FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání diplomové práce

Název:	Multi-agentní hledání cest se spojitým časem založené na celočíselném lineárním programování
Student:	Bc. Yana Zabrodskaya
Vedoucí:	doc. RNDr. Pavel Surynek, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Znalostní inženýrství
Katedra:	Katedra aplikované matematiky
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Cílem práce je prozkoumat možnosti kompilace multi-agentního hledání cest se spojitým časem (MAPF-R) jako celočíselného lineárního programu. Jedním z možných přístupů je nahlížet vznikající kolize mezi agenty líným způsobem, tj. přidávat omezení na eliminaci kolize až poté, co je kolize detekována. Celočíselný program modelující úlohu MAPF-R by tedy nebyl definovaný předem, ale vznikal v průběhu řešení úlohy. Úkoly pro uchazeče budou následující:

1. Prostudujte algoritmy pro řešení úlohy MAPF-R [1] a současně se seznamte s celočíselným lineárním programováním.
2. Na základě získaných poznatků navrhněte celočíselný model úlohy MAPF-R, případně smíšený celočíselný model, předpokládáme i využití prvků líné kompilace.
3. Svůj návrh implementujte formou softwarového prototypu a otestujte jej na relevantních testovacích úlohách dostupných na movingai.com.

[1] Anton Andreychuk, Konstantin S. Yakovlev, Dor Atzmon, Roni Stern: Multi-Agent Pathfinding with Continuous Time. IJCAI 2019: 39-45

Elektronicky schválil/a Ing. Karel Klouda, Ph.D. dne 2. prosince 2021 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Multi-agentní hledání cest se spojitým časem založené na celočíselném lineárním programování

Bc. Yana Zabrodskaya

Katedra aplikované matematiky
Vedoucí práce: doc. RNDr. Pavel Surynek, Ph.D.

2. května 2022

Poděkování

Ráda bych tímto poděkovala svému vedoucímu doc. RNDr. Pavlu Surynkovi, Ph.D., za veškerou pomoc při tvorbě diplomové práce. Také bych chtěla poděkovat mamince, babičce a kamarádce Polině za podporu.

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisu. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisu, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programu, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli zpusobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelum). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným zpusobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 2. května 2022 . . . . . .

Odkaz na tuto práci

Zabrodskaya, Yana. *Multi-agentní hledání cest se spojitým časem založené na celočíselném lineárním programování*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2022.

Abstrakt

Tato práce se zabývá problémem hledání nejkratší cesty pro několik robotů tak, aby se tito roboti nesrazili. Za účelem řešení výše uvedeného problému byl vyvinut algoritmus, který je založen na celočíselném lineárním programování s prvky líné kompilace. Následně je algoritmus experimentálně vyhodnocen.

Klíčová slova multi-agentní hledání cest, MAPF, Gurobi, celočíselné lineární programování, konfliktní prohledávání, robot, líná kompilace

Abstract

The thesis describes the problem of finding the shortest path for several robots so that these robots do not collide. In order to find a solution to this problem and basing on integer linear programming with a lazy compilation elements, an algorithm has been developed. The experimental evaluation of the algorithm is followed.

Keywords multi-agent pathfinding, MAPF, Gurobi, integer linear programming, conflict-based search, robot, lazy compilation

Obsah

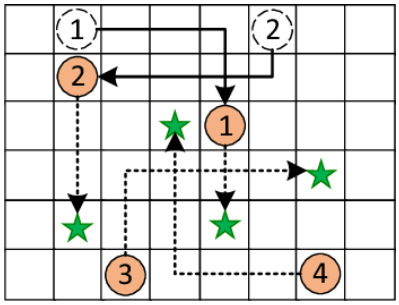
Seznam obrázků

Seznam tabulek

Seznam algoritmů

Úvod

Multi-agentním hledáním cest (MAPF) se zabývají především obory umělé inteligence a robotiky. Úkolem MAPF je naleznutí cesty pro několik agentů tak, aby se agenti nesrazili, a zároveň minimalizace délky těchto cest nebo jejich doby trvání. Agenti se pohybují mapou (grafem) přes vrcholy, každý agent má vlastní vrchol startu a cíle.



Obrázek 0.1: Příklad diskrétního MAPF problému [1]

Obrázek 0.2: Příklad spojitého MAPF problému

Prostředí, ve kterém se agenti nacházejí, je většinou spojité [2] a může mít složitý geometrický tvar, proto tato práce je zaměřena právě na problém multi-agentního hledání cest se spojitým časem (MAPF_R).

Cíl práce

Hlavním cílem práce je návrh a implementace algoritmu pro multi-agentní hledání cest se spojitým časem jako celočíselného lineárního programování.

Vedlejší cíle jsou:

- Analýza postupu při hledání cest více agentů
- Analýza dvou základních přístupů kompilací
- Analýza a způsoby řešení různých konfliktů při srážkách
- Popis navrženého algoritmu
- Experimentální vyhodnocení algoritmu a interpretace výsledků

Od navrženého algoritmu se očekávají následující výstupy:

- Bude nalezena cesta pro dva a více agentů
- Nalezena cesta bude bezkolizní
- Vyhnutí se srážkám bude optimální z hlediska celkového času

Struktura práce

Práce je rozdělena do tří kapitol. V první kapitole je představena teoretická část a algoritmy, ze kterých vychází vyvinuté řešení. Další kapitola zahrnuje popis a vysvětlení vlastního postupu nalezení nejkratších cest se spojitým časem pro několik agentů pomocí celočíselného lineárního programování. V poslední kapitole se ukazuje princip fungování navrženého algoritmu na několika mapách pro různé scénáře a uvádí se analýza výsledků výpočtů.

Kapitola 1

Východiska

Tato kapitola je zaměřena na definovaní multi-agentního hledání cest se spojitým časem ve spojitém prostoru, a kromě toho také na popsání řešení tohoto problému a metrik vyhodnocení jeho kvality. Následně je uveden algoritmus pro konfliktní prohledávání ve spojitém prostoru a několik algoritmů, které sloužily inspirací k tvorbě popsaného v další kapitole modelu pro řešení MAPF_R úlohy. Kapitola zahrnuje také i popis různých druhů kompilace.

1.1 Multi-agentní hledání cest ve spojitém prostoru

Problém multi-agentního hledání cest ve spojitém prostoru (MAPF_R) spočívá v nalezení cest pro k agentů tak, aby nedošlo ke srazu. Agent může mít libovolný tvar, avšak v této práci jsou agenti kruhoví.

Definice 1: *$MAPF_R$* problém je definován trojicí $\langle G, S, T \rangle$, kde *G(V, E)* je neorientovány graf s vrcholy V a hranami E. Vrcholy jsou body se souřadnicemi v prostoru R^2 a hrany jsou křivky, jež je spojují. Délka hrany $e_{ij} \in E$ se počítá jako Euklidova vzdálenost mezi vrcholy i a j. $S \in V$ je množina startovních vrcholů a $T \in V$ – množina cílových vrcholů. Platí, že žádní dva agenti nemohou začít nebo skončit svou cestu ve stejném vrcholu, tj. $|S| = |T| = k$ [3, 1, 4, 5].

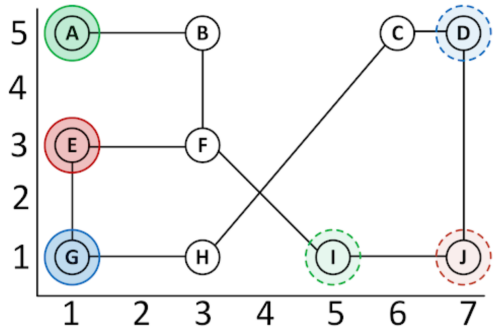
Úkolem agenta je provést posloupnost akcí tak, aby se dostal ze svého startovního vrcholu $s_i \in S$ do svého cílového vrcholu $t_i \in T$. Akce je relace $a : V \to V$ taková, že $a(v) = v'$, tj. pokud $v = v'$, jedná se o akci čekání ve vrcholu v, jinak agent provede akci přechodu z vrcholu v do v'. Pohyb mezi vrcholy je spojitý přesun hranou $e_{vv'} \in E$.

Výsledná posloupnost akcí π je cestou agenta. Tato cesta musí být nejkratší a bezkolizní. Délka cesty se počítá dvěma základními způsoby [4, 5]:

- *Součet cen*[1] – součet délek všech hran v cestě: $\sum_{i \in \pi} |\pi_i|$;
- *Délka trvání cesty*[2] – rozdíl v čase mezi odjezdem ze startovního vrcholu a příjezdem do cílového: $\max_{i \in \pi} |\pi_i|$.

Bezkolizní cesta znamená, že se agent musí vyhnout konfliktům s ostatními agenty. Srážka nastává kdykoliv, kdy se agenti překrývají [4, 6].

Řešením MAPF$_R$ problému je nalezení bezkolizní cesty pro každého agenta a minimalizace optimalizační funkce (SOC nebo makespan) [1].



Obrázek 1.1: Příklad: MAPF$_R$ problému [4]

1.2 Konfliktní prohledávání ve spojitém prostoru

Jedním z algoritmů pro nalezení řešení MAPF$_R$ problému je konfliktní prohledávání ve spojitém prostoru[3]. Jde o dvouúrovňový algoritmus založený na hledání konfliktů a vypočítání nebezpečných intervalů. Nebezpečný interval je časový úsek, kdy agent nesmí vykonávat žádnou akci, aby předešel konfliktu.

Horní vrstva algoritmu obsahuje binární strom, tzv. Constraint Tree (CT). V uzlech stromu jsou omezení pro agenty, řešení a výsledek optimalizační funkce pro dané řešení.

Kořen stromu má prázdnou množinu omezení. Pokud v řešení v uzlu N nastane konflikt mezi agentem a_i v čase $[t_0, t_+)$ pohybujícím se hranou e_{uv} a agentem a_j v čase $[t'_0, t'_+)$ pohybujícím se hranou $e_{u'v'}$, spočítá se nebezpečný interval (unsafe interval) $[\tau_0, \tau_+)$ a vytvoří se dva následnici [8].

Levý následník zdědí všechna omezení z rodičovského uzlu a přidá nové omezení pro agenta a_i ve tvaru $(a_i, e_{uv}, [\tau_0, \tau_+))$ a pravý následník po zdědění předchozích omezení přidá další $(a_j, e_{u'v'}, [\tau_0, \tau_+))$. Toto omezení říká, že agent a_i nesmí použít hranu e_{uv} v čase $[\tau_0, \tau_+)$, pro agenta a_j platí totéž [7, 8].

[1] Sum of costs (SOC)
[2] Makespan
[3] Conflict-based search with continuous time (CCBS) [4, 7, 8]

Spodní vrstva hledá pro jednoho agenta novou nejkratší cestu s ohledem na získaný od horní vrstvy seznam omezení.

Řešení se poté nachází v uzlu s minimální hodnotou optimalizační funkce a cestami, ve kterých se nevyskytuje žádný konflikt.

Pseudokód horní vrstvy je popsán v algoritmu 1.1 [7].

Algoritmus 1.1 CCBS: konfliktní prohledávání ve spojitém prostoru (horní vrstva)

$R.constraints \leftarrow \varnothing$
$R.solution \leftarrow$ najdi cestu pro každého agenta
$R.\mu \leftarrow max_\mu(N.solution(a_i))$
$OPEN \leftarrow R$
while $OPEN \neq \varnothing$ **do**
  $N \leftarrow min_\mu(OPEN)$
  $collisions \leftarrow$ konflikty v $N.solution$
  if $collisions = \varnothing$ **then**
    return $N.solution$
  end if
  let $(a_i, \{u, v\}, [t_0, t_+)) \times (a_j, \{u', v'\}, [t'_0, t'_+)) \in collisions$
  $[\tau_0, \tau_+) \leftarrow$ computeUnsafeInterval()
  for *each* $(a, \{w, z\} \in \{(a_i, \{u, v\}), (a_j, \{u', v'\})\})$ **do**
    $P \leftarrow$ nový uzel
    $P.constraints \leftarrow N.constraints \cup \{(a, \{w, z\}, [\tau_0, \tau_+))\}$
    $P.solution \leftarrow N.solution$
    $P.solution(a) \leftarrow$ cesta pro agenta a
    $P.\mu \leftarrow \sum_{i=1}^{k} \mu(P.solution(a_i))$
    $OPEN \leftarrow OPEN \cup P$
  end for
  $OPEN.pop()$
end while

1.3 Kompilace pomocí výrokové splnitelnosti

Jedním ze způsobů řešení MAPF_R úloh je převod plánování do SAT[4].

1.3.1 Výroková logika

SAT řešič očekává na vstupu booleovskou formuli v konjunktivní normální formě (CNF) a vrací výsledek s informací, zda je daná formule pravdivá.

Formule je v CNF, pokud je klauzulí nebo konjunkcí několika klauzulí.

- Literál je prvotní formule nebo její negace, např. x, $\neg x$.

[4]Problém splnitelnosti booleovské formule (Boolean satisfiability problem)

- Klauzule je literál nebo disjunkce několika literálů, např. $(\neg x)$, $(y \vee \neg z)$.

Příklad formule v CNF: $(\neg x \vee y) \wedge (x \vee \neg z) \wedge (y \vee z)$. Formule je splněna, pokud existuje takové ohodnocení literálů, při kterém je formule pravdivá, čili nabývá hodnoty 1. Ohodnocení $x = 0,\ y = 1,\ z = 0$ splňuje výše uvedenou formuli [9].

1.3.2 Klasická kompilace

Klasická kompilace je převod celého problému do booleovské formule. Pokud SAT řešič nalezne řešení formule, z výsledného ohodnocení literálů lze sestavit řešení plánovacího problému.

Jedním ze základních příkladů je přesun nákladu. Ten je definován následujícím způsobem: je potřeba přesunout krabici K z místa A do místa B pomocí auta T.

Zmíněný problém lze převést na výrokovou logiku – například pomocí jazyka PDDL. Tento jazyk používá operátory, které mají své podmínky a efekty.

Na začátku se krabice K nachází v místě A a auto T – v místě B. Toto lze popsat konjunkcí literálů *at(K, A), at(T, B), place(A), place(B), truck(T), cargo(K)*. Cílový stav je: krabice přemístěna na místo B – *at(K, B)*.

Operátory jsou:

- *move(T, A, B)* – přesun auta z místa A do místa B
  - podmínky: *truck(T), place(A), place(B), at(T, B)*
  - efekty: $\neg$*at(T, A), at(T, B)*
- *load(K, T, A)* – naložení krabice do auta v místě A
  - podmínky: *truck(T), place(A), cargo(K), at(T, A), at(K, A)*
  - efekty: $\neg$*at(K, A), at(K, T)*
- *unload(K, T, A)* – vykládka krabice z auta v místě A
  - podmínky: *truck(T), place(A), cargo(K), at(K, T), at(T, A)*
  - efekty: $\neg$*at(K, T), at(K, A)*

Potom řešením je posloupnost akcí *move(T, B, A), load(K, T, A), move(T, A, B), unload(K, T, B)*.

1.3.2.1 SATPlan

Klasickou kompilaci pro převod plánování z PDDL do SAT využívá například plánovač SATPlan [10, 11]. Verze SATPlan z roku 2006 vytváří plánovací graf o velikosti k, převádí dána grafem omezení na množinu klauzulí a poté řeší

tuto formuli SAT řešičem. Pokud řešení není nalezeno, zvětší se hloubka grafu k a postup se zopakuje.

Plánovací graf je sestaven takovým způsobem, že ve hloubce 0 se nachází počáteční stav problému a ve hloubce k – cíle. Potom hrany odpovídají provedeným akcím a vrcholy obsahují efekty, které nastaly po těmto akcím.

Pseudokód SATPlanu je popsán v algoritmu 1.2 [12]

Algoritmus 1.2 SATPlan (init, transition, goal, T_{max})

```
for t = 0 to T_max do
    cnf ← TRANSLATE-TO-SAT(init, transition, goal, t)
    model ← SAT-SOLVER(cnf)
    if model is not null then
        return EXTRACT-SOLUTION(model)
    end if
end for
return failure
```

Nevýhodou klasické kompilace je vliv velikosti formule na paměťovou náročnost a délka trvání výpočtu.

