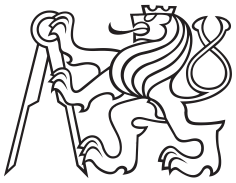
Diplomová práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra řídicí techniky

Plánování pro systém MoleMOD

Bc. Michal Urválek

Vedoucí: RNDr. Miroslav Kulich, Ph.D.
Obor: Kybernetika a robotika
Studijní program: Kybernetika a robotika
Leden 2020

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Urválek** Jméno: **Michal** Osobní číslo: **434937**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra řídicí techniky**

Studijní program: **Kybernetika a robotika**

Studijní obor: **Kybernetika a robotika**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Plánování pro systém MoleMOD

Název diplomové práce anglicky:

Planning for the MoleMOD system

Pokyny pro vypracování:

Cílem práce je navrhnout plánovací algoritmus pro systém MoleMOD, který vzniká na Fakultě architektury ČVUT v Praze v ateliéru doc. Floriána. MoleMOD je revoluční systém speciálně navržených robotů, který umožní autonomně skládat stavební moduly ('cihly') do složitějších konstrukcí. Systém je unikátní v tom, že v modulech jsou cesty, kterými jsou roboty ve tvaru červa schopny procházet a moduly přemisťovat.
V práci by student řešil situaci, kdy jsou na 'staveniště' přiveženy stavební moduly složené v kompaktním tvaru (např. krychle) a ty musí být postupným přemisťováním roboty poskládány do požadovaného finálního tvaru. Úkolem studenta bude navrhnout plánovací algoritmus, který vytvoří plán pohybu pro jednotlivé roboty realizujících toto přemístění. Konkrétní úkoly jsou pak následující:
1. Seznámit se problematikou prohledávání stavového prostoru a zejména s návrhem heuristik pro prohledávací algoritmy.
2. Realizovat grafické rozhraní pro systém MoleMOD.
3. Navrhnout vhodnou reprezentaci stavového prostoru pro systém MoleMOD.
4. Navrhnout a implementovat prohledávací algoritmus a ověřit jeho chování.
5. Získané poznatky a navržené postupy zdokumentovat.

Seznam doporučené literatury:

[1] Petrš, J.: MoleMOD. http://www.studioflorian.com/projekty/347-jan-petrs-molemod (2017)
[2]. Petrš, J., Havelka, J., Florián, M., Novák, J.: MoleMOD - on design specification and applications of a self-reconfigurable constructional robotic system. In: Fioravanti, A., Cursi, S., Elahmar, S., Gargaro, S., Loffreda, G., Novembri, G., Trento, A. (eds.) ShoCK! - Sharing Computational Knowledge! - Proceedings of the 35th eCAADe Conference. vol. 2, pp. 159–166. Fioravanti, A, Cursi, S, Elahmar, S, Gargaro, S, Loffreda, G, Novembri, G, Trento, A (eds.) (2017)
[3] M. Brejchová, M. Kulich, J. Petrš, and L. Přeučil: Modelling, simulation, and planning for the MoleMOD system, Proceedings of Modelling and Simulation for Autonomous Systems (MESAS) Conference 2018

Jméno a pracoviště vedoucí(ho) diplomové práce:

RNDr. Miroslav Kulich, Ph.D., inteligentní a mobilní robotika CIIRC

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **24.01.2019** Termín odevzdání diplomové práce: **24.05.2019**

Platnost zadání diplomové práce: **20.09.2020**

RNDr. Miroslav Kulich, Ph.D.	prof. Ing. Michael Šebek, DrSc.	prof. Ing. Pavel Ripka, CSc.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

Poděkování

Tímto bych chtěl poděkovat vedoucímu diplomové práce RNDr. Miroslavu Kulichovi, Ph.D. za připomínky a cenné rady při řešení zadané úlohy. Dále bych chtěl poděkovat rodičům a přátelům za podporu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 7. ledna 2020

. . . . . .

Abstrakt

Tato diplomová práce se zabývá návrhem plánovacího algoritmu pro modulární robotický systém MoleMOD. V úvodní části obsahuje popis systému MoleMOD. V další části je zavedena terminologie použitá pro popis problematiky týkající se omezení a pravidel návrhu algoritmu. Následuje návrh samotného algoritmu, který je rozdělen do tří částí: třídění, plánování a optimalizace. V závěrečných kapitolách je popis implementace algoritmu a zhodnocení výsledků testování daného algoritmu.

Klíčová slova: robot, plánování, A*, heuristiky, MoleMOD

Abstract

This diploma thesis deals with design of planning algorithm for modular robotic system MoleMOD. The introductory part contains a description of the MoleMOD system. The next part introduces terminology used to describe problems related to constraints and algorithm design rules. The following part describes the design of the algorithm itself, which is divided into three parts: sorting, planning and optimization. In the final chapters there is a description of the implementation of the algorithm and evaluation of the results of the algorithm testing.

Keywords: robot, planning, A*, heuristics, MoleMOD

Title translation: Planning for the MoleMOD system

Obsah

Obrázky

Tabulky

Kapitola 1

Úvod

Žijeme v době, kdy roboti pomalu, avšak jistě začínají přebírat práci, která až dosud byla doménou lidí. Robotizace již velmi pokročila v průmyslové výrobě. Roboti pomáhají v lékařství, a již delší dobu jsou nepostradatelní pro vojenské účely. Stroje přebírají mechanickou činnost i kancelářským pracovníkům napříč obory. Veškerá tato automatizace dává lidem možnost se věnovat více tvůrčí práci a kreativním činnostem.

Tato diplomová práce se zabývá systémem MoleMOD, který vznikl na fakultě architektury ČVUT v Praze. Tento systém přináší automatizaci do oboru stavebnictví, architektury a má velmi široké uplatnění. Jediné co je třeba, je dostatek vhodného stavebního materiálu. S pomocí systému MoleMOD lze vytvořit stavby od provizorních mostů po nejmodernější architekturu. MoleMOD se skládá z aktivních prvků - robotů a pasivních prvků - bloků.

Tato práce se zabývá úlohou, kdy je na staveniště dovezena skupina bloků a robotů. Cílem práce je navrhnout plánovací algoritmus pro efektivní sestavení pasivních prvků do požadovaného objektu. Tato práce nezávisle navazuje na předchozí práci, která se zabývala plánováním pro systém MoleMOD ve 2D prostoru [3].

Klíčovým přínosem této práce je podrobný matematický popis celého systému, který lze použít jako základ pro další úlohy spojené se systémem MoleMOD. Dalším přínosem je návrh plánovacího algoritmu ve 3D prostoru, který byl následně implementován a rovněž bylo vytvořeno prostředí pro vizualizaci simulací, za použití jazyka C++.

V následující Kapitole 2 je podrobný popis samotného systému MoleMOD. Kapitola 3 zavádí potřebnou terminologii, definuje a popisuje omezení systému. Na základě zavedené terminologie popisuje systém MoleMOD a jeho možné operace. V Kapitole 4 je popsán navržený plánovací algoritmus, skládající se ze tří fází: třídění, plánování pohybů bloků/robotů a optimalizace. V Kapitole 5 je popis implementace a simulačního prostředí. Výsledky navržených experimentů jsou zdokumentovány v Kapitole 6. Celá práce je shrnuta v Kapitole 7.

Kapitola 2

MoleMOD

Zvláště v posledním desetiletí se zvýšil zájem o modulární robotické systémy [11]. Modulární robotické systémy mají výhody, jako je nízká cena, robustnost a univerzálnost. Jsou vhodné pro hromadnou výrobu, čímž se snižují náklady. Typickým představitelem robotického modulárního systému je had, který vznikne spojením jednotlivých bloků (robotů). Tyto části mohou, ale nemusejí být stejné. Každá z těchto částí obsahuje mechanické a elektronické části. Roboti se mohou pohybovat samostatně nebo jako celek. Tím získávají robotické systémy tvarovou neurčitost a jsou velmi flexibilní pro různé aplikace.

Systém MoleMOD je specifický druh modulárního systému. Je složen z aktivních prvků a pasivních prvků. Aktivními prvky jsou roboti (Mole), které mají tvar červa a pohybují se uvnitř bloků, kterými také mohou pohybovat. Pasivní části jsou bloky/moduly (MOD). Rozdílem oproti typickému modulárnímu systému je, že výsledná stavba neobsahuje žádné elektronické části. Mechanických částí je minimum a jedná se pouze o šrouby spojující jednotlivé bloky. Veškerá elektronika od řídících jednotek po motory a senzory je pouze v robotických červech. Tím, že systém MoleMOD obsahuje menší počet částí náchylných na poškození, je jeho výhodou snížení nákladů a zvýšení robustnosti,

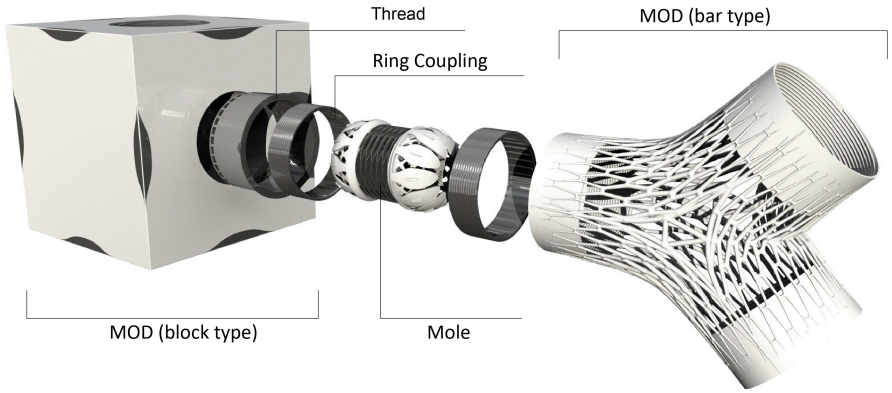
2.1 Konstrukce

Aktivní prvky, kterými jsou roboti se skládají ze tří částí: měkkého těla, mechanické hlavy a rotátorů. Hlavní účel otočné hlavy je otáčení speciálních šroubů sloužících pro spojení jednotlivých bloků. Další funkcí otočné hlavy je umožnit robotům se pohybovat vně systému bloků. Pro pohyb uvnitř bloků slouží měkké tělo, které zároveň zaručuje přesný pohyb robota. Sekundární

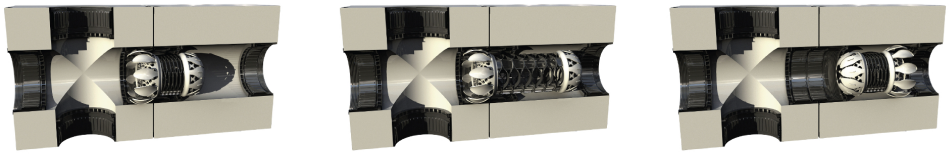
funkcí je manipulace s samotnými bloky. Celá konstrukce je poháněna pomocí rotátorů.

Pasivní prvky systému bloky tvoří výslednou stavbu. Výhodou tohoto systému je, že jednotlivé bloky nemusí mít stejný tvar. Podmínkou je, aby byly průchozí pro roboty a mohly být připojeny k ostatním blokům pomocí průchozích šroubů. Tyto šrouby slouží ke spojování dvou sousedních bloků a jsou navrženy tak, aby skrz tyto šrouby mohli procházet roboti.

Bloky lze vyrobit z různých materiálů od recyklovaných plastů přes dřevo, hliník apod. Hmotnost modulů je vzhledem ke konstrukci robotů hraje důležitou roli, lehčí materiál je výhodnější pro snazší manipulaci.



