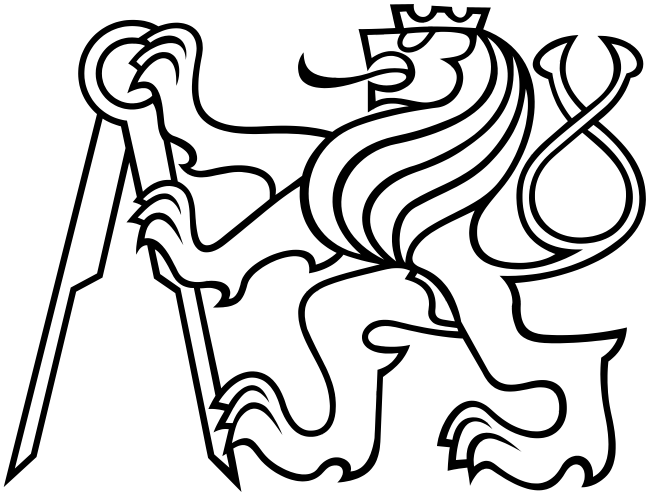
České Vysoké Učení Technické v Praze
Fakulta Elektrotechnická

BAKALÁŘSKÁ PRÁCE



Ondřej Hrstka

Inteligentní hlídkování v námořním provozu

Katedra kybernetiky
Vedoucí bakalářské práce: **Ing. Ondřej Vaněk**

Praha, 2010

Prohlášení

Prohlašuji, že jsem svou bakalářskou práci vypracoval samostatně a použil jsem pouze podklady (literaturu, projekty, SW atd.) uvedené v přiloženém seznamu.

V Praze dne 27. 5. 2010 .....

.....
podpis

České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Ondřej H r s t k a

Studijní program: Elektrotechnika a informatika (bakalářský), strukturovaný

Obor: Kybernetika a měření

Název tématu: Inteligentní hlídkování v námořním provozu

Pokyny pro vypracování:

1. Seznamte se s principy modelování rizika pomocí pravděpodobnostních rozložení. Seznamte se s problémem maximálního pokrytí (maximum coverage problem) a způsoby jeho řešení.
2. Seznamte se se současným stavem pirátství v Adenském zálivu a vytvořte mapu rizika kombinací vybraných pravděpodobnostních rozložení.
3. Implementujte a vyhodnoťte sadu algoritmů pro pokrytí oblasti N hlídkami o rozdílných poloměrech působnosti.
4. Vytvořte agentní, decentralizovaný koordinační algoritmus pro pokrytí prostoru N hlídkami z bodu (4).
5. Porovnejte kvalitu a výpočetní náročnost decentralizovaného (5) a centralizovaných (4) řešení.
6. Integrujte řešení problému do platformy AgentC.

Seznam odborné literatury:

[1] Russell, S. And Norvig, P.: Artificial Intelligence: A Modern Approach, Englewood Cliffs, New Jersey, 1995.
[2] Cohen, R., Katzir, L.: The Generalized Maximum Coverage Problem, Information Processing Letters, Volume 108, Issue 1, Pages 15-22, 2008.
[3] Tsvetkova B.: Securitizing Piracy Off the Coast of Somalia, Central European Journal of International and Security Studies, 2008.
[4] Vybrané články from Kuala Lumpur Conference on Piracy and Crimes on Sea, 2009.

Vedoucí bakalářské práce: Ing. Ondřej Vaněk

Platnost zadání: do konce zimního semestru 2010/2011

prof. Ing. Vladimír Mařík, DrSc.
vedoucí katedry

doc. Ing. Boris Šimák, CSc.
děkan

V Praze dne 12. 1. 2010

Abstrakt

Tato práce reaguje na zvýšenou pirátskou aktivitu v oblasti Adenského zálivu a zabývá se problémem optimálního rozmístění a patrolování protipirátských hlídek v námořním provozu. V první fázi je vytvořena mapa rizika, která modeluje riziko přepadení transportní lodě piráty. Ve druhé fázi je tato mapa použita nejprve k optimálnímu rozmístění hlídek, poté k naplánovaní trasy hlídky minimalizující riziko přepadení. Navržené algoritmy jsou integrovány s multi-agentní platformou AgentC, která dovoluje testovat jejich efektivitu. Výsledné algoritmy jsou testovány vzhledem k různým vlastnostem a různým počtům hlídek a vzhledem ke změnám vlastností prostředí. Ukazuje se, že je možné riziko snížit, ale za reálných podmínek není možné toto riziko úplně eliminovat.

Abstract

This thesis responses to increased acts of piracy in the area of Gulf of Aden. It deals with problem of optimal deployment and routing of anti-pirates patrols in maritime traffic. In the first phase there is created a risk map modeling risk of pirates raid on transport ship. This map is then used in the second phase for optimal patrol deployment and for planning patrol route in order to minimize risk of pirates incident. Designed algorithms are integrated into multi-agent platform AgentC, which allows measuring of their efficiency. Algorithms are tested with different number of patrols and with various features. As a result there is shown that risk can be lowered, but under real conditions it cannot be eliminated entirely.

Děkuji Ing. Ondřejovi Vaňkovi za výborné vedení práce, Ing. Michalovi Jakobovi, Ph.D. za cenné rady a Ing. Janu Faiglovi za poskytnutí LaTeX šablony pro tuto práci.

Obsah

Seznam tabulek

Seznam obrázků

Úvod

Pirátství je námořní loupežná činnost se kterou se lidstvo potýká již po několik tisíciletí. Tento nebezpečný jev je v dnešní době na vzestupu [8]. Oblastí s významným podílem pirátství je pobřeží Somálska, zvláště pak Adenský záliv.

Protože Somálsko nedisponuje námořními vojenskými silami a jeho vláda je velmi slabá, musí země mezinárodního společenství potírat pirátství na širém moři. Naneštěstí je oblast Adenského zálivu a západního Indického oceánu velmi rozlehlá, a tak mezinárodní vojenské síly stále nezvládají zabránit všem útokům na dopravní lodě.

Zvýšit šance mezinárodního úsilí by mohla implementace algoritmů umělé inteligence. Cílem této práce je prozkoumat a implementovat algoritmy pro optimální rozmisťování a hlídkování proti-pirátských hlídek v Adenském zálivu.

Program, zvláště pak jeho část zabývající se hlídkováním, je implementován jako součást platformy *AgentC* vyvíjenou v Agent Technology Center na Fakultě elektrotechnické ČVUT.

V kapitole 1 jsou uvedeny základní reálie a pojmy problematiky. Následují 3 kapitoly, kde v každé je řešena část problému. V kapitole 2 je řešena problematika modelování rizika pomocí mapy rizika v oblasti zadané oblasti. V kapitole 3 jsou zpracovány algoritmy na optimální rozmístění nepohybujících se hlídek na risk mapě. V kapitole 4 je pak řešena otázka, jak co nejefektivněji hlídkovat v určité oblasti za použití risk mapy.

Kapitola 1

Přehled problematiky

1.1 Definice pojmů

V této kapitole je uvedeno několik pojmů, které jsou nadále používány v této práci:

Transportní loď je civilní loď, která je ohrožena piráty. Jedná se tedy převážně o nákladní, dopravní a rybářské lodě.

Hlídka je vojenská loď, jejíž cílem je potlačení pirátských aktivit a ochrana transportních lodí.

Pirát organizovaná skupina, která pomocí jednoho nebo několika rychlých člunů přepadává a bere do zajetí transportní lodě.

1.2 Mezinárodní proti-pirátské úsilí

Tato práce se zabývá hlídkováním v Adenském zálivu. Adenský záliv se rozkládá jižně od Rudého moře a na západ od Arabského moře. Touto oblastí propluje denně 50 lodí, což činí jednu desetinu celkové světové přepravy [12]. Téměř 12 % světové produkce ropy je přepravováno zálivem [5]. Úspěšný pirátský útok působí škodu komerčním subjektů, které přepravují náklad. Tyto škody se mohou výrazně projevit i ve světové ekonomice [12].

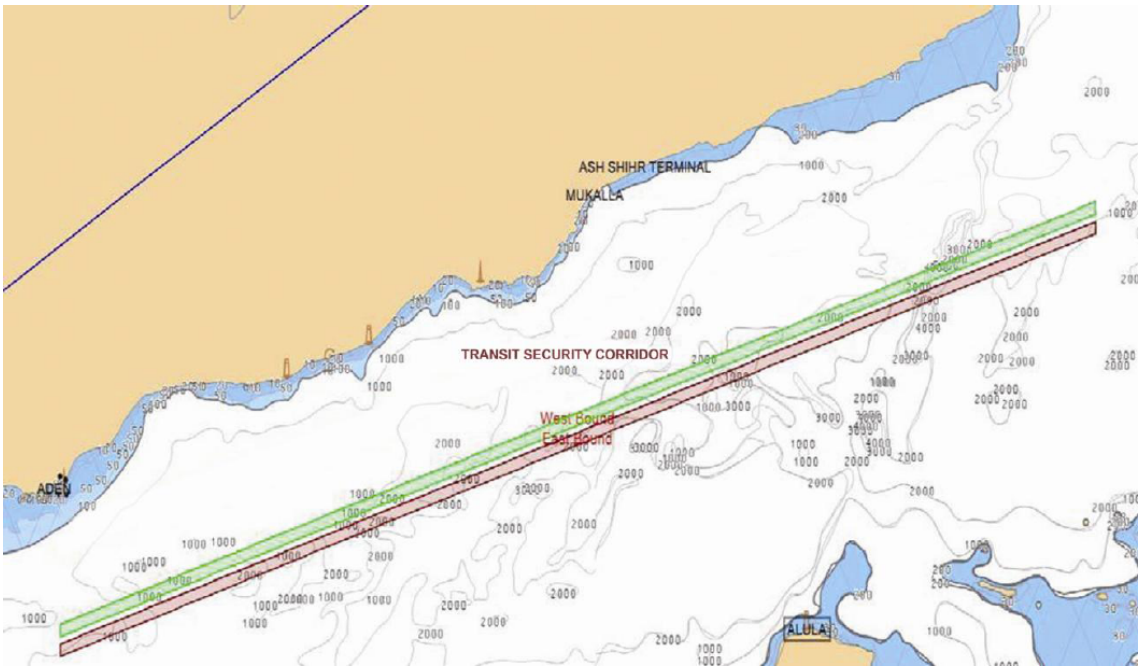
Jako reakci na zvýšenou pirátskou aktivitu různé země a mezinárodní organizace poslali své vojenské síly hlídkovat do Adenského zálivu a přilehlého Indického oceánu. Zároveň byla zřízena centra, která se zaměřují na monitorování ilegálních aktivit v dané oblasti, koordinují civilní provoz a vydávají doporučení, jak se při plavbě chovat. Mezi tato centra patří také Maritime Security Centre (Horn of Africa) a (UK) Maritime Trade Operation.

Maritime Security Centre (Horn of Africa) - MSC(HOA) bylo zřízeno Evropskou Unií v září 2008. Jedná se o koordinační centrum, které poskytuje služby námořníkům v Adenském zálivu a přilehlém Indickém oceánu. Toto centrum také stojí za proti-pirátskou operací ATLANTA (viz kapitola 1.2.2)[9].

(UK) Maritime Trade Operation - UKMTO poskytuje spojení mezi vojenskými silami a námořním průmyslem. Sídlí v Dubaji. Sleduje polohu lodí v Adenském zálivu a tyto informace předává vojenským silám a velení v EU. Zároveň je schopno předávat důležité informace přímo zúčastněným lodím, namísto jejich distribucí přes dopravní společnosti [1].

1.2.1 IRTC koridor

Internationally Recommended Transit Corridor (IRTC) je doporučená trasa pro transportní lodě skrz Adenský záliv. Byl ustanoven MSC(HOA) 1.února 2009 [3]. Tento koridor obsahuje samostatnou trasu pro východ-západní a západo-východní směr.



Obrázek 1.1: IRTC koridor [11]

IRTC je následovníkem *Maritime Security Patrol Area (MSPA)* [13]. Oproti MSPA nezasahuje do Jemenských národních vod, což činilo problémy především pro armádní lodě. Větší vzdálenost od Jemenského pobřeží také stěžuje činnost pirátům, které ho využívali jako opěrný bod [11]. Dále je IRTC také přímá trasa, což usnadňuje vojenským hlídkám snadnější ochranu civilní dopravy.

1.2.2 Přehled vojenských sil

V této sekci je uvedeno několik vojenských uskupení, která se snaží potlačit pirátskou aktivitu v Adenském zálivu.

Combined Task Force 150 - CTF 150 je mezinárodní operační skupina zřízená zřízená k podpoře boje proti terorismu. Od roku 2006 se zapojuje i do bojů proti pirátským aktivitám v okolí Somálska [14].

Combined Task Force 151 - CTF 151 je mezinárodní operační skupina zřízená v lednu 2009 speciálně pro potírání pirátských aktivit ve výše zmíněné oblasti [15].