1.3.3 Líná kompilace

Líná kompilace na rozdíl od klasické využívá postupné přidávání klauzulí do formule. Na začátku se formule skládá jen z částí podformulí a nalezení pravdivého ohodnocení neznamená správnost řešení problému. Naopak, když formule s podmnožinou klauzulí není splnitelná, platí, že řešení problému neexistuje.

Po nalezení pravdivého ohodnocení se provede kontrola vynechaných klauzulí. Pokud nějaká klauzule není splněna, přidá se do formule, což bude pokračovat, dokud nebudou splněny všechny klauzule nebo řešič nevrátí odpověď, že žádné ohodnocení nesplňuje právě zkoumanou podmnožinu klauzulí.

1.3.3.1 Algoritmus SMT-CBS

Jednou z metod založených na líné kompilaci je SMT-CBS algoritmus, jehož pseudokód je popsán v algoritmu 1.3 [7], [13].

SMT-CBS sjednocuje dva základní přístupy k řešení MAPF problémů a využívá jak konfliktní prohledávání (CBS), tak převod do výrokové logiky[5].

SMT-CBS používá na horní úrovni algoritmus CBS pro teorie splnitelnosti, kdežto na dolní úrovni – kódovaní do SAT. Místo větvení stromu po nalezení konfliktu v CBS se do modelu přidá disjunktivní omezení.

Toto omezení znamená, že jednomu nebo druhému agentu bude zakázáno se srazit. Omezení je formule ve tvaru CNF, tj. seznám klauzulí, což v interpretaci lineárního programu lze chápat jako seznám řádků. Přidávaní disjunktiv-

[5]Satisfiability modulo theories (SMT)

ních omezení se podobá generování řádků, které bude popsáno v souvisejících metodách.

Následně je zpřesněný model ověřen SAT řešičem.

Výsledkem je neúplný výrokový model $H(\xi)$, který je splnitelný, když problém Σ má řešení s cenou ξ.

Algoritmus 1.3 SMT-CBS $(\Sigma = (G, A, \alpha_0, \alpha_+), \xi)$

```
H(ξ) ← encode-Basic(ξ, Σ)
while TRUE do
    assignment ← consult-SAT-Solver(H(ξ))
    if assignment ≠ UNSAT then
        paths ← extract-Solution(assignment)
        collisions ← validate(paths)
        if collisions ≠ ∅ then
            return paths
        end if
        for each (a_i, a_j, v, t) ∈ collisions do
            H(ξ) ← H(ξ) ∪ {¬X_v^t(a_i) ∨ ¬X_v^t(a_j)}
        end for
    end if
    return UNSAT
end while
```

1.4 Kompilace pomocí lineárního programování

Lineární programování je technikou matematického modelování, ve které je lineární funkce minimalizována nebo maximalizována a je ovlivněna různými omezeními [14].

Řešením problému lineárního programování je nalezení optimální hodnoty lineárního výrazu (optimalizační funkce)

$$f = c_1 x_1 \dots c_n x_n$$

za podmínek

$$a_{l1} x_1 + \dots + a_{ln} x_n \leq b_1$$

$$\vdots$$

$$a_{m1} x_1 + \dots + a_{mn} x_n \leq b_m, \forall x_i \geq 0$$

Pomocí lineárního programování lze například spočítat hledání nejkratší cesty, dokonce i několika způsoby.

První možnost: máme graf $G = (V, E)$, kde je každé hraně přidělena váha w – její délka. Úkolem je najít nejkratší cestu z vrcholu s do vrcholu t. Ať d_t je součet vah cesty z s do t, potom program vypadá následovně [14]:

$$\text{minimalizovat } d_t$$

za podmínek

$$d_s = 0$$

$$d_v \leq d_u + w(u, v), \forall (u, v) \in E$$

Ve výsledku se každá proměnná d_v bude rovnat součtu vah nejkratší cesty z s do v.

Druhá možnost, skrze kterou lze zapsat stejný problém, je označit, která hrana se vyskytuje ve výsledné cestě [14]:

$$\text{minimalizovat} \sum_{\forall (u,v) \in E} w(u, v) x_{u,v}$$

za podmínek

$$\sum_{\forall (u,v) \in E} x_{u,v} = \sum_{\forall (v,w) \in E} x_{v,w}, \forall v \in V \setminus \{s, t\}$$

$$\sum_{\forall (u,v) \in E} x_{u,v} = 1$$

Proměnná $x_{u,v}$ nabývá hodnoty 0 nebo 1 v souvislosti s tím, zda se tato hrana vyskytuje v cestě.

Jiná formulace označování navštívených hran je [15]:

$$\text{minimalizovat} \sum_{\forall (u,v) \in E} w(u, v) x_{u,v}$$

za podmínek

$$\sum_{\forall (u,s) \in E} x_{u,s} - \sum_{\forall (s,w) \in E} x_{s,w} = 1$$

$$\sum_{\forall (u,t) \in E} x_{u,t} - \sum_{\forall (t,w) \in E} x_{t,w} = -1$$

$$\sum_{\forall (u,v) \in E} x_{u,v} - \sum_{\forall (v,w) \in E} x_{v,w} = 0, \forall v \in V \setminus \{s, t\}$$

$$x_{u,v} \geq 0, \forall (u, v) \in E$$

1.4.1 Tok v síti

Definice 2: *Síť* je ohodnocený orientovaný graf $G = (V, E)$ s hodnoticí funkci $c : E \to \mathbb{R}_0^+$, která přiřazuje hranám kapacitu. Dále jsou vrcholy rozděleny do tří disjunktních množin: zdroje, stoky a vnitřní vrcholy $V = V^+ \cup V^- \cup V^0$. Předpokládáme, že pro každou hranu e_{uv} existuje opačná hrana e_{vu}. Pokud taková hrana chybí, lze ji přidat s nulovou kapacitou [14, 16].

Definice 3: *Tok v síti* je funkce $f : E \to \mathbb{R}_0^+$, která splňuje následující podmínky [14, 16]:

- tok hrany je omezen její kapacitou:

$$\forall e \in E : f(e) \leq c(e)$$

- tok, který vtéká do vnitřního vrcholu (přítok), se musí rovnat toku, který z něj odtéká (odtok):

$$\forall v \in V^0 : \sum_{e \in \to v} f(e) = \sum_{e \in v \to} f(e)$$

- odtok ve zdrojích musí být větší než přítok:

$$\forall v \in V^+ : \sum_{e \in \to v} f(e) \leq \sum_{e \in v \to} f(e)$$

- přítok ve stocích musí být větší než odtok:

$$\forall v \in V^- : \sum_{e \in v \to} f(e) \leq \sum_{e \in \to v} f(e)$$

Základní úlohou je nalezení maximálního toku, tak aby odtok ze zdrojů byl maximální možný. Tento problém lze také vyřešit pomocí lineárního programování, kde optimalizační funkce je

$$\text{maximalizovat} \sum_{\forall (i,j) \in E} x_{i,j}$$

za podmínek

$$0 \leq x_{i,j} \leq c_{i,j}, \forall (i,j) \in E$$

$$\sum_{\forall (u,i) \in E} x_{u,i} = \sum_{\forall (i,w) \in E} x_{i,w}, \forall i \in V^0$$

Úkolem nalezení minimálního toku byl inspirován lineární program uvedený v následující kapitole.

1.5 Související metody

Poslední podkapitola uvádí několik metod pro řešení velkých problémů, primárně se jedná o nalezení přípustného řešení velkých soustav lineárních rovnic.

1.5.1 Column-and-row-generation

Column gereration [17], neboli generování sloupců, je algoritmus pro řešení velkých soustav lineárních rovnic.

Počet proměnných je někdy příliš velký na to, aby je bylo možné explicitně vygenerovat. Cílem algoritmu je generování jenom podmnožiny proměnných, což snižuje velikost zkoumaného problému.

Algoritmus přeformuluje problém na master problém a jeden nebo více podproblémů. Každý sloupec v master problému reprezentuje řešení jednoho podproblému. Řešením podproblému je poté nalezení přípustného řešení části lineární soustavy. Iterativně se hledá řešení master problému, které zajistí, že všechna omezení mezi sloupci jsou splněna. Podproblém následně vygeneruje nové sloupce, které lze přidat do master problému.

Row generation [17], neboli generování řádků, přidává řádky do master problému, což je založeno na podobném principu jako přidání omezení v líné kompilaci.

1.5.2 Counterexample-guided Abstraction Refinement

Ověření velkého modelu je dalším složitým problémem. Metoda CEGAR[6] [18] umožňuje vytvářet menší abstraktní model, kontrolovat a zpřesňovat ho v případě, že kontrola selže

Nalezený při kontrole protipříklad může být výsledkem provedené aproximace a nemusí být přítomen v původním modelu. Daný protipříklad se používá pro zpřesnění abstraktního modelu z toho důvodu, aby v následující iteraci k chybnému chování již nedocházelo.

1.5.3 Elipsoidová metoda

Další podobnou metodou je elipsoidová metoda [19], [20]. Jedná se o iterativní metodu pro minimalizaci konvexních funkcí, jejímž cílem je hledání optimálního řešení v polynomiálním čase.

Počáteční elipsoid obsahuje množinu všech řešení soustavy lineárních nerovnic a jeho střed je kandidátem pro přípustné řešení. Splňuje-li tento bod všechny nerovnice, algoritmus skončí s nalezeným řešením. V opačném případě budou nesplněné nerovnice použity pro konstrukci dalšího elipsoidu o menší velikosti a s jiným centrem. Postup se opakuje, dokud nebude nalezeno řešení soustavy nebo nebude překročen maximální počet iterací.

[6] Counterexample-guided Abstraction Refinement (zpřesnění abstrakce vedené protipříkladem)

Kapitola 2

Vyvinuté metody

Níže uvedená kapitola nejdříve definuje řešení z hlediska jeho struktury a složení. Následně jsou uvedeny vyvinuté lineární programy pro nalezení nejkratších bezkolizních cest.

2.1 Reprezentace řešení

Řešením je struktura, která se skládá z množiny k agentů, k cest (kde i-tá cesta odpovídá i-tému agentu), množiny omezení a hodnoty optimalizační funkce pro dané řešení. Jako optimalizační funkce byl zvolen makespan, neboli součet časových kroků mezi začátkem a koncem cesty přes každého agenta.

2.1.1 Reprezentace mapy

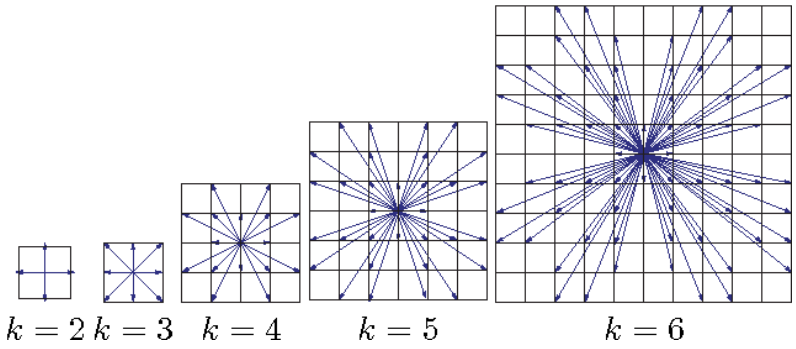
Mapa je struktura, která je složena z n vrcholů, kde každý vrchol má svůj index z množiny 0 až $n-1$ a souřadnice x, y. Vrcholy jsou spojeny podle technologie 2^k Neighborhoods (obrázek 2.1) [21]. Bylo vyzkoušeno $k = 2$, 3 a 4. Například pro $k = 4$, platí, že pokud souřadnice vrcholu jsou (i, j), bude spojen s vrcholy $(i,\ j \pm 1)$, $(i \pm 1,\ j)$, $(i \pm 1,\ j \pm 1)$, $(i \pm 1,\ j \pm 2)$, $(i \pm 2,\ j \pm 1)$, pokud takové existují.

2.1.2 Reprezentace agenta

Mapou se pohybuje k agentů. Každý agent má index z množiny $\{0, 1, ..., k-1\}$, vlastní a odlišní počáteční a koncový vrchol, rychlost pohybu a poloměr. Všichni agenti jsou kruhoví.

2.1.3 Reprezentace omezení

Omezení značně ovlivňují hledání cesty a jsou přidána po detekci konfliktu. Program rozlišuje 3 druhy omezení:



Obrázek 2.1: 2^k Neighborhoods

- Srážka mezi dvěma agenty i a j nastala ve stejném vrcholu v v čase t. Pak omezení bude mít tvar $(agent_i, agent_j, v, t)$;
- Srážka nastala při pohybu stejnou hranou mezi vrcholy v a v' v čase t, potom omezeni je $(agent_i, agent_j, e_{vv'}, t)$;
- Srážka byla detekována kdekoliv jinde, přičemž se agent i nacházel mezi vrcholy v a v'. V takovém případě omezení vypadá jako $(agent_i, agent_j, v, t')$, kde t' je čas, který agent i musí počkat ve vrcholu před konfliktem, aby mu předešel.

Po přidání omezení je nalezena nová cesta, přičemž i-tá cesta musí splňovat všechna omezení, ve kterých se vyskytuje i-tý agent na první pozici omezení.

2.2 Překryvný konflikt a jeho řešení

Konflikt je detekován kdykoliv, kdy se dva agenti dotýkají nebo překrývají. V takovém případě je vzdálenost mezi centry agentů menší než součet jejich poloměrů: $R_i + R_j \leq \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$, kde (x_i, y_i) a R_i je poloha agenta i v čase t a jeho poloměr, to samé platí i pro agenta j.

2.2.1 Konflikt ve vrcholu

Pokud se oba agenti srazí ve stejném vrcholu v současně, bude přidáno omezení, že jeden agent nesmí vstoupit do tohoto vrcholu dříve, než druhý ho opustí. Potom čas čekání je $t' = \frac{R_i + R_j}{s_j}$, který se rovná součtu poloměrů agentů děleného rychlostí druhého agenta. Každý agent si pamatuje čas příjezdu a odjezdu z vrcholu a podle tohoto času lze detekovat vrcholový konflikt. Bude-li se agent i nacházet ve vrcholu v v době mezi t^i_{arrive} a t^i_{depart} a agent j – t^j_{arrive} a t^j_{depart}, potom konflikt nastane, když:

- oba agenti dorazili současně a nikdo nečeká (obrázek 2.2), neboli

$$t^i_{arrive} = t^i_{depart} = t^j_{arrive} = t^j_{depart} = t$$

- jeden z agentů navštívil vrchol, zatímco druhý ve vrcholu čekal (obrázek 2.3):

$$t^i_{arrive} \leq t^j_{arrive} \leq t^j_{depart} \leq t^i_{depart}$$

  nebo

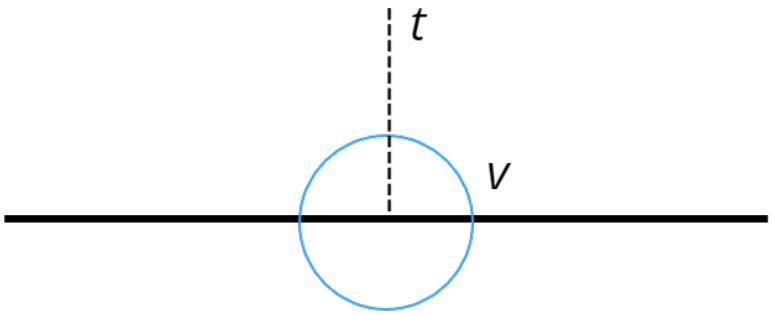
$$t^j_{arrive} \leq t^i_{arrive} \leq t^i_{depart} \leq t^j_{depart}$$

- oba agenti čekají (obrázek 2.4), tj.