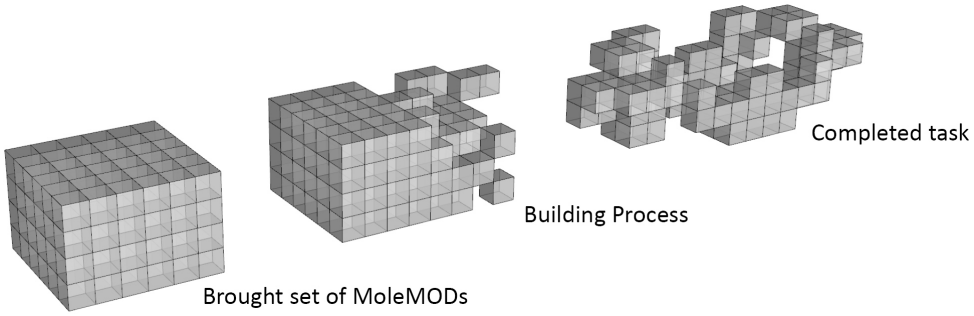
Obrázek 2.1: Jednotlivé části systému MoleMOD (od Jana Petrše [1])



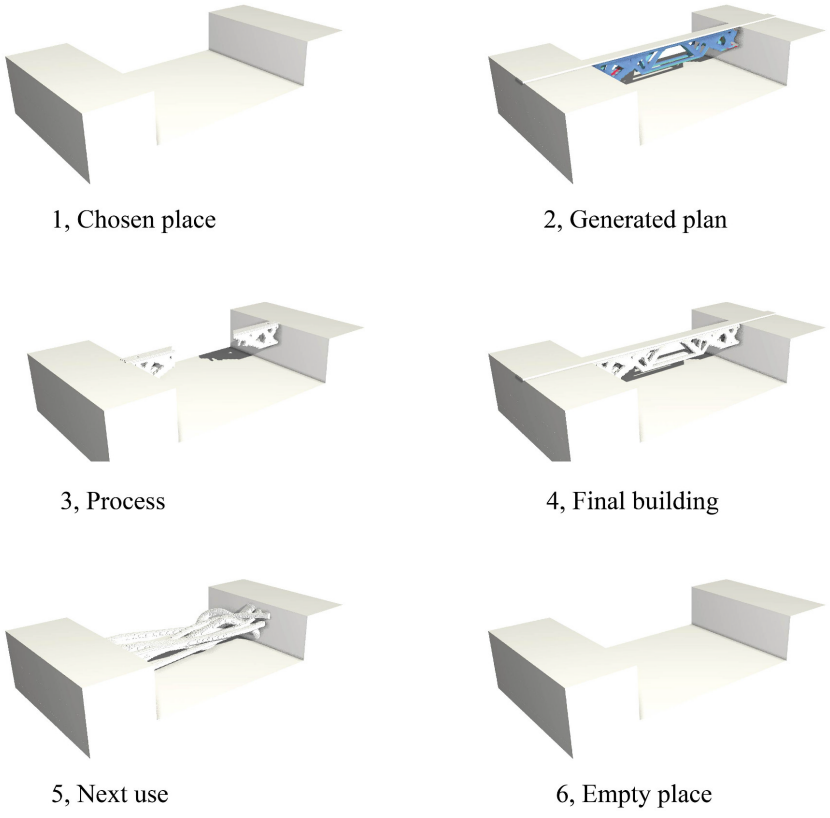
Obrázek 2.2: Řez a umístění robota v kostce (od Jana Petrše [1])

2.1.1 Aplikace

Systém MoleMOD se primárně zaměřuje na budoucí architekturu, která by měla být adaptivní a pohyblivá. Mimo architekturu lze tento systém také využít například při stavbu mostů, viz Obrázek 2.4, a dalších staveb na těžko dostupných místech.



Obrázek 2.3: Princip aplikace systému MoleMOD (od Jana Petrše [1])



Obrázek 2.4: Využití systému MoleMOD (od Jana Petrše [1])

Kapitola 3

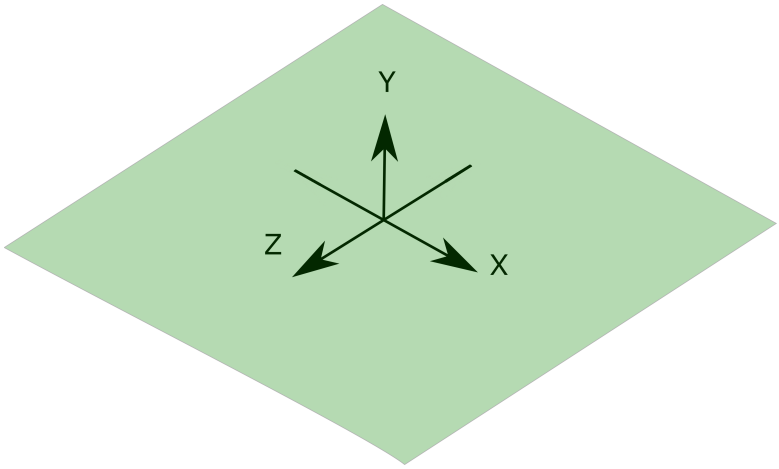
Definice úlohy

3.1 Definice pojmů a terminologie

K popisu řešení úlohy a plánovacího algoritmu je použita terminologie nadefinovaná níže. Plánování probíhá ve 3D prostoru.

Blok: Pasivní prvek systému MoleMOD ve tvaru krychle, která je průchozí ve všech směrech dále označována jako $c \in C$, kde C je množina všech bloků. Počet bloků označíme $N_c \in \mathbb{N}$.
Robot: Aktivní prvek systému MoleMOD ve tvaru červa, který prolézá skrz bloky a manipuluje s nimi, dále označován jako $r \in R$, kde R je množina všech robotů. Robot se pohybuje pouze uvnitř bloků. Počet robotů označíme $N_r \in \mathbb{N}$.
Souřadnicový systém: S ohledem na vizualizační aplikaci byl převzat souřadnicový systém, který používá námi zvolená grafická knihovna, viz Obrázek 3.1. Jedná se o pravotočivý kartézský souřadnicový systém. Osy x a z leží v rovině podstavy. Osa y je kolmá k podstavě a směřuje nahoru.
Stav robota: Stav robota s^r definuje vektor $s^r = (x, y, z)$ určující pozici robota, kde $x, z \in \mathbb{Z}$ a $y \in \mathbb{N}_0$, $s^r \in S_R$, kde S_R je množina všech stavů robota.
Stav bloku: Stav bloku s^c definuje vektor $s^c = (x, y, z)$ určující pozici bloku, kde $x, z \in \mathbb{Z}$ a $y \in \mathbb{N}_0$, $s^c \in S_C$, kde S_C je množina všech stavů bloku.
Stav: Stav (systému) $\mathscr{S}$ je množina stavů kostek a robotů. $\mathscr{S} =< s^{c1}, s^{c2}, ..., s^{cN_c}, s^{r1}, s^{r2}, ..., s^{rN_r} >$, kde s^{ci} je stav i-té kostky a s^{ri} je stav i-tého robota.
Akce robota: Mějme stavy systému $\mathscr{S}_i$ a $\mathscr{S}_{i+1}$. Akce a_r je definována jako zobrazení v množině stavů robotů $a_r : S_R \to S_R$ ze stavu $s_i^r \in \mathscr{S}_i$ do stavu $s_{i+1}^r \in \mathscr{S}_{i+1}$ pro které platí:

$$a_r(s_i^{rk}) = s_{i+1}^{rk} \tag{3.1}$$



Obrázek 3.1: Souřadnicový systém

Přechody mezi stavy robotů respektují pravidla a omezení popsané v následujících kapitolách.
Akce kostky: Mějme stavy systému $\mathscr{S}_i$ a $\mathscr{S}_{i+1}$. Akce a_c je definována jako zobrazení v množině stavů bloků $a_c : S_C \rightarrow S_C$ ze stavu $s_i^c \in \mathscr{S}_i$ do stavu $s_{i+1}^c \in \mathscr{S}_{i+1}$ pro které platí:

$$a_c(s_i^{ck}) = s_{i+1}^{ck} \tag{3.2}$$

Přechody mezi stavy bloků jsou prováděny pomocí akcí robotů, které respektují pravidla a omezení popsané v následujících kapitolách.
Plán robota: Mějme stavy robota s_A^r a s_B^r. Plán robota $route_R(s_A^r, s_B^r)$ je posloupnost stavů, kterými musí robot projít ze stavu A do stavu B.
$route_R(s_A^r,\ s_B^r) = < s_0^r = s_A^r, s_1^r, ..., s_{n-1}^r, s_n^r = s_B^r >$, pro které platí $a_r(s_i^r) = s_{i+1}^r$, kde $i \in < 0, n-1 >$.
Plán bloku: Mějme stavy bloků s_A^c a s_B^c. Plán bloku $route_C$ je posloupnost stavů, kterými musí robot projít ze stavu A do stavu B. $route_C(s_A^c,\ s_B^c) = < s_0^c = s_A^c, s_1^c, ..., s_{n-1}^c, s_n^c = s_B^c >$, pro které platí $a_c(s_i^c) = s_{i+1}^c$, kde $i \in < 0, n-1 >$.
Plán úlohy/stavby: Mějme počáteční (zadaný) stav $\mathscr{S}_0$ a cílový (požadovaný) stav $\mathscr{S}_F$. Plán plánovací úlohy $\mathscr{S}_C$ je posloupnost stavů, kterými musí systém projít k dosažení cílového stavu. $\mathscr{S}_C = (\mathscr{S}_0, \mathscr{S}_1, ..., \mathscr{S}_F)$.

3.2 Pravidla a omezení systému MoleMOD

Pro řešení plánovací úlohy byla aplikována následující pravidla a omezení (omezení vychází ze samotné konstrukce systému MoleMOD a byla částečně upravena pro řešení této úlohy):

1) *Pohyb robota uvnitř bloku*: Samotný robotický systém je navržen tak, že je možné, aby se roboti pohybovali mimo stavbu. Bylo dáno omezení, že se roboti smějí pohybovat pouze uvnitř stavby. Mohlo by se stát, že v důsledku okolního prostředí se robot poškodí nebo se dostane mimo dosah stavby.
2) *Uvnitř bloku maximálně jeden robot*: Pro jednodušší provádění akcí bloků a robotů bylo stanoveno, že v každém bloku bude nejvýše jeden robot. Předejde se tím uvíznutí a zaseknutí robotů.
3) *Všechny bloky jsou stejné*: Systém MoleMOD umožňuje použít různé druhy bloků. Mohou být krychlové, ale i se zaoblenými rohy nebo jinak upravené, aby bylo možné dělat designové architektonické návrhy. Z důvodu možnosti obecnějšího řešení a zajištění existence řešení bylo zavedeno opatření, že všechny bloky musí být stejné a navíc splňovat podmínky uvedené v dalších bodech.
4) *Krychlové bloky*: Jako unifikovaný tvar byla zvolena krychle. Důvodem je, že je to v systému MoleMOD nejčastěji používaný prvek. Mimo jiné nabízí nejširší možnosti pohybů a tím lze snadno dosáhnout všech požadovaných staveb.
5) *Průchodnost bloků ve všech směrech*: Teoreticky systém MoleMOD umožňuje využití bloků, které mají některé stěny neprůchozí, ale vzhledem k požadavku na velikost stavby a také, aby roboti neopouštěli bloky, bylo zavedeno toto omezení. Mimo jiné se tím i zabrání situaci, kdy algoritmus nemůže najít další možný pohyb jakýmkoliv blokem.
6) *Nelze přesouvat blok s robotem*: Stav žádného z robotů nesmí být shodný se stavem bloku vykonávajícího akci: máme $\mathscr{S}_A = (s_A^{c1}, ..., s_A^{cN_c}, s_A^{r1}, ..., s_A^{rN_r})$, potom $a_c(s_A^{ci}) = s_B^{ci}$, pokud $\forall s_A^{rj} \in \mathscr{S}_A : \ s_A^{rj} \neq s_A^{ci} \wedge s_A^{rj} \neq s_B^{ci}$, kde $a_c(s_A^{ci})$ je $i-tý$ blok ve stavu A, s_A^{rj} je $j-tý$ robot ve stavu A a s_B^{ci} je $i-tý$ blok ve stavu B. Důvodem je, že robot, který se nachází uvnitř přesouvaného bloku, by mohl blokovat prostor pro robota/y, kteří vykonávají pohyb daného bloku. Při nedodržení doporučení, že by hmotnost jednotlivých bloků měla být co nejmenší, by mohlo dojít k nepředvídatelnému chování, jako například k vypadnutí kostky z držení robotů.
7) *Celistvost stavby*: V jakémkoliv kroku algoritmu musí platit, že existuje cesta robota mezi dvěma libovolnými bloky stavby: $\forall(s_k^{ci}, s_k^{cj} \in \mathscr{S}_k) \ \exists route_R(s_k^{ci}, \ s_k^{cj})$. Tedy není dovolené, aby nějaký z bloků ležel mimo stavbu a nebylo s ním možné manipulovat, aniž by některý z robotů opustil blok, viz pravidlo 1).
8) *Maximální počet robotů*: V souvislosti s předchozími pravidly musí být zajištěno, aby počet robotů byl minimálně o jedna menší než je počet bloků $N_C > N_R$. V opačném případě by nebylo možné na základě pravidel 6) a 1) pohnout žádnou kostkou.
9) *Minimální počet robotů*: Design samotného robotického systému předurčuje, že pro provedení všech možných pohybů blokem je minimální počet robotů 2 $N_R \geq 2$.
10) *Blokování akcí bloků*: Akci pohybu bloku lze provést pouze v případě, že blok není ve svém pohybu blokován ostatními bloky. Všechny přípustné pohyby včetně jejich omezení jsou popsány v Kapitole 3.3.2.

