Optimální kooperativní hledání cest řešené pomocí výrokové splnitelnosti

Pavel Surynek
Univerzita Karlova v Praze, Matematicko-fyzikální fakulta
Malostranské náměstí 25, 118 00 Praha 1, Česká republika
pavel.surynek@mff.cuni.cz

Abstrakt. V kooperativním hledání cest je úkolem pro každého mobilního agenta ze skupiny agentů nalézt cestu spojující jeho počáteční pozici se zadanou cílovou pozicí tak, aby nedocházelo ke srážkám s překážkami a mezi agenty navzájem. Agenty se přitom pohybují v jistém prostředí, které je obvykle modelováno jako neorientovaný graf s vrcholy reprezentujícími pozice a hranami reprezentujícími možnost pohybu na sousední pozici. V každém vrcholu grafu se nachází nejvýše jeden agent a pohybovat se může do sousedního volného vrcholu. Článek ukazuje metodu řešení založenou na kódování úlohy jako výrokové formule, která je následně řešena výkonnými řešícími systémy pro výrokovou splnitelnost (SAT). Navrženou metodou lze generovat řešení, která jsou optimální vzhledem k délce vykonání plánu.

Klíčová slova: kooperativní hledání cest, optimální řešení, výroková splnitelnost, výrokové kódování, SAT

1 Úvod a výzkum v současném kontextu

Kooperativní hledání cest stale více přitahuje pozornost komunity zabývající se umělou inteligencí. Tento zájem je motivován širokými možnostmi, kde může být kooperativní hledání cest aplikováno (robotika, počítačové hry, optimalizace dopravy, atd.), ale také výzkumnými výzvami, které problém skýtá. Úloha spočívá v nalezení cest v prostoru a čase pro jisté agenty, kde každý agent usiluje o dosažení určité polohy. Mezi agenty nesmí docházet ke srážkám a stejně tak se agenty musí vyhýbat překážkám. Jeden z nejdůležitějších průlomů v řešení úlohy představuje algoritmus WHCA* [8], který hledání kooperativního plánu rozkládá na hledání plánů pro jednotlivé agenty. Nedávno se dokonce objevily podobné metody, které řeší úlohu optimálně [10]. Společným nedostatkem metod používajících rozklad úlohy je fakt, že jsou reálně použitelné pouze pro instance s malým podílem prostoru/prostředí obsazeného agenty.

Na opačné straně spektra algoritmů řešících úlohu jsou úplné sub-optimální metody [4, 11]. Tyto algoritmy jsou schopné najít řešení bez ohledu na podíl prostoru obsazeného agenty. Zvlášť dobrých výsledků bylo dosaženo pro instance hustě obsazené

Tato práce je podporována z grantů poskytnutých Grantovou agenturou ČR (číslo projektu GAP103/10/1287) a Univerzitou Karlovou v Praze (projekt PRVOUK P46).
Děkuji recenzentům za pečlivé přečtení článku, zajímavé postřehy a opravy.

agenty. Na druhou stranu, u řídce obsazených instancí tyto algoritmy často generují příliš dlouhá řešení.

Jiné metody se zase snaží využít struktury prostředí [7] a struktury aktuálního rozmístění agentů [12]. Tyto metody ovšem vyžadují dostatek volného prostoru v prostředí (například pro dočasné odložení agentů nebo vyhledání volné cesty) a navíc jsou v některých případech neúplné.

V této práci se snažíme přispět k případu s vysokou obsazeností prostředí s požadavkem na pokud možno co nejkratší dobu vykonání řešení (makespan). Námi navrhované řešení je svým založením úplné. Novým způsobem zde používáme technologii řešení výrokové splnitelnosti (SAT). Nejprve je nějakým existujícím rychlým algoritmem vygenerováno řešení, které je vzhledem k délce neoptimální. Toto neoptimální řešení je následně rozděleno na krátké posloupnosti akcí/pohybů, které jsou poté nahrazovány optimálními posloupnostmi nalezenými pomocí řešení převodu úlohy na SAT. Proces je iterován, dokud délka aktuálního řešení nezkonverguje k pevnému bodu. Rozklad původní úlohy na mnoho menších dovoluje využít nejsilnější stránku současných systémů pro SAT, a sice jejich schopnost vyřešit relativně malou avšak stále dost složitou instanci velmi rychle.

V článku je nejprve rozebráno kooperativní hledání cest formálně. Dále je popsáno námi navržené speciální doménově závislé kódování úlohy jako SATu, na které pak navazuje popis techniky optimalizující sub-optimální řešení. Nakonec je ukázáno experimentální srovnání s existujícími technikami rovněž používajícími řešení SATu.