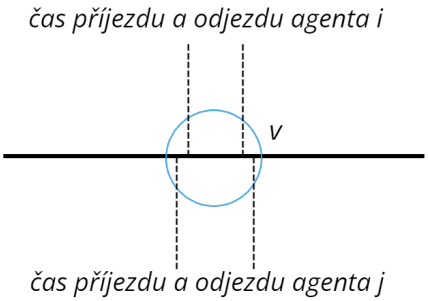
$$t^i_{arrive} \leq t^j_{arrive} \leq t^i_{depart} \leq t^j_{depart}$$

  nebo

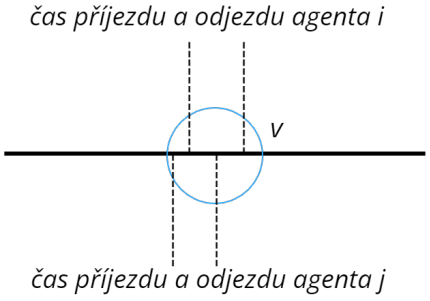
$$t^j_{arrive} \leq t^i_{arrive} \leq t^j_{depart} \leq t^i_{depart}$$



Obrázek 2.2: Bez čekání



Obrázek 2.3: Jeden čeká

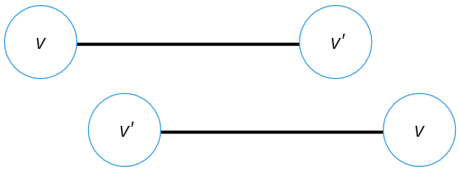


Obrázek 2.4: Oba čekají

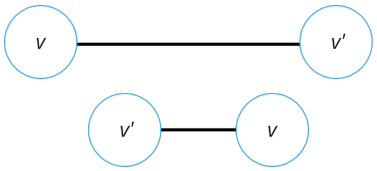
2.2.2 Konflikt na hraně

Hranový konflikt může mít podobu dvou případů:

- Prohození agentů, tj. jeden agent se pohybuje hranou $e_{vv'}$, zatímco druhý jede stejnou hranou v opačném směru – $e_{v'v}$ (obrázky 2.5 2.6).
- Předjíždění agenta – při různých rychlostech může docházet k situaci, kdy rychlejší agent předhoní pomalejšího při přepravě stejnou hranou (obrázek 2.7).



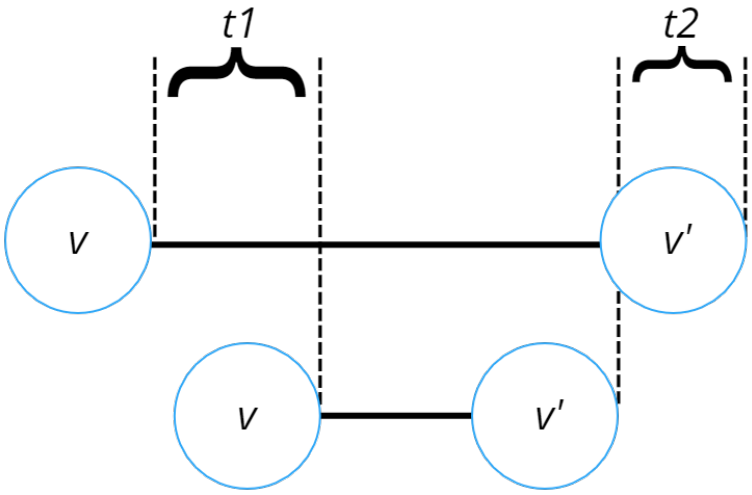
Obrázek 2.5: Stejná rychlost



Obrázek 2.6: Různá rychlost

První druh konfliktů se řeší tak, že pro vyhnutí se agent použije jinou hranu, než tu, na které nastala srážka nebo ze které právě přijel. Má-li vrchol v pouze stupeň 2, bude vrácen výsledek, že žádná cesta nemůže splnit dané omezení.

Předjíždění se řeší podobným způsobem jako vrcholový konflikt. Čas čekání potom je $t' = t_1 + t_2$, kde t_1 je rozdíl v časech odjezdů z vrcholu v a t_2 je rozdíl pro vrchol v'. Pomalejší agent počká čas t' ve vrcholu předcházejícímu v.



Obrázek 2.7: Předjíždění

2.2.3 Konflikt bez společného vrcholu nebo hrany

Rozdíl řešení od ostatních konfliktů je v tom, že je nutné najít polohu každého agenta v čase t. Ať agent j počká a agent i pokračuje ve své cestě. Výpočet času čekání je následující:

1. Najít souřadnice každého z agentů v čase t. Předpokládejme, že se agent j nacházel mezi vrcholy v a v', z vrcholu v odjel v čase t_v a vzdálenost mezi v a v' je $d_{vv'}$, potom souřadnice jsou:

$$(x_j, y_j) = [(x_v, y_v) \times (d_{vv'} - \frac{t - t_v}{s_j}) + (x_{v'}, y_{v'}) \times \frac{t - t_v}{s_j}] \times d_{vv'}^{-1}$$

2. Najít úhel, ve kterém se momentálně nachází:

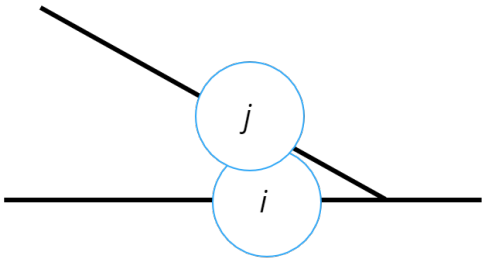
$$a = \text{atan2}(y_{v'} - y_v, x_{v'} - x_v)$$

3. Najít čas čekání – $t' = \frac{-b+\sqrt{b^2-4c}}{2}$, kde

$$b = -2\cos a \times (x_j - x_i) - 2\sin a \times (y_j - y_i)$$

$$c = (x_j - x_i)^2 + (y_j - y_i)^2 - (R_i + R_j)^2$$

Příklad: dejme tomu, že konflikt nastal v čase $t = 4$ (obrázek 2.8).



Obrázek 2.8: Detekce konfliktu

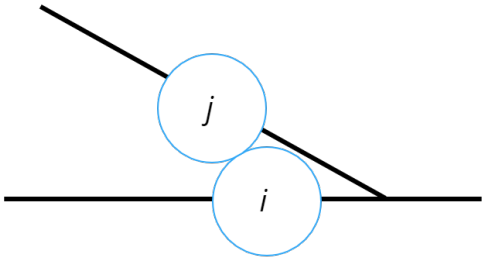
- Agent i se pohyboval mezi vrcholy v_0 a v_1, z v_0 odjel v čase 0, v_0 se nachází na (0,0) a v_1 – na (5,0). Agent má rychlost 1 a poloměr 0.5.
- Agent j se nacházel mezi vrcholy v_2, souřadnice jsou (1,3), a v_1, z v_2 odjel také v čase 0. Poloměr je 0.5 a rychlost se rovná 1.

Potom:

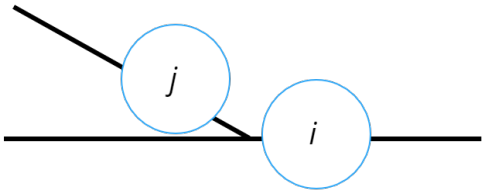
1. $(x_j, y_j) = [(1,3) \times (5 - \frac{4-0}{1}) + (5,0) \times \frac{4-0}{1}] \times 5^{-1} = (\frac{21}{5}, \frac{3}{5})$
2. $(x_i, y_i) = [(0,0) \times (5 - \frac{4-0}{1}) + (5,0) \times \frac{4-0}{1}] \times 5^{-1} = (4,0)$

3. $a = \text{atan2}(0-3, 5-1) = -0.643$
4. $b = -2\cos(-0.643) \times (4.2-4) - 2\sin(-0.643) \times (0.6-0) = 0.4$
5. $c = (4.2-4)^2 + (0.6-0)^2 - (0.5+0.5)^2 = -0.6$
6. konečně $t' = \frac{-0.4+\sqrt{0.4^2+4\times 0.6}}{2} = 0.581$

Nakonec konflikt nenastane, pokud agent j počká čas 0.581 ve vrcholu v_2 (obrázky 2.9, 2.10).



Obrázek 2.9: Po čekání



Obrázek 2.10: Výsledek

2.3 Lineární program č. 1: Hranový model

Použitý model byl inspirován problémem toku v síti a vychází z předpokladu, že počty vstupů a výstupů z vrcholů se musí rovnat, avšak z počátečního vrcholu vede jeden výstup navíc a do koncového – jeden dodatečný vstup. Program zvládne nalézt nejkratší cestu pro jednoho agenta s vyhnutím se všem potenciálním hranovým konfliktům.

Definice 4: hranový model je lineární program, jehož optimalizační funkcí je minimalizace délky cesty a proměnné jsou orientované hrany $e_{ij} \in N_+$ (hrana z vrcholu i do vrcholu j), hodnota kterých udává, kolikrát agent použil tuto hranu ve své cestě, a orientované hrany $\delta_{ij} \in R_+$, jejichž hodnota je délka hrany z i do j . Na model jsou kladena následující omezení, kde N je množina vrcholů v mapě:

- $\sum_k^N e_{ki} = \sum_k^N e_{ik}$, pokud i není počáteční ani koncový vrchol
- $\sum_k^N e_{ki} - \sum_k^N e_{ik} = -1$, pokud i je počáteční vrchol
- $\sum_k^N e_{ki} - \sum_k^N e_{ik} = 1$, pokud i je koncový vrchol

Daný model také řeší hranové konflikty a to takovým způsobem, že při konfliktu na hraně e_{ij} budou zvoleny sousední vrcholy s vrcholem i kromě vrcholu j a vrcholu předcházejícímu i v této cestě. Ať tyto vrcholy tvoří množinu M, potom jsou přidána další omezení:

- $e_{ij} = 0$, pokud $M \in \varnothing$
- $\sum_k^M e_{ik} = 1$, jinak

Jako poslední omezení je pro každou hranu spočítána její délka, která je následně vynásobena počtem průchodů touto hranou: $\delta_{ij} = e_{ij} \times d_{ij}$, kde $d_{ij} \in R_+$ je Euklidova vzdálenost mezi vrcholy i a j.

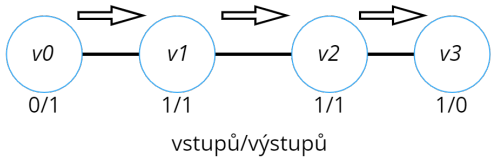
Tvrzení: přípustné řešení je nejkratší spojitá cesta.
Důkaz:

- $\sum_k^N e_{ki} = \sum_k^N e_{ik}$ znamená, že počet vstupů do vrcholu i se musí rovnat počtu výstupů z něho
- $\sum_k^N e_{ki} - \sum_k^N e_{ik} = -1$, pokud i je počáteční vrchol, jelikož má o 1 vstup do startovního vrcholu méně
- $\sum_k^N e_{ki} - \sum_k^N e_{ik} = 1$, pokud i je koncový vrchol, protože má o 1 výstup z cílového vrcholu méně

Výsledná cesta je řešením, jelikož se jedná o spojitou cestu začínající se počátečním vrcholem a končící se cílovým. Délka cesty je minimální kvůli optimalizační funkci, jinak by porušovala minimalizační podmínku.

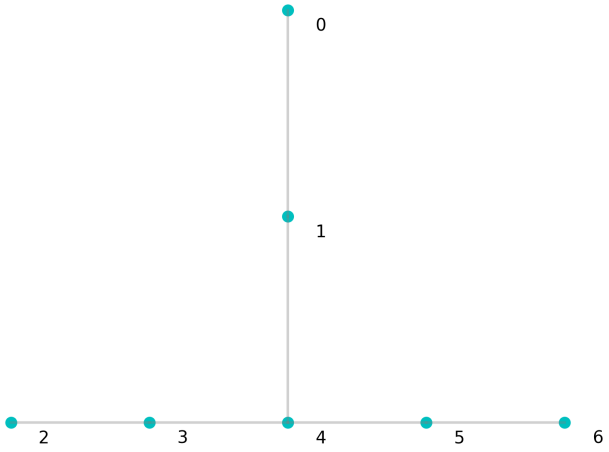
Řešení je cesta. Řešení je posloupnost hran $(e_1, e_2, ..., e_{n-1})$ pro kterou existuje posloupnost vrcholů $(v_1, v_2, ..., v_n)$ a platí $\forall i \in 1, 2, .., n-1 : e_i = (v_i, v_{i+1})$ a v_{i+1} je následníkem v_i, což je definici cesty.

Tvrzení: proměnné e_{ij} jsou celočíselné.
Důkaz: ať se cesta skládá ze čtyřech vrcholů: v_0 (startovní), v_1, v_2 a v_3 (cílový). Potom $e_{v_0v_1} = 1$, $e_{v_1v_0} = 0$, $e_{v_2v_3} = 1$ a $e_{v_3v_2} = 0$ což odpovídá podmínkám pro počáteční a koncový vrchol. Zbývají hrany $e_{v_1v_2}$ a $e_{v_2v_1}$. Jelikož musí platit první podmínka modelu, $e_{v_1v_2}$ se rovná $e_{v_2v_1}$ a navíc se rovná $e_{v_0v_1}$ a $e_{v_2v_3}$, jinak by došlo k porušení omezení $\sum_k^N e_{ki} = \sum_k^N e_{ik}$ (obrázek 2.11). Důkaz obdobně platí i pro delší cestu.



Obrázek 2.11: Důkaz

Příklad: pro mapu z obrázku 2.12 byl vytvořen scénář se dvěma agenty: první se pohybuje z vrcholu 2 do vrcholu 6, druhý – z 6 do 2. Každý má jednotkovou rychlost a poloměr 0.4.



Obrázek 2.12: Příklad: mapa

Ze začátku pro každého agenta byla vypočítána nejkratší cesta bez ohledu na kolize, které jsou znázorněny v tabulce 2.1. Následně je v tabulce 2.2 zobrazeno optimální řešení po vyhnutím se všem konfliktům.

agent 0		agent 1	
hrana	počet návštěv	hrana	počet návštěv
e_{23}	1	e_{65}	1
e_{34}	1	e_{54}	1
e_{45}	1	e_{43}	1
e_{56}	1	e_{32}	1

Tabulka 2.1: Ohodnocení proměnných: počáteční cesta

agent 0		agent 1	
hrana	počet návštěv	hrana	počet návštěv
e_{01}	1	e_{65}	1
e_{12}	1	e_{54}	1
e_{23}	1	e_{41}	1
e_{34}	1	e_{14}	1
		e_{43}	1
		e_{32}	1

Tabulka 2.2: Ohodnocení proměnných: řešení

2.4 Lineární program č. 2: Model se spojitým časem

Další model je o něco náročnější: využívá předchozí a rozšiřuje je o časovou proměnnou.

Definice 5: model se spojitým časem je lineární program s optimalizační funkcí, která minimalizuje dobu trvání cesty. Proměnné jsou opět orientované hrany $\tau_{ij} \in R_+$, jejich hodnotou je avšak čas výstupu z vrcholu j. Omezení jsou, pokud hrana $e_{ij} > 0$, tj. ve hranovém modelu táto hrana má kladnou hodnotu, což znamená, že daná hrana je použitá v cestě agenta:

- $\tau_{ij} \geq \frac{d_{ij}}{s}$, pokud i je počáteční vrchol
- $\tau_{ij} \geq \tau_{ki} + \frac{d_{ij}}{s}$ pro $k \in N$ jinak

Vzdálenost $d_{ij} \in R_+$ se počítá jako Euklidova vzdálenost mezi vrcholy i a j, rychlost s je rychlostí pohybu agenta.

Pokud dojde k vrcholovému konfliktu v čase $t \in R_+$ a vrcholu j, bude přidáno další omezení pro předchozí vrchol i: $\tau_{ij} \geq t + w$, kde $w \in R_+$ je čas, který agent musí vyčkat ve vrcholu i, aby konflikt byl vyloučen.

Další omezení se týká opakované návštěvy vrcholu. Jelikož hranový model vrací jenom počet použití hrany, resp. počet vstupů a výstupů z vrcholu, není možné z ohodnocení hrany sestavit posloupnost těchto kroků. Při opakované návštěvě vrcholu k se vytvoří dvě množiny: k_{in} – hrany vstupující do k, a k_{out} – hrany vystupující z k. Z těchto množin bude poté zvolena právě jedna kombinace vstupů-výstupů. Počet takových kombinací je $|k_{in}|!$. Z hranového modelu vyplývá, že $|k_{in}| = |k_{out}|$, pokud k není počáteční ani cílový vrchol.

Pseudokód je popsán v algoritmu 2.1. Tento případ je rozebrán dále v textu na konkretním příkladu.