Operace ATLANTA - ATLANTA je mise zřízená MSC(HOA) 8. prosince 2008 v souladu s rezolucemi 1814, 1816 a 1838 Rady Bezpečnosti OSN. Jejím cílem je ochrana humanitární potravinové pomoci Somálsku, ochrana plavidel plujících skrz Adenský záliv a potlačování pirátství a loupeží u pobřeží Somálska [10].

Operace Ocean Shield - Ocean Shield je mise pod záštitou Severoatlantické Aliance. Spuštěna byla 17. srpna 2009 [7].

Mezi další síly patří například lodě Čínské lidové republiky nebo Ruské federace.

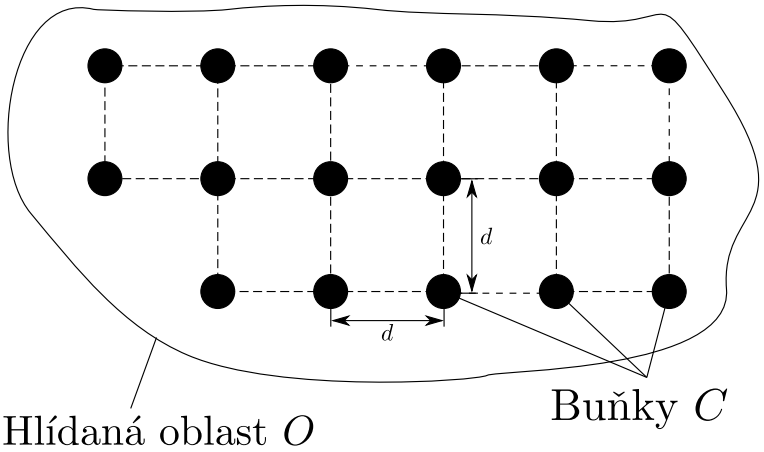
Kapitola 2

Mapa rizika

Mapa rizika je způsob reprezentace dat pro algoritmy zajišťující rozmístění hlídek. Popisuje *míru ohrožení* v hlídané oblasti O.

Hlídaná oblast O je souvislá ohraničená plocha na zemském povrchu. Jedná se o polygon, jehož okrajové body jsou dány zeměpisnými souřadnicemi. Protože je obsah této plochy zanedbatelný vůči obsahu Země, je možné souřadnice místo sférických považovat za kartézské a tím aproximovat eliptický prostor eukleidovským prostorem E_2[1]. Vznikne tak souřadný systém S, kde souřadnice x je pak rovna zeměpisné délce, souřadnice y odpovídá zeměpisné šířce.

Následně je tato oblast diskretizována a vznikne tak množina uspořádaných dvojic $\{[x_1, y_1], [x_2, y_2], [x_3, y_3], ..., [x_n, y_n]\} = C, C \in \mathbb{R}^2$, kde $[x_i, y_i]$ je bod uvnitř hlídané oblasti O. Tyto souřadnice pak tvoří čtvercovou síť, jejíž body jsou od sebe vzdáleny o diskretizační konstantu d (viz obr. 2.1). Tyto body jsou označovány jako *buňky*.



Obrázek 2.1: Diskretizace oblasti O

[1]Vzhledem k poloze Adenského zálivu není třeba se zabývat singularitami jako póly, rovník atd.

Mapa rizika je pak množina těchto bodů a množina hodnot těchto bodů, která je určená funkcí $f(x, y) : C \mapsto \mathbb{R}_0^+$. Čím je hodnota funkce $f(x, y)$ vyšší, tím vyšší je riziko útoku pro transportní loď na pozici x, y. Mapa rizika je značena symbolem Θ.

Dále je pro mapu rizika je zavedeno několik pojmů:

- $O(\Theta)$ je oblast, kterou pokrývá Θ. Jedná se o spojitý 2 dimenzionální prostor.
- $C(\Theta)$ je množina buněk c risk mapy Θ.
- $val(c)$ je hodnota rizika buňky c. Jestliže platí $c = [x, y]$, pak také platí $val(c) = f(x, y)$.
- $val(\Theta)$ je hodnota rizika Θ. Je určená vztahem $val(\Theta) = \sum_{c \in P(\Theta)} val(c)$.

Mapa rizika vytvořená v této práci je složena ze dvou jednodušších map rizik. Jedná se o mapu rizika založenou na IRTC koridoru (Θ_{IRTC}) a mapu rizika založenou na znalostech pozice a kapacity pirátských přístavů (Θ_P).

Hlídaná oblast $O(\Theta)$ a tedy i množina $C(\Theta)$ je v celé této práci oblast Adenského zálivu.

2.1 Mapa rizika podle IRTC

Po zavedení IRTC došlo k přesunu výskytu pirátských útoků jižněji v souladu se změnou trasy většiny transportních lodí [13] (viz obr. 2.2). Na základě tohoto jevu je možné předpokládat, že piráti, kteří samozřejmě o IRTC ví, se soustředí právě do této oblasti. Na tomto předpokladu je vytvořena i mapa rizika podle IRTC (Θ_{IRTC}).

2.1.1 Matematický model

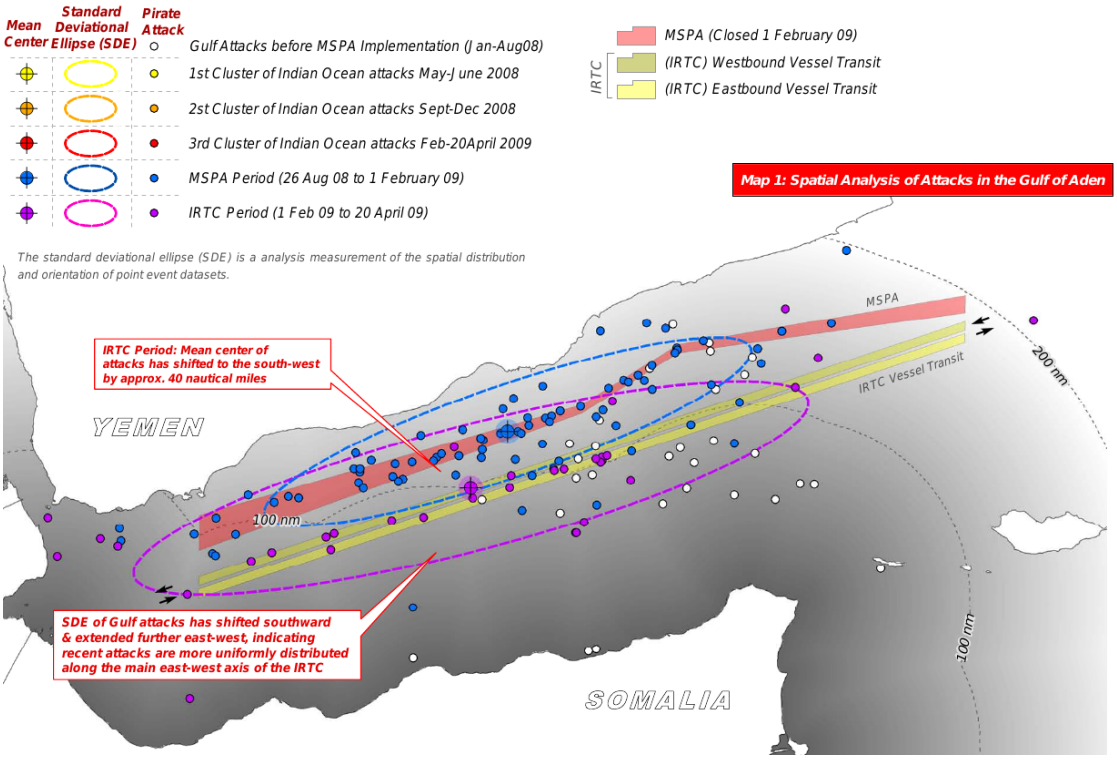
Matematický model Θ_{IRTC} využívá *normálního rozložení*[2]. Hustota pravděpodobnosti normálního rozložení je dána vztahem

$$f_{gauss}(x; \sigma, \mu) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \sigma > 0. \tag{2.1}$$

Parametr μ je v Θ_{IRTC} $\mu = 0$. Toto rozložení je generováno podél každé trasy $\tau \in T$ v koridoru IRTC. Funkce $f(x, y)$ Θ_{IRTC} je pak definována vztahem

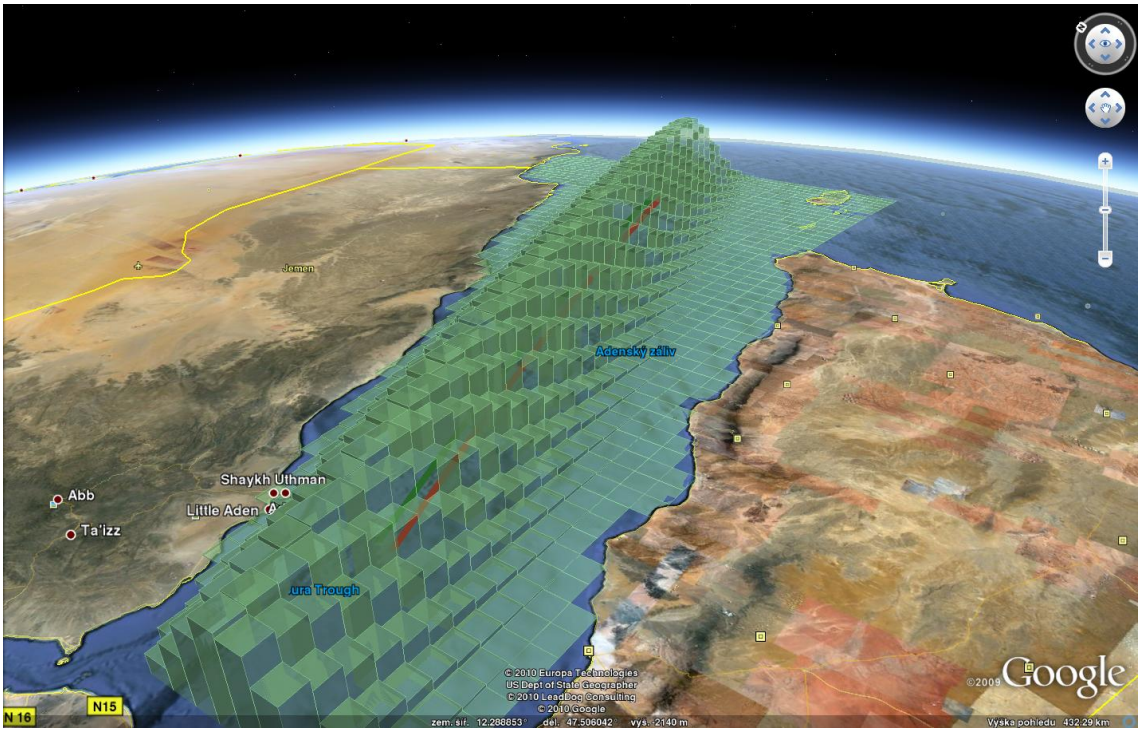
$$f_{IRTC}(x, y) = \sum_{\tau \in T} f_{gauss}(d(x, y, \tau)), \tag{2.2}$$

[2]Přestože se při tvorbě map rizika pracuje s pravděpodobnostními rozloženími samotná mapa rizika nepopisuje pravděpodobnost, ale obecnou míru rizika.



Obrázek 2.2: Změna výskytu útoků po 1. 2. 2009 [11]

kde $d(x, y, \tau)$ je vzdálenost bodu $[x, y]$ od trasy τ. Protože IRTC obsahuje dvě trasy, vznikne Θ_{IRTC} jako součet dvou normálních rozložení podél dvou tras.



Obrázek 2.3: Vizualizace mapy rizika podle IRTC

2.2 Mapa rizika podle přístavů

Mapa rizika podle přístavů (Θ_P) vychází z předpokladu, že pro piráta je výhodnější útočit blízko svého přístavu. Pokud totiž pirát uspěje v útoku na transportní loď, hrozí mu protiútok ze strany hlídky. Osvobození unesené transportní lodi je snazší na volném moři než v pirátském přístavu. Druhý důvod je čistě ekonomický. Čím dále od pobřeží se musí pirát plavit, tím více spotřebuje paliva.

2.2.1 Matematický model

Pro generování risk mapy přístavů je použito dvou pravděpodobnostních rozložení. Jedná se o normální rozložení a *weibullovo rozložení*.

Weibullovo pravděpodobností rozložení

Hustota weibullova pravděpodobnostního rozložení je dána vztahem:

$$f_{weibull}(x;\lambda,k) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k}, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{2.3}$$

V vztahu (2.3) je $k > 0$ tvarový parametr a $\lambda > 0$ představuje měřítko. Právě díky parametru k může weibullovo rozložení nabývat rozličných tvarů. Pro potřeby této práce byl použit parametr $k = 1.5$.