2 Kooperativní hledání cest formálně

Jako model prostředí lze vzít libovolný **neorientovaný graf**. Nechť $G = (V, E)$ je takový graf, kde V je konečná množina vrcholů a $E \subseteq \binom{V}{2}$ je množina hran.

Rozložení agentů v prostředí je modelováno přiřazením vrcholů grafu. Nechť $A = \{a_1, a_2, \dots, a_n\}$ je konečná množina *agentů*. *Rozložení* agentů ve vrcholech grafu G je popsáno pomocí funkce *umístění* $\alpha: A \longrightarrow V$. Význam funkce je takový, že agent $a \in A$ je umístěn ve vrcholu $\alpha(a)$. **Nejvýše jeden agent** se může nacházet ve vrcholu grafu; to znamená, že α je prostá funkce. Zobecněná inverze k funkci α bude označena jako $\alpha^{-1}: V \longrightarrow A \cup \{\perp\}$ a bude vracet informaci o tom, který agent se nachází v zadaném vrcholu, případně $\perp$, jestliže je vrchol prázdný.

Definice 1 (KOOPERATIVNÍ HLEDÁNÍ CEST). Instance problému *kooperativního hledání cest* je čtveřice $\Sigma = [G = (V, E), A, \alpha_0, \alpha_+]$, kde funkce umístění α_0 a α_+ určují počáteční a cílové rozložení agentů z množiny A v grafu G. □

Dynamika modelu předpokládá diskrétní čas rozdělený do časových kroků. Rozložení agentů α_i v i-tém časovém kroku může být instantně změněno pomocí pohybů agentů na nové rozložení α_{i+1}. Agenty se přitom pohybují po hranách grafu a vzájemně nekolidují. Formálně musí α_{i+1} splňovat následující *pohybové podmínky*:

(i) $\forall a \in A$ platí buď $\alpha_i(a) = \alpha_{i+1}(a)$, nebo $\{\alpha_i(a), \alpha_{i+1}(a)\} \in E$
(agenty se pohybují podél hran, nebo zůstávají na místě),

(ii) $\forall a \in A \;\; \alpha_i(a) \neq \alpha_{i+1}(a) \Rightarrow \alpha_i^{-1}(\alpha_{i+1}(a)) = \perp$
(agenty se pohybují pouze do volných vrcholů), a

(iii) $\forall a, b \in A \;\; a \neq b \Rightarrow \alpha_{i+1}(a) \neq \alpha_{i+1}(b)$
(žádné dva agenty se nepohybují do stejného cílového vrcholu).

Úkolem je změnit rozložení α_0 konečně mnoha pohyby splňujícími pohybové podmínky na rozložení α_+.

Definice 2 (ŘEŠENÍ, DÉLKA/MAKESPAN). *Řešení délky* (*makespan*) m instance problému kooperativního hledání cest $\Sigma = [G, A, \alpha_0, \alpha_+]$ je posloupnost rozložení $\vec{s} = [\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_m]$ kde $\alpha_m = \alpha_+$ a pro každé $i = 1,2, \ldots, m-1$ je α_{i+1} výsledek změny α_i splňující pohybové podmínky. □

Jestliže je otázkou, zda existuje řešení Σ nejvýše určité zadané délky, potom hovoříme o *omezené variantě* problému. Všimněme si, že díky akcím, které neprovádějí žádnou změnu, je tato otázka ekvivalentní otázce, zda existuje řešení přesně zadané délky.

3 SAT kódování a omezená varianta problému

Cílem je vyvinout SAT kódování omezené varianty kooperativního hledání cest, které je vhodné pro instance s relativně vysokou obsazeností prostředí. Současně požadujeme, aby výsledné kódování bylo pokud možno co nejkompaktnější. Vycházeli jsme z klasických kódování *Graphplan* [2, 3] v tom směru, že rovněž kódujeme každý časový krok.

Využití vícehodnotových stavových proměnných

Primárně nás inspirovala kódování SATPLAN [3] a SASE [2]. Ovšem na rozdíl od těchto obecných kódování my pracujeme se specifickou doménou, takže jsme schopni využít znalost této domény k návrhu speciálního kódování. A priorně známe kandidáty na *vícehodnotové stavové proměnné* – je to funkce umístění a její inverze. Pomocí technik navržených Rintanenem [6] může být každá stavová proměnná kódována pomocí logaritmického počtu výrokových proměnných vzhledem k velikosti své domény. Další zajímavou otázkou je, jak kódovat změny rozložení tak, aby byly zachování pohybové podmínky.