Při vícenásobné návštěvě startovního vrcholu, postup se nepatrně liší. Množina k_{out} bude mít o jednu hranu více. Proto je potřeba projít všemi hranami z k_{out}, předpokládat, že hrana, která je na řadě, je počáteční a z ostat-

Algoritmus 2.1 Opakovaná návštěva vrcholu k

$k_{in} \leftarrow e_{*k}$
$k_{out} \leftarrow e_{k*}$
$possibilities \leftarrow$Combine(k_{in}, k_{out})
$y \leftarrow [[\,] \times |possibilities|]$
$i \leftarrow 0$
for $p \in possibilities$ **do**
  if $y[i_{++}] = 1$ **then**
    AddConstraint(p)
  end if
end for
AddConstraint$(\sum y = 1)$

ních vytvářet kombinace. Potom počet možností je $|k_{in}|! \times |k_{out}|$. Pseudokód je uveden v algoritmu 2.2.

Algoritmus pro návrat do cílového vrcholu je podobný: $|k_{in}| - |k_{out}| = 1$ a tyto dvě množiny se musí prohodit. Místo podmínky, že zkoumána hrana je počáteční, bude podmínka, že hodnota této hrany je maximální, což zaručí, že bude poslední v cestě. Algoritmus je předveden níže.

Algoritmus 2.2 Opakovaná návštěva startovního vrcholu k

$k_{in} \leftarrow e_{*k}$
$k_{out} \leftarrow e_{k*}$
$y \leftarrow [[\,] \times (|k_{in}|! \times |k_{out}|)]$
$i \leftarrow 0$
for $x \in k_{out}$ **do**
  $k_{out'} \leftarrow k_{out} \setminus \{x\}$
  $possibilities \leftarrow$Combine$(k_{in}, k_{out'})$
  for $p \in possibilities$ **do**
    if $y[i_{++}] = 1$ **then**
      AddConstraint$(start = x)$
      AddConstraint(p)
    end if
  end for
end for
AddConstraint$(\sum y = 1)$

Tvrzení: přípustné řešení je nejkratší cesta se spojitým časem a vyhnutím se všem vrcholovým konfliktům.

Důkaz: omezení $\tau_{ij} \geq \tau_{ki} + \frac{d_{ij}}{s}$ říká, že čas vstupu do vrcholu j přes vrchol i se rovná času vstupu do i a času přechodu mezi těmito vrcholy. Jelikož optimalizační funkcí je minimalizace doby trvání cesty, k nerovnosti dochází

Algoritmus 2.3 Opakovaná návštěva cílového vrcholu k

$k_{in} \leftarrow e_{*k}$
$k_{out} \leftarrow e_{k*}$
$y \leftarrow [[\,] \times (|k_{out}|! \times |k_{in}|)]$
$i \leftarrow 0$
for $x \in k_{in}$ **do**
  $k_{in'} \leftarrow k_{in} \setminus \{x\}$
  $possibilities \leftarrow$Combine$(k_{in'}, k_{out})$
  for $p \in possibilities$ **do**
    if $y[i_{++}] = 1$ **then**
      AddConstraint$(x \geq p)$
      AddConstraint(p)
    end if
  end for
end for
AddConstraint$(\sum y = 1)$

pouze při vrcholovém konfliktu, když agent musí počkat nějaký čas navíc.

Řešení je cesta, protože využívá cestu z hranového modelu, žádným způsobem ji neupravuje a jenom přiřazuje hranám časovou složku.

Cesta je řešením: optimalizační funkce daného modelu zajišťuje nejkratší dobu čekání při vrcholovém konfliktu a díky omezením jsou hrany ve správném pořadí.

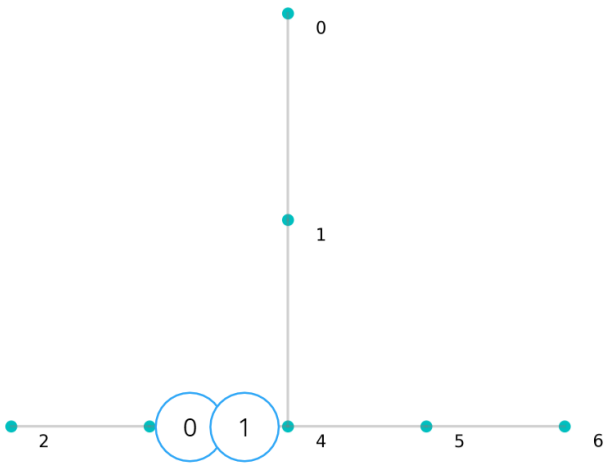
Příklad: pro ukázku práce tohoto modelu byl použit stejný scénář a mapa 2.12.

V tabulce 2.3 je ohodnocení počáteční cesty a lze z toho pozorovat vrcholový konflikt v čase 2 ve vrcholu 4, proto jeden z agentů (v tomto případě agent 0) počká nějaký čas v předchozím vrcholu, ve výsledku ale dojde ke hranovému konfliktu na hraně z vrcholu 3 do vrcholu 4 (obrázek 2.13).

agent 0		agent 1	
hrana	čas výstupu z hrany e_{ij}	hrana	čas výstupu z hrany e_{ij}
e_{23}	1	e_{65}	1
e_{34}	2	e_{54}	2
e_{45}	3	e_{43}	3
e_{56}	4	e_{32}	4

Tabulka 2.3: Ohodnocení proměnných: počáteční cesta

Agent 1 se musí vyhnout danému konfliktu a odbočit z vrcholu 4. Jediná možnost je vrchol 1. Návrat z vrcholu 1 způsobí opakovanou návštěvu vrcholu 4. V tomto okamžiku model ví, jak má ohodnotit hrany e_{65} (6 je počáteční



Obrázek 2.13: Příklad: hranový konflikt

vrchol, takže se rovná vzdálenosti z 6 do 5) a e_{54} (následuje po hraně e_{65}), ale neví, která hrana je další: e_{41} nebo e_{43}. V tomto případě množina k_{in} se rovná [e_{54}, e_{14}] a množina k_{out} – [e_{41}, e_{43}]. Z těchto množin se vytvoří dvě možné posloupností:

$$\tau_{43} \geq \tau_{54} + d_{43}$$
$$\tau_{41} \geq \tau_{14} + d_{41}$$

a

$$\tau_{41} \geq \tau_{54} + d_{41}$$
$$\tau_{43} \geq \tau_{14} + d_{43}.$$

První nedává spojitou cestu, jelikož do e_{14} se dá dostat jenom přes e_{41} a tato kombinace říká, že e_{41} následuje po e_{14}. Momentálně model ví, že správná posloupnost je druhá. Výsledek je v tabulce 2.4.

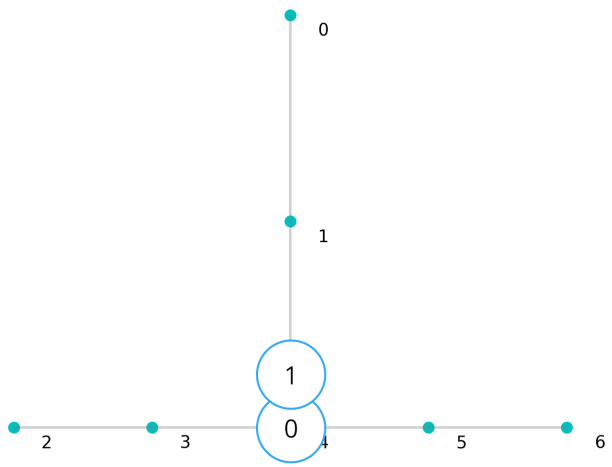
Nakonec, jakmile vrcholové a hranové konflikty jsou vyřešeny, je zkontrolováno, zda nedošlo ke konfliktům mimo společné úseky cest. V čase 2.8, kdy agent 0 byl ve vrcholu 4 a agent 1 se pohyboval hranou e_{41}, nastala srážka. Agent 0 proto počkal navíc ve vrcholu 3 (tabulka 2.5, obrázek 2.14).

Poslední konflikt nastal v čase 4: agent 0 se pohyboval hranou e_{34}, agent 1 se nacházel ve vrcholu 4. V tomto případě agent 1 musel počkat nějaký čas ve vrcholu 1.

Po vyřešení všech konfliktů bylo spočítáno optimální řešení (tabulka 2.6) a z toho byla zrekonstruována bezkolizní cesta pro každého agenta (tabulka 2.7).

agent 0		agent 1	
hrana	čas výstupu z hrany e_{ij}	hrana	čas výstupu z hrany e_{ij}
e_{23}	1.8	e_{65}	1
e_{34}	2.8	e_{54}	2
e_{45}	3.8	e_{41}	3
e_{56}	4.8	e_{14}	4
		e_{43}	5
		e_{32}	6

Tabulka 2.4: Ohodnocení proměnných: po vyřešení vrcholového a hranového konfliktu



Obrázek 2.14: Příklad: překryvný konflikt

agent 0		agent 1	
hrana	čas výstupu z hrany e_{ij}	hrana	čas výstupu z hrany e_{ij}
e_{23}	2.125	e_{65}	1
e_{34}	3.125	e_{54}	2
e_{45}	4.125	e_{41}	3
e_{56}	5.125	e_{14}	4
		e_{43}	5
		e_{32}	6

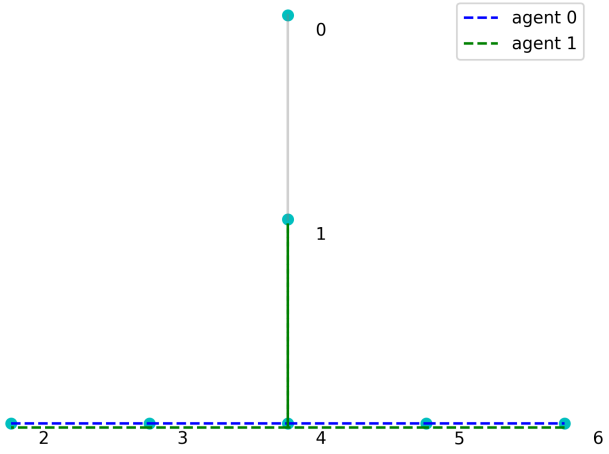
Tabulka 2.5: Ohodnocení proměnných: po vyřešení překryvného konfliktu

agent 0		agent 1	
hrana	čas výstupu z hrany e_{ij}	hrana	čas výstupu z hrany e_{ij}
e_{23}	2.125	e_{65}	1
e_{34}	3.125	e_{54}	2
e_{45}	4.125	e_{41}	3.206
e_{56}	5.125	e_{14}	4.206
		e_{43}	5.206
		e_{32}	6.206

Tabulka 2.6: Ohodnocení proměnných: řešení

agent 0			agent 1		
vrchol	čas příjezdu	čas odjezdu	vrchol	čas příjezdu	čas odjezdu
2	–	0	6	–	0
3	1	2.125	5	1	1
4	3.125	3.125	4	2	2
5	4.125	4.125	1	3	3.206
6	5.125	–	4	4.206	4.206
			3	5.206	5.206
			2	6.206	–

Tabulka 2.7: Výsledek



Obrázek 2.15: Příklad: řešení

Kapitola 3

Experimentální vyhodnocení

Cílem teto kapitoly je představit vyvinutý program na konkretních příkladech, detailněji popsat několik z nich, porovnat s jiným algoritmem pro stejný problém a na závěr – vyhodnotit dobu běhu algoritmu pro různý počet agentů a různé propojení map.

3.1 Gurobi implementace

Algoritmus se skládá ze dvou částí: první odpovídá horní vrstvě algoritmu CCBS, ve které se postaví binární strom se získanými omezeními, druhá část řeší hledání cesty pomocí lineárních programů popsaných v 2. kapitole.

Modely představené v předchozí části byly implementovány pomocí Gurobi optimalizátora.

Většina akcí v Gurobi rozhraní se provádí voláním metod na Gurobi objekty. Nejčastěji používaným objektem je Model. Gurobi model se skládá ze tří částí:

- **Objective** [22]: každý optimalizační model má za cíl minimalizovat nebo maximalizovat určitou funkci a při větším počtu řešení právě tato funkce říká, které řešení je nejlepší. Tato funkce může být lineární nebo kvadratická.
- **Variables** [23]: rozhodovací proměnné, ve kterých se nachází výsledek optimalizace. Pro optimální řešení platí, že tyto proměnné musí splňovat všechna omezení. Každá proměnná má dolní mez, horní mez a typ (spojitá, binární atd.).
- **Constraints** [24]: omezení, která musí splňovat proměnné. V modelech jsou použitá pouze lineární omezení (rovnost a neostrá nerovnost).

Líná kompilace se projevuje línými omezeními. Takovými jsou omezení pro hranový konflikt při předjíždění a vrcholový a překryvný konflikty, jinými

slovy ty, u kterých dochází k čekání. To znamená, že se nejdříve najde cesta bez zapojení těchto omezení a v případě, že nalezená cesta poruší nějaké z nich, přidá se toto omezení do aktivních a zahájí se hledání nového řešení.

3.2 Experimenty

Všechny mapy jsou inspirovány nebo poskytnuty z webu *movingai.com* [25]. Každý scénář je otestován na mapě s propojením podle 2^k Neighborhoods, kde k je 3 nebo 4.

Dále budou rozebrány 3 scénáře pro každou mapu. Scénáře byly zvoleny z náhodných spouštění při počtu agentů 5 a k=3 a byly otestovány i na k=4. Primárním cílem byl výběr zajímavých ukázek, kde došlo ke konfliktům.

Každý scénář obsahuje popis vzniklých konfliktů, postup jejich řešení a obrázek s výslednými cestami.

V experimentech se počítá doba běhu algoritmu pro příslušný scénář, počet volání, průměrný a celkový počet vrcholů v cestách, průměrný počet proměnných a omezení pro oba modely.

Parametry agenta se generovaly náhodně:

- počáteční a koncový vrcholy – z množiny všech vrcholů s podmínkou, že se nesmí opakovat ani se rovnat
- poloměr – v rozmezí necelých čísel 1 až 5
- rychlost – z množiny celých čísel 1 až 10

Vyvinutý algoritmus byl následně porovnán s algoritmem pro hledání multi-agentních cest založeným na SMT-CBS [13] (pseudokód 1.3).

3.3 Mapa č. 1

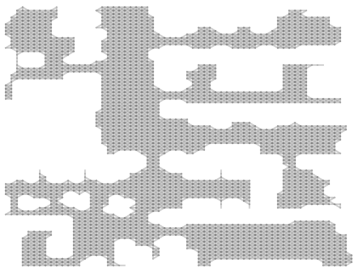
První mapa odpovídá mapě den312d.

Parametry mapy s k=3 (obrázek 3.1):

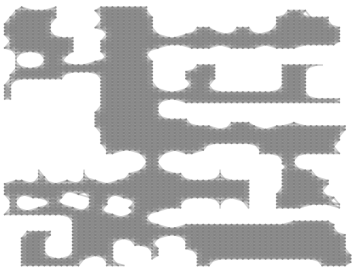
- počet vrcholů je 2442
- počet hran je 16910

Parametry mapy s k=4 (obrázek 3.2):

- počet vrcholů je 2442
- počet hran je 31772



Obrázek 3.1: Mapa, $k = 3$

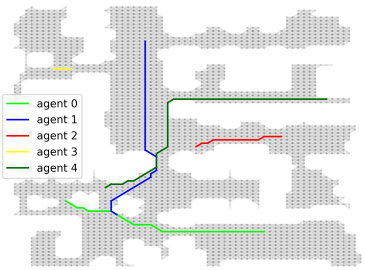


Obrázek 3.2: Mapa, $k = 4$

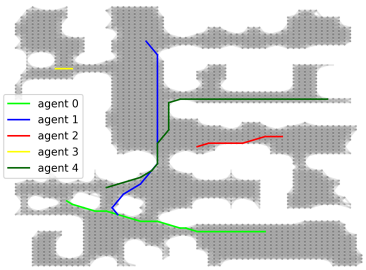
3.3.1 Scénář č. 1

První scénář se z hlediska konfliktů výrazně neliší pro propojení k=3 a k=4: při prvotním vybudování cest jsou detekovány dva konflikty mezi agenty 1 a 4 ve vrcholech jdoucích po sobě a 7 překryvných. Čekání agenta 4 v prvním z problematických vrcholů eliminuje všechny další srážky. U k=4 byl detekován pouze jeden vrcholový konflikt a 9 překryvných ve stejných místech.