3.3 Popis akcí bloků a robotů

Všechny akce popsané v této části pro svou proveditelnost předpokládají platnost všech pravidel zmíněných v Kapitole 3.2.

3.3.1 Akce robotů

Roboti mohou provádět dva typy akcí: změnu svého stavu a pohyb blokem. Samotná realizace pohybu bloku není součástí plánování. Tato kapitola popisuje pouze změnu stavu robota.

Z pohledu akcí robotů je celá stavba ortogonální soustavou bloků. Roboti mohou provádět následujících 6 akcí za předpokladu, že pro oba stavy robota $s^r_{x_i,y_i,z_i} \in \mathscr{S}_i$ a $s^r_{x_{i+1},y_{i+1},z_{i+1}} \in \mathscr{S}_{i+1}$ jsou bloky ve stavech $\{s^c_{x_i,y_i,z_i}, s^c_{x_{i+1},y_{i+1},z_{i+1}}\} \in \mathscr{S}_i \wedge \{s^c_{x_i,y_i,z_i}, s^c_{x_{i+1},y_{i+1},z_{i+1}}\} \in \mathscr{S}_{i+1}$:

$$s^r_{x,y,z} \in \mathscr{S}_i : s^c_{x,y,z} \in \mathscr{S}_i \wedge s^c_{x,y,z} \in \mathscr{S}_{i+1} \tag{3.3}$$

$$a^{x+}_r(s^r_{x,y,z}) = s^r_{x+1,y,z} \in \mathscr{S}_{i+1}\ pokud\ s^c_{x+1,y,z} \in \mathscr{S}_i,\ \wedge s^c_{x+1,y,z} \in \mathscr{S}_{i+1} \tag{3.4}$$

$$a^{x-}_r(s^r_{x,y,z}) = s^r_{x-1,y,z} \in \mathscr{S}_{i+1}\ pokud\ s^c_{x-1,y,z} \in \mathscr{S}_i \wedge s^c_{x-1,y,z} \in \mathscr{S}_{i+1} \tag{3.5}$$

$$a^{y+}_r(s^r_{x,y,z}) = s^r_{x,y+1,z} \in \mathscr{S}_{i+1}\ pokud\ s^c_{x,y+1,z} \in \mathscr{S}_i \wedge s^c_{x,y+1,z} \in \mathscr{S}_{i+1} \tag{3.6}$$

$$a^{y-}_r(s^r_{x,y,z}) = s^r_{x,y-1,z} \in \mathscr{S}_{i+1}\ pokud\ s^c_{x,y-1,z} \in \mathscr{S}_i \wedge s^c_{x,y-1,z} \in \mathscr{S}_{i+1} \tag{3.7}$$

$$a^{z+}_r(s^r_{x,y,z}) = s^r_{x,y,z+1} \in \mathscr{S}_{i+1}\ pokud\ s^c_{x,y,z+1} \in \mathscr{S}_i \wedge s^c_{x,y,z+1} \in \mathscr{S}_{i+1} \tag{3.8}$$

$$a^{z-}_r(s^r_{x,y,z}) = s^r_{x,y,z-1} \in \mathscr{S}_{i+1}\ pokud\ s^c_{x,y,z-1} \in \mathscr{S}_i \wedge s^c_{x,y,z-1} \in \mathscr{S}_{i+1} \tag{3.9}$$

kde $s^r_{x,y,z}$ je počáteční stav robota, $\mathscr{S}_i$ je aktuální stav systému a $\mathscr{S}_{i+1}$ je stav systému po provedení akce a_r. Musí platit, že se v obou stavech systému nachází bloky ve stavu $s^c_{x,y,z}$ a ve stavu, kterého dosáhne robot po provedení akce.

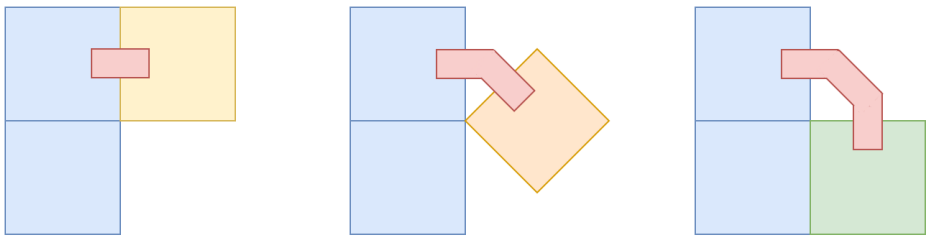
3.3.2 Akce bloků

Pro definici všech možných akcí bloku předpokládejme blok v počátečním stavu $s^c_{x,y,z} \in \mathscr{S}_i$. Po provedení akce se bude blok nacházet ve stavu $s^c_{x_1,y_1,z_1} \in \mathscr{S}_{i+1}$.

Samotná akce bloku je vykonávána pomocí jednoho nebo dvou robotů (dle typu akce bloku), kteří musí být v definovaných stavech, viz ukázka na Obrázku 3.2. Robot je na obrázku znázorněn červenou barvou. Ostatní barvy použité v obrázku odpovídají popisu barev uvedenému níže. U dalších ukázek

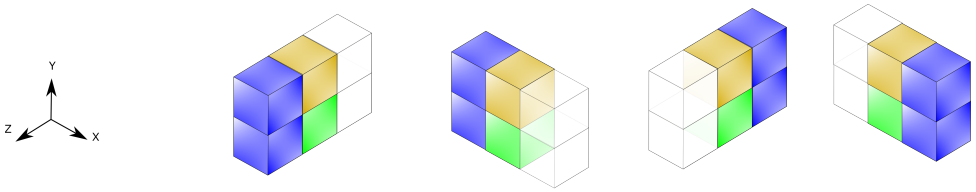
akcí bloků není rozmístění robotů kvůli přehlednosti zobrazeno v jednotlivých ilustracích. Je pouze uvedeno ve výčtu požadavků a omezení nutných pro provedení dané akce bloku.

Pro popis akcí bloků (grafický i matematický) použijeme následující značení:
1) Množina C_M, označená na následujících obrázcích modrou barvou, je množina všech stavů bloků, pro kterou platí: $C_M \subset \mathscr{S}_i \wedge C_M \subset \mathscr{S}_{i+1}$. Tyto stavy bloků musí být obsazené v obou stavech systému, aby bylo možné provést akci bloku.
2) Množina C_P, označená na následujících obrázcích zobrazená jako průsvitné bloky, je množina všech stavů bloků, pro kterou platí: $C_P \cap \mathscr{S}_i = \emptyset \wedge C_P \cap \mathscr{S}_{i+1} = \emptyset$. Tyto stavy bloků musí být volné v obou stavech systému, aby bylo možné provést akci bloku.
3) Stav bloku $s^c_{x,y,z}$, označen na následujících obrázcích žlutou barvou, je počáteční stav bloku, pro kterou platí: $s^c_{x,y,z} \in \mathscr{S}_i \wedge s^c_{x,y,z} \notin \mathscr{S}_{i+1}$.
4) Stav bloku $s^c_{x_1,y_1,z_1}$, označen na následujících obrázcích zelenou barvou, je cílový stavu bloku, pro který platí: $s^c_{x_1,y_1,z_1} \notin \mathscr{S}_i \wedge s^c_{x_1,y_1,z_1} \in \mathscr{S}_{i+1}$.
5) Množina označená C_R, která na následujících obrázcích není znázorněna, je množina všech stavů robotů, pro kterou platí: $C_R \subset \mathscr{S}_i \wedge C_R \subset \mathscr{S}_{i+1}$. Toto rozestavení robotů je nezbytné pro provedení akce kostky.



Obrázek 3.2: Příklad provedení akce za použití robota

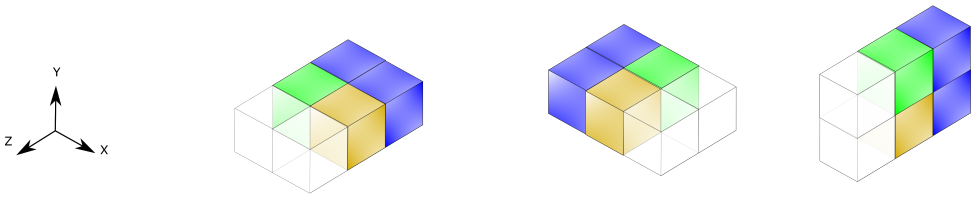
V následujících tabulkách jsou popsány všechny možné akce bloku včetně všech omezení. Každá z akcí má více možných způsobů, jak ji lze provést. Pro popis akcí je v tabulkách použito následující názvosloví: rotace nahoru/dolu znamená posunutí bloku ve směru osy y, rotace doprava/doleva znamená posunutí bloku ve směru osy x a rotace dopředu/dozadu znamená posunutí bloku ve směru osy z, Pro příklad jsou na Obrázku 3.3 uvedeny všechny způsoby pro provedení jedné akce. U ostatních akcí je graficky znázorněn jenom jeden ze způsobů provedení dané akce.



Obrázek 3.3: Všechny způsoby provedení akce A

Akce A - rotace o 90° dolu	
#	$a_c^A(s_{x,y,z}^c) = s_{x,y-1,z}^c$
1	$C_M = \{s_{x,y,z+1}^c, s_{x,y-1,z+1}^c\}$ $C_P = \{s_{x,y,z-1}^c, s_{x,y-1,z-1}^c\}$ $C_R = \{s_{x,y,z+1}^r, s_{x,y-1,z+1}^r\}$
2	$C_M = \{s_{x-1,y,z}^c, s_{x-1,y-1,z}^c\}$ $C_P = \{s_{x+1,y,z}^c, s_{x+1,y-1,z}^c\}$ $C_R = \{s_{x-1,y,z}^r, s_{x-1,y-1,z}^r\}$
3	$C_M = \{s_{x,y,z-1}^c, s_{x,y-1,z-1}^c\}$ $C_P = \{s_{x,y,z+1}^c, s_{x,y-1,z+1}^c\}$ $C_R = \{s_{x,y,z-1}^r, s_{x,y-1,z-1}^r\}$
4	$C_M = \{s_{x+1,y,z}^c, s_{x+1,y-1,z}^c\}$ $C_P = \{s_{x-1,y,z}^c, s_{x-1,y-1,z}^c\}$ $C_R = \{s_{x+1,y,z}^r, s_{x+1,y-1,z}^r\}$

Tabulka 3.1: Popis stavů bloků a robotů pro akci A, viz Obrázek 3.3



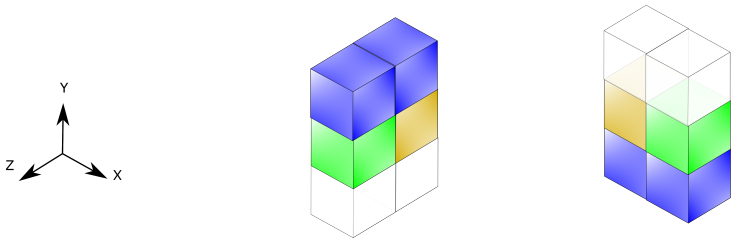
Obrázek 3.4: Příklad akcí B až D: rotace o 90°