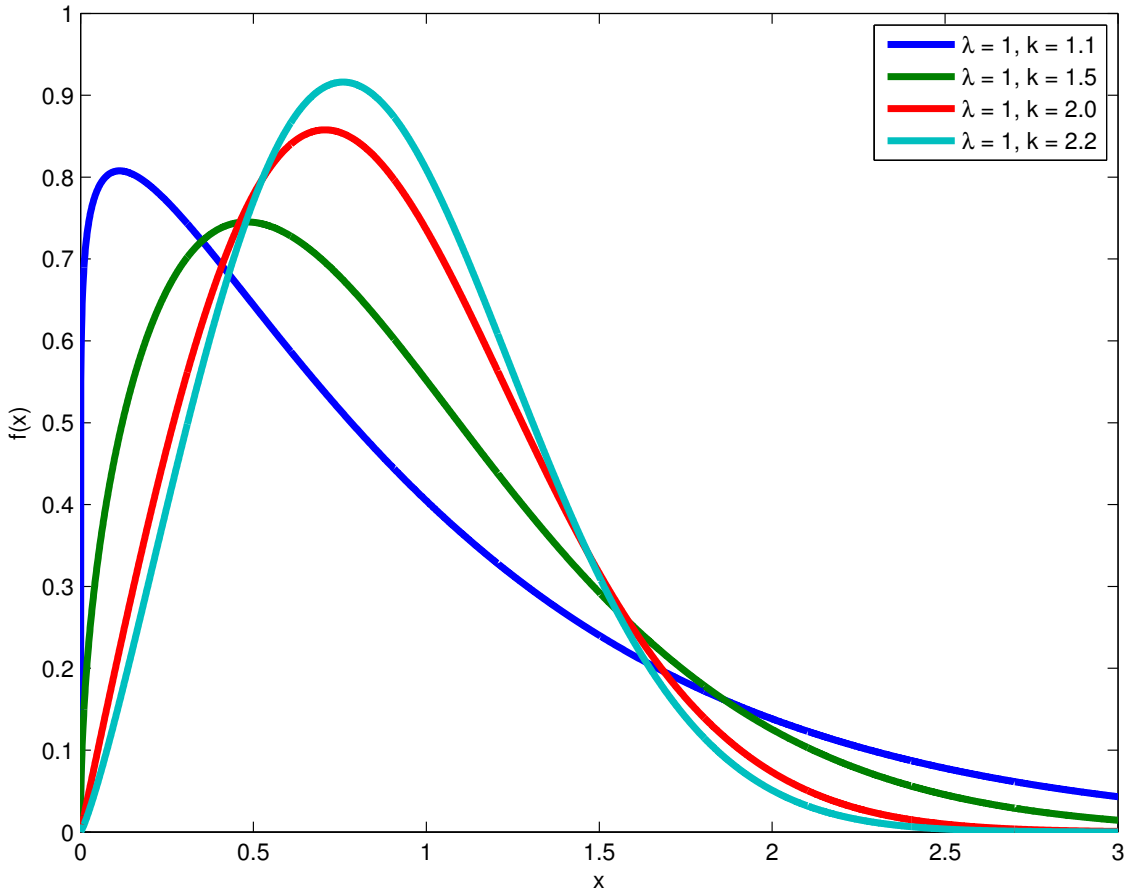
Tvorba mapy rizika podle přístavů

Θ_P je sestavena z menších elementárních map rizik, kde každá reprezentuje právě jeden pirátský přístav. Pirátské přístavy tvoří množinu $H = \{h_1, h_2, h_3, ..., h_n\}$, kde h_i je uspořádaná dvojice $[x, y]$, která udává souřadnice přístavu. V H jsou reprezentovány pouze přístavy, které jsou na jih od $IRTC$ a které jsou v Adenském zálivu.

Pro vytvoření této elementární mapy rizika je nejprve potřeba si zadefinovat nový souřadný systém. Jedná se o kartézský pravotočivý systém S' s počátkem v místě přístavu. Směr osy x' je kolmý na IRTC koridor. Kladná orientace je pak směrem k tomuto koridoru.

Výše popsaný souřadný systém je zaveden pro každý přístav $h \in H$. Souřadnice x', y' vzniknout transformací ze souřadného systému $S \rightarrow S'$. Pro každý tento přístav je pak definována funkce

$$f_{P,el}(x, y, h) = f_{weibull}(x')f_{gauss}(y')c(h), \tag{2.4}$$



Obrázek 2.4: Weibullovo rozložení pro různé hodnoty parametru k

kde $c(h)$ je kapacita přístavu h, která je přímo úměrná počtu pirátů schopných vést z tohoto přístavu útok.

Celková Θ_P je pak dána jako součet jednotlivých vlivů přístavů

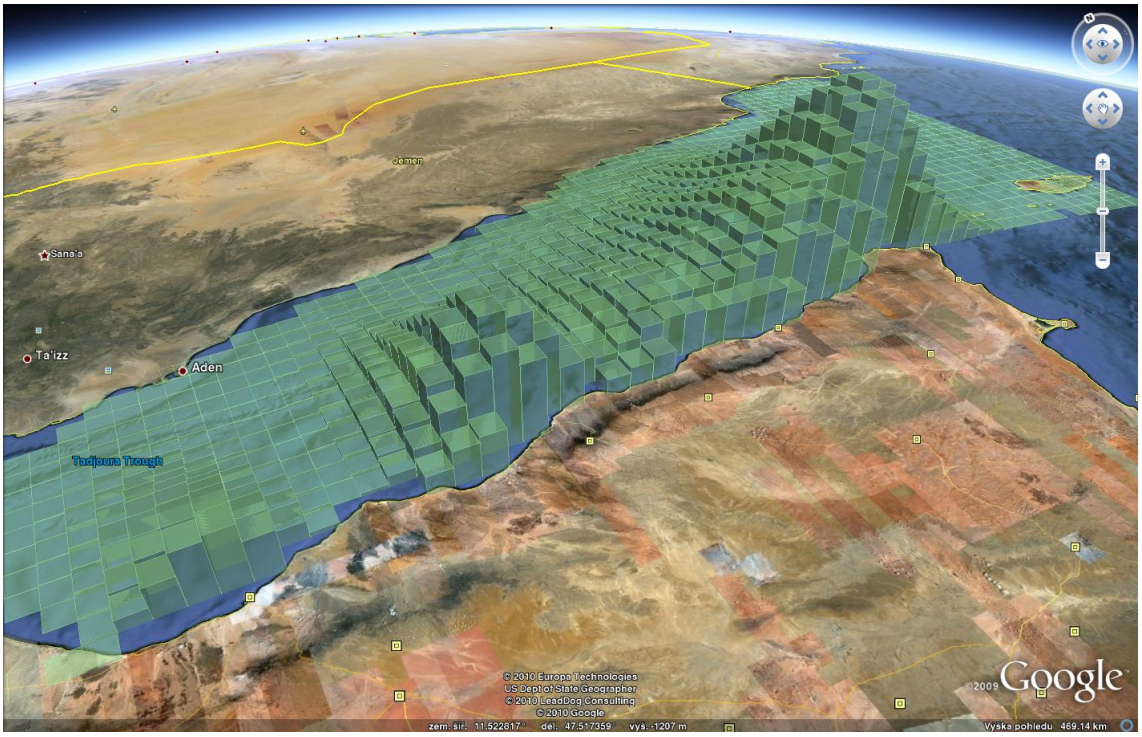
$$f_P(x, y) = \sum_{h \in H} f_{harbor,el}(x, y, h). \tag{2.5}$$

Vizualizace této mapy rizika je na obr. 2.5.

2.3 Složená mapa rizika

Jak již bylo uvedeno výše, výsledná mapa rizika vznikne složení Θ_{IRTC} a Θ_P. Složení těchto map rizik je provedeno pomocí násobení. Toto lze provést na základě následující úvahy.

Jestliže se pirát pohybuje v oblasti, kde se nevyskytuje žádná transportní loď, je neškodný a hlídku nemusí jeho přítomnost vůbec zajímat. Naopak pokud se transportní loď pohybuje v oblasti, kde se nevyskytují piráti, není ohrožena a hlídka se



Obrázek 2.5: Vizualizace mapy rizika podle přístavů

jí tedy taky nemusí zabývat. Hlídku tedy zajímají pouze oblasti, kde se zároveň vyskytují transportní lodě a piráti. Pouze v těchto oblastech je tedy riziko útoku na transportní lodě. Tuto vlastnost dobře splňuje operace násobení.

Výsledná mapa rizika je tedy dána funkcí

$$f_{final}(x, y) = f_{IRTC}(x, y) \cdot f_P(x, y). \tag{2.6}$$

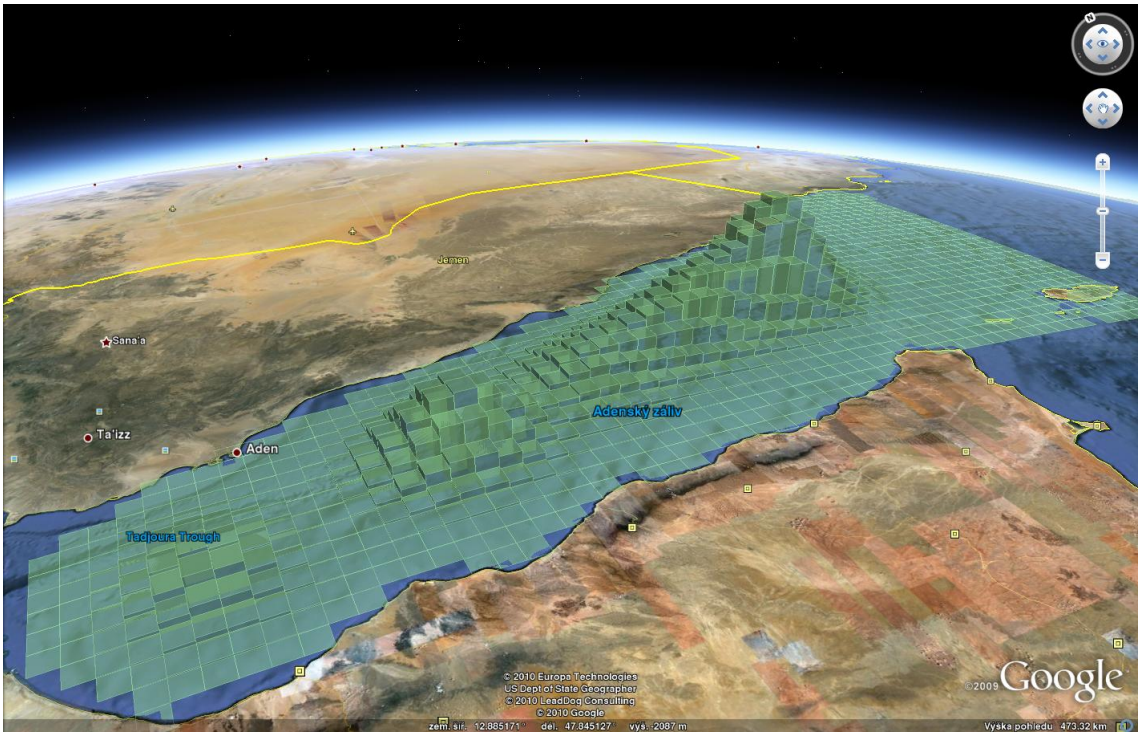
Vizualizace takto vytvořené mapy je na obr. 2.6. Při porovnání této vizualizace s vizualizací útoků od února 2009 na obr. 2.2 je patrné, že tato mapa má vztah s reálnou situací.

2.4 Alternativní mapa rizika

Výše zmíněná mapa rizika je vytvořena na základě apriorní informace. Lze ale také vytvořit mapu rizika na základě aposteriorní informace, v tomto případě na základě znalosti o pirátských útocích z období od 1. 1. 2009 do 1. 6. 2009.

Tvorba alternativní mapy rizika (Θ_A) je inspirována *kernel density estimation (KDE)* [16]. Přestože KDE pracuje s pravděpodobností, lze tento postup použít i pro mapu rizika, která je v korelaci s pravděpodobností útoku.