Vyjádření rozleţení pomocí **inverzního umístění** (to jest, máme stavovou proměnnou pro každý vrchol) nám umožnilo kódovat změny efektivně a úsporně. Jsou použity dvě tzv. *primitivní akce* pro každou hranu příslušející danému vrcholu, které odpovídají pohybu agenta po hraně v každém směru, plus navíc jedna primitivní akce na každý vrchol, která odpovídá neprovedení žádného pohybu. Polovina primitivních akcí odpovídajících vrcholu je vyhrazena na příchozí agenty z okolí, zatímco druhá polovina primitivních akcí je vyhrazena pro odchozí agenty. Jestliže je zvolena odchozí primitivní akce v nějakém vrcholu je nutno v cílovém sousedním vrcholu zvolit odpovídající příchozí akci. Jestliže jsou výběry primitivních akcí implementovány pomocí vícehodnotových stavových proměnných, pak je automaticky zajištěna platnost podmínek (i) a (iii). Všimněme si také, že stupeň vrcholů v grafu G je v reálných instancích typicky nízký, neboť se odehrávají ve světě, který je dvourozměrný, resp. trojrozměrný, což indukuje nízkou konektivitu modelujících grafu, a tedy výběr akce ve vrcholu lze uskutečnit pomocí malého počtu výrokových proměnných.

Nechť $\Sigma = [G = (V, E), A, \alpha_0, \alpha_+]$ je kooperativní instance a nechť $k \in \mathbb{N}$ omezení na délku řešení. Navržené kódování potom bude mít vrstvy číslované $0,1, \ldots, k$. Předpokládejme, že sousední vrcholy daného vrcholu jsou pevně uspořádány. To jest, $\forall v \in V$ mame funkci $\sigma_v: \{u | \{v, u\} \in E\} \longrightarrow \{1,2, \ldots, \deg_G(v)\}$ a její inverzi σ_v^{-1}.

Definice 3 (VRSTEVNATÉ KÓDOVÁNÍ). i-tá regulární vrstva kódování sestává z následujících intervalových celočíselných **stavových proměnných**:

- $\mathcal{A}_i^v \in \{0,1,2,\dots,n\}$ pro všechna $v \in V$ tak, že
  $\mathcal{A}_i^v = j$ iff $\alpha_i(a_j) = v$
- $\mathcal{T}_i^v \in \{0,1,2,\dots,2\deg_G(v)\}$ pro všechna $v \in V$ tak, že
  $\mathcal{T}_i^v = 0$ jestliže ve v nebyl proveden žádný pohyb;
  $\mathcal{T}_i^v = \sigma_v(u)$ jestliže byla ve v vybrána odchozí primitivní akce s cílem $u \in V$;
  $\mathcal{T}_i^v = \deg_G(v) + \sigma_v(u)$ jestliže byla ve v vybrána příchozí primitivní akce s $u \in V$ jako se zdrojovým vrcholem.

a **podmínek**:

- $\mathcal{T}_i^v = 0 \Rightarrow \mathcal{A}_{i+1}^v = \mathcal{A}_i^v$ pro všechna $v \in V$ (**žádný** pohyb);
- $0 < \mathcal{T}_i^v \leq \deg_G(v) \Rightarrow \mathcal{A}_i^u = 0 \wedge \mathcal{A}_{i+1}^u = \mathcal{A}_i^v \wedge \mathcal{T}_i^u = \sigma_u(v) + \deg_G(u)$, kde $u = \sigma_v^{-1}(\mathcal{T}_i^v)$ pro všechna $v \in V$ (**odchozí** případ);
- $\deg_G(v) < T_i^v \leq 2\deg_G(v) \Rightarrow \mathcal{T}_i^u = \sigma_u(v)$, kde $u = \sigma_v^{-1}(\mathcal{T}_i^v - \deg_G(v))$ pro všechna $v \in V$ (**příchozí** případ). □

Stavové proměnné $\mathcal{A}_i^v$ pro $v \in V$ vyjadřují inverzní funkci umístění v časovém kroku i. Analogicky stavové proměnné $\mathcal{T}_i^v$ pro $v \in V$ vyjadřují primitivní akce vybrané ve vrcholech v časovém kroku i. Podmínky vyjadřují pohybová omezení nad navrženou celočíselnou reprezentací. Poslední vrstva kódování je neregulární, neboť obsahuje pouze stavové proměnné a žádné podmínky. Počáteční a cílové rozmístění agentů je vyjádřeno pomocí jednoduchých rovností mezi stavovými proměnnými a konstantami v první a poslední vrstvě.