Akce B - rotace o 90° doleva		Akce C - rotace o 90° dozadu	
#	$a_c^B(s_{x,y,z}^c) = s_{x-1,y,z}^c$	#	$a_c^C(s_{x,y,z}^c) = s_{x,y,z-1}^c$
1	$C_M = \{s_{x,y,z-1}^c, s_{x-1,y,z-1}^c\}$ $C_P = \{s_{x,y,z+1}^c, s_{x-1,y,z+1}^c\}$ $C_R = \{s_{x,y,z-1}^r, s_{x-1,y,z-1}^r\}$	1	$C_M = \{s_{x-1,y,z}^c, s_{x-1,y,z-1}^c\}$ $C_P = \{s_{x+1,y,z}^c, s_{x+1,y,z-1}^c\}$ $C_R = \{s_{x-1,y,z}^r, s_{x-1,y,z-1}^r\}$
2	$C_M = \{s_{x,y,z+1}^c, s_{x-1,y,z+1}^c\}$ $C_P = \{s_{x,y,z-1}^c, s_{x-1,y,z-1}^c\}$ $C_R = \{s_{x,y,z+1}^r, s_{x-1,y,z+1}^r\}$	2	$C_M = \{s_{x+1,y,z}^c, s_{x+1,y,z-1}^c\}$ $C_P = \{s_{x-1,y,z}^c, s_{x-1,y,z-1}^c\}$ $C_R = \{s_{x+1,y,z}^r, s_{x+1,y,z-1}^r\}$
3	$C_M = \{s_{x,y+1,z}^c, s_{x-1,y+1,z}^c\}$ $C_P = \{s_{x,y-1,z}^c, s_{x-1,y-1,z}^c\}$ $C_R = \{s_{x,y+1,z}^r, s_{x-1,y+1,z}^r\}$	3	$C_M = \{s_{x,y-1,z}^c, s_{x,y-1,z-1}^c\}$ $C_P = \{s_{x,y+1,z}^c, s_{x,y+1,z-1}^c\}$ $C_R = \{s_{x,y-1,z}^r, s_{x,y-1,z-1}^r\}$
4	$C_M = \{s_{x,y-1,z}^c, s_{x-1,y-1,z}^c\}$ $C_P = \{s_{x,y+1,z}^c, s_{x-1,y+1,z}^c\}$ $C_R = \{s_{x,y-1,z}^r, s_{x-1,y-1,z}^r\}$	4	$C_M = \{s_{x,y+1,z}^c, s_{x,y+1,z-1}^c\}$ $C_P = \{s_{x,y-1,z}^c, s_{x,y-1,z-1}^c\}$ $C_R = \{s_{x,y+1,z}^r, s_{x,y+1,z-1}^r\}$

Tabulka 3.2: Popis stavů bloků a robotů pro akce B a C, viz Obrázek 3.4

Akce D - rotace o 90° nahoru	
#	$a_c^D(s^c_{x,y,z}) = s^c_{x,y+1,z}$
1	$C_M = \{s^c_{x,y,z+1}, s^c_{x,y+1,z+1}\}$ $C_P = \{s^c_{x,y,z-1}, s^c_{x,y+1,z-1}\}$ $C_R = \{s^r_{x,y,z+1}, s^r_{x,y+1,z+1}\}$
2	$C_M = \{s^c_{x,y,z-1}, s^c_{x,y+1,z-1}\}$ $C_P = \{s^c_{x,y,z+1}, s^c_{x,y+1,z+1}\}$ $C_R = \{s^r_{x,y,z-1}, s^r_{x,y+1,z-1}\}$
3	$C_M = \{s^c_{x-1,y,z}, s^c_{x-1,y+1,z}\}$ $C_P = \{s^c_{x+1,y,z}, s^c_{x+1,y+1,z}\}$ $C_R = \{s^r_{x-1,y,z}, s^r_{x-1,y+1,z}\}$
4	$C_M = \{s^c_{x+1,y,z}, s^c_{x+1,y+1,z}\}$ $C_P = \{s^c_{x-1,y,z}, s^c_{x-1,y+1,z}\}$ $C_R = \{s^r_{x+1,y,z}, s^r_{x+1,y+1,z}\}$

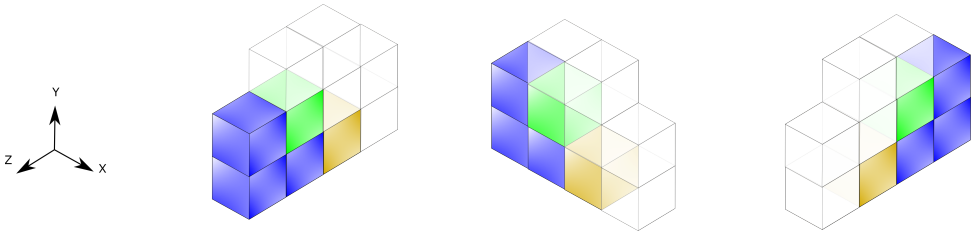
Tabulka 3.3: Popis stavů bloků a robotů pro akci D, viz Obrázek 3.4



Obrázek 3.5: Příklad akcí E a F: rotace o 90°

Akce E - rotace o 90° dopředu		Akce F - rotace o 90° doprava	
#	$a_c^E(s^c_{x,y,z}) = s^c_{x,y,z+1}$	#	$a_c^F(s^c_{x,y,z}) = s^c_{x+1,y,z}$
1	$C_M = \{s^c_{x-1,y,z}, s^c_{x-1,y,z+1}\}$ $C_P = \{s^c_{x+1,y,z}, s^c_{x+1,y,z+1}\}$ $C_R = \{s^r_{x-1,y,z}, s^r_{x-1,y,z+1}\}$	1	$C_M = \{s^c_{x,y+1,z}, s^c_{x+1,y+1,z}\}$ $C_P = \{s^c_{x,y-1,z}, s^c_{x+1,y-1,z}\}$ $C_R = \{s^r_{x,y+1,z}, s^r_{x+1,y+1,z}\}$
2	$C_M = \{s^c_{x+1,y,z}, s^c_{x+1,y,z+1}\}$ $C_P = \{s^c_{x-1,y,z}, s^c_{x-1,y,z+1}\}$ $C_R = \{s^r_{x+1,y,z}, s^r_{x+1,y,z+1}\}$	2	$C_M = \{s^c_{x,y-1,z}, s^c_{x+1,y-1,z}\}$ $C_P = \{s^c_{x,y+1,z}, s^c_{x+1,y+1,z}\}$ $C_R = \{s^r_{x,y-1,z}, s^r_{x+1,y-1,z}\}$
3	$C_M = \{s^c_{x,y+1,z}, s^c_{x,y+1,z+1}\}$ $C_P = \{s^c_{x,y-1,z}, s^c_{x,y-1,z+1}\}$ $C_R = \{s^r_{x,y+1,z}, s^r_{x,y+1,z+1}\}$	3	$C_M = \{s^c_{x,y,z+1}, s^c_{x+1,y,z+1}\}$ $C_P = \{s^c_{x,y,z-1}, s^c_{x+1,y,z-1}\}$ $C_R = \{s^r_{x,y,z+1}, s^r_{x+1,y,z+1}\}$
4	$C_M = \{s^c_{x,y-1,z}, s^c_{x,y-1,z+1}\}$ $C_P = \{s^c_{x,y+1,z}, s^c_{x,y+1,z+1}\}$ $C_R = \{s^r_{x,y-1,z}, s^r_{x,y-1,z+1}\}$	4	$C_M = \{s^c_{x,y,z-1}, s^c_{x+1,y,z-y1}\}$ $C_P = \{s^c_{x,y,z+1}, s^c_{x+1,y,z+1}\}$ $C_R = \{s^r_{x,y,z-1}, s^r_{x+1,y,z-1}\}$

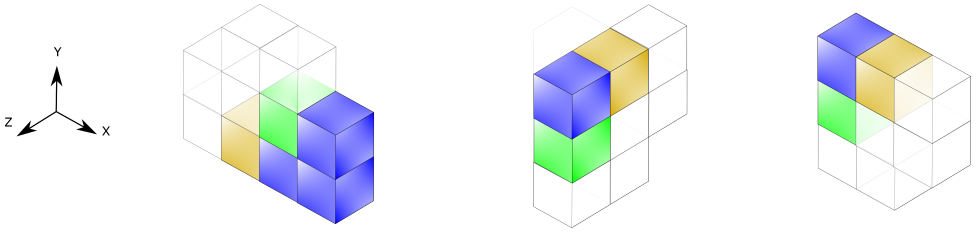
Tabulka 3.4: Popis stavů bloků a robotů pro akce E a F, viz Obrázek 3.5



Obrázek 3.6: Příklad akcí G až I: rotace o 180°

Akce G - rotace o 180° dopředu a nahoru	
#	$a_c^G(s^c_{x,y,z}) = s^c_{x,y+1,z+1}$
1	$C_M = \{s^c_{x,y,z+1}\}$ $C_P = \{s^c_{x,y,z-1}, s^c_{x,y+1,z-1}, s^c_{x,y+1,z}, s^c_{x,y+2,z}, s^c_{x,y+2,z+1}\}$ $C_R = \{s^r_{x,y,z+1}\}$
2	$C_M = \{s^c_{x,y+1,z}\}$ $C_P = \{s^c_{x,y-1,z}, s^c_{x,y-1,z+1}, s^c_{x,y,z+1}, s^c_{x,y,z+2}, s^c_{x,y+1,z+2}\}$ $C_R = \{s^r_{x,y+1,z}\}$
Akce H - rotace o 180° doleva a nahoru	
#	$a_c^H(s^c_{x,y,z}) = s^c_{x-1,y+1,z}$
1	$C_M = \{s^c_{x-1,y,z}\}$ $C_P = \{s^c_{x+1,y,z}, s^c_{x+1,y+1,z}, s^c_{x,y+1,z}, s^c_{x,y+2,z}, s^c_{x-1,y+2,z}\}$ $C_R = \{s^r_{x-1,y,z}\}$
2	$C_M = \{s^c_{x,y+1,z}\}$ $C_P = \{s^c_{x,y-1,z}, s^c_{x-1,y-1,z}, s^c_{x-1,y,z}, s^c_{x-2,y,z}, s^c_{x-2,y+1,z}\}$ $C_R = \{s^r_{x,y+1,z}\}$
Akce I - rotace o 180° dozadu a nahoru	
#	$a_c^I(s^c_{x,y,z}) = s^c_{x,y+1,z-1}$
1	$C_M = \{s^c_{x,y,z-1}\}$ $C_P = \{s^c_{x,y,z+1}, s^c_{x,y+1,z+1}, s^c_{x,y+1,z}, s^c_{x,y+2,z}, s^c_{x,y+2,z-1}\}$ $C_R = \{s^r_{x,y,z-1}\}$
2	$C_M = \{s^c_{x,y+1,z}\}$ $C_P = \{s^c_{x,y-1,z}, s^c_{x,y-1,z-1}, s^c_{x,y,z-1}, s^c_{x,y,z-2}, s^c_{x,y+1,z-2}\}$ $C_R = \{s^r_{x,y+1,z}\}$

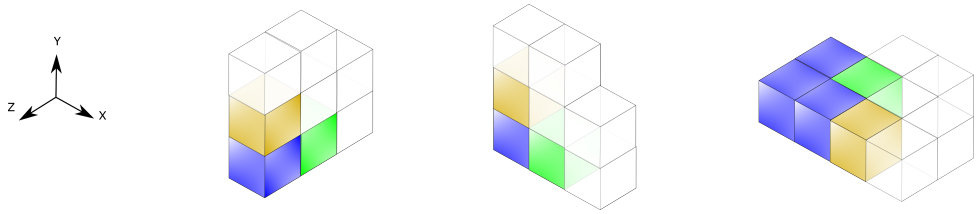
Tabulka 3.5: Popis stavů bloků a robotů pro akce G až I, viz Obrázek 3.6



Obrázek 3.7: Příklad akcí J až L: rotace o 180°

Akce J - rotace o 180° doprava a nahoru	
#	$a_c^J(s^c_{x,y,z}) = s^c_{x+1,y+1,z}$
1	$C_M = \{s^c_{x+1,y,z}\}$ $C_P = \{s^c_{x-1,y,z}, s^c_{x-1,y+1,z}, s^c_{x,y+1,z}, s^c_{x,y+2,z}, s^c_{x+1,y+2,z}\}$ $C_R = \{s^r_{x+1,y,z}\}$
2	$C_M = \{s^c_{x,y+1,z}\}$ $C_P = \{s^c_{x,y-1,z}, s^c_{x+1,y-1,z}, s^c_{x+1,y,z}, s^c_{x+2,y,z}, s^c_{x+2,y+1,z}\}$ $C_R = \{s^r_{x,y+1,z}\}$
Akce K - rotace o 180° dopředu a dolu	
#	$a_c^K(s^c_{x,y,z}) = s^c_{x,y-1,z+1}$
1	$C_M = \{s^c_{x,y,z+1}\}$ $C_P = \{s^c_{x,y,z-1}, s^c_{x,y-1,z-1}, s^c_{x,y-1,z}, s^c_{x,y-2,z}, s^c_{x,y-2,z+1}\}$ $C_R = \{s^r_{x,y,z+1}\}$
2	$C_M = \{s^c_{x,y-1,z}\}$ $C_P = \{s^c_{x,y-1,z+2}, s^c_{x,y,z+2}, s^c_{x,y,z+1}, s^c_{x,y+1,z+1}, s^c_{x,y+1,z}\}$ $C_R = \{s^r_{x,y-1,z}\}$
Akce L - rotace o 180° doleva a dolu	
#	$a_c^L(s^c_{x,y,z}) = s^c_{x-1,y-1,z}$
1	$C_M = \{s^c_{x-1,y,z}\}$ $C_P = \{s^c_{x+1,y,z}, s^c_{x+1,y-1,z}, s^c_{x,y-1,z}, s^c_{x,y-2,z}, s^c_{x-1,y+2,z}\}$ $C_R = \{s^r_{x-1,y,z}\}$
2	$C_M = \{s^c_{x,y-1,z}\}$ $C_P = \{s^c_{x,y+1,z}, s^c_{x-1,y+1,z}, s^c_{x-1,y,z}, s^c_{x-2,y,z}, s^c_{x-2,y-1,z}\}$ $C_R = \{s^r_{x,y-1,z}\}$