Řešení tohoto scénáře lze prohlédnout na obrázcích 3.3, 3.4.



Obrázek 3.3: Řešení, $k = 3$



Obrázek 3.4: Řešení, $k = 4$

Z tabulky 3.1 plyne, že cesty při k=4 jsou kratší, avšak doba běhu algoritmu je naopak delší. Doba běhu je závislá především na počtu hran a konfliktů, resp. počtu proměnných a omezení v obou modelech. Z tabulky je také kromě toho patrné, že počet proměnných hranového modelu se rovná dvojnásobnému počtu hran a počet omezení – součtu počtů hran a vrcholů. Další rozdíl spočívá v parametrech časového modelu: kratší cesty produkují menší

počet proměnných a omezení v časovém modelu.

Statistika	k=3	k=4
Doba běhu (s)	50.669	121.293
Počet volání	48	95
Počet konfliktů	9	10
Průměrný počet vrcholů v cestách	33	27.8
Celkový počet vrcholů v cestách	165	139
Průměrný počet proměnných v hranovém modelu	33820	63544
Průměrný počet omezení v hranovém modelu	19352	34214
Průměrný počet proměnných v časovém modelu	52.31	47
Průměrný počet omezení v časovém modelu	53.9	49.08

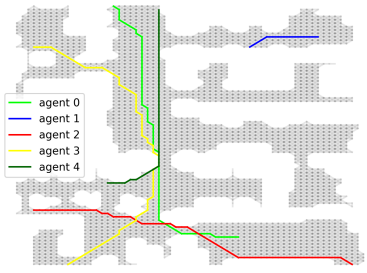
Tabulka 3.1: Statistika, mapa č. 1, scénář č. 1

Algoritmus SMT-CBS nalezl stejné počáteční cesty pro k=3, ale selhal při hledání řešení kolizí. Pro k=4 nalezl horší počáteční cesty a také selhal.

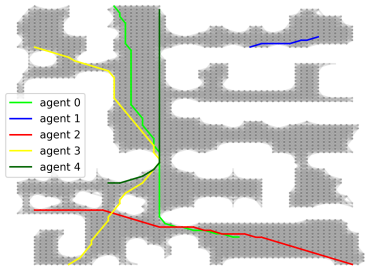
3.3.2 Scénář č. 2

V tomto scénáři dochází k několika překryvným konfliktům: při k=3 (obrázek 3.5) se agent 3 sráží s agentem 4, při k=4 (obrázek 3.6) srážka nastává také mezi agenty 3 a 4, ale později a ve dvou různých částech mapy.

V prvním případě jsou konflikty vyřešeny za podmínky, že agent 3 počká potřebný čas ve vrcholu před srážkami. V druhé situaci musel čekat nejdříve agent 3, následně při srážce v jiném místě počkal agent 4.



Obrázek 3.5: Řešení, $k = 3$



Obrázek 3.6: Řešení, $k = 4$

V tabulce 3.2 jsou představeny výsledky výpočtu. Při větším propojení mapy algoritmus běžel déle, nicméně na dobu běhu měl vliv i počet konfliktů.

Algoritmus SMT-CBS překročil nastavený časový limit, počáteční cesty byly delší.

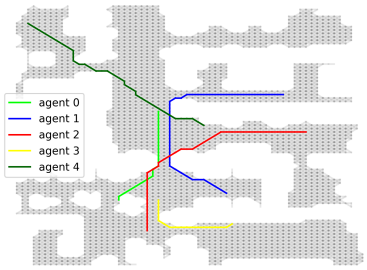
Statistika	k=3	k=4
Doba běhu (s)	52.971	78.731
Počet volání	52	58
Počet konfliktů	18	26
Průměrný počet vrcholů v cestách	55	42.8
Celkový počet vrcholů v cestách	275	214
Průměrný počet proměnných v hranovém modelu	33820	63544
Průměrný počet omezení v hranovém modelu	19352	34214
Průměrný počet proměnných v časovém modelu	62.83	48.4
Průměrný počet omezení v časovém modelu	66.04	54.36

Tabulka 3.2: Statistika, mapa č. 1, scénář č. 2

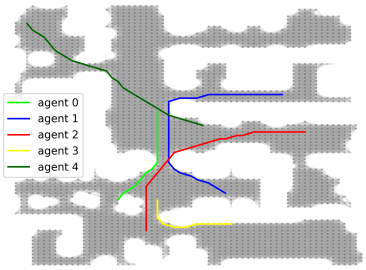
3.3.3 Scénář č. 3

Další scénář zahrnuje hranový konflikt. Při k=3 (obrázek 3.7) mezi agenty 0 a 2 nastal hranový konflikt, když se agenti pohybovali různými směry. Řešením bylo, aby agent 2 odbočil do sousedního vrcholu.

V mapě s propojením k=4 (obrázek 3.8) ke konfliktům nedošlo.



Obrázek 3.7: Řešení, $k = 3$



Obrázek 3.8: Řešení, $k = 4$

Tabulka 3.3 ukazuje, že se změnil počet omezení v hranovém modelu oproti předchozím scénářům – byla tam přidána omezení pro hranový konflikt. Také je vidět závislost parametrů časového modelu na počtu omezení a délce cesty.

SMT-CBS algoritmus nalezl horší počáteční cesty a selhal.

3.3.4 Závěr

Konflikty byly detekovány v 32.7% případů.

Na závěr byla prozkoumána závislost doby běhu algoritmu a počtu volání na počtu agentů a typu propojení mapy. Každá kombinace byla spouštěna

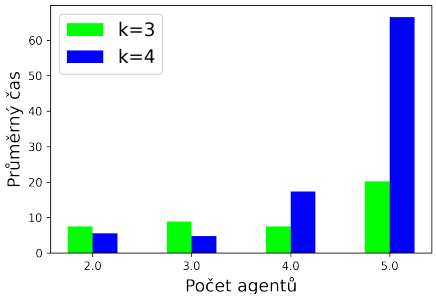
Statistika	k=3	k=4
Doba běhu (s)	8.367	6.33
Počet volání	7	5
Počet konfliktů	1	0
Průměrný počet vrcholů v cestách	36.2	29.4
Celkový počet vrcholů v cestách	181	147
Průměrný počet proměnných v hranovém modelu	33820	63544
Průměrný počet omezení v hranovém modelu	19352.01	34214
Průměrný počet proměnných v časovém modelu	44.86	29.4
Průměrný počet omezení v časovém modelu	48.16	28.4

Tabulka 3.3: Statistika, mapa č. 1, scénář č. 3

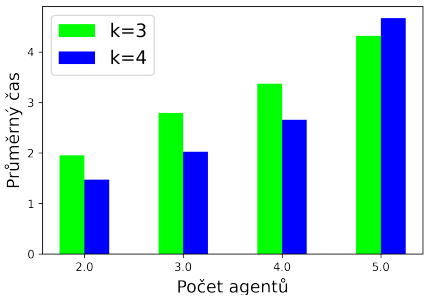
15-krát na náhodných agentech a výsledek se zprůměroval.

Graf 3.9 předvádí průměrný čas běhu pro instance, kde došlo k jakémukoliv konfliktu. Značný rozdíl nastal při k=4 a 5 agentech. To je způsobeno větším počtem hran v mapě a častými překryvnými konflikty mezi agenty.

Na grafu 3.10 jsou potom znázorněny scénáře bez konfliktů. Ten čas je stráven konstrukcí počátečních cest a ověřováním, zda obsahují srážky. Propojení k=4 je ve výsledku rychlejší, protože cesty mají menší počet vrcholů a kvůli tomuto i kontrola trvá méně.



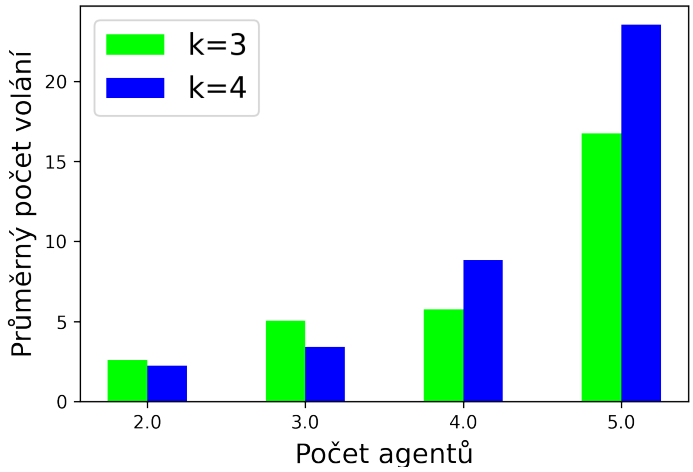
Obrázek 3.9: S konflikty



Obrázek 3.10: Bez konfliktů

Další graf 3.11 ukazuje počet volání lineárního programu. Tento graf se nerozlišuje na konfliktní a bezkonfliktní scénáře, jelikož bezkonfliktní mají vždy počet volání rovný počtu agentů.

Při malém počtu agentů (2 a 3) se téměř nedocházelo ke konfliktům, naopak při 4 a 5 agentech se srážky detekovaly častěji.



Obrázek 3.11: Počet volání

3.4 Mapa č. 2

Druhá mapa vychází z první a je upravena takovým způsobem, že přechody mezi ostrůvky jsou tenčí. Motivací bylo vyvolat více hranových konfliktů.

Parametry mapy s k=3 (obrázek 3.12):

- počet vrcholů je 2315
- počet hran je 15578

Parametry mapy s k=4 (obrázek 3.13):

- počet vrcholů je 2315
- počet hran je 29194

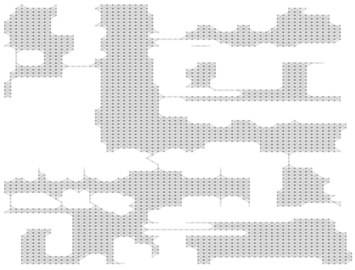
3.4.1 Scénář č. 1

V dánem scénáři v obou verzích mapy (obrázky 3.14, 3.15) nastal hranový konflikt mezi agenty 2 a 4 a překryvný mezi agenty 1 a 4 ve stejných místech.

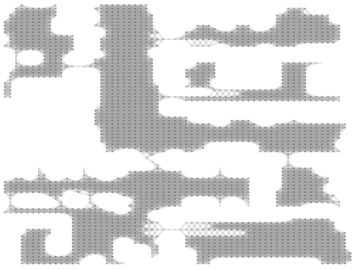
Hranový konflikt vyřešil agent 2 a překryvný byl za agentem 1.

V tabulce 3.4 jsou představeny výsledky výše uvedeného scénáře. Větší počet volání u k=4 souvisí s tím, že bylo nalezeno více možných cest: některé nesplňovaly omezení a jiné byly horší z hlediska optimalizační funkce.

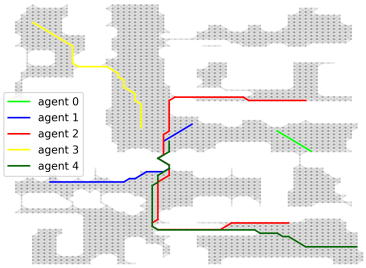
SMT-CBS algoritmus nalezl stejné počáteční cesty pro k=3, ale horší pro k=4, následně selhal pro obě propojení.



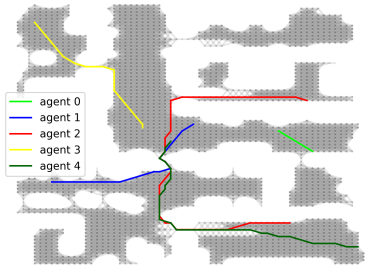
Obrázek 3.12: Mapa, $k = 3$



Obrázek 3.13: Mapa, $k = 4$



Obrázek 3.14: Řešení, $k = 3$



Obrázek 3.15: Řešení, $k = 4$

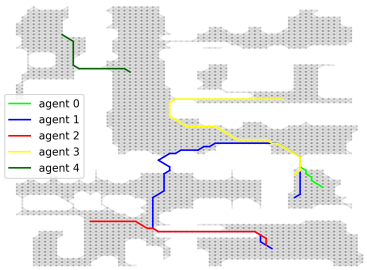
Statistika	k=3	k=4
Doba běhu (s)	7.339	13.2
Počet volání	16	11
Počet konfliktů	2	2
Průměrný počet vrcholů v cestách	45.4	35.8
Celkový počet vrcholů v cestách	227	179
Průměrný počet proměnných v hranovém modelu	31156	58388
Průměrný počet omezení v hranovém modelu	17893.31	31509.36
Průměrný počet proměnných v časovém modelu	56.25	43.18
Průměrný počet omezení v časovém modelu	55.69	42.36

Tabulka 3.4: Statistika, mapa č. 2, scénář č. 1

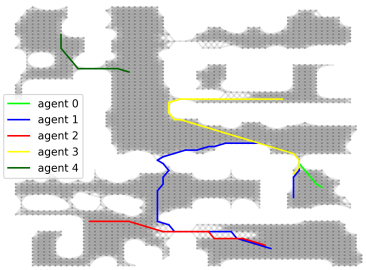
3.4.2 Scénář č. 2

V druhém scénáři při k=3 došlo k 8 překryvným konfliktům mezi agenty 0 a 1 v jejich společném úseku cest. Agent 0 počkal dvakrát ve vrcholech před společnou části a tímto byly konflikty vyřešeny (obrázek 3.16). Při k=4 se agenti 0 a 1 srazili pouze sedmkrát ve stejné části mapy.

Řešení bylo podobné: agent 0 počkal jenom jednou (obrázek 3.17).



Obrázek 3.16: Řešení, $k = 3$



Obrázek 3.17: Řešení, $k = 4$

V tabulce 3.5 jsou znázorněny výsledky zkoumaného scénáře. Častější volání algoritmu je způsobeno jedním konfliktem navíc.

Statistika	k=3	k=4
Doba běhu (s)	19.182	21.146
Počet volání	29	18
Počet konfliktů	8	7
Průměrný počet vrcholů v cestách	39.4	29.6
Celkový počet vrcholů v cestách	197	132
Průměrný počet proměnných v hranovém modelu	31156	58388
Průměrný počet omezení v hranovém modelu	17893	31509
Průměrný počet proměnných v časovém modelu	39.07	30.56
Průměrný počet omezení v časovém modelu	40.38	31.44

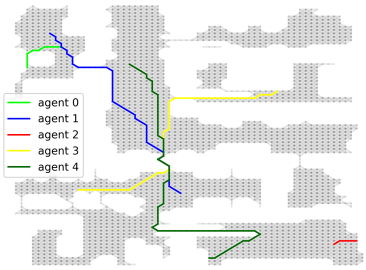
Tabulka 3.5: Statistika, mapa č. 2, scénář č. 2

Algoritmus SMT-CBS překročil nastavený časový limit, pro k=3 spočítal stejné počáteční cesty a pro k=4 – horší.

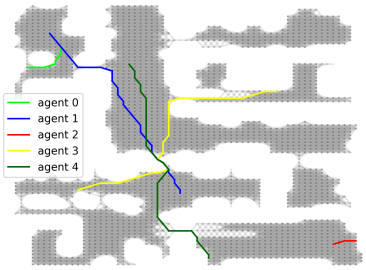
3.4.3 Scénář č. 3

Poslední scénář nemá hranové konflikty, nýbrž 11 překryvných pro k=3 a žádný konflikt pro k=4 (obrázky 3.18, 3.19).

Konflikty nastaly mezi agenty 3 a 4 v jednom vrcholu. Čekáním agenta 3 ve vrcholu před ním byly najednou vyřešeny. Také se srazili agenti 0 a 1. Agent 1 měl počkat hned ve druhém vrcholu své cesty.



Obrázek 3.18: Řešení, $k = 3$



Obrázek 3.19: Řešení, $k = 4$

Z tabulky 3.5 stojí za zmínku počet volání u k=3 – je výsledekem hledání nejlepšího řešení pro překryvné konflikty několika agentů, u k=4 – počet proměnných a omezení v časovém modelu se rovnají počtu vrcholů v cestě, protože tam nenastal ani vrcholový, ani překryvný konflikt.