Překlad navržené celočíselné reprezentace do výrokových proměnných je přímočarý. Kvůli zmenšení výsledné formule v konjunktivně normálním tvaru (CNF) se doporučuje využít Tseitinova hierarchického překladu s pomocnými proměnnými.

Většina klauzulí generovaných v navrženém kódování má aritu $\lceil \log_2(2\deg_G(v) + 1) \rceil + 1$ pro nějaké $v \in V$ nebo $\lceil \log_2 |A| \rceil + 1$. Srovnání s kódováními založenými na plánovacím grafu, které se využívají v plánovacím systému SATPLAN, je ukázáno v Tabulka 1. Navržené doménově závislé kódování je zřejmě menší, přičemž rozdíl roste s narůstajícím počtem agentů.

Tabulka 1. *Porovnání velikostí kódování.* Výsledky jsou ukázány pro nejmenší počet vrstev, kdy SATPLAN nemohl detekovat nedosažitelnost cíle bez prohledávání pouhým uvažováním nad plánovacím grafem. Tento počet vrstev označujeme jako *cílovou hladinu* – byla použita jako mez na délku řešené.

|**Agenti**| v 4-propojené mřížce 8x8	**Cílová hladina**	**SATPLAN** kódování		Navržené **doménově závislé** kódování	
		|**Proměnné**|	|**Klauzule**|	|**Proměnné**|	|**Klauzule**|
4	8	5864	55330	9432	55008
8	8	10022	165660	11968	70400
12	8	14471	356410	11968	68352
16	10	30157	1169198	18490	112580
24	10	43451	2473813	18490	107360
32	14	99398	8530312	32116	200768

4 COBOPT: Nový přístup k plánování (cest)

Námi navržená nová technika pro kooperativní hledání cest nazvaná COBOPT využívá technologii řešení SATu [1] nikoli k vytvoření celého řešení, ale k optimalizaci již existujícího řešení vzhledem k jeho délce. Abychom mohli využít systémy pro řešení SATu tímto způsobem, je třeba nejprve získat alespoň nějaké (neoptimální) řešení. Dále budeme toto řešení nazývat *výchozím řešením*. Jak již bylo zmíněno, v současnosti existuje mnoho řešících technik pro kooperativní hledání cest [7], [8] (WHCA*); [11] (BIBOX); [4] (PUSH-SWAP); [10] - OD+ID; [12] (MAPP). Libovolná z nich může být použita k vygenerování výchozího řešení. Navržená optimalizační technika je v tomto smyslu zcela modulární. Je ovšem potřeba mít na zřeteli, že konkrétní řešící algoritmy jsou vhodné pro řešení určitých tříd problému, zatímco na instancích mimo danou třídu mohou poskytovat horší výkon. Typickým nedostatkem například u metod založených na rozkladu úlohy (WHCA*, MAPP) je, že dovolují, aby některé agenty vůbec nedorazily do svých cílových vrcholů [8, 12].

Algoritmus 1. COBOPT: *Na SATu založená optimalizace řešení kooperativního hledání cest – základní schéma s binárním vyhledáváním. Řešení SATu je provedeno voláním externí procedury Solve-SAT.*

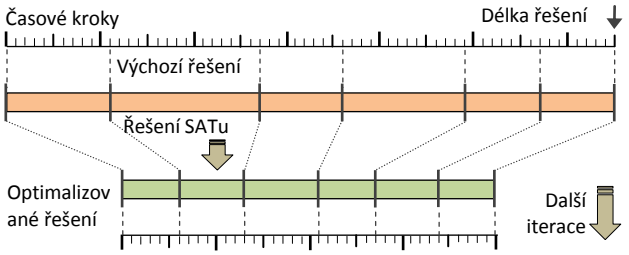
function *COBOPT-Optimize-Cooperative-Plan* $(\Sigma, \vec{s}, k^+)$: **solution**
1: $\vec{s}_+ \leftarrow \vec{s}$
2: **do**
3: $\quad \vec{s}_- \leftarrow \vec{s}_+$
4: $\quad$ **let** $\vec{s}_- = [\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_m]$
5: $\quad t \leftarrow 0$; $\vec{s}_+ \leftarrow []$
6: $\quad$ **while** $t < m$ **do**
7: $\quad\quad t^+ \leftarrow$ *Find-Last-Reachable-Arrangement*$(\Sigma, \alpha_t, \vec{s}_-, k^+)$
8: $\quad\quad \vec{s}_+ \leftarrow \vec{s}_+$.*Compute-Optimal-Solution*$(\Sigma, \alpha_t, \alpha_{t^+})$
9: $\quad\quad t \leftarrow t^+$
10: **while** $|\vec{s}_-| > |\vec{s}_+|$
11: **return** $\vec{s}_+$

