.	Heuristika 1		Heuristika 2		Heuristika 3		Heuristika 4		
10	Vygenerovaných plánů	1064	492	46%	504	47%	504	47%	576	54%	Scénář 1
	Odeslaných plánů	136	147	108%	143	105%	143	105%	233	171%	
	Prům. hodhota utility	63,4512	66,7492	105%	63,7415	100%	63,7415	100%	64,5483	102%	
	Počet iterací	76	38	50%	21	28%	21	28%	12	16%	
	Počet vyřešených kolizí	27	18	67%	14	52%	14	52%	10	37%	
	Procent validních plánů	30,36	55,69	183%	54,17	178%	54,17	178%	80,90	267%	
16	Vygenerovaných plánů	1680	1472	88%	1200	71%	1200	71%	1200	71%	Scénář 2
	Odeslaných plánů	155	306	197%	263	170%	263	170%	413	266%	
	Prům. hodhota utility	69,0143	76,6071	111%	74,8479	108%	74,8479	108%	65,6350	95%	
	Počet iterací	120	115	96%	50	42%	50	42%	25	21%	
	Počet vyřešených kolizí	31	46	148%	28	90%	28	90%	23	74%	
	Procent validních plánů	17,92	40,15	224%	46,50	260%	46,50	260%	71,83	401%	
	Časový evaluátor	**Původní alg.**	**Heuristika 1**		**Heuristika 2**		**Heuristika 3**		**Heuristika 4**		
10	Vygenerovaných plánů	490	534	109%	528	108%	528	108%	1200	245%	Scénář 3
	Odeslaných plánů	85	147	173%	159	187%	159	187%	464	546%	
	Prům. hodhota utility	514370	577770	112%	513194	100%	513194	100%	525698	102%	
	Počet iterací	35	42	120%	22	63%	22	63%	25	71%	
	Počet vyřešených kolizí	17	15	88%	15	88%	15	88%	23	135%	
	Procent validních plánů	35,31	51,87	147%	61,55	174%	61,55	174%	75,67	214%	
16	Vygenerovaných plánů	2170	1224	56%	1320	61%	1260	58%	1824	84%	Scénář 4
	Odeslaných plánů	235	244	104%	312	133%	289	123%	731	311%	
	Prům. hodhota utility	573896	572864	100%	524383	91%	522590	91%	512682	89%	
	Počet iterací	155	96	62%	55	35%	52,5	34%	38	25%	
	Počet vyřešených kolizí	42	36	86%	37	88%	35	83%	35	83%	
	Procent validních plánů	20,18	41,01	203%	48,11	238%	47,94	237%	72,70	360%	
	Průměrné hodnoty	**Původní alg.**	**Heuristika 1**		**Heuristika 2**		**Heuristika 3**		**Heuristika 4**		
	Vygenerovaných plánů	1351	930,5	69%	888	66%	873	65%	1200	89%	
	Odeslaných plánů	152,75	211	138%	219,25	144%	213,5	140%	460,25	301%	
	Hodhota utility (palivo)	66,2328	71,6781	108%	69,2947	105%	69,2947	105%	65,0917	98%	
	Hodnota utility (čas)	544133	575317	106%	518788	95%	517892	95%	519190	95%	
	Počet iterací	96,5	72,75	75%	37	38%	36,375	38%	25	26%	
	Počet vyřešených kolizí	29,25	28,75	98%	23,5	80%	23	79%	22,75	78%	
	Procent validních plánů	25,94	47,18	182%	52,58	203%	52,54	203%	75,28	290%	

Tab. 1. Srovnání výsledků testování. Hodnoty jsou v každém řádku srovnány barevně. Nejhorší hodnoty jsou červené, nejlepší hodnoty jsou zelené. Procentuální hodnoty značí změnu hodnoty oproti původnímu algoritmu.

Nejnadějnější navržené rozšíření, vhodné pro další rozvoj, je nejspíše Heuristika 2 (viz obrázek 7). Při přijatelném zvýšení objemu komunikace jsou její výsledky vyvážené a lepší, než v výsledky základního algoritmu. Objem komunikace se nepodařilo v daných scénářích snížit. Navržené algoritmy tak nejsou vhodné pro optimalizaci objemu komunikace.



Obr. 7. Příklad vyřešené kolize deseti letadel pomocí heuristiky 2. Barevně jsou znázorněny trajektorie výsledných bezkolizních letových plánů.

7. Závěr

Cílem této práce byla implementace deterministické varianty algoritmu IPPCA a jejího rozšíření. Nejprve bylo představeno, jak algoritmus IPPCA řeší kolize s ostatními bezpilotními letadly. Poté byla navržena a implementována jeho deterministická varianta. Práce se věnovala zejména detekční, správní a řešící části algoritmu. Návrh těchto částí vycházel z existující nedeterministické varianty algoritmu. Hlavní změnou bylo použití nového druhu komunikace letadel, pro jejíž funkčnost byly navrženy postupy pro korektní navázání a ukončení komunikace bezpilotních letadel.