Statistika	k=3	k=4
Doba běhu (s)	69.456	5.958
Počet volání	103	5
Počet konfliktů	11	0
Průměrný počet vrcholů v cestách	40.4	28.4
Celkový počet vrcholů v cestách	202	142
Průměrný počet proměnných v hranovém modelu	31156	58388
Průměrný počet omezení v hranovém modelu	17893	31509
Průměrný počet proměnných v časovém modelu	51.83	28.4
Průměrný počet omezení v časovém modelu	53.18	27.4

Tabulka 3.6: Statistika, mapa č. 2, scénář č. 3

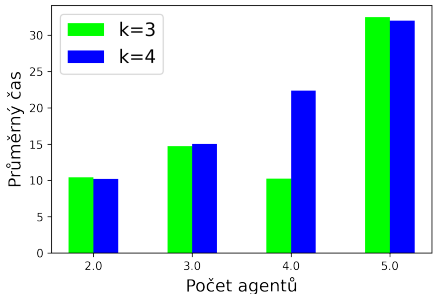
SMT-CBS vrátil stejné cesty pro k=3 a delší pro k=4, následně selhal.

3.4.4 Závěr

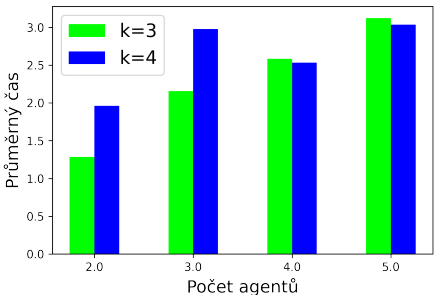
Srážky nastávaly v 41.7% scénářů.

Na grafu 3.20 jsou zobrazeny výsledky instancí s konflikty. Tenčí přechody mezi ostrůvky vyvolávaly častější srážky u k=3, když u k=4 tyto přechody byly rozšířeny kvůli lepšímu propojení.

Další graf 3.21 znázorňuje dobu běhu scénářů bez konfliktů: časy se rovnají bez ohledu na to, že cesty při k=4 jsou o něco kratší, trvá to déle je najít kvůli většímu počtu hran.

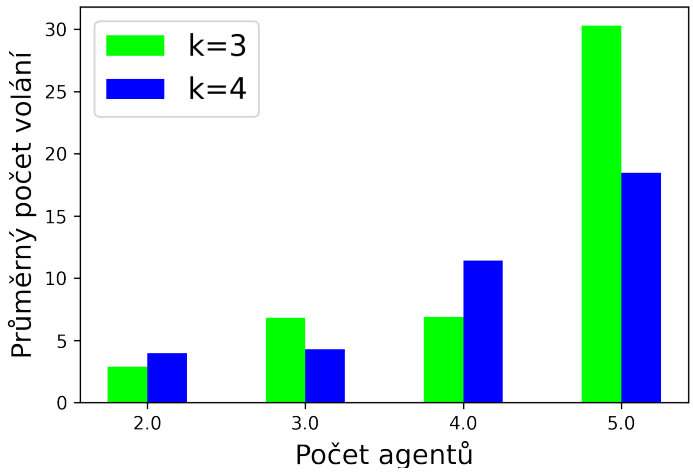


Obrázek 3.20: S konflikty



Obrázek 3.21: Bez konfliktů

Graf 3.22 ukazuje počet volání: čím více agentů se pohybuje mapou a čím tenčí jsou přechody, tím častěji vznikají srážky.



Obrázek 3.22: Počet volání

3.5 Mapa č. 3

Tato mapa je oříznutá část z mapy Berlín.

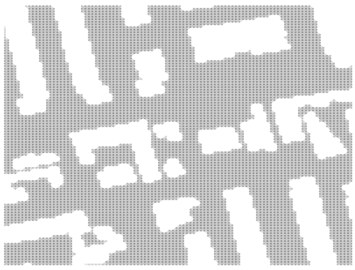
Parametry mapy s k=3 (obrázek 3.23):

- počet vrcholů je 6614

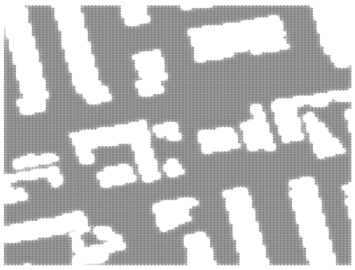
- počet hran je 48398

Parametry mapy s k=4 (obrázek 3.24):

- počet vrcholů je 6614
- počet hran je 93142



Obrázek 3.23: Mapa, $k = 3$



Obrázek 3.24: Mapa, $k = 4$

3.5.1 Scénář č. 1

V tomto scénáři při k=3 (obrázek 3.25) nastal nejdříve hranový konflikt mezi agenty 3 a 4, pak v následujícím vrcholu i překryvný. Při k=4 – pouze hranový (obrázek 3.26).

Oba případy byly vyřešeny tak, že agent 3 nalezl novou cestu, kde již ke srážce nedošlo.

V grafu 3.7 je vidět, že čas hledání alternativní cesty při hranovém konfliktu záleží na počtu hran vedoucích z konfliktního vrcholu, proto při k=4 výpočet trval dvakrát déle.

SMT-CBS spočítal stejné cesty pro k=3, delší pro k=4, poté selhal.

3.5.2 Scénář č. 2

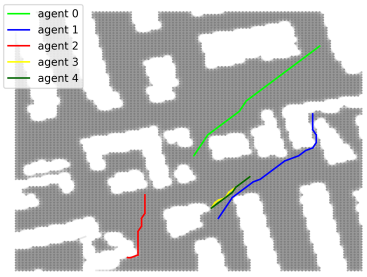
V dalším scénáři při k=3 byl detekován hranový konflikt mezi agenty 0 a 1, ale při k=4 tento konflikt již nenastal.

Pro vyřešení konfliktu agent 0 využil jinou hranu a srážka byla eliminována (obrázky 3.27, 3.28).

Z tabulky 3.8 je patrné, že výpočet řešení s delšími cestami a jedním konfliktem trvá méně než výpočet při větším počtu hran v mapě.

Algoritmus SMT-CBS nalezl stejné cesty při k=3, horší při k=4, poté selhal.

Obrázek 3.25: Řešení, $k = 3$

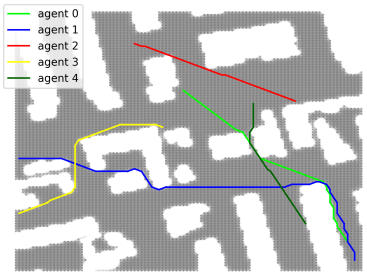


Obrázek 3.26: Řešení, $k = 4$

Statistika	k=3	k=4
Doba běhu (s)	21.942	43.719
Počet volání	13	12
Počet konfliktů	2	1
Průměrný počet vrcholů v cestách	28.4	23.8
Celkový počet vrcholů v cestách	142	119
Průměrný počet proměnných v hranovém modelu	96796	186284
Průměrný počet omezení v hranovém modelu	55012.38	99756.17
Průměrný počet proměnných v časovém modelu	19.08	20.75
Průměrný počet omezení v časovém modelu	18.54	20.17

Tabulka 3.7: Statistika, mapa č. 3, scénář č. 1

Obrázek 3.27: Řešení, $k = 3$



Obrázek 3.28: Řešení, $k = 4$

Statistika	k=3	k=4
Doba běhu (s)	7.214	16.996
Počet volání	7	5
Počet konfliktů	1	0
Průměrný počet vrcholů v cestách	68.6	49.2
Celkový počet vrcholů v cestách	343	246
Průměrný počet proměnných v hranovém modelu	96796	186284
Průměrný počet omezení v hranovém modelu	55012.29	99756
Průměrný počet proměnných v časovém modelu	76	49.2
Průměrný počet omezení v časovém modelu	75	48.2

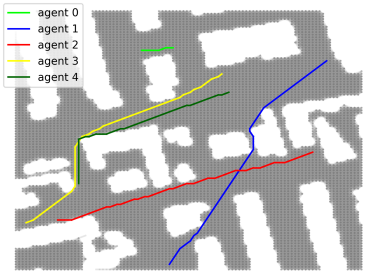
Tabulka 3.8: Statistika, mapa č. 3, scénář č. 2

3.5.3 Scénář č. 3

V následující situaci nastal hranový konflikt mezi agenty 3 a 4 při k=3 (obrázek 3.29), ale s propojením k=4 se agenti již vyhnuli (obrázek 3.30).

Pro nalezení bezkolizní cesty agent 3 využil sousední vrchol místo toho, kde došlo ke srážce.

Obrázek 3.29: Řešení, $k = 3$



Obrázek 3.30: Řešení, $k = 4$

Statistická data tohoto scénáře (tabulka 3.9) jsou podobná předchozímu: kontrola a konstrukce kratších bezkolizních cest trvá déle při větším počtu hran v mapě než při delších cestách s konflikty, ale řídkým propojením.

Algoritmus SMT-CBS zvládl scénář pro k=3, nalezl cesty, které trvají stejný čas, ale vedou přes jiné vrcholy. Nehledě na to, konflikty také nastaly mezi agenty 3 a 4. Řešení je zobrazeno na obrázku 3.31. Pro k=4 výpočet překročil časový limit a počáteční cesty byly delší.

Statistika	k=3	k=4
Doba běhu (s)	15.709	5.248
Počet volání	7	5
Počet konfliktů	1	0
Průměrný počet vrcholů v cestách	59.8	41.8
Celkový počet vrcholů v cestách	299	209
Průměrný počet proměnných v hranovém modelu	96796	186284
Průměrný počet omezení v hranovém modelu	55012.29	99756
Průměrný počet proměnných v časovém modelu	62.14	41.8
Průměrný počet omezení v časovém modelu	61.14	40.8

Tabulka 3.9: Statistika, mapa č. 3, scénář č. 3

Obrázek 3.31: SMT-CBS, $k = 3$

3.5.4 Závěr

Agenti se srazili pouze v 18.5% instancí.

Na grafu 3.32 jsou vidět časy běhu algoritmu, když nastaly konflikty: většinou to jsou jedinečné situace, jelikož polovina srážek se týká scénářů s pěti agenty.

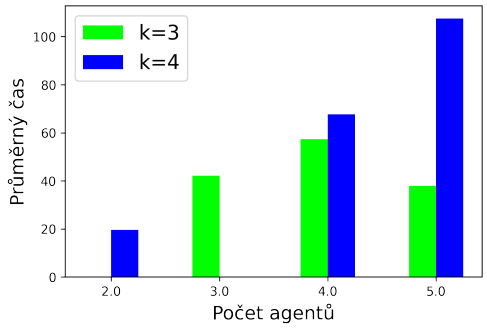
Na grafu 3.33 jsou pak časy bez srážek: při k=4 a počtu agentů 2 až 4 kratší cesty měly větší dopad na dobu běhu než čas strávený kontrolou, s pěti agenty to bylo naopak.

Poslední graf 3.34 zobrazuje počty volání algoritmu: z grafu lze pozorovat skutečnost, že při počtu agentů 2 a 3 konflikty téměř nenastávaly, mezi čtyřmi bylo menší množství srážek, a většina připadá na počet agentů 5.

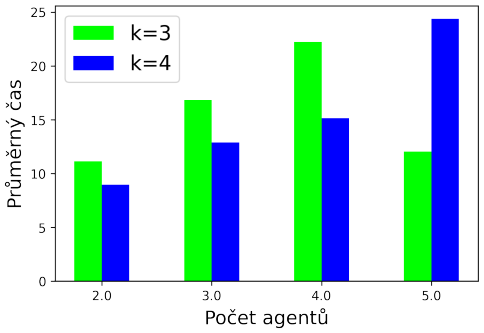
3.6 Mapa č. 4

Další mapa vychází z mapy kaple (chantry) a je o něco oříznuta shora a zdola.

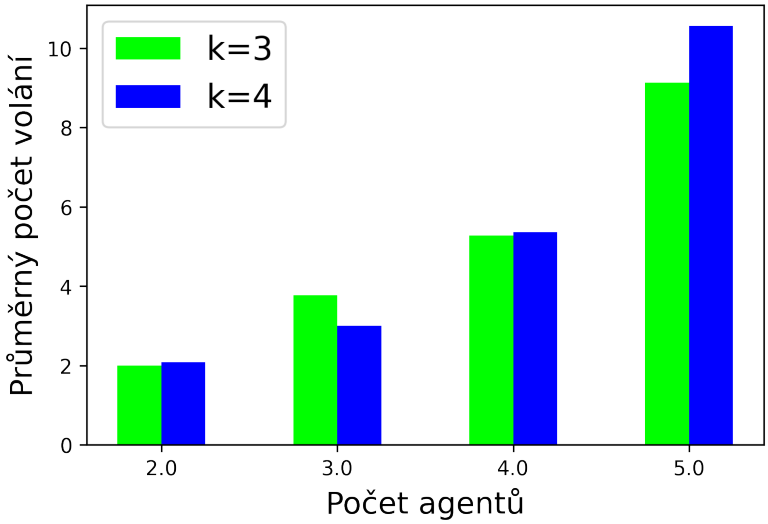
Parametry mapy s k=3 (obrázek 3.35):



Obrázek 3.32: S konflikty



Obrázek 3.33: Bez konfliktů

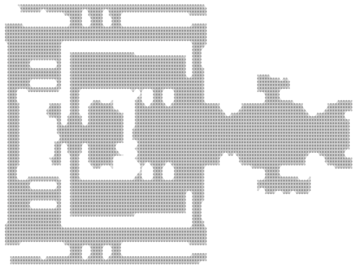


Obrázek 3.34: Počet volání

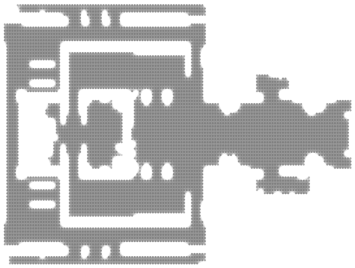
- počet vrcholů je 5675
- počet hran je 40772

Parametry mapy s k=4 (obrázek 3.36):

- počet vrcholů je 5675
- počet hran je 77416



Obrázek 3.35: Mapa, $k = 3$

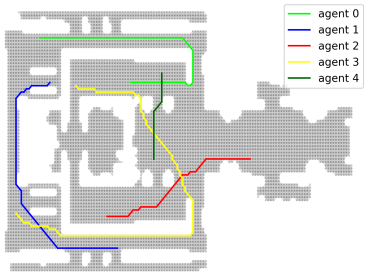


Obrázek 3.36: Mapa, $k = 4$

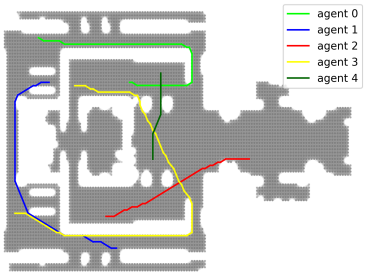
3.6.1 Scénář č. 1

V této situaci s okolím k=3 došlo k 15 překryvným konfliktům mezi agenty 1 a 3: 14 z nich bylo vyřešeno čekáním agenta 1 a jeden – agenta 3 (obrázek 3.37).

Při okolí k=4 bylo detekováno 10 překryvných konfliktů – opět mezi agenty 1 a 3: 9 bylo eliminováno agentem 1 a jeden konflikt – agentem 3 (obrázek 3.38).



Obrázek 3.37: Řešení, $k = 3$



Obrázek 3.38: Řešení, $k = 4$

V tabulce 3.10 jsou znázorněny výsledky výše popsaného scénáře. Výpočet s širším okolím trval déle i přes to, že měl menší počet srážek. Je to kvůli tomu, že konflikty se řešily stejným způsobem a délky cest se výrazně nelišily (398 proti 327 pro celkový počet vrcholů), proto největší rozdíl byl v dvojnásobném počtu hran.