function *Find-Last-Reachable-Arrangement* $(\Sigma, \alpha_t, \vec{s}, k^+)$: **integer**
1: **let** $\vec{s} = [\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_m]$
2: $l \leftarrow t$; $u \leftarrow m + 1$
3: **while** $u - l > 1$ **do**
4: $\quad r \leftarrow (u + l)/2$
5: $\quad k \leftarrow \min(m - t, k^+)$
6: $\quad$ **if** *Check-Reachability*$(\Sigma, \alpha_t, \alpha_r, k)$ **then**
7: $\quad\quad \Xi \leftarrow$*Encode*$(\Sigma, \alpha_t, \alpha_r, k)$
8: $\quad\quad$ **if** *Solve-SAT* (Ξ) **then** $l \leftarrow r$
9: $\quad\quad$ **else** $u \leftarrow r$
10: $\quad$ **else**
11: $\quad\quad u \leftarrow r$
12: **return** l

function *Check-Reachability* $(\Sigma, \alpha_t, \alpha_r, k)$: **boolean**
1: **let** $\Sigma = [G, A, \alpha_0, \alpha_+]$
2: **for each** $a \in A$ **do**
3: $\quad$ **if** $\mathrm{dist}_G(\alpha_t(a), \alpha_r(a)) > k$ **then return** *FALSE*
4: **return** $TRUE$

V našich počátečních experimentech se ukázalo, že vyřešit úlohu kooperativního hledání cest se stává dramaticky obtížnější, jak se zvyšuje mez na délku řešení v kódované instanci. Přesněji řečeno, řešící systém pro SAT má obvykle potíže vyřešit instanci se 100 vrcholy, 30 agenty a mezí na délku řešení 10 dříve než za několik minut na současném komoditním PC s použitím navrženého kódování. V případě použití kódování systému SATPLAN je situace mnohem horší – řešícímu systému trvá několik minut už pouhé generování výrokové formule, kterou má následně řešit. Tento nález ukazuje, že používat řešící systémy pro SAT s kódováním SATPLAN [2, 3] na vyřešení celé úlohy kooperativního plánování užitečné velikosti není v současnosti realistické, jelikož i optimální plán může vyžadovat až stovky časových kroků. Přístup využívaný v systému SATPLAN má však jednu nespornou výhodu – a sice tu, že vygenerované řešení je optimální vzhledem k délce.

Po vytvoření výchozího řešení je toto předáno řešícímu systému pro SAT k optimalizaci. Je stanovena maximální mez na délku řešení k^+. Poté jsou podposloupnosti výchozího řešení nahrazeny vypočtenými optimálními částečnými řešeními. Předpokládejme, že proces optimalizace probíhá na časovém kroku t. Je spočteno, jaký je nejvyšší časový krok $t^+ > t$ takový, že rozložení v kroku t^+ lze dosáhnout z rozložení v časovím kroku t za nejvýše k^+ časových kroků. Poté je část výchozího řešení od časového kroku t do kroku t^+ nahrazena optimálním částečným řešením, které je získáno od řešícího systému pro SAT. Optimalizační proces poté pokračuje v časovém kroku t^+ dokud není zpracováno celé výchozí řešení.



Obrázek 1. *Ilustrace optimalizačního procesu.* Je ukázána jedna iterace – proces iteruje, dokud se délka řešení zkracuje, nebo není vyčerpán časový limit.

Optimalizační proces může být iterován tak, že výsledné částečně optimalizované řešení je položeno jako nové výchozí řešení, což se opakuje, dokud se délka řešení zkracuje. K nalezení časového kroku t^+ se používá binárního vyhledávání, což snižuje počet dotazů na SAT řešící systém – viz Algoritmus 1, který základní metodu COBOPT popisuje formálně.

Postup optimalizace řešení je ještě znázorněn na Obrázek 1. Všimněme si, že místa, kde je výchozí řešení rozděleno, jsou hledána hladovým způsobem – optimalizace pokračuje vždy na ještě nezpracovaném časovém kroku. Toto nemusí být nutně dobrý způsob, proto jsme vyzkoušeli najít optimální rozdělení výchozího řešení pomocí techniky dynamického programování. Ukázalo se, že optimální rozdělení výchozího řešení přináší jej nepatrně lepší výsledky, ale je zaplaceno extrémními časovými nároky, neboť je nutné najít optimální náhrady za všechny souvislé podposloupnosti výchozího řešení. Z těchto důvodů lze tedy hladový přístup považovat za dostatečný.