Pirátské útoky jsou reprezentovány množinou $A = \{a_1, a_2, a_3, ..., a_n\}$, kde a_i je uspořádaná dvojice $[x, y]$ udávající polohu útoku. Stejně jako Θ_P je i tato mapa



Obrázek 2.6: Vizualizace složené mapy rizika

rizika složena z menších elementárních map rizik. Každý element je reprezentován vztahem

$$f_{test,el}(p, a) = f_{gauss}(d(x, y, a)), \tag{2.7}$$

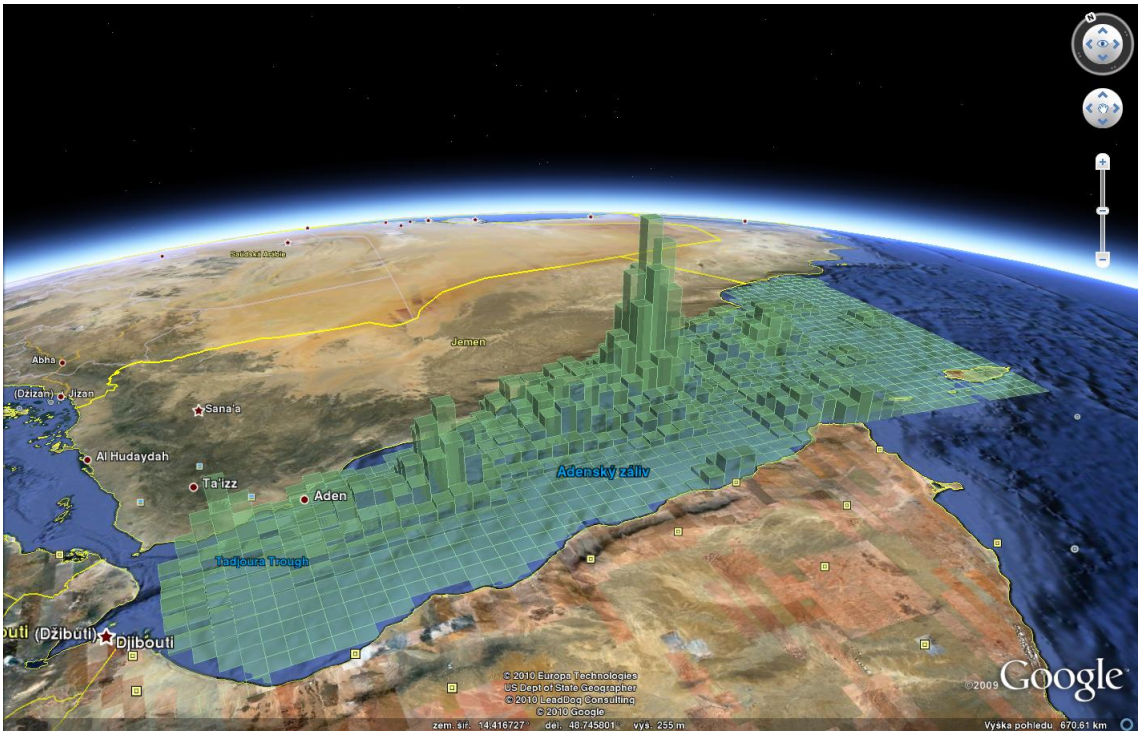
kde $d(x, y, a)$ je vzdálenost polohy útoku a od bodu $[x, y]$.

Výsledná Θ_A pak vznikne jako součet jednotlivých elementů:

$$f_{test}(p) = \sum_{a \in A} f_{test,el}(p, a) \tag{2.8}$$

Vizualizace Θ_A je na obr. 2.7.

Tato mapa rizika je komplikovanější a proto je použita na testování algoritmů popsaných v následujících kapitolách.



Obrázek 2.7: Vizualizace alternativní mapy rizika

Kapitola 3

Rozmisťování hlídek

Cílem této kapitoly je nalézt a prozkoumat algoritmy, které co nejlépe rozmístí hlídky na dané mapě rizika.

3.1 Požadavky na algoritmus

Vstupy pro algoritmus jsou mapa rizika Θa seznam hlídek.

Seznam hlídek je seznam $H = [h_1, h_2, h_3, ..., h_n]$. Na začátku běhu algoritmu má každá hlídka zadán *poloměr účinnosti* $r(h)$.

Výstupem algoritmu je seznam pozic $K = [k_1, k_2, k_3, ..., k_n]$, kde pozice k_i je uspořádána dvojici $[x, y]$ a udává pozici hlídky h_i. Pro zjednodušení může být hlídka umístěna pouze do středu některé buňky $c \in C(\Theta)$.

Cílem algoritmu je tedy každé hlídce $h \in H$ přiřadit pozici, při které dosáhnou hlídky nejlepšího pokrytí mapy rizika Θ. Určení pokrytí mapy rizika hlídkou je uvedeno na konci kapitoly 3.2.1.

3.2 Vliv hlídky na mapu rizika

3.2.1 Případ jedné hlídky

Lze předpokládat, že pirát se vyhýbá setkání s hlídkou. Samotná hlídka je pak schopna zasáhnout proti pirátům, zneškodnit jejich útok. Z tohoto lze vyvodit, že tam kde se vyskytuje hlídka bude nižší riziko útoku pirátů.

Přítomnost hlídky tedy zmenšuje hodnotu jednotlivých buněk $c \in C(\Theta)$. Hlídka tak vytváří novou mapu rizika Θ_n z původní mapy rizika Θ_o.

Pro určení, jak hlídka ovlivní hodnotu v buňce c je zadefinována *funkce vlivu*. Jedná se o funkci $v(h, k, c)$, kde h je daná hlídka, k je její pozice a c je příslušná

buňka. Obor hodnot této funkce je $< 0; 1 >$. Jestliže funkce nabývá hodnoty 1, hlídka v buňce c úplně eliminuje riziko výskytu a útoků pirátů, jestliže nabývá hodnoty 0, pak nemá v této buňce žádný vliv.

V této práci je pro funkci $v(h, k, c)$ použito normální rozložení, které je normováno tak, aby maximální hodnota byla 1. Funkce má pak tvar

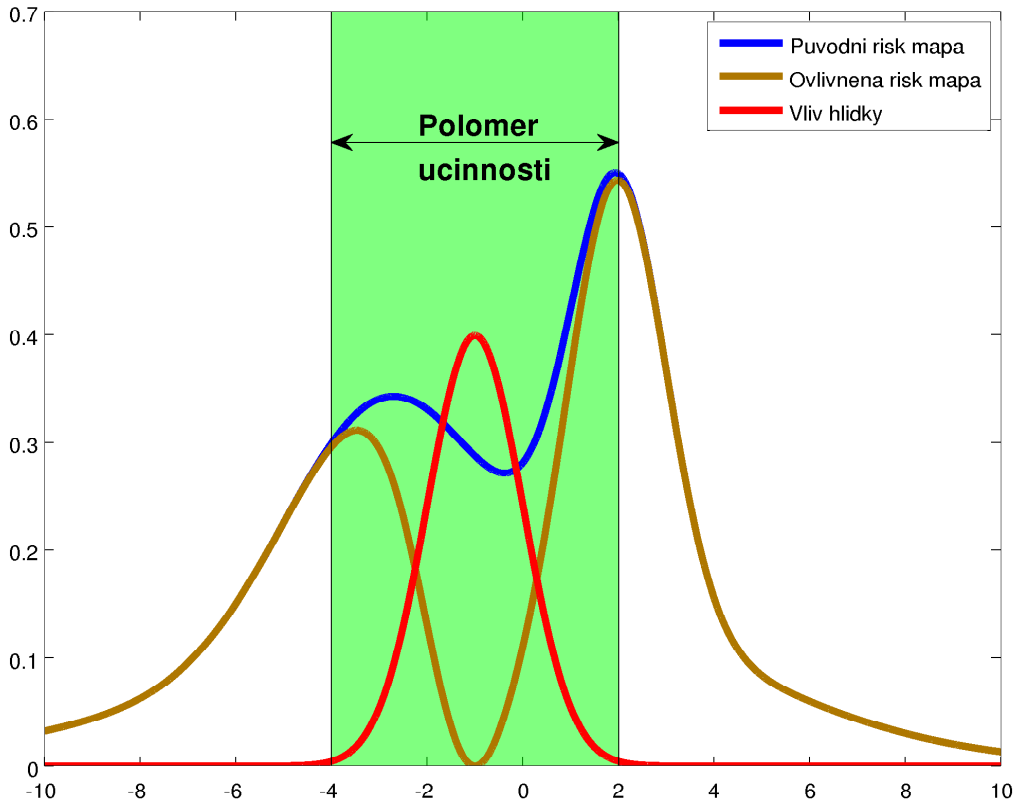
$$v(h, k, c) = f_{gauss}(dist(k, pos(c))) \cdot norm, \tag{3.1}$$

kde $dist(k, pos(c))$ je vzdálenost buňky c od pozice k a $norm$ je normalizační konstanta. Konstanta μ je určena poloměrem účinnosti $\mu = r(h)$.

Nová hodnota buňky c označená jako $val(c_n)$ je pak určena vztahem

$$val(c_n) = val(c_o) \cdot (1 - f(h, k, c_o)). \tag{3.2}$$

Na obr. 3.1 je zobrazen vliv hlídky na risk mapu. Pro zvýšení přehlednosti je příklad převeden do jednorozměrné spojité varianty.



Obrázek 3.1: Vliv umístění hlídky na risk mapu

3.2.2 Případ více hlídek

Umístění více hlídek lze aplikovat pomocí výše zmíněného umístění jedné hlídky. Stačí pouze upravit rovnici (3.2) na:

$$val(c_n) = \max\{0, val(c_o)(1 - \sum_{h \in H}(v(h, k, c_o)))\} \tag{3.3}$$

3.2.3 Hodnotící funkce

Hodnotící funkce umožňuje porovnávat různá řešení různých algoritmů pro stejné vstupy. Jedná se o zobrazení $e : \Xi \mapsto \mathbb{R}$, kde Ξ je množina všech řešení K. Čím je hodnota funkce $e(K)$ nižší, tím lepší je algoritmus.

Vztah pro hodnotící funkci je rozdíl hodnoty původní mapy rizika $val(\Theta_o)$ a hodnoty mapy rizika po rozmístění hlídek $val(\Theta_n)$:

$$e(K) = val(\Theta_o) - val(\Theta_n) \tag{3.4}$$

Čím je tedy hodnota $e(K)$ vyšší, tím více je eliminováno riziko a tím lepší je tedy rozmístění hlídek K.

3.3 Návrh algoritmů

Ze zadání úlohy vyplývá, že algoritmus vybírá z konečné množiny řešení. Bohužel počet všech řešení odpovídá variaci bez opakování $\frac{n!}{(n-k)!}$, kde n je počet buněk risk mapy a k je počet hlídek, které rozmisťujeme. Pokud zjednodušíme zadání tak, že hlídky mají stejný poloměr účinnosti, pak jsou zaměnitelné a počet všech řešení odpovídá kombinaci bez opakování $\binom{n}{k}$. V obou případech je ale náročnost algoritmu pomocí hrubé síly *faktoriálová* a proto nepřichází toto řešení v úvahu.

V této práci jsou zkoumány celkem 3 algoritmy na rozmisťování hlídek:

- genetický algoritmus
- greedy algoritmus
- greedy + local search algoritmus

3.3.1 Genetický algoritmus

Genetický algoritmus patří do skupiny evolučních algoritmů a je založen na základě pozorování evoluce v přírodě. Algoritmus pracuje s *populací*, což je množina

jedinců. Jeden jedinec je reprezentován určitým seznamem vlastností (*genů*), které tvoří jeho *genom*. Genom pak představuje jedno možné řešení úlohy. Genom v této práci je seznam pozic hlídek. Na počátku běhu algoritmu jsou jedinci vytvořeni náhodně. Následně se opakuje sekvence kroků:

1. selekce
2. křížení
3. mutace

Selekce

Selekce je výběr jedinců, kteří postoupí do dalšího kroku zpracování algoritmu. Vychází z principu přirozeného výběru. Jako kritérium zde slouží hodnotící funkce e. Předpokládá se totiž, že geny s lepší hodnotící funkcí mají větší šanci vytvořit řešení. Zároveň ale dává i šanci (třebaže menší) méně kvalitním genům, protože i v nich se může nacházet část řešení.

Selekce použita v této práci je popsána pseudokódem v algoritmu 1. Jedinci vybraní tímto postupem dále postupují ke křížení.

Křížení

Křížení je prohození částí několika jedinců. Dá se interpretovat tak, že původní jedinci (rodiče) křížením svých genů vytvoří jedince nové (potomky). V této práci je použito jednoduché křížení ilustrované na příkladě. Jsou dány rodiče $R_1 = a, b, c, d, R_2 = e, f, g, h$. Potomci pak vzniknou tak, že 1. potomek je tvořen 1. polovinou genomu 1. rodiče a 2. polovinou genomu 2. rodiče. Druhý potomek je pak tvořen 2. polovinou genomu 1. rodiče a 1. polovinou genomu 2. rodiče. Křížením rodičů R_1, R_2 jsou tedy potomci $P_1 = a, b, g, h, P_2 = e, f, c, d$.

V této práci je křížení aplikováno na všechny jedince, tak aby každý prošel křížením právě jednou. Z algoritmu 1 je patrné, že silný jedinec (mající dobrou hodnotící funkci) může být ve výstupu selekce vícekrát. Tedy je možné, aby byl křížen i sám se sebou a výsledkem takového křížení je pak identická dvojice jedinců. Tímto postupem se dobří jedinci zachovají v nezměněné podobě a nedochází tak ke ztrátě dobrých řešení, ale naopak je zde šance, že se v dalším kroku vytvoří další, jiný, dobrý jedinec a křížením těchto dvou vznikne ještě lepší řešení.

Mutace

Mutace je náhodná změna některého z genů. Nastává pouze u některých jedinců. Tímto postupem se zamezuje uváznutí algoritmu v lokálním minimu.