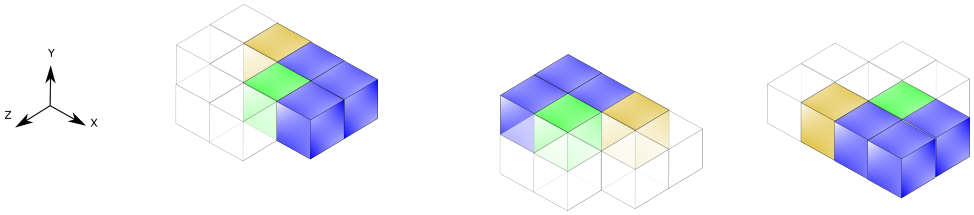
Tabulka 3.6: Popis stavů bloků a robotů pro akce J až L, viz Obrázek 3.7



Obrázek 3.8: Příklad akcí M až O: rotace o 180°

Akce M - rotace o 180° dozadu a dolu	
#	$a_c^M(s^c_{x,y,z}) = s^c_{x,y-1,z-1}$
1	$C_M = \{s^c_{x,y,z-1}\}$ $C_P = \{s^c_{x,y,z+1}, s^c_{x,y-1,z+1}, s^c_{x,y-1,z}, s^c_{x,y-2,z}, s^c_{x,y-2,z-1}\}$ $C_R = \{s^r_{x,y,z-1}\}$
2	$C_M = \{s^c_{x,y-1,z}\}$ $C_P = \{s^c_{x,y-1,z-2}, s^c_{x,y,z-2}, s^c_{x,y,z-1}, s^c_{x,y+1,z-1}, s^c_{x,y+1,z}\}$ $C_R = \{s^r_{x,y-1,z}\}$
Akce N - rotace o 180° doprava a dolu	
#	$a_c^N(s^c_{x,y,z}) = s^c_{x+1,y-1,z}$
1	$C_M = \{s^c_{x+1,y,z}\}$ $C_P = \{s^c_{x-1,y,z}, s^c_{x-1,y-1,z}, s^c_{x,y-1,z}, s^c_{x,y-2,z}, s^c_{x+1,y+2,z}\}$ $C_R = \{s^r_{x+1,y,z}\}$
2	$C_M = \{s^c_{x,y-1,z}\}$ $C_P = \{s^c_{x,y+1,z}, s^c_{x+1,y+1,z}, s^c_{x+1,y,z}, s^c_{x+2,y,z}, s^c_{x+2,y-1,z}\}$ $C_R = \{s^r_{x,y-1,z}\}$
Akce O - rotace o 180° doleva a dozadu	
#	$a_c^O(s^c_{x,y,z}) = s^c_{x-1,y,z-1}$
1	$C_M = \{s^c_{x-1,y,z}\}$ $C_P = \{s^c_{x+1,y,z}, s^c_{x+1,y,z-1}, s^c_{x,y,z-1}, s^c_{x,y,z-2}, s^c_{x-1,y,z-2}\}$ $C_R = \{s^r_{x-1,y,z}\}$
2	$C_M = \{s^c_{x,y,z-1}\}$ $C_P = \{s^c_{x,y,z+1}, s^c_{x-1,y,z+1}, s^c_{x-1,y,z}, s^c_{x-2,y,z}, s^c_{x-2,y,z-1}\}$ $C_R = \{s^r_{x,y,z-1}\}$

Tabulka 3.7: Popis stavů bloků a robotů pro akce M až O, viz Obrázek 3.8



Obrázek 3.9: Příklad akcí P až R: rotace o 180°

Akce P - rotace o 180° doprava a dopředu	
#	$a_c^P(s^c_{x,y,z}) = s^c_{x+1,y,z+1}$
1	$C_M = \{s^c_{x+1,y,z}\}$ $C_P = \{s^c_{x-1,y,z}, s^c_{x-1,y,z+1}, s^c_{x,y,z+1}, s^c_{x,y,z+2}, s^c_{x+1,y,z+2}\}$ $C_R = \{s^r_{x+1,y,z}\}$
2	$C_M = \{s^c_{x,y,z+1}\}$ $C_P = \{s^c_{x,y,z-1}, s^c_{x+1,y,z-1}, s^c_{x+1,y,z}, s^c_{x+2,y,z}, s^c_{x+2,y,z+1}\}$ $C_R = \{s^r_{x,y,z+1}\}$
Akce Q - rotace o 180° doleva a dopředu	
#	$a_c^Q(s^c_{x,y,z}) = s^c_{x-1,y,z+1}$
1	$C_M = \{s^c_{x-1,y,z}\}$ $C_P = \{s^c_{x+1,y,z}, s^c_{x+1,y,z+1}, s^c_{x,y,z+1}, s^c_{x,y,z+2}, s^c_{x-1,y,z+2}\}$ $C_R = \{s^r_{x-1,y,z}\}$
2	$C_M = \{s^c_{x,y,z+1}\}$ $C_P = \{s^c_{x,y,z-1}, s^c_{x-1,y,z-1}, s^c_{x-1,y,z}, s^c_{x-2,y,z}, s^c_{x-2,y,z+1}\}$ $C_R = \{s^r_{x,y,z+1}\}$
Akce R - rotace o 180° doprava a dozadu	
#	$a_c^R(s^c_{x,y,z}) = s^c_{x+1,y,z-1}$
1	$C_M = \{s^c_{x+1,y,z}\}$ $C_P = \{s^c_{x-1,y,z}, s^c_{x-1,y,z-1}, s^c_{x,y,z-1}, s^c_{x,y,z-2}, s^c_{x+1,y,z-2}\}$ $C_R = \{s^r_{x+1,y,z}\}$
2	$C_M = \{s^c_{x,y,z-1}\}$ $C_P = \{s^c_{x,y,z+1}, s^c_{x+1,y,z+1}, s^c_{x+1,y,z}, s^c_{x+2,y,z}, s^c_{x+2,y,z-1}\}$ $C_R = \{s^r_{x,y,z-1}\}$

Tabulka 3.8: Popis stavů bloků a robotů pro akce P až R, viz Obrázek 3.9

Kapitola 4

Plánovací algoritmus

Cílem plánovací úlohy je nalezení plánu stavby $\mathscr{S}_C$ na základě vstupního stavu $\mathscr{S}_0$ a výstupního stavu $\mathscr{S}_F$ systému. Tvorba plánu se řídí pravidly popsanými v předchozích kapitolách. Plánovací algoritmus je rozdělen do tří částí (třídění, plánování, optimalizace), viz Diagram 4.1, podrobněji popsaných v následujících kapitolách.

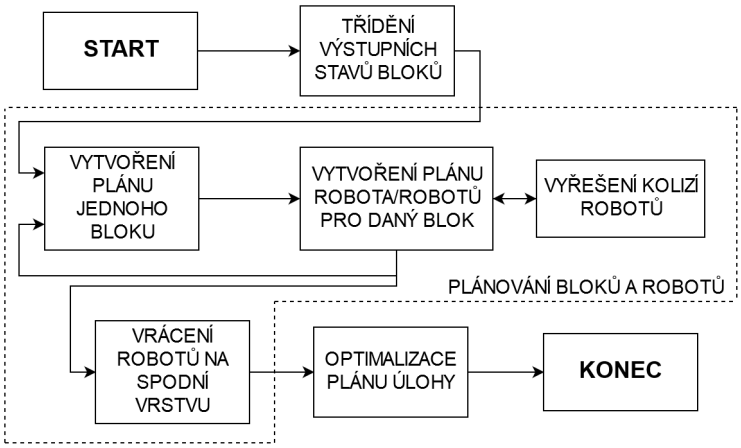
Stavy bloků v množině $\mathscr{S}_0$ jsou uspořádány tak, že přibližně tvoří jednu velkou krychli. Roboti mohou být umístěni libovolně, ale musí se nacházet uvnitř nějakého bloku, tj.: $\forall s_0^{ri} \in \mathscr{S}_0 \exists s_0^{cj} \in \mathscr{S}_0$, kde $i \in < 0, 1, ...N_r >$ a $j \in < 0, 1, ...N_c >$.

Celá stavba probíhá na podložce, která je rovnoběžná s osami x a z, přičemž $y = 0$.

Prvním krokem algoritmu je třídění výstupních stavů bloků. Účelem třídění je určit pořadí ve kterém budou cílové stavy bloků obsazovány. Díky tomu se zrychlí běh algoritmu.

Druhým krokem je vytvoření plánu úlohy. Tato část je na diagramu na Obrázku 4.1 označena jako plánování bloků a robotů. Jde o sekvenční plánování. Nejdříve se vytvoří plán j-tého bloku $route_C(s_A^{cj}, s_B^{cj})$. Na základě tohoto plánu jsou vytvořeny plány dvou robotů $route_R(s_A^{ri}, s_B^{ri})$ a $route_R(s_A^{ri+1}, s_B^{ri+1})$, kteří slouží k vykonání jednotlivých akcí bloku. Spojením těchto tří plánů vznikne kompletní plán, na základě kterého lze dostat blok ze stavu A do stavu B. Výsledný plán splňuje veškeré požadavky a omezení uvedené výše.

Posledním krokem je optimalizace plánu úlohy. Cílem optimalizace je paralelizovat činnost robotů. Účelem paralelizace je zkrácení celkové doby stavění. Optimalizace je provedena překrýváním jednotlivých plánů bloků a robotů.



Obrázek 4.1: Vývojový diagram plánovacího algoritmu

4.1 Třídění výstupních stavů bloků

Účelem třídící fáze je vytvořit z výstupních stavů bloků $\mathscr{S}_F$ posloupnost stavů S^c_{sort}. Tato posloupnost určuje pořadí, v jakém budou obsazovány výstupní stavy bloků.

Posloupnost bloků je vytvářena pomocí následujících pravidel:
1) První stav bloku v posloupnosti $\mathscr{S}_{sort}$ je ten, který je nejblíže počátku souřadné soustavy a současně se nachází ve stavu s nejnižší hodnotou souřadnice y.
2) Pro každý $i+1$ stav bloku v posloupnosti musí platit, že existuje plán robota ze stavu bloku z $i+1$ do stavu 0 (první stav posloupnosti). Rovněž platí, že robot smí provádět akce pouze ve stavech bloků, které byly v posloupnosti uvedené dříve.

Samotné třídění bloků popisuje Algoritmus 1. Nejdříve vybere všechny stavy bloku s^{cj} z výstupních stavů $\mathscr{S}_F$ a vytvoří z nich posloupnost S^c_{sort}.

První krok třídění porovnává velikosti souřadnic y pro dva sousedící stavy v posloupnosti $s^{cj}_{y_j}$ $s^{cj+1}_{y_{j+1}}$. V případě, že jsou velikosti y ve druhém stavu bloku menší, prohodí tyto dva stavy v posloupnosti, viz řádky 6 a 7. Tato fáze končí ve chvíli, kdy jsou všechny stavy bloků seřazené podle souřadnice y (řádky 2-11).

Druhý krok třídění vychází z výsledků prvního kroku třídění. Porovnává Manhattanské vzdálenosti od prvního prvku posloupnosti S^c_{sort}, viz Rovnice 4.1, zvlášť pro každou podmnožinu množiny S^c_{sort}, pro kterou platí, že hodnota souřadnice y je konstantní pro všechny prvky podmnožiny.

$$\begin{gathered} s^{c0}_{xA,yA,zA}, s^{cj}_{xB,yB,zB} \in S^c_{sort} \\ d_M(A,B) = |xA - xB| + |yA - yB| + |zA - zB| \end{gathered} \tag{4.1}$$

V tomto případě je $s^{c0}_{xA,yA,zA}$ první prvek posloupnosti S^c_{sort} a $s^{cj}_{xB,yB,zB} \in S^c_{sort}$ je j-tý prvek posloupnosti. Mějme dva sousedící stavy v posloupnosti $s^{cj}_{y_j}$, $s^{cj+1}_{y_{j+1}}$, pro které platí výše zmíněné. V případě, že je vzdálenost j+1 bloku menší než vzdálenost j-tého bloku od počátku, dojde k prohození těchto dvou stavů posloupnosti (řádky 18 a 19). Tato fáze končí ve chvíli, kdy jsou všechny stavy bloků seřazené podle vzdálenosti pro jednotlivá y (řádky 13-24).