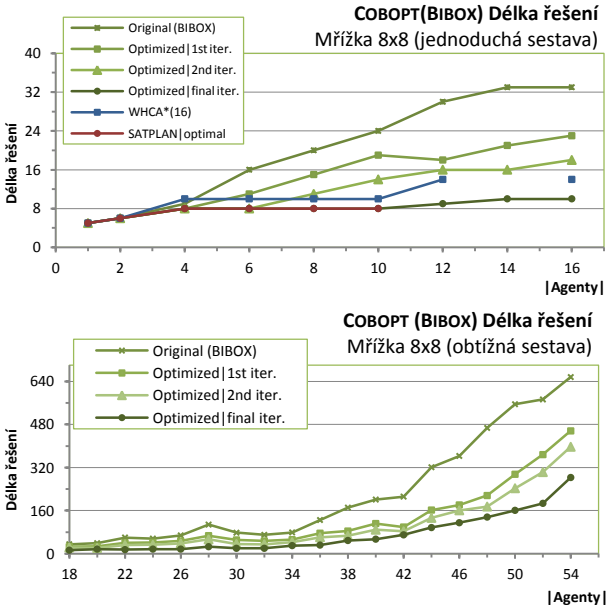
5 Experimentální vyhodnocení

Implementovali jsme optimalizační metodu COBOPT v C++, abychom mohli provést experimenty. Srovnání bylo prováděno vůči třem existujícím řešícím algoritmům – WHCA*, SATPLAN a BIBOX. WHCA* byl vybrán, protože je často používán jako referenční metoda a standardní benchmark pro kooperativní hledání cest.

Tabulka 2. *Optimální řešení získaná systémem* SATPLAN. Více agentů nebyl systém SATPLAN schopen v daném časovém limitu 7200 vteřin vyřešit.

|Agenty|	4-propojená mřížka 8x8		4-propojená mřížka 16x16	
	Optimální délka	Čas běhu (s)	Optimální délka	Čas běhu (s)
1	5	0.0	4	0.68
4	6	0.15	21	195.5
8	8	19.85	15	1396.07

Jelikož vhodná implementace WHCA* nebyla k dispozici, implementovali jsme jej sami v C++. SATPLAN je metoda velmi podobná námi navržené a má důležitou vlastnost, že generuje optimální řešení – zde jsme použili implementaci poskytovanou autory. Nakonec BIBOX byl vybrán jako hlavní metoda pro generovaná výchozích řešení, neboť je známo, že zvládá řešit i obtížné instance úlohy.

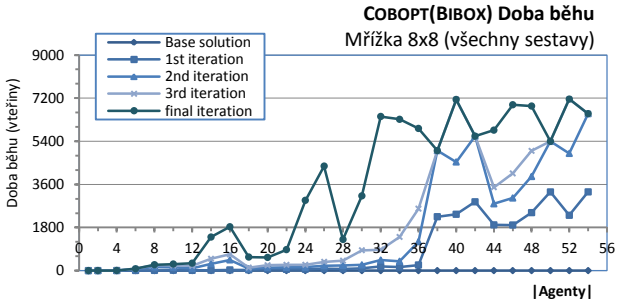


Obrázek 2. *Optimalizace délky řešení na 4-propojené mřížce 8×8.* Je ukázáno srovnání s optimálním systémem SATPLAN a skoro optimálním WHCA* (zvládají pouze jednoduchou sestavu s málo agenty).

Algoritmus BIBOX má polynomiální časovou složitost (výchozí řešení ke všem prezentovaným úlohám byla vygenerována za méně než 0.1 vteřiny) a generuje sice

neoptimální řešení, ovšem řešení dobré kvality – společně s algoritmem PUSH-SWAP autorů Luna a Berky [4] je to jediný algoritmus zvládající vyřešit obtížné instance úlohy. COBOPT využívající BIBOX jako výchozí generátor řešení bude odkazován jako COBOPT(BIBOX). Pro řešení instancí zakódovaných jako SAT jsme ve všech experimentech využívali MINISAT 2.2 [1].

Standardní benchmark pro kooperativní hledání cest sestává ze 4-propojené mřížky a náhodně rozmístěných agentů v počátečním a cílovém stavu. Tento benchmark jsme rovněž použili. Byly analyzovány různé parametry metody COBOPT(BIBOX) v závislosti na vzrůstajícím počtu agentů. Byla použita sestava s mřížkou o velikosti 8×8 s počtem agentů od 1 do 54. Dále byl použit časový limit 240s pro jedno zavolání řešícího systému SAT. Přitom limit na délku řešení byl nastaven na 8. Celkový limit na dobu běhu byl stanoven na 7200 vteřin (2 hodiny), po jehož uplynutí byla optimalizace zastavena a jako výsledné řešení bylo vráceno poslední nejkratší nalezené.