Algoritmus 1: Selekce

Vstup: Populace popIn
Výstup: Nová populace popOut vybraná selekcí

```
// Setřídění populace vzestupně podle hodnotící funkce
popIn ← sortByFitness(popIn);
sum ← 0;
foreach member in popIn do                    // Součet hodnotících funkcí
    sum ← sum+fitness(member);
end
foreach member in popIn do
    probability ← sum/fitness(member);
    // Nastav member zadanou pravděpodobnost probability
    setProbability(member, probability) ;
end
while popOut is not full do
    rand ← random(0,1) ;                  // Náhodné číslo z int. < 0; 1).
    foreach member in popIn do
        if rand > getProbability(mem) then
            add(popOut, member) ;              // Přidej member do popOut
        end
    end
end
return popOut
```

V této práci má mutace dva druhy, které nastávají se stejnou pravděpodobností. V prvním případě se hodnota genu změní na jinou úplně náhodnou. Tedy jedna z hlídek je přesunuta na úplně jinou pozici. V druhém případě se pak hlídka přesune na jednu z 8 sousedních pozic.

Četnost mutace je jedním z parametrů genetického algoritmu. V kapitole 5.1 je mimo jiné vyhodnocen vliv tohoto parametru na kvalitu výstupu.

Výměna části populace

V genetickém algoritmu je možné některé z jeho částí odebrat nebo přidat. V této práci je před selekci navíc přidána funkce, která nahradí nejhorších 10% populace novou náhodně vygenerovanou populací.

Ukončení běhu algoritmu

Ukončení běhu algoritmu nastane po předem daném počtu iterací. Při běhu je pak uchováváno nejlepší dosažené řešení, které je nakonec výstupem.

Shrnutí

Genetický algoritmus umožňuje nalézt řešení, aniž by měl k dispozici podrobnější informaci o problému. Stačí mu hodnotící funkce a správná reprezentace genomu. Toto je jeho výhodou ale i nevýhodou. Pokud jsou totiž k dispozici informace o problému, které je možno využít, lze nalézt algoritmus, který daný problém řeší lépe.

3.3.2 Greedy algoritmus

Greedy algoritmus se snaží najít řešení pomocí postupného umisťování hlídek na nejvýhodnější pozice. Algoritmus tedy běží n kroků, kde n je počet hlídek v seznamu hlídek H.

Před během algoritmu jsou hlídky seřazeny sestupně podle jejich poloměru účinnosti $r(h)$. Poté je tento seznam procházen postupně v krocích. V každém kroku i vybere pozici k_i takovou, že hodnotící funkce $e(K)$ dosahuje maxima. Seznam K má v kroku i právě i prvků, obsahuje tedy pouze pozice hlídek umístěných v předchozích krocích a pozici hlídky umisťované v současném kroku. Celý tento postup je zapsán v pseudokódu v algoritmu 2.

Výhodou greedy algoritmu je, že v tomto případě běží velmi krátkou dobu oproti genetickému algoritmu. Greedy musí vyzkoušet $n \cdot k$ kombinací, kde n je počet buněk mapy rizika Θa k je počet hlídek h.

Algoritmus 2: Greedy

Vstup: Seznam hlídek patrolsList, mapa rizika riskMap
Výstup: Seznam pozic positionList

```
// Sestupné setřídění podle r(h)
patrolsList ← sortByRadius(patrolsList)
foreach patrol in patrolsList do
    position ← findBestPosition(patrol, riskMap);
    // Úprava mapy rizika podle vlivu hlídky
    riskMap ← place(riskMap, patrol, position);
    add(positionList, position) ;          // Přidej position do positionList
end
return positionList;
```

Jeho nevýhodou je snadné uváznutí v lokálním maximu. Tuto nevýhodu se snaží potlačit následující algoritmus.

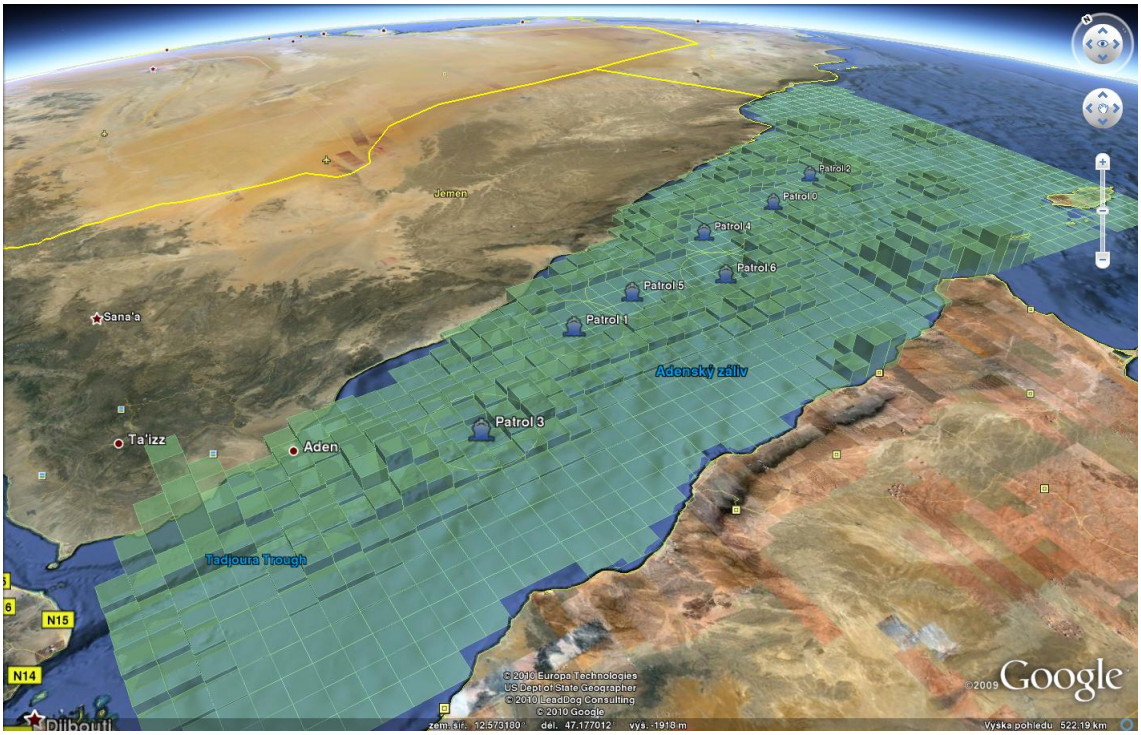
3.3.3 Greedy + local search algoritmus

Jak již název napovídá, tento algoritmus vychází z greedy algoritmu. Po provedení greedy algoritmu je velmi malá šance, že by bylo nalezeno optimální řešení. Proto je výsledek zpracován local search algoritmem, který iterativně prohledává okolí pozice každé hlídky a snaží se v tomto okolí najít výhodnější pozici. Algoritmus je reprezentován níže uvedeným pseudokódem.

Algoritmus 3: Greedy + local search

Vstup: Seznam hlídek patrolsList, mapa rizika riskMap
Výstup: Seznam pozic positionList

```
positionList ← greedy(patrolsList);
repeat
    improve ← false;
    foreach patrol in patrolsList do
        // Pro každou sousední buňku
        foreach positionNeighbor in getNeighborPositionList(patrol) do
            riskMapTemp ← place(riskMap, patrol, positionNeighbor) if
            fitness(riskMapTemp) > fitness(riskMap) then
                riskMap ← riskMapTemp;
                replace(positionList, position, positionNeighbor) ;
                improve ← true ;
            end
        end
    end
until improve == true ;
```



Obrázek 3.2: Ukázka rozmístění 10 hlídek pomocí algoritmu Greedy + Local search

Kapitola 4

Hlídkování

Tato kapitola se zabývá návrhem algoritmů, které budou plánovat *trasu* τ pro hlídku, tak aby co nejlépe pokryla danou risk mapu.

Trasa τ je seznam pozic $[[x_1, y_1], [x_2, y_2], [x_3, y_3], ..., [x_n, y_n]] = \tau$. Tyto pozice tvoří navigační body pro hlídku. Hlídka se plaví postupně směrem k jednotlivým bodům $[x_i, y_i]$. Jestliže k danému bodu dopluje, plaví se k dalšímu bodu $[x_{i+1}, y_{i+1}]$.

Až hlídka dopluje na poslední bod v trase τ, zavolá opět plánovací algoritmus, který vygeneruje trasu novou.

4.1 Požadavky na algoritmus

Vstupy algoritmu je mapa rizika Θ a hlídka h. Hlídce je zadán poloměr účinnosti $r(h)$ a její výchozí pozice $pos(h)$.

Výstupem algoritmu je trasa τ.

4.2 Vliv pohybující se hlídky na mapu rizika

Jestliže hlídka proplouvá nějakou oblastí, dá se předpokládat, že pirát se této oblasti vyhne. Protože pirát nezná strategii hlídky a neví tedy, jestli se v oblasti nechce pohybovat i nadále, může trvat nějaký čas, než se do této oblasti opět odváží. Na těchto předpokladech funguje vyhodnocování pohybu hlídky vzhledem k mapě rizika.

Aby byla simulace pohybu hlídky a jejího vlivu na mapu rizika realizovatelná, je třeba časový průběh simulace *diskretizovat*. V této práci je časový skok nastaven na interval 5 min. Tedy každých 5 minut se vyhodnocuje vliv hlídky na risk mapu.

V každém vyhodnocení mapy proběhnou 2 události:

1. Obnoví se mapa rizika.
2. Vyhodnotí se vliv hlídky na mapu rizika.

Obnovení mapy rizika je navyšování hodnoty buněk $C(\Theta)$ na jejich původní hodnotu. Rychlost tohoto navyšování je dána *časovou konstantou* T a jeho průběh je lineární. Tato konstanta udává čas, za který se buňka dostane z nulové hodnoty na její původní. Jestliže je tedy časová konstanta například 30 min, pak buňka c, která má v čase $t = 0\,\text{min}$ hodnotu $val(c) = 0$ získá svojí původní hodnotu za 30 minut simulačního času. Jestliže by buňka měla polovinu původní hodnoty, obnoví se plně za 15 minut atd.

Vyhodnocení vlivu hlídky probíhá naprosto stejně jako v kapitole 3.2. Mapa rizika Θ_o je tedy nahrazena novou mapou rizika Θ_n.

4.3 Návrh algoritmů

V této práci jsou popsány 2 algoritmy řešící výše zadanou úlohu:

- Greedy algoritmus
- Zig-Zag algoritmus

4.3.1 Greedy algoritmus

Úloha hlídkování po mapě rizika se dá převést na problém *Continuous Area Sweeping*, který je popsán v [2]. V tomto článku je také popsána myšlenka použít greedy algoritmus.

V každé fázi plánování algoritmus vytvoří 72 přímých tras radiálně rovnoměrně směrem od robota. Tyto trasy mají všechny délku d. Každá tato trasa je vyhodnocena z hlediska potenciálního zisku. Výsledná trasa je ta s největším možným ziskem.

Vyhodnocování potenciálního zisku

Při vyhodnocování potenciálního zisku je možno použít několik metod.

Jednoduché vyhodnocení - při tomto vyhodnocení se provede simulace plavby hlídky po dané trase s tím, že mapa není obnovována.

Vyhodnocení s discount faktorem - při použití této metody se postupuje stejně jako při použití jednoduchého vyhodnocení, s tím rozdílem, že zisk z daných kroků se snižuje podle gama funkce [4]. Tato funkce je daná vztahem

$$f(k) = \gamma^k, \tag{4.1}$$

kde k je krok plánování a $\gamma \in <0;1)$ je discount faktor.

Gama funkce zohledňuje fakt, že informace o stavu risk mapy v čase zastarává a že by tedy měl algoritmus brát v potaz hlavně bližší okolí hlídky.

Vyhodnocení s temporal faktorem - toto vyhodnocení je stejné jako vyhodnocení s discount faktorem, nicméně zde se počítá s tím, že mapa rizika je obnovována. Při plánování tedy probíhá simulace, jako by hlídka plánovanou trasou skutečně proplouvala, ale zisk v každém kroku je opět upravován podle vztahu (4.1).

4.3.2 Zig-Zag algoritmus

Zig-Zag algoritmus v dané oblasti (nejlépe obdélníkové) trasu, která „cik-cak“ prochází celý obdélník. Je velmi vhodný pro naplánování trasy, která pokryje celou plochu.

V tomto případě by nicméně takováto trasa byla moc dlouhá a zbytečně by vedla hlídku do oblastí, kde je riziko napadení malé. Proto je třeba tuto trasu nějak zkrátit.

Inspirací pro tento algoritmus byla implementace algoritmu zig-zag v [6]. V této práci bylo třeba navrhnout plánování trasu pro bezpilotní letoun tak, aby v co nejkratším čase zmonitoroval celou zadanou oblast. Algoritmus navrhl trasu a následně z ní odebral jednoznačně redundantní části.

V případě hlídkování na moři byla pak využita myšlenka postupného zkracování předem naplánované trasy.

Pro zjednodušení je v tomto postupu hlídaná oblast tvaru obdélníku. Nejprve jsou v této oblasti vytvořeny *řádky*. Tyto řádky jsou úseky trasy rovnoběžné s kratší stranou obdélníku. Řádky jsou od sebe vzdáleny o dvojnásobek poloměru účinnosti zadané hlídky.