Poslední fáze vychází z výsledků předchozích dvou částí, ale už nemá žádné omezení, které by se týkalo prohazování prvků. Kontroluje, zda existuje plán robota z prvního stavu bloku posloupnosti S^c_{sort} do každého j-tého stavu bloku. Tento plán robota musí procházet pouze stavy bloků na pozicích 0 až j-1 v posloupnosti S^c_{sort}. V případě, že plán robota neexistuje, dojde k prohození dvou stavů bloků v posloupnosti S^c_{sort} (řádky 25-35).

V případě, že nelze seřadit výstupní stavy bloků dle těchto pravidel, nesplňuje zadaná množina výstupních stavů pravidla a omezení, viz Kapitola 3.2.

4.2 Plánování bloků a robotů

Plán úlohy $\mathscr{S}_C$ je posloupnost stavů robotů a bloků vytvořena na základě provádění akcí robotů a bloků. Plán úlohy se skládá z dílčích plánů robotů a bloků. Plány bloků a robotů vnikají nezávisle. Během plánování jsou respektována všechna pravidla a omezení uvedená v Kapitole 3.

Dílčí plán systému je podmnožina $\mathscr{S}_C$ mezi dvěma stavy $\mathscr{S}_A$, $\mathscr{S}_B$. ($\mathscr{S}_A$, $\mathscr{S}_{A+1}, ..., \mathscr{S}_{B+1}, \mathscr{S}_B) \subset \mathscr{S}_C$, přičemž $\mathscr{S}_A =< s^c_A, s^{r1}_{A_1}, s^{r2}_{A_2}, ... >$ a $\mathscr{S}_B =< s^c_B, s^{r1}_{B_1}, s^{r2}_{B_2}, ... >$. Pro přechod mezi těmito stavy platí, že existuje $route_C(s^c_A, s^c_B) = < s^c_0 = s^c_A, s^c_1, ..., s^c_N = s^c_B >$. Pro každou akci bloků $a_c(s^c_k) = s^c_{k+1}$ existují plány robotů $route_R(s^{r1}_{k1}, s^{r1}_{k1+1})$ a $route_R(s^r2_k, s^{r2}_{k2+1})$, kde $k_1, k_2 \in (0, 1, ..., N)$ a N je počet prvků $route_C(s^c_A, s^c_B)$.

Tvorba dílčího plánu probíhá následovně. Z posloupnosti stavů S^c_{sort} vybereme první neobsazený stav bloku $s^c_{xB,yB,zB} \in S^c_{sort}$ v aktuálním stavu systému $\mathscr{S}_k$, $s^c_{xB,yB,zB} \notin \mathscr{S}_A$. V $\mathscr{S}_A$ nalezneme stav bloku s^c_A, který splňuje pravidla a možnosti provádění akcí bloků uvedené v Kapitole 3.2 a jehož Manhattanská vzdálenost do stavu $s^c_{xB,yB,zB}$ je nejkratší. Pro tento blok následně vytvoříme plán bloku $route_C(s^c_A, s^c_B)$.

Plán bloku se skládá ze stavů bloků s^c_k, s^c_{k+1}, mezi kterými se přechází pomocí akce $a_c(s^c_k) = s^c_{k+1}$. Pro provedení každé akce bloku je nutné naplánovat plány robota/ů $route_R(s^{ri}_{k-1}, s^{ri}_k)$, aby byla požadovaná akce bloku proveditelná.

Tvorba plánu robotů probíhá tak, že se jednomu stavu bloku $s^c_k \in \mathscr{S}_k$ přiřadí jeden nebo dva stavy robotů ze stavu systému $\mathscr{S}_k$. Robot/i jsou přiřazováni k blokům postupně na základě předem definovaného pořadí. Tím je zajištěno, že budou všichni roboti využiti rovnoměrně. Pro robota/y je následně vytvořeno několik plánů, které přesouvají roboty mezi jednotlivými

Algorithm 1 Třídění výstupní stavů bloků

1: $S^c_{sort} = \forall s^{cj} \in \mathscr{S}_F,\ notSorted = true$
2: *# třídění vzestupně podle hodnoty souřadnice y*
3: **while** $notSorted$ **do**
4: $notSorted = false$
5: **for** $\forall s^{cj} \in S^c_{sort}$ **do**
6: **if** $s^{cj}_{y_j} > s^{cj+1}_{y_{j+1}}$ **then**
7: $switchPosition(s^{cj}_{y_j}, s^{cj+1}_{y_{j+1}})$
8: $notSorted = true$
9: **end if**
10: **end for**
11: **end while**
12: *# třídění podle vzdálenosti od počátku respektující předchozí třídění*
13: $notSorted = true$
14: **while** $notSorted$ **do**
15: $notSorted = false$
16: **for** $\forall s^{cj} \in S^c_{sort}$ **do**
17: **if** $y1 \in s^{cj}_{x1,y1,z1} = y2 \in s^{cj}_{x2,y2,z2}$ **then**
18: **if** $d_M(s^{cj}_{x1,y,z1}) > d_M(s^{cj+1}_{x2,y,z2})$ **then**
19: $switchPosition(s^{cj}_{x1,y,z1}, s^{cj+1}_{x2,y,z2})$
20: $notSorted = true$
21: **end if**
22: **end if**
23: **end for**
24: **end while**
25: *# třídění založené na existenci cesty robota* $route_R(s^{c0}, s^{cj})$
26: $notSorted = true$
27: **while** $notSorted$ **do**
28: $notSorted = false$
29: **for** $\forall s^{cj} \in S^c_{sort}$ **do**
30: **if** $(not\ route_R(s^{c0}, s^{cj})$ **then**
31: $switchPosition(s^{cj}, s^{cj+1})$
32: $notSorted = true$
33: **end if**
34: **end for**
35: **end while**

stavy potřebnými pro provádění akcí bloků. Během tvorby plánů robotů dochází ke kolizím více robotů tak, jak je popsané v Kapitole 4.2.1.

Výše popsaná tvorba dílčího plánu je postupně aplikována na všechny výstupní stavy v pořadí, v jakém byly setříděné. Jakmile jsou všechny cílové stavy bloků obsazené, jsou vytvořeny plány robotů do stavů bloků s nejnižší hodnotu souřadnice y.

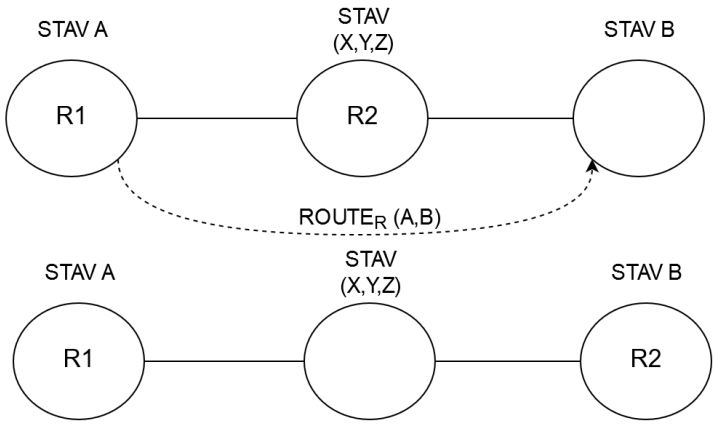
4.2.1 Kolize robotů

Navržený systém tvorby plánů samotných bloků se pro testované případy jevil bezkolizní. Nicméně při tvorbě plánů robotů ke kolizím dochází. Hlavním důvodem kolizí je pravidlo, že stav jednoho bloku může být totožný nejvýše s jedním stavem robota v každém stavu systému. Jsou dva možné typy kolizí: 1) Robot ve stavu s_A^{r1} má jediný existující plán robota $route_R(s_A^{r1}, s_B^{r1})$, který vede přes stav bloku $s_{x,y,z}^c$, který je již obsazen robotem ve stavu $s_{x,y,z}^{r2}$. Naopak robot ve stavu $s_{x,y,z}^{r2}$ nebyl přidělen pro vykonávání akcí právě přesouvaného bloku. V takovémto případě dojde k prohození robotů. Robot ve stavu s_A^{r1} zůstane v tomto stavu. Pro robota ve stavu $s_{x,y,z}^{r2}$ je vytvořen plán $route_R(s_{(x,y,z)}^{r2}, s_B^{r2})$. Toto prohození zůstává pro zbytek plánu stavu bloku s_B^c. Viz Obrázek 4.2 a Tabulka 4.1, kde R1 až R5 jsou roboti a C je blok, kterým pohybují roboti R1 a R3. Jelikož vznikla kolize s robotem, který nebyl přidělen pro přesouvání bloku C, je role robota R1 prohozena s rolí robota R2. Robot R2 následně ve spolupráci s robotem R3 pohybují blokem C.

před kolizí	R1	R3	R4	R2	R5
po kolizi	R2	R3	R4	R1	R5
přidělený blok	C	C	–	–	–

Tabulka 4.1: Tabulka přidělených robotů pro jednotlivé bloky v případě kolize 1

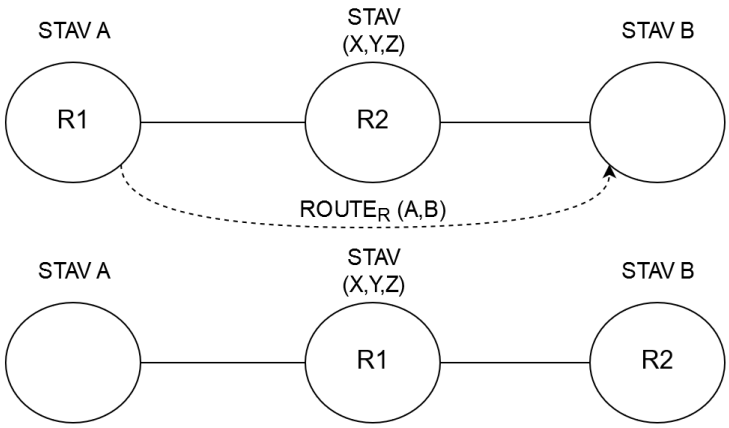
2) Situace je podobná jako v prvním případě. Rozdílem je, že stav blokujícího robota je současně i stavem robota použitého pro vykonání akcí bloku. Na rozdíl od prvního případu dojde k posunutí obou robotů. Pro robota ve stavu s_A^{r1} je vytvořen plán $route_R(s_A^{r1}, s_{x,y,z}^{r2})$ do původního stavu robota $s_{x,y,z}^{r2}$. A pro robota ve stavu $s_{(x,y,z)}^{r2}$ je vytvořen plán $route_R(s_{x,y,z}^{r2}, s_B^{r1})$ do stavu robota s_B^{r1}. Při vykonávání plánů obou robotů je nutné zajistit, aby robot ve stavu $s_{x,y,z}^{r2}$ tento stav opustil a následně může být obsazen druhým robotem. Viz Obrázek 4.3 a Tabulka 4.2, kde R1 až R5 jsou roboti a C je blok, kterým pohybují roboti R1 a R2. V tomto případě mezi sebou kolidují pouze roboti, kteří už byli přiděleni pro přesouvání bloku C. Nedochází tedy ke změně přidělených robotů pro přesun bloku C.



Obrázek 4.2: Znázornění prvního případu kolize robotů

před kolizí	R1	R2	R3	R4	R5
po kolizi	R1	R2	R3	R4	R5
přidělený blok	C	C	–	–	–

Tabulka 4.2: Tabulka přidělených robotů pro jednotlivé bloky v případě kolize 2



Obrázek 4.3: Znázornění druhého případu kolize robotů

4.2.2 Algoritmus A*

K nalezení plánu bloku/robota byl použit prohledávací algoritmus A*. A* je grafový prohledávací algoritmus sloužící k nalezení nejkratší cesty. Princip algoritmu je podobný prohledávání do šířky/hloubky. Na rozdíl od prohledávání do šířky/hloubky se jedná o informované prohledávání. Optimalita algoritmu je dána volbou heuristické funkce.

Heuristická funkce $h(s^{c/r})$ slouží k nastavení chování A* algoritmu. Je to minimální odhad počtu stavů z aktuálního stavu $s_i^{c/r}$ do konečného stavu $s_F^{c/r}$.