Obrázek 3. *Doba běhu různého počtu iterací v mřížce 8×8.* Výchozí řešení bylo ve všech případech vygenerováno za méně než 0.1 vteřiny.

Pomocí WHCA* jsme zjistili, že sestavy s nejvýše 20% obsazenými vrcholy jsou ve skutečnosti jednoduché a k jejich vyřešení je potřeba pouze velmi omezené kooperace mezi agenty. Toto pozorování z našeho srovnání vyřadilo metodu OD+ID [9, 10], neboť ta efektivně funguje pouze v sestavách s obsazeností vrcholů menší 10%. Zde se zajímáme o sestavy s obsazeností v rozmezí 20% - 50%, které jsou obtížnější, protože k jejich vyřešení je zapotřebí intenzivní kooperace mezi agenty.

Abychom zjistili, jaké jsou optimální délky řešení testovaných instancí, pokusili jsme je vyřešit systémem SATPLAN (Tabulka 2). Ukázalo se ale, že SATPLAN byl schopen vygenerovat optimální řešení pouze pro instance s malým počtem agentů. Hlavním limitujícím faktorem přitom byla neefektivita používaného doménově nezávislého kódování (Tabulka 1).

V dalších experimentech jsme využili metodu WHCA*. Očekávaně se ukázalo, že tato metoda umí generovat skoro optimální řešení (Obrázek 2), jelikož jsou skoro optimální cesty hledané pro každého agenta zvlášť. Nicméně, tato metoda principiálně neumí vyřešit instance, kde je nutná intenzivnější kooperace mezi agenty. WHCA* jsme použili na klasifikaci instancí na **jednoduché** a **obtížné** – jednoduché jsou řešitelné pomocí WHCA*.

Na rozdíl od SATPLANu a WHCA* je COBOPT úspěšnější; dokáže vyřešit každou instanci, kterou dokáže vyřešit algoritmus na generování výchozího řešení – v případě algoritmu BIBOX to byly všechny instance v testované sadě.

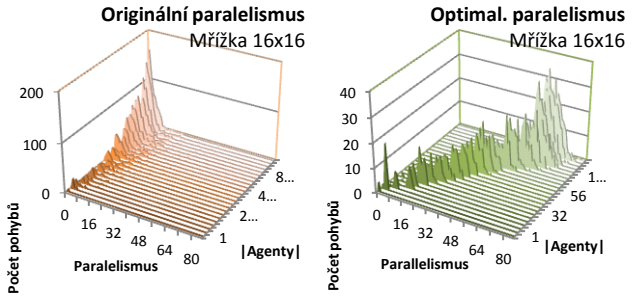
Na mřížce 8×8 COBOPT(BIBOX) generoval skoro optimální řešení pro jednoduché sestavy (stejné jako SATPLAN; stejné nebo lepší než WHCA*) - Obrázek 2. Nicméně,

nejzajímavějších výsledků bylo dosaženo pro obtížné sestavy, kde docházelo ke kompresi až faktoru $\frac{1}{3}$ vzhledem k původnímu výchozímu řešení. Ačkoli nelze říci, zda bylo dosaženo optimum, či jak daleko od něj se výsledné řešení nacházelo, výsledky demonstrují, že optimalizační proces COBOPT dokáže původní řešení výrazně zkrátit.

Počet iterací, za který bylo dosaženo pevného bodu, se pohyboval mezi 1 a 20 s mediánem 7 na mřížce 8×8. Počet řešení SATu se pohyboval mezi 13 a 194. Preferováno je tedy relativně četné řešení menších instancí SATu.

Doba běhu[1] je ukázána na Obrázek 3. Navzdory mnohým voláním řešícího systému pro SAT je celková doba běhu přijatelná. Navíc je nutno uvážit, že metoda COBOPT je velmi snadno implementovatelná ve vícevláknovém prostředí. Škálovatelnost je tedy pozitivním aspektem navržené metody. Jestliže je algoritmus pro vygenerování výchozího řešení dostatečně rychlý, lze COBOPT považovat za *any-time* metodu – to znamená, že v libovolném okamžiku může být výpočet přerušen a uživatel přesto obdrží nějaké řešení.

Abychom zjistili, co se děje při optimalizaci pomocí navržené metody, zkoumali jsme distribuci počtu pohybů, které jsou vykonány paralelně – Obrázek 4 (výsledky jsou ukázány pro větší mřížku velikosti 16×16). Je zřejmé, že původní výchozí řešení trpí tím, že obsahuje mnoho uváznutých agentů, které musí čekat, než se jim uvolní cesta. V optimalizovaných řešeních se pokud možno co nejvyšší počet agentů pohybuje směrem k cíli – agenty využívají téměř všechen dostupný prostor, aby se mohly pohybovat směrem cíli.