Dále je každý okrajový bod řádku ohodnocen. V tomto případě je ohodnocení potenciální zisk hlídky umístěné v tomto bodě. Bod s nejmenší hodnotou je následně odebrán. Přidružený řádek je tedy zkrácen. Délka zkrácení odpovídá vzdálenosti, kterou hlídka urazí za dobu jednoho časového skoku.

Výše popsaný postup je iterativně opakován po konstantní počet kroků. Poté jsou zbývající řádky propojeny a je z nich vytvořena trasa. Algoritmus 4 popisuje tento postup pomocí pseudokódu a na obr. 4.1 je zobrazena vizualizace pro 3 kroky.

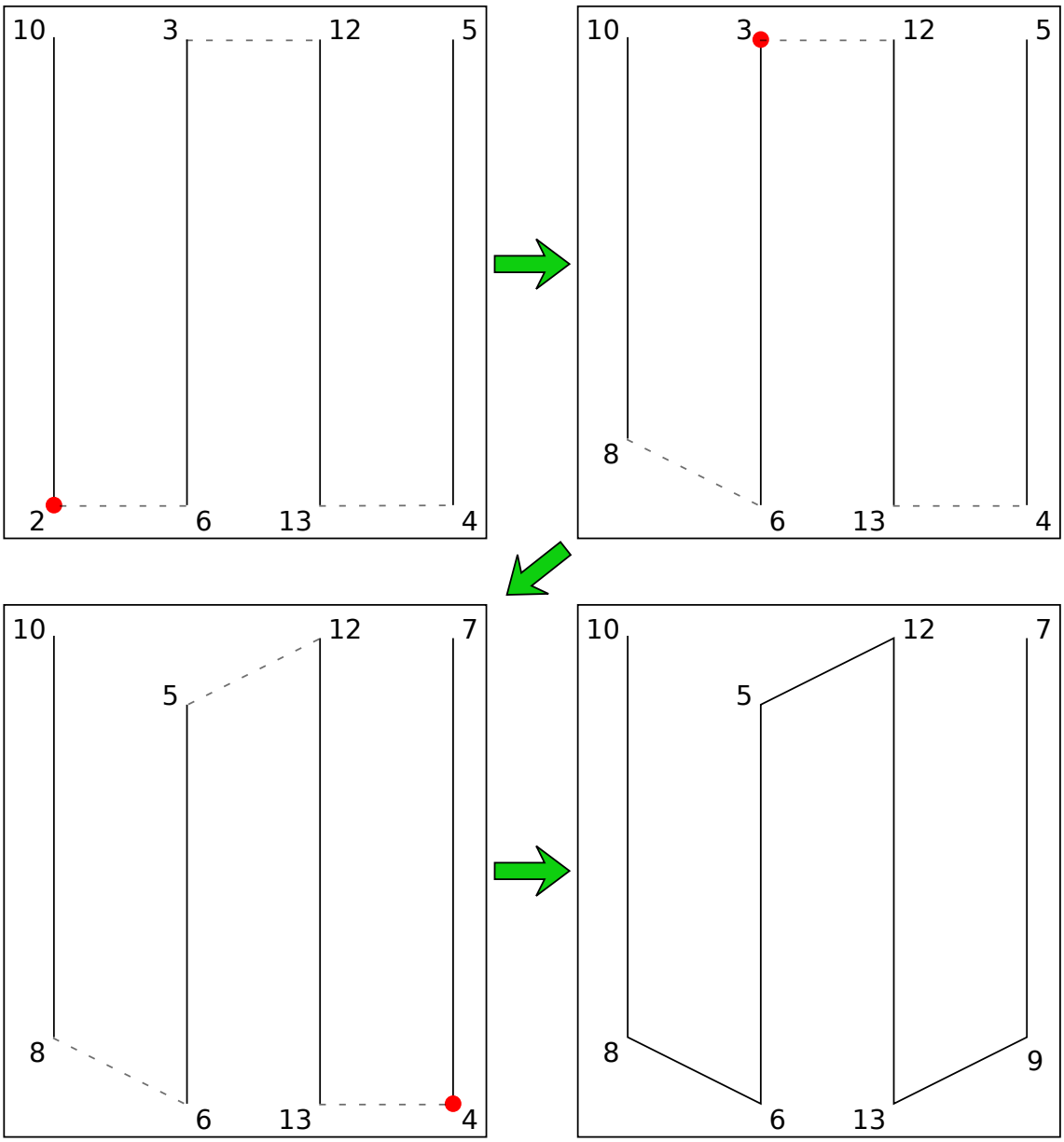
Algoritmus 4: Zig-Zag

Vstup: Hlídka patrol, mapa rizika riskMap
Výstup: Trasa path

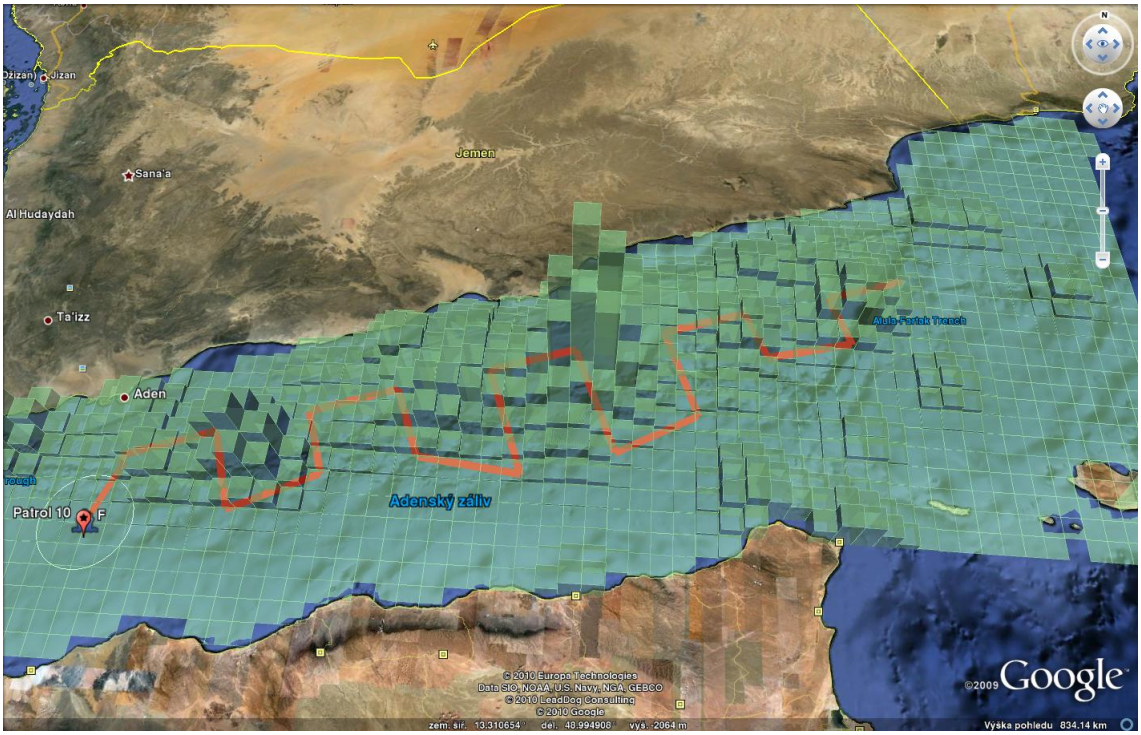
```
rows ← makeRows(riskMap) ;                          // Vytvoření řádků
for i = 0 to R do                          // Konstanta R udává počet iterací
    // Oříznutí okraje řádku s nejnižším potenciálním ziskem
    cutRowEdge(rows) ;
end
path ← makePathFromRows(rows);
return path
```

Počet iterací (konstanta R) je v této práci určena experimentálně. Byla spouštěna simulace pro různě velké hodnoty konstanty R. V simulaci hlídka vždy jednou projela trasu vytvořenou pomocí zig-zag algoritmu. Byla zaznamenána doba trvání simulace t_{sim} a zisk hlídky rev_{sim}. Výsledná konstanta R byla pak vybrána pro nejvyššího hodnotu rev_{sim}/t_{sim}.

Na rozdíl od zig-zag algoritmu uvedeném v [6] není v tomto případě třeba spojit počáteční a koncový bod trasy, aby vznikla smyčka. Jestliže hlídka dopluje na konec trasy, stačí prostě naplánovat novou trasu v opačném směru. Protože se risk mapa obnovuje postupně, vezme tyto hodnoty v potaz i algoritmu zig-zag.



Obrázek 4.1: Tvorba trasy pomocí algoritmu zig-zag. Čísla udávají hodnoty potenciálního zisku v okrajových bodech.



Obrázek 4.2: Ukázka naplánované trasy pomocí algoritmu Zig-Zag

Kapitola 5

Zhodnocení algoritmů

V této kapitole jsou zhodnoceny algoritmy z kapitol 3 a 4.

Všechny algoritmy byly testovány na mapě rizika popsané v kapitole 2.4.

5.1 Algoritmy pro rozmisťování hlídek

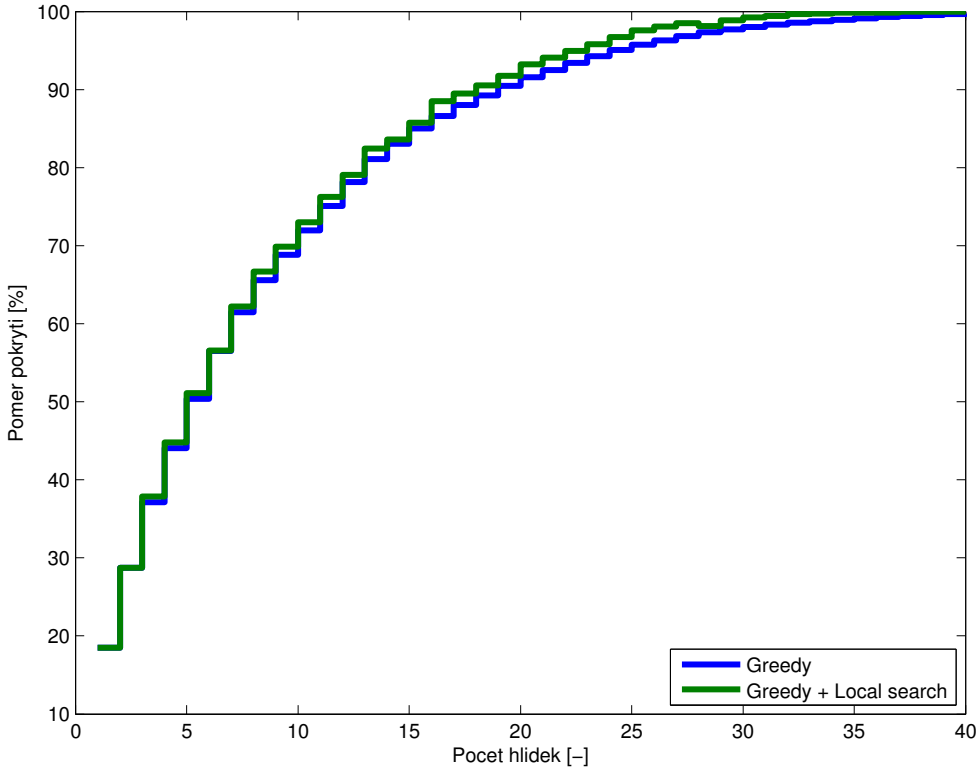
5.1.1 Způsob testování

Výsledky těchto algoritmů závisí na dvou parametrech: poloměr účinnosti hlídek a počet hlídek. Pro tyto parametry byly tedy testovány. Nejprve pro konstantní počet hlídek $n = 15$ a pro různé poloměry účinnosti. Poté pro konstantní poloměr účinnosti $r(h) = 135\,\text{km}$ a různé počty hlídek.