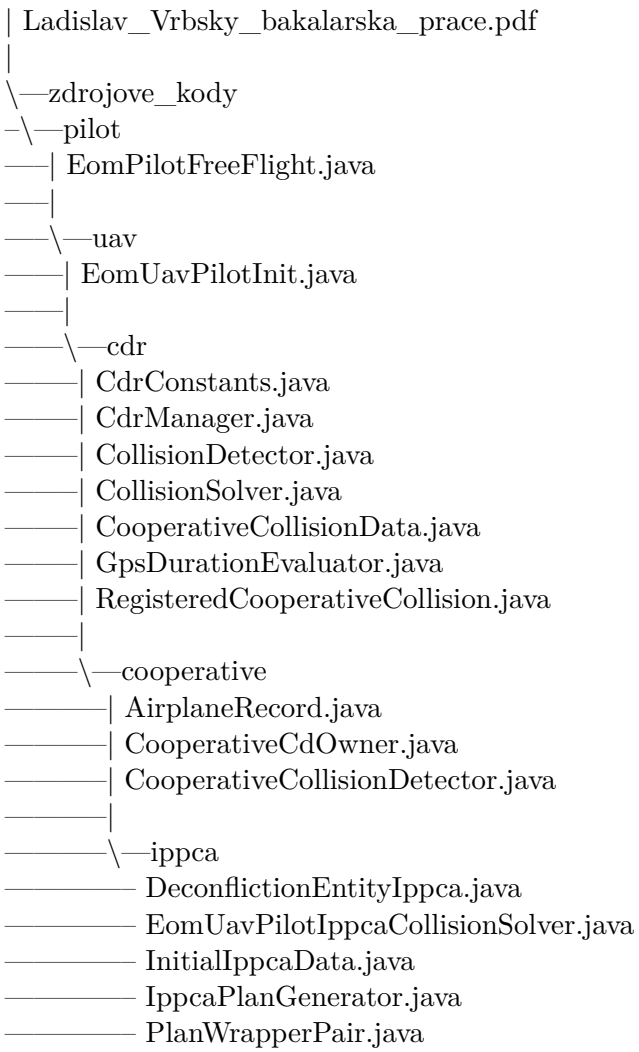
Algoritmus byl dále rozšířen o heuristický generátor letových plánů. Čtyři konfigurace rozšířené varianty algoritmu byly porovnány ve čtyřech scénářích a výsledky těchto scénářů byly vyhodnoceny. Heuristické generování letových plánů přineslo pozitivní výsledky zejména v počtu vygenerovaných letových plánů. Tento počet se podařilo omezit přibližně na dvě třetiny oproti původní variantě. Objem komunikace (počet odeslaných letových plánů) ovšem vzrostl. Hodnotu utilit letových plánů se podařilo udržet na podobných hodnotách, zejména díky použití větší velikosti sady generovaných letových plánů v každé iteraci algoritmu.

Hlavní náplní této bakalářské práce byla implementace algoritmu a jejího rozšíření. Zadání úkolu vyžadovalo zejména v počátku hlubší studium pro získání orientace v této problematice. Bylo nutné porozumnět návrhu projektu AgentFly a jeho komplexní struktuře. Celkový návrh a implementace jednotlivých komponent si vyžádal řádově stovky hodin. Pro celkovou determiničnost simulovaného letu byly implementovány deterministické verze jednak detekční a jednak řešící části algoritmu.

Celkově byl návrh deterministické verze algoritmu i jeho implementace úspěšný. Navržené rozšíření algoritmu optimalizuje počet vygenerovaných letových plánů pro nalezení řešení kolize. Algoritmus je pro nasazení v simulovaném provozu dostačující a dále může být vylepšen o sofistikovanější přístup pro generování validních letových plánů.

Příloha A.

Obsah CD



Bibliografie

[1] Boeing Co. *Current Market Outlook 2013-2032*. 2013.

[2] Agent Technology Center. 22.5. 2014. URL: `www.fel.cvut.cz/vv/tymy/atg.html`.

[3] Přemysl Volf. "Multiagent Simulation of Air Space and Air Traffic Management". Dis. Czech Technical University in Prague, 2013.

[4] Michal Pěchouček et al. *AGENTFLY: Autonomous Agents in Air-Traffic Control January 2008 Final Report*. Tech. zpr. Agent Technology Group, Gerstner Laboratory, Czech Technical University in Prague, 2008.

[5] Michal Pěchouček a David Šišlák. "Agent-Based Approach to Free-Flight Planning, Control, and Simulation". In: (2009).

[6] Manish Bhojasia. 22.5. 2014. URL: `http://www.sanfoundry.com/java-program-gaussian-elimination-algorithm/`.