Statistika	k=3	k=4
Doba běhu (s)	50.509	87.604
Počet volání	39	27
Počet konfliktů	15	10
Průměrný počet vrcholů v cestách	79.6	65.4
Celkový počet vrcholů v cestách	398	327
Průměrný počet proměnných v hranovém modelu	81544	154832
Průměrný počet omezení v hranovém modelu	46447	83091
Průměrný počet proměnných v časovém modelu	106.1	83.81
Průměrný počet omezení v časovém modelu	109.08	85.67

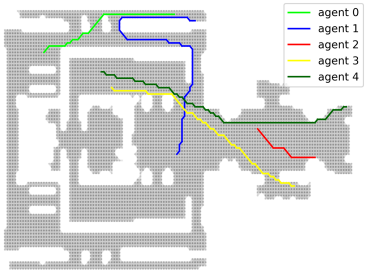
Tabulka 3.10: Statistika, mapa č. 4, scénář č. 1

Algoritmus SMT-CBS vyhledal stejné cesty pro k=3 a horší cesty pro k=4, následně selhal.

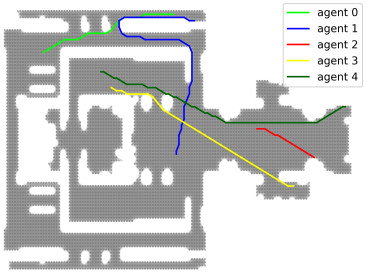
3.6.2 Scénář č. 2

Tento scénář obsahuje 3 překryvné konflikty mezi agenty 1 a 4 při k=3 (obrázek 3.39), ale při k=4 již k žádné srážce nedošlo (obrázek 3.40).

V prvním případě konflikt byl vyřešen tím, že agent 4 počkal ve vrcholech před detekovanými srážkami.



Obrázek 3.39: Řešení, $k = 3$



Obrázek 3.40: Řešení, $k = 4$

Uvedený scénář ukazuje stejnou statistiku (tabulka 3.11) jako předchozí: čas běhu je nejvíce ovlivněn propojením mapy, resp. počtem proměnných a omezení v modelech, pokud se počet konfliktů výrazně neliší.

SMT-CBS nalezl stejnou počáteční cestu pro k=3 a delší pro k=4, poté selhal.

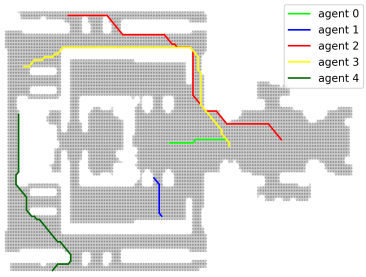
Statistika	k=3	k=4
Doba běhu (s)	9.422	15.34
Počet volání	11	5
Počet konfliktů	3	0
Průměrný počet vrcholů v cestách	66.8	50.8
Celkový počet vrcholů v cestách	334	254
Průměrný počet proměnných v hranovém modelu	81544	154832
Průměrný počet omezení v hranovém modelu	46447	83091
Průměrný počet proměnných v časovém modelu	81.91	50.8
Průměrný počet omezení v časovém modelu	81.73	49.8

Tabulka 3.11: Statistika, mapa č. 4, scénář č. 2

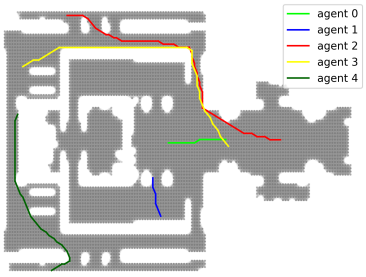
3.6.3 Scénář č. 3

Poslední scénář ukazuje, že i při širším okolí může docházet k většímu počtu konfliktů. Například zde při k=4 nastalo 10 srážek mezi agenty 2 a 3 (obrázek 3.41), ale při k=3 – jenom jedna (obrázek 3.42).

V první situaci konflikt vyřešil agent 3, ve druhé – nejvíce čekal agent 2, ale i agent 3 zůstával ve vrcholu krátkou dobu, aby se srážce vyhnul.



Obrázek 3.41: Řešení, $k = 3$



Obrázek 3.42: Řešení, $k = 4$

Doba běhu výpočtu (tabulka 3.12) je zase závislá na parametru k, ale v tomto případě při k=4 došlo k více konfliktům, proto je výpočet čtyřikrát pomalejší.

Algoritmus SMT-CBS překročil časový limit, pro k=3 cesty byly stejné, pro k=4 – horší.

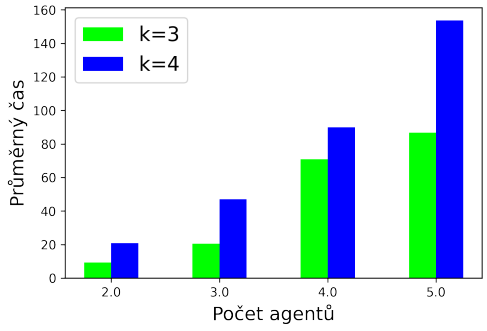
Statistika	k=3	k=4
Doba běhu (s)	18.834	92.61
Počet volání	11	29
Počet konfliktů	1	10
Průměrný počet vrcholů v cestách	57.2	46.6
Celkový počet vrcholů v cestách	286	233
Průměrný počet proměnných v hranovém modelu	81544	154832
Průměrný počet omezení v hranovém modelu	46447	83091
Průměrný počet proměnných v časovém modelu	78.64	73
Průměrný počet omezení v časovém modelu	78.45	74.76

Tabulka 3.12: Statistika, mapa č. 4, scénář č. 3

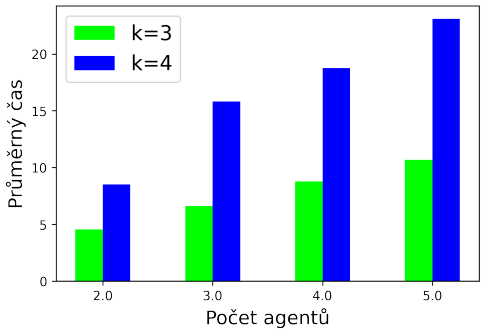
3.6.4 Závěr

Srážky nastaly v 29.5% spuštění.

Z grafů 3.43 a 3.44 je vidět, že pro bezkonfliktní řešení doba běhu je dvakrát delší pro k=4, než pro k=3. To odpovídá i statistice z rozebraných scénářů.



Obrázek 3.43: S konflikty



Obrázek 3.44: Bez konfliktů

Naopak počet volání algoritmu (graf 3.45) se výrazně neliší pro různá k.

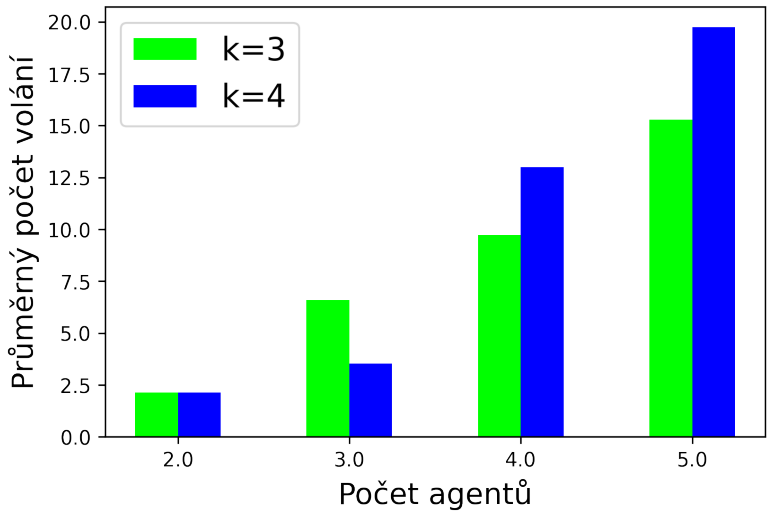
3.7 Mapa č. 5

Poslední mapa odpovídá prostředí skladu (warehouse).

Parametry mapy s k=3 (obrázek 3.46):

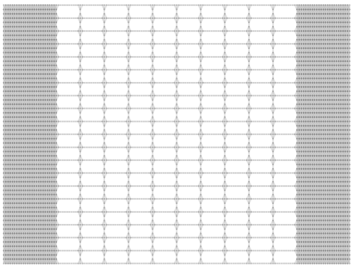
- počet vrcholů je 5699
- počet hran je 30676

Parametry mapy s k=4 (obrázek 3.47):

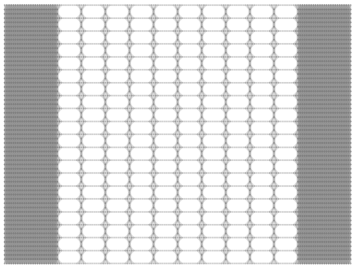


Obrázek 3.45: Počet volání

- počet vrcholů je 5699
- počet hran je 56404



Obrázek 3.46: Mapa, $k = 3$

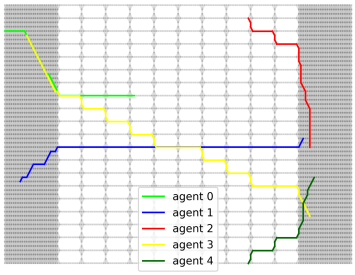


Obrázek 3.47: Mapa, $k = 4$

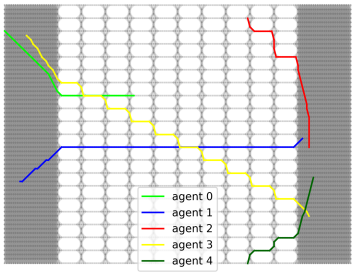
3.7.1 Scénář č. 1

V prvním scénáři došlo k 10 překryvným konfliktům mezi agenty 1 a 3 při k=3 (obrázek 3.48) a 9 srážkám mezi stejnými agenty při k=4 (obrázek 3.49).

Všechny konflikty byly vyřešený tím, že agent 1 počkal potřebný čas ve vrcholu před kolizemi.



Obrázek 3.48: Řešení, $k = 3$



Obrázek 3.49: Řešení, $k = 4$

Počet volání a konfliktů byl téměř stejný, proto dobu běhu nejvíce ovlivňoval počet hran v mapě, což zpomalilo výpočet, a počet vrcholů v cestách, což ho naopak zrychlilo. Z tabulky 3.13 je vidět, že počet hran měl větší vliv na rychlost nalezení řešení.

Statistika	k=3	k=4
Doba běhu (s)	42.479	56.868
Počet volání	25	23
Počet konfliktů	10	9
Průměrný počet vrcholů v cestách	83.2	69.8
Celkový počet vrcholů v cestách	416	349
Průměrný počet proměnných v hranovém modelu	61352	112808
Průměrný počet omezení v hranovém modelu	36375	62103
Průměrný počet proměnných v časovém modelu	123.84	106.35
Průměrný počet omezení v časovém modelu	125.44	107.7

Tabulka 3.13: Statistika, mapa č. 5, scénář č. 1

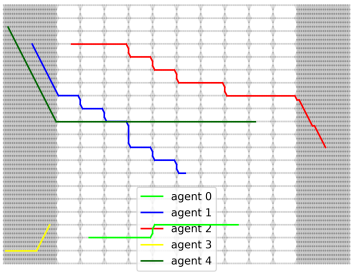
SMT-CBS algoritmus spočítal stejnou cestu pro k=3 a delší pro k=4, potom selhal.

3.7.2 Scénář č. 2

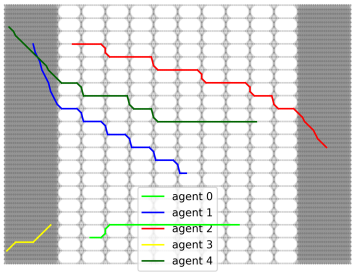
V tomto scénáři agenty 1 a 4 se srazili jenom v mapě s okolím k=3 (obrázek 3.50). Při k=4 agent 4 využil jiné chodbičky ve své cestě a tím předešel konfliktům (obrázek 3.51).

Agent 4 jednou počkal ve vrcholu před konflikty a tím se jich zbavil.

V tabulce 3.14 jsou porovnány výsledky kolizního a bezkolizního řešení. U k=4 ověření cest proběhlo relativně rychle, zatímco u k=3 se agenti museli vyhnout při 15 překryvných konfliktech.



Obrázek 3.50: Řešení, $k = 3$



Obrázek 3.51: Řešení, $k = 4$

Statistika	k=3	k=4
Doba běhu (s)	59.019	11.876
Počet volání	35	5
Počet konfliktů	15	0
Průměrný počet vrcholů v cestách	81	70.8
Celkový počet vrcholů v cestách	405	354
Průměrný počet proměnných v hranovém modelu	61352	112808
Průměrný počet omezení v hranovém modelu	36375	62103
Průměrný počet proměnných v časovém modelu	94.29	70.8
Průměrný počet omezení v časovém modelu	97.14	69.8

Tabulka 3.14: Statistika, mapa č. 5, scénář č. 2

Algoritmus SMT-CBS spočítal stejné počáteční cesty pro k=3 a delší pro k=4, poté selhal.

3.7.3 Scénář č. 3

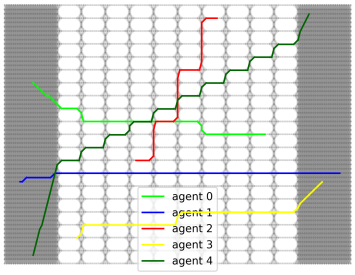
Poslední situace obsahuje jeden hranový konflikt a 15 překryvných, všechny nastaly mezi agenty 0 a 4 při k=3 (obrázek 3.52). Při k=4 se agenti pohybovali jinak a úsek jejich společné cesty se nacházel také jinde, proto ke srážkám nedošlo (obrázek 3.53).

Hranový konflikt a 14 překryvných vyřešil agent 0, poslední srážka byla za agentem 4.

Statistika rozebraného případu (tabulka 3.15) ukazuje, že více druhů konfliktů produkuje větší počet volání algoritmu, což prodlužuje i dobu běhu.

SMT-CBS algoritmus našel shodné počáteční cesty pro k=3 a delší pro k=4, poté selhal.

Obrázek 3.52: Řešení, $k = 3$



Obrázek 3.53: Řešení, $k = 4$

Statistika	k=3	k=4
Doba běhu (s)	172.788	12.429
Počet volání	99	5
Počet konfliktů	16	0
Průměrný počet vrcholů v cestách	116.2	104.8
Celkový počet vrcholů v cestách	581	524
Průměrný počet proměnných v hranovém modelu	61352	112808
Průměrný počet omezení v hranovém modelu	36375.48	62103
Průměrný počet proměnných v časovém modelu	126.64	104.8
Průměrný počet omezení v časovém modelu	129.39	103.8

Tabulka 3.15: Statistika, mapa č. 5, scénář č. 3

3.7.4 Závěr

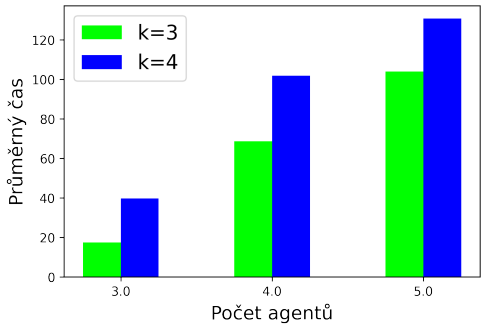
Konflikty byly nalezeny v 14.1% scénářů, přičemž u žádného pro dva agenty (graf 3.54).

Na grafu 3.55 jsou zobrazeny časy hledání řešení pro bezkolizní cesty. Výpočet trval téměř stejnou dobu pro různá k, jelikož centrální část obou map se výrazně neliší a většina doplňkových hran při k=4 se nachází v bočních úsecích.

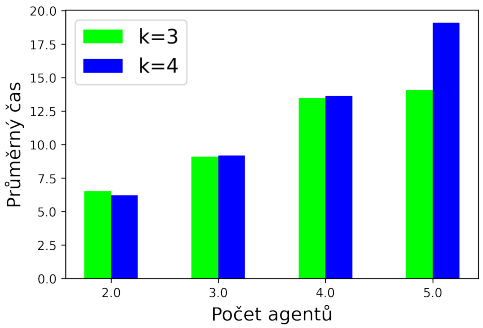
Počet volání (graf 3.56) je závislý na volbě počátečních cest: při k=4 agenti častěji volili cesty, u kterých nedocházelo ke srážkám, proto i počet volání je nižší.