Obrázek 4. *Distribuce paralelismu na mřížce 16×16.* Téměř všechen volný prostor je využit k pohybu v optimalizovaném řešení.

6 Diskuse a závěr

Byla představena nová technika nazvaná COBOPT na řešení úlohy kooperativního hledání cest založená na řešení SATu. Aby bylo možné používat k řešení kooperativního hledání cest systém pro SAT, navrhli jsme speciální doménově závislé kódování úlohy jako výrokové formule. Navržené kódování využívá vlastností úlohy, aby bylo dosaženo jeho co nejmenší velikosti a efektivity řešení.

Bylo ukázáno, že metoda COBOPT dokáže generovat řešení, která jsou délkově blízká optimu, nebo jsou alespoň dobré kvality v případech, kdy je prostředí vysoce obsazené agenty, což jsou právě ty případy, kdy ostatní existující metody selhávají.

[1] Všechna experimentální měření probíhala na systému s 6 jádrovým CPU Intel Xeon 2.0GHz s 12GiB RAM pod operačním systémem Linux (jádro 2.6.24-19).

V experimentech se podařilo uspokojivě vyřešit instance na 4-propojené mřížce velikosti 8×8 s obsazeností až 80%. Jedním z pozitivních aspektů nové metody je možnost snadného přizpůsobení pro více-procesorové architektury.

Reference

1. Eén, N., Sörensson, N. *An Extensible SAT-solver.* Proceedings of Theory and Applications of Satisfiability Testing (SAT 2003), pp. 502-518, LNCS 2919, Springer, 2004.
2. Huang, R., Chen, Y., Zhang, W. *A Novel Transition Based Encoding Scheme for Planning as Satisfiability.* Proceedings AAAI 2010, AAAI Press, 2010.
3. Kautz, H., Selman, B. *Unifying SAT-based and Graph-based Planning.* Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI 1999), pp. 318-325, Morgan Kaufmann, 1999.
4. Luna, R., Berkis, K., E. *Push-and-Swap: Fast Cooperative Path-Finding with Completeness Guarantees.* Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011), pp. 294-300, IJCAI/AAAI Press, 2011.
5. Ratner, D., Warmuth, M. K. *Finding a Shortest Solution for the N × N Extension of the 15-PUZZLE Is Intractable.* Proceedings of AAAI 1986, pp. 168-172, Morgan Kaufmann, 1986.
6. Rintanen, J. *Compact Representation of Sets of Binary Constraints.* Proceedings of the 17th European Conference on Artificial Intelligence (ECAI 2006), pp. 143-147, IOS Press, 2006.
7. Ryan, M. R. K. *Exploiting Subgraph Structure in Multi-Robot Path Planning.* Journal of Artificial Intelligence Research (JAIR), Volume 31, pp. 497-542, AAA Press, 2008.
8. Silver, D. *Cooperative Pathfinding.* Proceedings of the 1st Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE 2005), pp. 117-122, AAAI Press, 2005.
9. Standley, T. S. *Finding Optimal Solutions to Cooperative Pathfinding Problems.* Proceedings of the 24th Conference on Artificial Intelligence (AAAI 2010), AAAI Press, 2010.
10. Standley, T. S., Korf, R. E. *Complete Algorithms for Cooperative Pathfinding Problems.* Proceedings of IJCAI 2011, 668-673, IJCAI/AAAI Press, 2011.
11. Surynek, P. *A Novel Approach to Path Planning for Multiple Robots in Bi-connected Graphs.* Proceedings of the International Conference on Robotics and Automation (ICRA 2009), pp. 3613-3619, IEEE Press, 2009.
12. Wang, K. C., Botea, A. *MAPP: a Scalable Multi-Agent Path Planning Algorithm with Tractability and Completeness Guarantees.* JAIR, Volume 42, pp. 55-90, AAAI Press, 2011.

Annotation:

Optimal Cooperative Path Planning through Satisfiability Solving

A novel approach to cooperative path-planning is presented. A SAT solver is used not to solve the whole instance but for optimizing the makespan of a sub-optimal solution. This approach is trying to exploit the ability of state-of-the-art SAT solvers to give a solution to relatively small instance quickly. A sub-optimal solution to the instance is obtained by some existent method first. It is then submitted to the optimization process which decomposes it into small subsequences for which optimal solutions are found by a SAT solver. The new shorter solution is subsequently obtained as concatenation of optimal sub-solutions.