Správná volba heuristiky je nezbytná pro nalezení optimální cesty. Heuristika se používá pro výpočet odhadu počtu stavů k dosažení cílového stavu pomocí vzorce:

$$f(s^{c/r}) = h(s^{c/r}) + g(s^{c/r}), \tag{4.2}$$

kde $f(s^{c/r})$ je odhadovaný počet stavů pro dosažení cílového stavu, $g(s^{c/r})$ je počet aktuálně projitých stavů. Funkce $f(s^{c/r})$ se používá k určení pořadí, v jakém se budou procházet jednotlivé stavy.

Aby se jednalo o algoritmus A*, je nutné použití přípustné heuristiky. Pro přípustnou heuristiku platí $0 \leq h() \leq C^*$, kde C^* je skutečný počet stavů pro dosažení cílového stavu, přičemž přípustná heuristika zaručuje optimálnost A* algoritmu.

Pro plánování byly použity dvě heuristiky (Euklidovská vzdálenost a Manhattanská vzdálenost):
Euklidovská vzdálenost, viz Rovnice 4.3, je nejkratší vzdálenost mezi jednotlivými stavy.

$$\begin{gathered} s^{c0}_{xA,yA,zA} \in \mathscr{S}_A, s^{c0}_{xB,yB,zB} \in \mathscr{S}_B \\ d_E(A,B) = \sqrt{(xA - xB)^2 + (yA - yB)^2 + (zA - zB)^2} \end{gathered} \tag{4.3}$$

Tato heuristika patří mezi přípustné heuristiky a je vždy menší nebo rovna vzdálenosti do cílového stavu. Vzhledem k charakteru úlohy a způsobu provádění jednotlivých akcí bloků/robotů by se zdálo vhodnější použít jako heuristickou funkci Manhattanskou vzdálenost, viz Rovnice 4.1. Je také přípustnou heuristikou, a navíc je Manhattanská vzdálenost častěji rovna skutečné vzdálenosti. Porovnání výsledků je uvedeno v Kapitole 6.

A* je založen na práci se dvěma množinami stavů. Množina uzavřených stavů robotů/bloků obsahuje všechny již navštívené stavy robotů/bloků, pro který již byly vygenerovány všechny následující stavy. Množina otevřených stavů obsahuje vygenerované a ještě neexpandované stavy, ze kterých jsou následně generovány nové stavy bloků/robotů. Princip algoritmu A* je takový, že vezme stav z množiny otevřených stavů, který má nejnižší hodnotu funkce $f()$ 4.2. Pro tento stav vygeneruje všechny následující stavy, které vloží dle podmínek, viz Algoritmus 2, do množiny otevřených stavů. Případně nahradí již existující stav v množině otevřených stavů nebo přesune stav z množiny uzavřených stavů do množiny otevřených stavů. Stav, ze kterého byly generovány nové stavy, se přesune z množiny otevřených stavů do množiny uzavřených stavů.

Pro popis Algoritmu 2 bylo použito značení, kde $s^{c/r}$ je stav bloku nebo robota. O je množina otevřených stavů bloků nebo robotů. C je množina všech uzavřených stavů bloků a robotů. Počáteční stav $s_0^{c/r}$ bloku/robota vložíme do množiny otevřených stavů O (řádek 1).
Následující část opakujeme, dokud není nalezen plán bloku/robota nebo dokud není fronta O prázdná (řádky 2 - 27):
Vezmeme stav ve frontě s nejnižší hodnotou funkce $f(s^{c/r}) = h(s^{c/r}) + g(s_a^{c/r})$ a označíme ho za aktuální stav bloku/robota $s_a^{c/r}$, kde $f(s_a^{c/r})$ je odhadovaný počet stavů (délka plánu robotu/robota) pro dosažení cílového stavu,

$g(s_a^{c/r})$ je počet aktuálně projitých stavů a $h(s_a^{c/r})$ je hodnota heuristické funkce pro odhad počtu kroků k dosažení cílového stavu (řádek 3). V případě, že je aktuální stav současně stavem cílovým $s_a^{c/r} = s_F^{c/r}$, je vytvořen plán $route_{C/R}(s_0^{c/r}, s_F^{c/r})$ obsahující všechny stavy potřebné pro dosažení cílového stavu (řádky 4,5). Plán $route_{C/R}(s_0^{c/r}, s_F^{c/r})$ je vytvářen průchodem nalezené cesty v opačném pořadí. Tj. pro stav $s_F^{c/r}$ je nalezen stav, ze kterého jsme se dostali do cílového stavu $s_F^{c/r}$. Dále je nalezen stav, který předcházel tomuto stavu. Takto se pokračuje, až se dostaneme do počátečního stavu $s_0^{c/r}$ a dostaneme kompletní plán cesty $route_{C/R}(s_0^{c/r}, s_F^{c/r})$. V opačném případě jsou pro aktuální stav $s_a^{c/r}$ vygenerovány všechny stavy $s_{si}^{c/r}$, do kterých se lze dostat provedením akce bloku/robota $a_{c/r}(s_a^{c/r}) = s_{si}^{c/r}$ (řádek 7).

Pro každý vygenerovaný stav $s_{si}^{c/r}$ jsou provedeny následující akce (řádky 8-25):

Pro stav $s_{si}^{c/r}$ je spočítána cena $(c/r)_{cost} = g(s_a^{c/r}) + w(s_a^{c/r}, s_{si}^{c/r})$, kde $(c/r)_{cost}$ je počet projitých stavů do stavu $s_{si}^{c/r}$ přes stav $s_a^{c/r}$. $w(s_a^{c/r}, s_{si}^{c/r})$ je cena přechodu ze stavu $s_a^{c/r}$ do stavu $s_{si}^{c/r}$ (řádek 9).

Pokud je stav $s_{si}^{c/r} \in O$ (řádek 10), je porovnán počet stavů, který je potřeba k dosažení $s_{si}^{c/r}$. Když je hodnota $g(s_{si}^{c/r}) \leq (c/r)_{cost}$, znamená to, že již dříve byl nalezen lepší způsob jak se dostat do stavu $s_{si}^{c/r}$ a nově vygenerovaný stav ignorujeme (řádky 11-12). Pokud nově vygenerovaný stav nabízí levnější cestu nastavíme $g(s_{si}^{c/r}) = (c/r)_{cost}$ pro daný stav v množině O (řádek 23).

Podobně pokud je stav $s_{si}^{c/r} \in C$ (řádek 14), je porovnán počet stavů, který je potřeba k dosažení $s_{si}^{c/r}$. Když je hodnota $g(s_{si}^{c/r}) \leq (c/r)_{cost}$, znamená to, že již dříve byl nalezen lepší způsob jak se dostat do stavu $s_{si}^{c/r}$ a nově vygenerovaný stav ignorujeme (řádky 15-17). Pokud nově vygenerovaný stav nabízí levnější cestu, přesunume stav $s_{si}^{c/r}$ z množiny C zpět do množiny O (řádek 18). Následně nastavíme $g(s_{si}^{c/r}) = (c/r)_{cost}$ pro daný stav v množině O (řádek 23).

Pokud stav $s_{si}^{c/r} \notin C \wedge s_{si}^{c/r} \notin O$, přidáme stav $s_{si}^{c/r}$ do množiny otevřených stavů O a pro přidaný stav spočítáme odhad $h(s_{si}^{c/r})$ (řádky 19-21). Nastavíme počet aktuálně projitých stavů, který je dán vztahem $g(s_{si}^{c/r}) = (c/r)_{cost}$. (řádek 23). Následně nastavíme stav $s_a^{c/r}$ jako rodiče a stav $s_{si}^{c/r}$ jako potomka. Toto slouží k sestavení plánu robota/kostky v případě nalezení cíle (řádek 24).

Nakonec je aktuální stav $s_a^{c/r}$ dán do množiny uzavřených stavů C (řádek 26).

V případě, že je nalezena posloupnost stavů, algoritmus vrátí tuto posloupnost (řádek 7). Pokud je množina otevřených stavů prázdná, ale algoritmus posloupnost stavů nenalezl je návratovou hodnotou algoritmu prázdná posloupnost (řádek 28).

Algorithm 2 A* algorithm

put $s_0^{c/r}$ into O with $f(s_0^{c/r}) = h(s_0^{c/r})$, $result = \{\}$
while O not empty **do**
    take top $s^{c/r}$ from O with lowest $f(s_a^{c/r}) = g(s_a^{c/r}) + h(s_a^{c/r})$
    if $s_a^{c/r} = s_F^{c/r}$ **then**
        $result = route_{C/R}(s_0^{c/r}, s_F^{c/r})$; break
    end if
    generate all $s_{si}^{c/r}$ from $s_a^{c/r}$
    for each $s_{si}^{c/r}$ of $s_a^{c/r}$ **do**
        set $(c/r)_{cost} = g(s_a^{c/r}) + w(s_a^{c/r}, s_{si}^{c/r})$
        if $s_{si}^{c/r}$ is in O **then**
            if $g(s_{si}^{c/r}) \leq (c/r)_{cost}$ **then**
                continue
            end if
        else if $s_{si}^{c/r}$ is in C **then**
            if $g(s_{si}^{c/r}) \leq (c/r)_{cost}$ **then**
                continue
            end if
            move $s_{si}^{c/r}$ from C to O
        else
            add $s_{si}^{c/r}$ to O
            set $h(s_{si}^{c/r})$
        end if
        set $g(s_{si}^{c/r}) = (c/r)_{cost}$
        set the parent of $s_{si}^{c/r}$ to $s_a^{c/r}$
    end for
    add $s_a^{c/r}$ to C
end while
return result

4.3 Optimalizace plánu úlohy

V předchozím kroku jsme vytvořili plán akcí pro jednotlivé bloky a roboty. Tento plán se skládá z dílčích posloupností. Každá dílčí posloupnost obsahuje právě jeden plán bloku $route_C(s_A^c, s_B^c)$ a několik plánů robotů pro jednoho nebo dva roboty $route_R(s_A^{r1}, s_B^{r1})$, případně $route_R(s_A^{r2}, s_B^{r2})$.

Význam symbolů v Tabulkách 4.3 a 4.4 je následující. Písmena A-J v Tabulce 4.3 reprezentují akce bloků/robotů $a_{r/c}(s_i^{r/c}) = s_{i+1}^{r/c}$. P1 a P2 je označení dílčích posloupností. Počet kroků algoritmu resp. počet provedených akcí je označen t. V Tabulce 4.4 je akce bloku označena C a reprezentuje $a_c(s_i^c) = s_{i+1}^c$. Akce robotů R1 a R2 reprezentují $a_{r1/r2}(s_i^{r1/r2}) = s_{i+1}^{r1/r2}$.

Optimalizací se v této úloze rozumí paralelizace provádění akcí bloků/robotů. Jsou zde využity dva kroky optimalizace, které jsou následující:
1) Máme dva dílčí plány P1 a P2, viz Tabulka 4.3. Bez optimalizace by se vykonal nejdříve první a potom druhý a jejich vykonání by trvalo čas t_1(řádky 2 a 3). Paralelizace těchto dvou plánů probíhá tak, že se vezme poslední akce, tj. E z P1 a porovná se s první akcí tj. F z P2. V případě, že tyto dvě akce nepotřebují k jejich provedení stejné stavy bloků/robotů, lze je vykonat najednou. Výsledkem je plán, jehož vykonání by trvalo $t_1 - 1$ a porovnávání se opakuje. V tomto případě se porovnává zda lze vykonat současně akce D, F a akce E, G. Tímto postupem se pokračuje dokud nedojde k případu, že by při vykonávání těchto dvou akcí došlo ke kolizím. Výsledkem je, že se části plánů P1 a P2 překrývají a celková doba vykonávání těchto akcí je t_2, kde platí, že $t_2 < t_1$ (řádky 4 a 5). To platí pro případ, že chceme držet dílčí plány celistvé a vykonávat je bez přerušení. Pokud na tomto netrváme, lze optimalizaci zlepšit. Jedná se pouze o návrh, který není implementován. Zlepšíme to tím, že porovnáváme akce dílčí části P2 samostatně s akcemi P1 (řádky 6 a 7). Avšak i v tomto případě je nutné dodržet pořadí vykování akcí v jednotlivých dílčích částech.