3.8 Výsledky

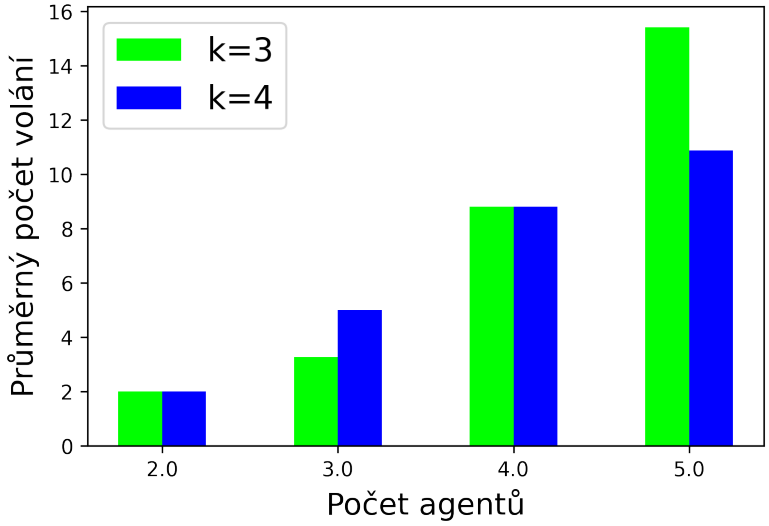
Z předvedených scénářů lze udělat závěr, že nalezené konflikty byly vyřešeny optimálním způsobem a srážky byly eliminovány.



Obrázek 3.54: S konflikty



Obrázek 3.55: Bez konfliktů



Obrázek 3.56: Počet volání

Sebrané statistické údaje se dají sjednotit následujícím způsobem:

- doba běhu je nejvíce ovlivněna počtem volání, dále také parametry mapy a délkou cest agentů;
- počet volání algoritmu je přímo závislý na počtu konfliktů a z některých scénářů navíc plyne, že při srážce jednoho agenta s více agenty během svého pohybu a/nebo při různých typech konfliktů narůstá počet volání rychleji;
- počet proměnných v hranovém modelu je vždy roven dvojnásobnému počtu hran v mapě;

- počet omezení v hranovém modelu se rovná součtu počtu hran a vrcholů, pokud nedošlo k hranovému konfliktu, jinak se k původnímu součtu přidá počet vrcholů, do kterých lze se posunout, aby se konflikt vyřešil, nebo jedno omezení pro každý konflikt, že tento vrchol nelze navštěvovat, pokud řešení konfliktu neexistuje;
- počet proměnných v časovém modelu je počet hran v cestě agentu plus počet kombinací při několikanásobné návštěvě stejného vrcholu;
- počet omezení v časovém modelu se rovná počtu hran v cestě a počtu vrcholových a překryvných konfliktů.

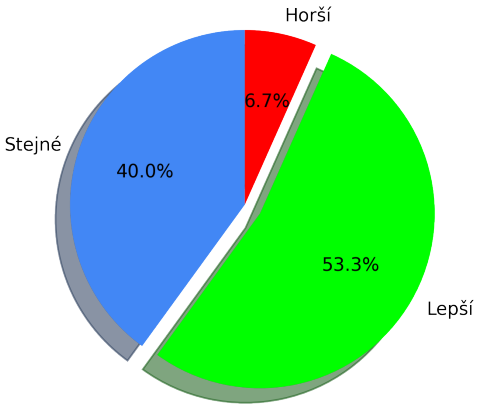
Co se týká srovnání s algoritmem SMT-CBS, nový algoritmus se ukázal jako lepší a zvládl všechny rozebrané scénáře, zatímco SMT-CBS – pouze jeden. SMT-CBS algoritmus nalezl vždy delší počáteční cesty také pro k=4.

Na dalších grafech 3.57, 3.58 jsou srovnány počáteční cesty nalezené algoritmem SMT-CBS a lineárním programem. Jelikož algoritmus SMT-CBS nalezl řešení jenom pro jeden scénář ze 30, není možné porovnávat kvalitu konečných cest, nýbrž pouze počátečních.

Cena počáteční cesty, neboli součet časů strávených pohybem přes každého agenta, je vždy nejmenší možná a výsledná cena bude buď stejná, nebo větší kvůli konfliktům.

Oba algoritmy spočítaly stejnou cenu pro 12 scénářů, SMT-CBS měl horši cesty v 16 případech a lepší ve dvou (graf 3.57).

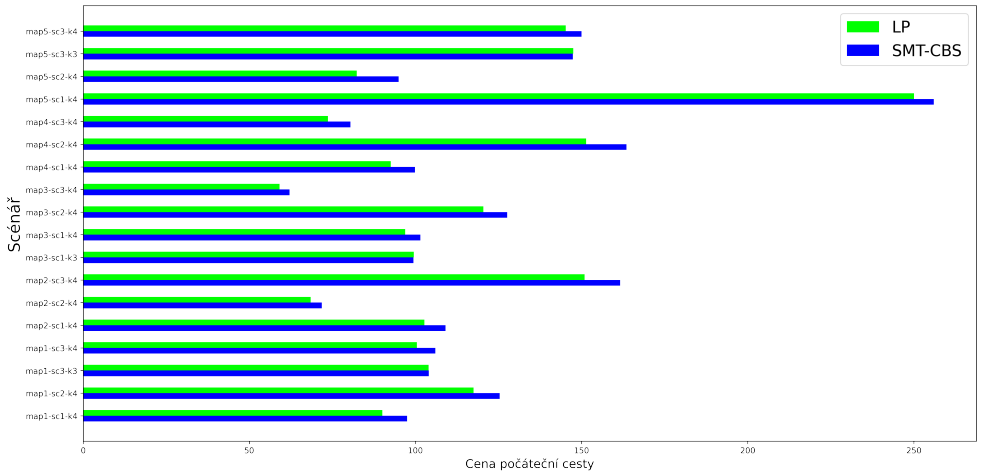
Tato informace může naznačovat, že nový algoritmus měl možnost najít lepší řešení pro 53.3% scénářů, kdyby SMT-CBS neselhal nebo nepřekročil časový limit.



Obrázek 3.57: Shodnost počátečních cest z hlediska LP algoritmu

Další graf 3.58 porovnává cenu počátečních cest pro scénáře, u kterých oba algoritmy spočítaly různé cesty.

Maximální rozdíl v ceně pro scénáře, ve kterých SMT-CBS se ukázal hůře, byl 12.621 vteřin a minimální – 0.031. Naopak SMT-CBS byl rychlejší o 0.145 a 0.118 vteřin, což není zásadní.



Obrázek 3.58: Porovnání ceny počátečních cest

Výhodou lineárního programu je jeho řešič, který je optimalizovány pro řešení takových problémů a umí dobře ořezávat stavový prostor. Nevýhodou algoritmu SMT-CBS je to, že musí generovat rozhodovací diagram, který je dost velký a proto tomu to trvá moc dlouho. Z toho potom plynou získané výsledky, že LP algoritmus byl lepší v půlce scénářů.

Závěr

Shrnutí práce

Práce se skládá ze tří částí.

První kapitola se zabývá teoretickými východisky: je v ní definováno multi-agentní hledání cest ve spojitém prostoru a konfliktní prohledávání ve spojitém prostoru, dále je vysvětlena kompilace pomocí výrokové splnitelnosti a lineárního programování, nakonec je zmíněno několik souvisejících metod a ty stručně popsány.

Druhá kapitola popisuje zkoumaný problém, uvádí definici překryvného konfliktu a postup jeho řešení. Kromě toho definuje dva lineární programy, které byly vyvinuty pro nalezení řešení problému.

V poslední kapitole je demonstrována ukázka práce vyvinutého algoritmu na pěti mapách. Každá mapa je následně vyhodnocena. Dokončuje kapitolu celkové vyhodnocení vytvořeného algoritmu.

Rekapitulace cílů

Tato práce měla několik cílů, a to jsou:

- Analýza postupu při hledání cest více agentů
- Analýza dvou základních přístupů kompilací
- Analýza a způsoby řešení různých konfliktů při srážkách
- Popis navrženého algoritmu
- Experimentální vyhodnocení algoritmu a interpretace výsledků

První dva cíle byly splněny v kapitole 1, dalším dvěma cílům se věnovala celá druhá kapitola a posledním cílem se zabývala 3. kapitola.

Výsledkem práce je navržený algoritmus, který nalezne nejkratší bezkolizní cesty pro několik agentů pohybujících se ve spojitém prostoru. Na tento algoritmus byla kladená následující očekávání:

- Bude nalezena cesta pro dva a více agenty
- Nalezena cesta bude bezkolizní
- Vyhnutí se srážkám bude optimální z hlediska celkového času

Z testovacích scénářů plyne, že algoritmus dokáže najít nejkratší cesty pro dva a více agenty a, pokud nastane srážka, spočítá optimálnější způsob, jak se jí vyhnout.

Možnosti budoucího rozvoje

Vyvinutý algoritmus lze ještě zlepšit. Další možnosti pro zdokonalení:

- používat agenty jiných tvarů, nejenom kruhové
- zvýšit výkon algoritmu, aby zvládal scénáře s vetším počtem agentů
- použit silnější variantu lineárního řešiče (např. paralelní)
- vyzkoušet výkonnější lineární řešič (např. CPLEX)

Literatura

[1] Stern, R.; Sturtevant, N. R.; Felner, A.; aj.: Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. *Symposium on Combinatorial Search (SoCS)*, 2019: s. 151–158.

[2] Krontiris, A.; Sajid, Q.; Bekris, K. E.: Towards using discrete multiagent pathfinding to address continuous problems. In *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[3] Wan, Q.; Gu, C.; Sun, S.; aj.: Lifelong Multi-Agent Path Finding in A Dynamic Environment. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, IEEE, 11 2018, s. 875–882, doi:10.1109/ICARCV.2018.8581181.

[4] Andreychuk, A.; Yakovlev, K.; Atzmon, D.; aj.: Multi-agent pathfinding with continuous time. *arXiv preprint arXiv:1901.05506*, 2019.

[5] Sharon, G.; Stern, R.; Felner, A.; aj.: Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, ročník 219, 2015: s. 40–66.

[6] Ryan, M. R. K.: Graph Decomposition for Efficient Multi-Robot Path Planning. In *IJCAI*, 2007, s. 2003–2008.

[7] Surynek, P.: Multi-agent Path Finding with Continuous Time Viewed Through Satisfiability Modulo Theories (SMT). *CoRR*, ročník abs/1903.09820, 2019, 1903.09820. Dostupné z: `http://arxiv.org/abs/1903.09820`

[8] Andreychuk, A.; Yakovlev, K.; Boyarski, E.; aj.: Improving Continuous-time Conflict Based Search. In *Proceedings of the International Symposium on Combinatorial Search*, ročník 12, 2021, s. 145–146.

[9] Trlifajová, K.; Vašata, D.: *Matematická Logika*. České vysoké učení technické, 2013.

[10] Kautz, H. A.; Selman, B.; aj.: Planning as Satisfiability. In *ECAI*, ročník 92, Citeseer, 1992, s. 359–363.

[11] Kautz, H.; Selman, B.; Hoffmann, J.: SatPlan: Planning as satisfiability. In *5th international planning competition*, 49, 2006, str. 156.

[12] Russell, S.; Norvig, P.: *Artificial intelligence*. Upper Saddle River, NJ: Pearson, třetí vydání, 12 2009.

[13] Surynek, P.: Unifying search-based and compilation-based approaches to multi-agent path finding through satisfiability modulo theories. In *International Symposium on Combinatorial Search*, ročník 10, 2019.

[14] Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; aj.: *Introduction to algorithms*. MIT press, 2009.

[15] Erickson, J.: *Algorithms*. S.L: S.N, 2019, ISBN 1792644833.

[16] Mareš, M.; Valla, T.: *PRŮVODCE labyrintem algoritmů*. CZ.NIC, z.s.p.o., 2017.

[17] Muter, I.; Birbil, Ş. İ.; Bülbül, K.: Benders decomposition and column-and-row generation for solving large-scale linear programs with column-dependent-rows. *European Journal of Operational Research*, ročník 264, č. 1, 2018: s. 29–45.

[18] Clarke, E.; Grumberg, O.; Jha, S.; aj.: Counterexample-guided abstraction refinement. In *International Conference on Computer Aided Verification*, Springer, 2000, s. 154–169.

[19] Goffin, J.-L.: Convergence rates of the ellipsoid method on general convex functions. *Mathematics of Operations Research*, ročník 8, č. 1, 1983: s. 135–150.

[20] Boyd, S.: Ellipsoid method. *Standford Univ., Stanford, CA, USA, Tech. Rep*, 2014.

[21] Hormazábal, N.; Díaz, A.; Hernández, C.; aj.: Fast and almost optimal any-angle pathfinding using the 2k neighborhoods. In *Tenth Annual Symposium on Combinatorial Search*, 2017.

[22] Gurobi Optimization: Objectives [online]. [Cit. 27.01.2022]. Dostupné z: `https://www.gurobi.com/documentation/9.5/refman/objectives.html`

[23] Gurobi Optimization: Variables [online]. [Cit. 27.01.2022]. Dostupné z: `https://www.gurobi.com/documentation/9.5/refman/variables.html`

[24] Gurobi Optimization: Constraints [online]. [Cit. 27.01.2022]. Dostupné z: https://www.gurobi.com/documentation/9.5/refman/constraints.html

[25] Sturtevant, N. R.: Benchmarks for grid-based pathfinding. *IEEE Transactions on Computational Intelligence and AI in Games*, ročník 4, č. 2, 2012: s. 144–148.

Příloha **A**

Seznam použitých zkratek

MAPF Multi-agentní hledání cest

MAPF$_R$ Multi-agentní hledání cest ve spojitém prostoru

SOC Součet cen

CCBS Konfliktní prohledávání ve spojitém prostoru

CT Constraint Tree

SAT Problém splnitelnosti booleovské formule

CNF Konjunktivní normální forma

PDDL The Planning Domain Definition Language

SMT Satisfiability modulo theories

LP Lineární programování

CEGAR Counterexample-guided Abstraction Refinement

Příloha **B**

Obsah přiložené SD-karty

- `readme.txt` ..... stručný popis obsahu SD-karty
- `exe` ..... adresář se spustitelnou formou implementace
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `DP_Zabrodskaya_Yana_2022.pdf` ..... text práce ve formátu PDF

Příloha C

Příloha

Program je napsán v jazyce Python s využitím knihovny gurobipy.

C.1 Popis algoritmů

Hlavní funkcí je *find_solution*, která potřebuje kořen CT stromu a strukturu mapy jako vstupní parametry. Vrátí řešení, pokud nalezne bezkolizní cesty pro každého agenta, nebo *None*, pokud řešení neexistuje nebo byl překročen nastavený časový limit.

Detekce konfliktů je implementována v rámci algoritmu *collisions*. Tato funkce pro každou dvojici agentů zkontroluje

- nejdříve vrcholové konflikty: funkce *vertex_conflict* vrátí *True* nebo *False*, bude-li ve dvou cestách stejný vrchol současně,
- pak hranové konflikty: *edge_conflict_no_swap* pro předjíždění a funkce *edge_conflict* pro prohození agentů
- a jako poslední překryvné konflikty

Algoritmus vrátí množinu nalezených srážek.

Následně algoritmus *find_path* implementuje model se spojitým časem popsaný v druhé kapitole, který volá algoritmus *find_edges* implementující hranový model. Tento algoritmus dostane na vstupu mapu a uzel CT stromu a vrátí cestu pro jednoho agenta s ohledem na jeho omezení.

Za výpočet času potřebného pro čekání odpovídá struktura *Solution* a funkce *calculate_waiting_time*.

C.2 Spouštění programu

Program se spouští přes příkazový řádek a požaduje soubor s mapou jako argumenty, které se nachází ve složce *maps*, číslo $k \in \{2, 3, 4\}$, soubor s definici

agentů ze složky *agents* nebo slovo *random* pro náhodné rozmístění a k tomu počet agentů.

Příklad:

`python main.py maps/berlin 3 agents/berlin_1` spustí mapu Berlína s okolím k=3 a prvním scénářem.

`python main.py maps/berlin 4 random 7` spustí mapu Berlína s okolím k=4 a sedmi agenty a náhodným rozmístěním.