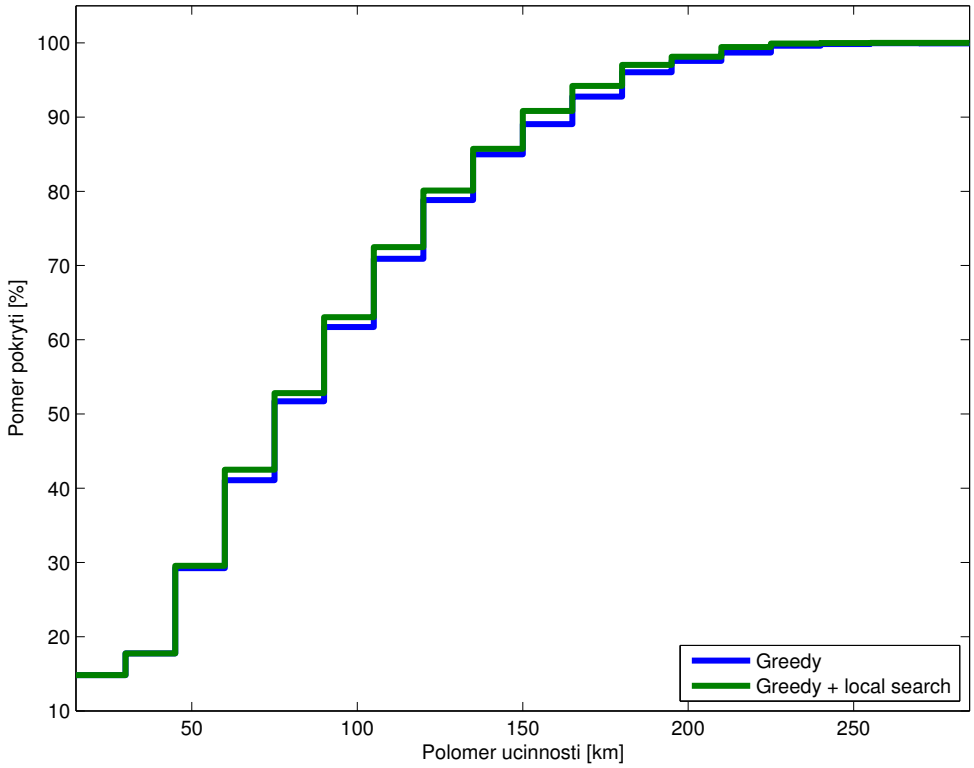
Výjimku tvoří genetický algoritmus. Tento algoritmus je časově velmi náročný a nebylo tedy možné ho otestovat pro tak velký počet případů. Nicméně, z vlastností genetického algoritmu je patrné, že poloměr účinnosti hlídek nebude mít na jeho výsledky ve srovnání s ostatními algoritmy výrazný vliv. Byl proto otestován pro několik počtů hlídek a porovnán s ostatními algoritmy se srovnatelnými vstupy.

Výstupem testů je *poměr pokrytí*. Jedná se o hodnotu hodnotící funkce vydělenou hodnotou původní mapy rizika Θ_o. Jestliže poměr pokrytí nabývá hodnoty 100%, algoritmus rozmístěním hlídek úplně eliminoval riziko.

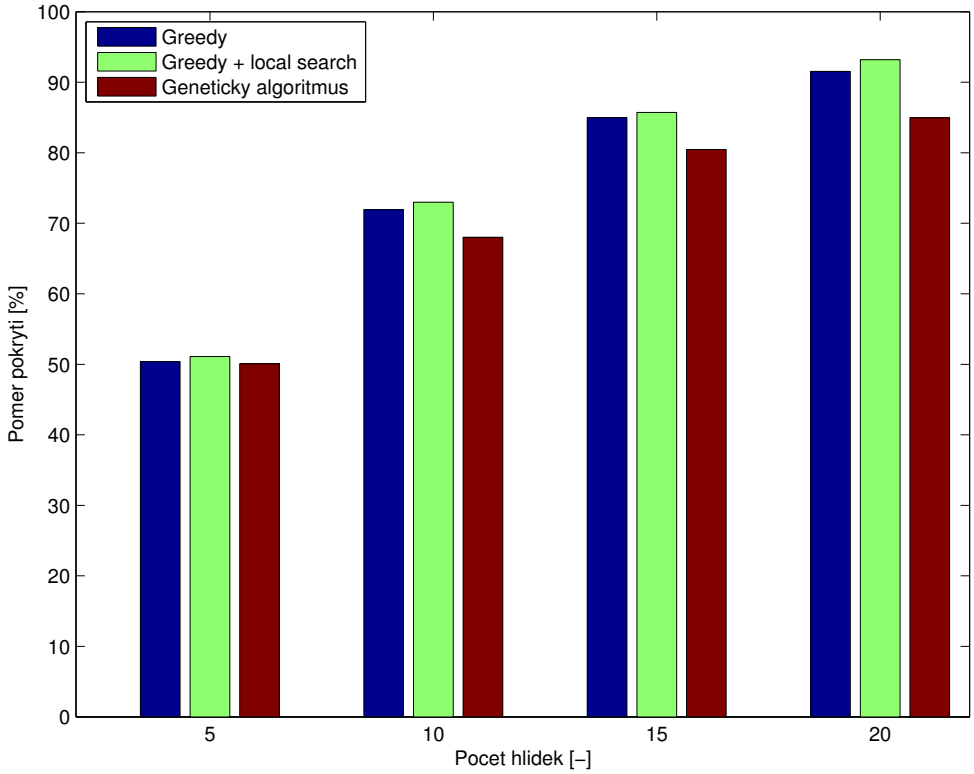
U genetického algoritmu byl dále otestován vliv četnosti mutace na kvalitu výsledku. Protože je genetický algoritmus stochastický, jeho opakovatelnost závisí především na použití generátoru náhodných čísel. V tomto testu byly spuštěny 3 běhy algoritmu pokaždé s jiným semínkem pro generátor náhodných čísel. Tímto postupem je eliminován vliv generátoru náhodných čísel na výsledek.



Obrázek 5.1: Porovnání algoritmů při změně počtu hlídek



Obrázek 5.2: Porovnání algoritmů při změně poloměru účinnosti



Obrázek 5.3: Srovnání genetického algoritmu

5.1.2 Výsledky testování

Z výsledků na obr. 5.3 vyplývá, že genetický algoritmus dosahuje horších výsledků než greedy a greedy + local search. Rozdíl mezi výsledky se zvyšuje s počtem hlídek.

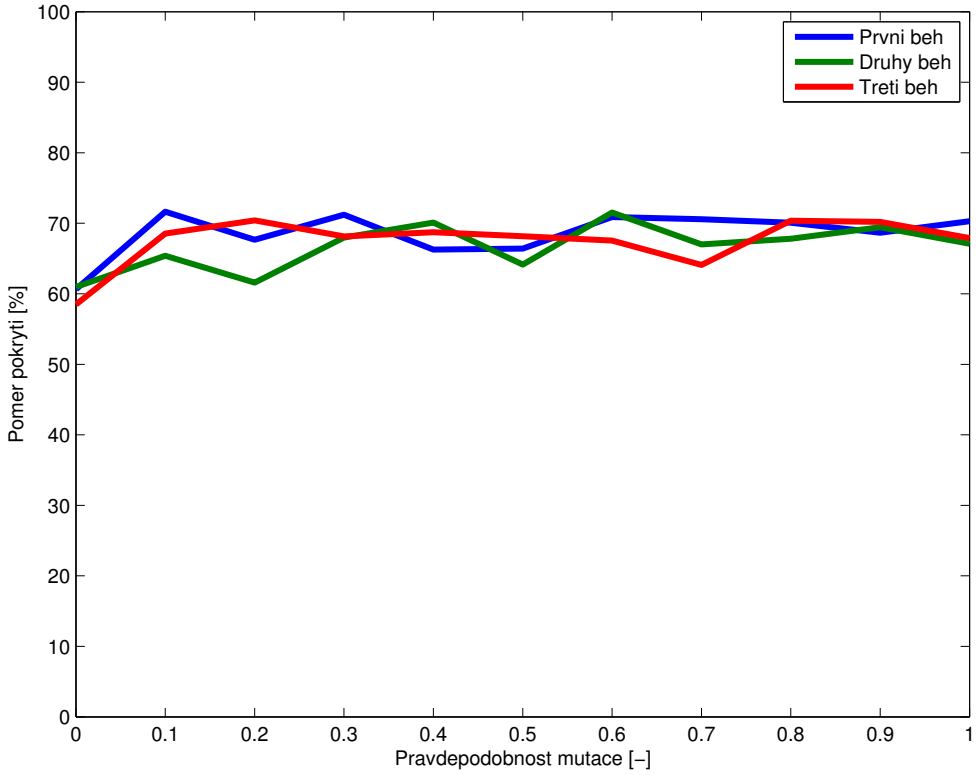
5.1.3 Diskuze

Z obr. 5.1 lze pozorovat, že se zvětšujícím se počtem hlídek roste rozdíl mezi greedy a greedy + local search (GLS) algoritmem. Pro jednu hlídku jsou pak tyto algoritmy stejně efektivní. Toto ostatně odpovídá i funkci GLS.

V případě jedné hlídky je totiž výstup z greedy algoritmu optimálním řešením problému. Postup greedy algoritmu je totiž identický s postupem řešení hrubou silou, tedy vyzkoušením všech možných řešení. GLS se pak snaží vylepšit výstup z greedy algoritmu, nicméně v tomto případě již není co vylepšovat.

Zvětšování náskoku GLS algoritmu se zvyšováním počtu hlídek je dáno faktem, že s vyšším počtem hlídek má algoritmus více míst, kde změnit řešení a nalézt tak lepší řešení.

Naopak snížení rozdílu u vyššího počtu hlídek je dáno faktem, že riziko je téměř celé eliminováno hlídkami a není tedy už co vylepšovat.



Obrázek 5.4: Vliv četnosti mutací na genetický algoritmus

Rozdíly na obr. 5.2 jsou pak dány redundancí v rozmístění hlídek, kterou se právě GLS snaží eliminovat. Tato redundance spočívá v jevu, kdy jsou na buňku mapy rizika c aplikovány dvě anebo více funkcí vlivu. Ve vztahu (3.3) je pomocí funkce max ošetřeno, že hodnota $val(c_n)$ není menší než 0. Pokud by se tam funkce max nevyskytovala, mohla by $val(c_n)$ nabývat záporných hodnot. Pokud by k tomuto došlo, funkce vlivů hlídek na tuto buňky by byly redundantní.

Jestliže mají hlídky malý poloměr účinnosti, pak je i malá pravděpodobnost, že výsledné řešení bude trpět výše zmíněnou redundancí. Toto vysvětluje postupné navyšování rozdílu s navyšováním poloměru účinnosti. Snižování rozdílu pro vysoký poloměr účinnosti je ze stejného důvodu, jako u snížení rozdílu pro vysoký počet hlídek.

Ze srovnání genetického algoritmu (GA) s greedy algoritmem a GLS a algoritmem na obr. 5.3 lze vidět, že GA nedosahuje kvalit greedy a GLS algoritmů. Tento rozdíl je patrnější s vyšším počtem hlídek. Toto je důsledkem faktu, že pro vyšší počet hlídek je v genomu i vyšší počet genů a je tedy více možných kombinací řešení. GA pak má menší šanci dojít ke správnému řešení.

Z obr. 5.4 je patrné, že mutace nemá na výsledek běhu genetického algoritmu žádný pozorovatelný vliv. Výjimku tvoří případ, kdy mutace nenastává vůbec, tehdy totiž absence mutace působí výrazně negativně na kvalitu řešení. Pozdější potlačení vlivu mutace je pravděpodobně dáno jednak zadáním problému, jednak pak nahra-

zováním desetiny nejhorší populace v každém kroku. Tento postup dodává do běhu algoritmu nová možná řešení a částečně tak vyrovnává různý vliv mutace.

5.2 Algoritmy pro hlídkování

5.2.1 Způsob testování

Tyto algoritmy byly testovány pouze pro případ nasazení jedné hlídky. O možnosti nasadit více hlídek se zmiňuje diskuze (5.2.3).

Vyhodnocování algoritmu probíhá pomocí záznamu celkové hodnoty mapy rizika $val(\Theta)$ v čase. Jediným měněným faktorem byla časová konstanta T (viz 4.2).

Během vývoje a zkoušení algoritmů se ukázalo, že průběh rizika v mapě je v delším časové úseku periodický. Právě tato perioda posloužila jako jedna z vlastností algoritmu. Naměřená data na obr. 5.5 totiž udávají závislost algoritmu na časové konstantě. Perioda byla určena pomocí autokorelační funkce.

Druhá měřená vlastnost byla průměrné riziko v risk mapě. Tato hodnota byla získána jako střední hodnota plovoucího průměru celkového rizika. Délka plovoucího průměru odpovídala velikosti periody.

5.2.2 Výsledky testování

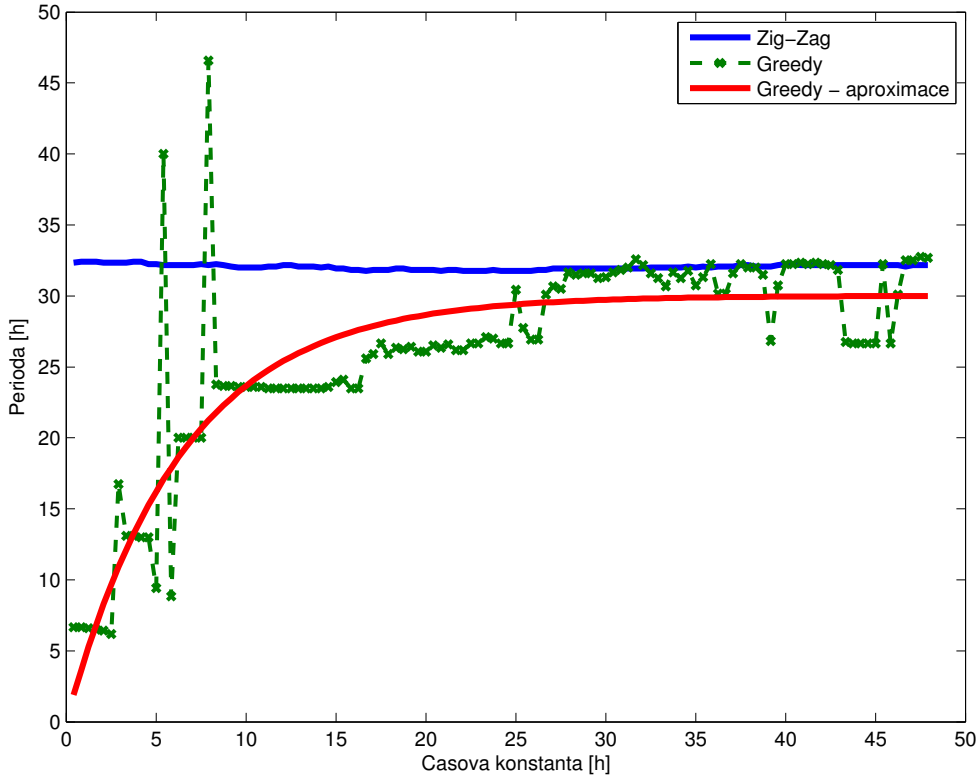
Z obr. 5.5 je vidět, že greedy algoritmus je velmi citlivý na délku časové konstanty. Zvláště pak při jejích nízkých hodnotách. Pro přehlednost byl greedy algoritmus aproximován funkcí $a \cdot e^{b \cdot x}$. Naproti tomu zig-zag algoritmus není délkou časové konstanty prakticky vůbec ovlivněn.

Z obr. 5.6 je patrné, že zvláště pro nízké hodnoty časové konstanty dává greedy algoritmus lepší výsledky, než zig-zag. Na druhou stranu, zig-zag je v závislosti na časové konstantě stabilnější.

5.2.3 Diskuze

Na obr. 5.5 lze pozorovat, že pro vysoké hodnoty časové konstanty se greedy algoritmus přiblížil k zig-zag algoritmu, oproti nízkým hodnotám, kdy je rozdíl velký. Tento jev ukazuje, že pro velkou hodnotu časové konstanty se greedy algoritmus chová podobně jako zig-zag, v tom smyslu, že hlídkuje přes celou hlídanou oblast.

Vysvětlení tohoto efektu je následující. Greedy algoritmus plánuje znatelně menší trasy τ než zig-zag. Díky malé časové konstantě se hodnoty mapy rizika obnovují velmi rychle. Pokud tedy hlídka propluje oblastí s velkým rizikem, toto riziko



Obrázek 5.5: Vliv časové konstanty na periodu součtu rizika v risk mapě

se tam obnoví velmi brzy, v době kdy je tato oblast ještě v „dosahu“ plánovacího algoritmu, který tam hlídku opět nasměruje.

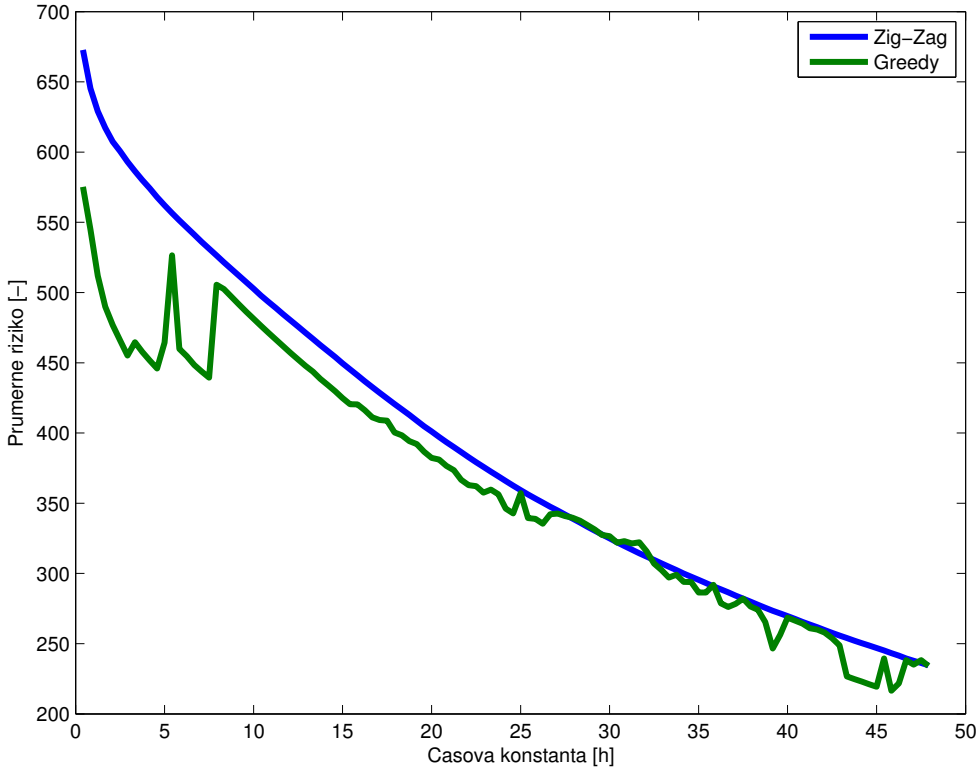
Naproti tomu pro velkou časovou konstantu nemá greedy důvod vracet hlídku do oblastí nedávno proplutých a proto plánuje trasu do oblastí nových. Takto postupně propluje celou hlídanou oblast.

Na obr. 5.6 lze pozorovat, že greedy dává lepší výsledky pro velký rozptyl časových konstant. Pro malé hodnoty časové konstanty jsou pak jeho výsledky výrazně lepší, než výsledky zig-zag. Toto je způsobené opět jevem, že pro rychlé obnovování mapy rizika se greedy drží v oblasti s nejvyšším rizikem, oproti zig-zag, který stále propátrává celou oblast.

Problém více hlídek

Jak již bylo uvedeno výše, plánovací algoritmy v této práci jsou testovány pouze pro jednu hlídku v hlídané oblasti. Tato sekce objasňuje a navrhuje řešení pro problém více hlídek.

Jedním z řešení, jak řídit hlídkování N hlídek je rozdělit oblast na N podoblastí a pro každou oblast přidělit jednu hlídku.



Obrázek 5.6: Vliv časové konstanty na průměrné riziko risk mapy

Další možností je nechat více hlídek hlídkovat v jedné oblasti nezávisle na sobě. Bohužel, pro greedy algoritmus tento způsob vede ke uváznutí hlídek v hodně blízkých trasách. Pro důkaz tohoto faktu stačí následující úvaha.

Pro zjednodušení je zaveden předpoklad, že všechny hlídky plánují ve stejný čas. To znamená, že hlídky vyberou cestu do oblasti s nejvyššími hodnotami risk mapy. Protože o sobě navzájem neví, neví ani o tom, že jejich přítomnost v oblasti je redundantní.

Jestliže by došlo k tomu, že hlídky by od sebe byly vzdálené natolik, že by každá hlídala jiné lokální maximum, pak je možné, že by nedošlo ke vzájemné konvergenci jejich tras. Zároveň by se ale defakto jednalo o rozdělení oblasti na N podoblastí, jak je uvedeno výše. Krom toho takovýto stav je nestabilní, protože stačí jedna příhodná situace, aby se hlídky dostaly k sobě. Naopak šance, aby se hlídky dostaly od sebe je velmi malá.

Nabízí se ještě možnost použití dvou algoritmů pro dvě hlídky, jednoho algoritmu typu zig-zag, druhého typu greedy. Tato možnost je funkční a je tedy možné ji použít. Bohužel počet dvou hlídek je nedostačující.

Řešením tohoto problému je implementovat vyjednávací protokol mezi jednotlivými hlídkami.

Závěr

V práci byl zpracován problém tvorby mapy rizika v zadané oblasti. Za pomoci této mapy bylo prezentováno několik algoritmů pro rozmístění hlídek

Při tvorbě mapy rizika byly vytvořeny dvě mapy: mapa rizika na základě apriorních znalostí a mapa rizika na základě aposteriorních znalostí. Pro testování algoritmů pak byla použita druhá z těchto map.

V práci jsou zpracovány tři algoritmy pro rozmístění hlídek. Nejlepších výsledků dosáhl algoritmus greedy + local search, který je vylepšením greedy algoritmu.

Pro plánování tras pro hlídky byl aplikován greedy algoritmus popsaný [2]. Dále byl vytvořen zig-zag algoritmus, založený na iterativním zkracování trasy.

Oproti greedy algoritmu vykazuje zig-zag algoritmus větší robustnost vůči změně časové konstanty pro obnovení mapy rizika. Greedy algoritmus oproti tomu dosahuje lepších výsledků pro většinu hodnot těchto konstant.

V budoucnu je možné mapu rizika rozšířit o podporu modelování dynamického rizika na základě pohybu transportních lodí. Taktéž je možné přidat podporu predikce vývoje mapy rizika. Pro použití plánovacích algoritmů pro více hlídek je třeba implementovat vyjednávací protokol mezi hlídkami. Dále je možné pokračovat ve vývoji zig-zag algoritmu například za pomoci přidání gama funkce nebo jiné funkce, která vezme v potaz zastarávání informace v čase.

Literatura

[1] Management Practices to deter Piracy in the Gulf of Aden and off the Coast of Somalia. [online], [cit. 18. 5. 2010].
URL `https://www.warrisk.no/?module=Files;action=File.getFile;ID=604`

[2] Ahmadi, M.; Stone, P.: Continuous Area Sweeping: A Task Definition and Initial Approach. V *The 12th International Conference on Advanced Robotics*, July 2005.

[3] AsianYachting Ventures: Gulf of Aden Pirate Corridor Waypoints - New coordinates came into effect. [online], [cit. 5. 5. 2010].
URL `http://asianyachting.com/news/PirateCorridor.htm`

[4] Buşoniu, L.; Babuška, R.; De Schutter, B.: A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, ročník 38, č. 2, Březen 2008: s. 156–172, doi:10.1109/TSMCC.2007.913919.

[5] Council, N. S.: Countering Piracy off the Horn of Africa: Partnership & Action Plan. [online], [cit. 13. 5. 2010].
URL `http://www.marad.dot.gov/documents/Countering_Piracy_Off_The_Horn_of_Africa_-_Partnership__Action_Plan.pdf`

[6] Jakob, M.; Semsch, E.; Pavlíček, D.; aj.: Occlusion-aware Multi-UAV Surveillance of Multiple Urban Areas. V *Proceedings of AAMAS 2010 Workshop on Agents In Traffic and Transportation*, 2010, [cit. 21. 5. 2010].

[7] MANW-NATO: Operation Ocean Shield. [online], [cit. 18. 5. 2010].
URL `http://www.manw.nato.int/page_operation_ocean_shield.aspx`

[8] Middleton, R.: Piracy in Somalia: Threatening Global Trade, Feeding Local Wars. [online], [cit. 13. 5. 2010].
URL `http://www.chathamhouse.org.uk/publications/papers/download/-/id/665/file/12203_1008piracysomalia.pdf`

[9] MSC(HOA): About The Maritime Security Centre - Horn of Africa. [online], [cit. 18. 5. 2010].
URL `http://www.mschoa.org/About.aspx`

[10] MSC(HOA): Force Information. [online], [cit. 18. 5. 2010].
URL http://www.mschoa.org/ForceInfo.aspx

[11] North of England P&I Association: *Piracy – Gulf of Aden*. [online], [cit. 10. 4. 2010].
URL http://www.nepia.com/cache/files/1576-1266327081/LPBriefing-Piracy-GulfofAden.pdf

[12] Tsvetkova, B.: Securiziting Piracy Off the Coast of Somalia. [online], [cit. 13. 5. 2010].
URL http://www.cejiss.org/assets/pdf/articles/vol3-1/Tsvetkova-Piracy_in_Somalia.pdf

[13] UNOSAT: Analysis of somali pirate activity in 2009. [online], [cit 5. 5. 2010].
URL http://unosat-maps.web.cern.ch/unosat-maps/SO/Piracy/2009/UNOSAT_Somalia_Pirates_Analysis_Q1_2009_23April09_v1.pdf

[14] Wikipedia, t. f. e.: Combined Task Force 150. [online], [cit. 18. 5. 2010].
URL http://en.wikipedia.org/wiki/Combined_Task_Force_150

[15] Wikipedia, t. f. e.: Combined Task Force 151. [online], [cit. 18. 5. 2010].
URL http://en.wikipedia.org/wiki/Combined_Task_Force_151

[16] Wikipedia, t. f. e.: Kernel density estimation. [online], [cit. 18. 5. 2010].
URL http://en.wikipedia.org/wiki/Kernel_density_estimation

Obsah CD

Přiložené CD obsahuje zdrojové kódy algoritmů a text diplomové práce ve formátu PDF. V následující tabulce je popsána struktura CD.

Adresář	Popis
`src`	zdrojové kódy knihovny
`agentC`	zkompilovaná platforma AgentC
`thesis.pdf`	text diplomové práce
`zadani.pdf`	oskenovaný zadávací formulář
`prohlaseni.pdf`	oskenované podepsané prohlášení prohlášení

Tabulka 1: Adresářová struktura na CD