#1	t	1	2	3	4	5	6	7	8	9	10
#2	P1	A	B	C	D	E					
#3	P2						F	G	H	I	J
#4	P1	A	B	C	D	E					
#5	P2			F	G	H	I	J			
#6	P1	A	B	C	D	E					
#7	P2		F	G		H	I	J			

Tabulka 4.3: Optimalizace dílčích posloupností

2) Máme dílčí plán P1, který můžeme rozdělit na akce robotů a bloku, viz Tabulka 4.4 (řádek 2). Samotné akce bloku nelze nijak optimalizovat. Lze ale uvažovat o paralelizaci přesunu robotů mezi jednotlivými akcemi bloků a to jen v případě, že jsou pro vykonání dané akce bloku potřeba dva roboti. Princip optimalizace je totožný jako v prvním případě. Jedná se znovu o porovnávání obsazenosti stavů potřebných pro vykonání dané akce.

#1	t	1	2	3	4	5	6	7	8	9	10
#2	P1	C	R1	R2	C	R1	R1	R2	R2	R2	C
#3	P1	C	R1 R2	C	R1 R2	R2	R1 R2	C			

Tabulka 4.4: Optimalizace v rámci jedné dílčí posloupnosti

Kapitola 5

Realizace

Algoritmus byl implementován v jazyce C++ s použitím grafické knihovny Ogre3D [9] pro vizualizaci nalezeného řešení. Jedná se o aplikaci, která zajišťuje načtení dat, naplánování cest, následnou vizualizaci a uložení výsledků.

5.1 Vývojový diagram programu

Blokový diagram celé aplikace je znázorněn na Obrázku 5.1.

V první fázi jsou načtena vstupní data ze dvou textových souborů, kde každý řádek v souboru reprezentuje jeden stav bloku/robota. První textový soubor obsahuje počáteční stav celé stavby. (Přestože je v Kapitole 3 zmíněno, že v počátečním stavu jsou bloky poskládány do velké krychle, je zde ponechána možnost pro libovolné počáteční rozestavení.) Tato data mají následující formát: V případě, že se jedná o blok, je zde uvedeno jeho pořadové číslo a následují hodnoty souřadnic x, y a z oddělené mezerou. V případě robota, je na začátek řádku přidán prefix R. Zbytek zůstává totožný. Je nutné dodržet to, aby bloky i roboti měli unikátní označení.

Druhý soubor obsahuje požadovaný/cílový stav stavby. Jsou v něm uvedeny pouze souřadnice bloků ve shodném tvaru jako v předchozím souboru. Načtená data slouží jako vstup do algoritmu a počáteční stav je následně použit pro vizualizaci na vykreslení počáteční scény.

Ukázka formátu vstupních dat:
Počáteční stav stavby (R/(nic), jméno, x, y, z):

```
0 10 5 2
1 15 2 2
```

```
R 2 10 5 2
...
```

Požadovaný/cílový stav (x, y, z):

```
8 5 2
6 5 7
4 5 6
...
```

Načtená data jsou zpracována pomocí algoritmu, jehož implementace byla provedena dle popisu v Kapitole 4. Pro samotnou implementaci byl zvolen procedurální způsob zápisu programu. Jako reprezentace stavového prostoru byla použita 3D matice společná pro bloky a roboty. Prvky této matice obsahují informaci, zda se na dané pozici nachází blok/robot nebo se jedná o podložku.

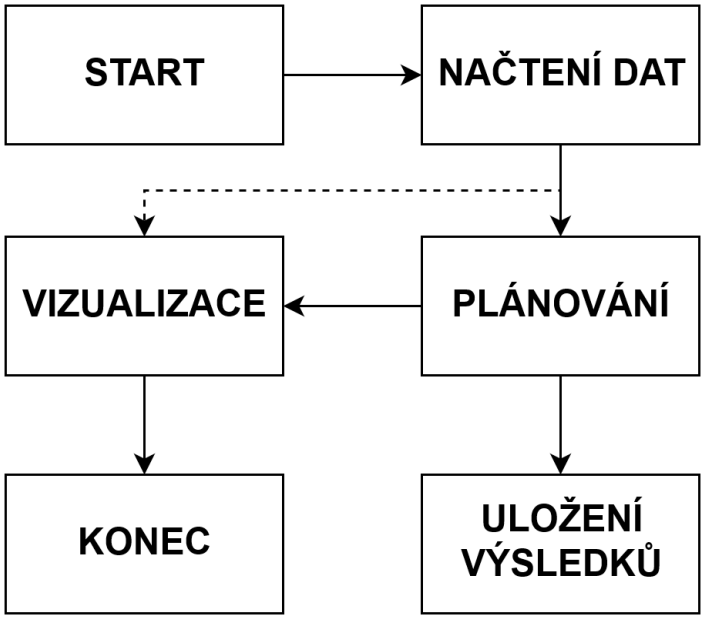
Výstupní posloupnost je uložena do souboru v následujícím formátu (každý řádek reprezentuje jednu provedenou akci a jednotlivé položky jsou oddělené mezerou): označení bloku/robota, počáteční souřadnice před provedením akce, koncové souřadnice po provedení akce a časovou značku. Časová značka je výsledkem optimalizace a říká, které všechny akce lze provést současně. Dále je výstupní posloupnost spolu s počáteční stavem předána do simulačního prostředí, viz Kapitola 5.2.

Výstup algoritmu má následující tvar (jméno, x1, y1, z1, x2, y2, z2, čas):

```
105 2 4 5 3 4 5 0
22 6 4 5 6 4 4 0
105 3 4 5 4 4 5 1
...
```

5.2 Simulační prostředí

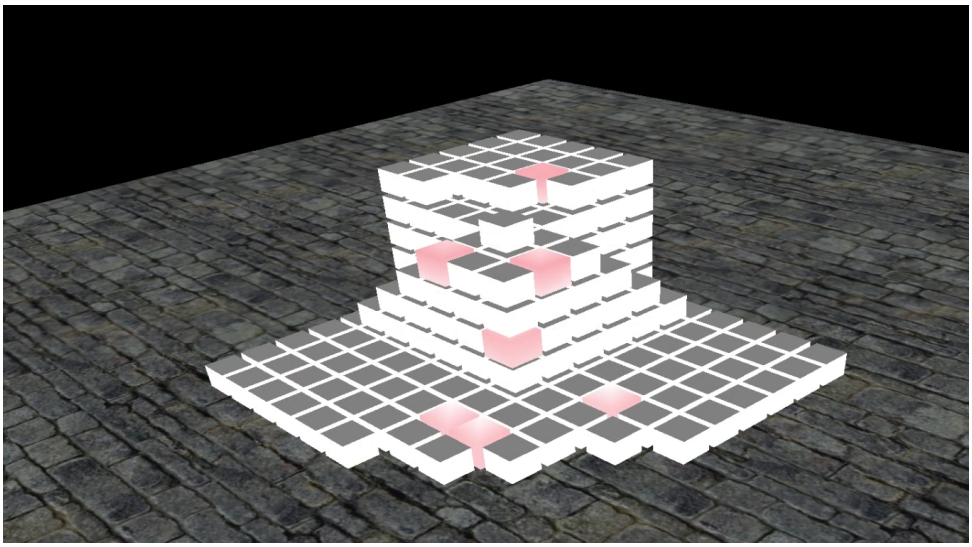
Pro vizualizaci byla zvolena 3D grafická knihovna Ogre3D [9]. Jedná se o volně dostupnou grafickou knihovnu, která je napsána v C++. Samotní tvůrci na svých stránkách uvádějí, že jejich knihovna byla použita pro projekty, jakými jsou například hry, simulátory, vzdělávací software, vědecká vizualizace, a další.



Obrázek 5.1: Vývojový diagram programu

Simulační prostředí je spuštěno automaticky po dokončení běhu algoritmu a po jeho ukončení se ukončí běh celé aplikace. Vzhled naimplementovaného prostředí je zobrazen na Obrázku 5.2. Jedná se o podložku, na které je textura. Na této podložce jsou bloky dvou barev. Bílé bloky označují stavy bloků, které neobsahují stavy robota. Naopak červené bloky, které jsou nepatrně větší a označují, že se uvnitř nachází robot ve stavu shodném se stavem bloku.

Samotné simulační prostředí nabízí dvě funkcionality. Tyto funkcionality se hodí nejen pro demonstraci řešení, ale i pro ladění a zlepšování algoritmu. První je "přehrání" vytvořené posloupnosti akcí. Posloupnost lze procházet krok po kroku, tj. po částech které vznikly optimalizací vytvořené posloupnosti. Rovněž je možné projít celou naplánovanou posloupnost najednou. Podobně jako pří přehrávaní videa, je zde i možnost pozastavení, a případně je zde i možnost vrátit se o libovolný počet kroků zpět. Druhou funkcionalitou je možnost nastavení pozice kamery.



Obrázek 5.2: Vzhled simulačního prostředí

Kapitola 6

Výsledky testování

Funkčnost a vlastnosti implementovaného algoritmu byly ověřeny sérií experimentů popsaných v této kapitole.

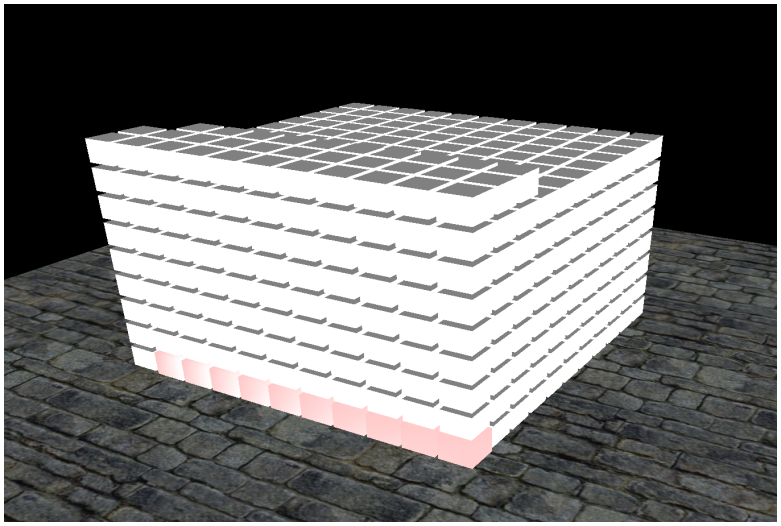
Všechny testy vycházely ze startovní pozice, viz Obrázek 6.1, kde jsou všechny bloky spojené a tvoří velkou neúplnou krychli. Roboti jsou na začátku umístěni do spodního patra krychle.

Ve všech tabulkách a grafech uvedených v této kapitole je použito značení, kde N_F značí počet pater pyramidy, N_c udává počet bloků, N_r je počet robotů, N_s je počet kroků algoritmu před optimalizací, $N_s^{c/r}$ jsou dílčí počty kroků bloků/robotů, N_s^{opt} je počet kroků po optimalizaci. Doba běhu algoritmu je označena t a je uvedena v sekundách. Pokud je uvedený čas v závorce, jedná se pouze o čas plánování bez optimalizace. Poslední sloupec v Tabulkách 6.1 a 6.2 značí efektivitu optimalizace. Hodnota v procentech říká, kolik % kroků zůstalo po provedení optimalizace.

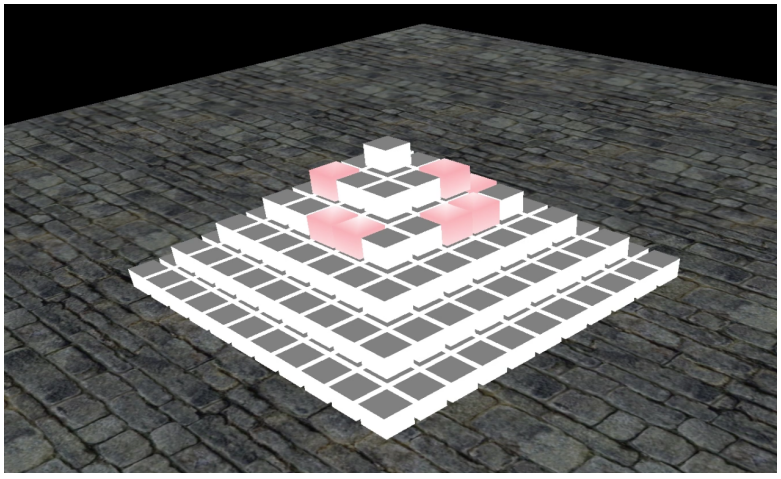
V rámci testování byla nahrána videa demonstrující funkci algoritmu s použitím vytvořeného simulačního prostředí, viz odkaz: MoleMOD videa .

6.1 Stavba pyramidy

Pro porovnání obou navržených heuristických funkcí v závislosti na počtu bloků a počtu robotů byla zvolena stavba pyramidy, viz Obrázek 6.2. Byla zvolena vzhledem k možnosti využít libovolného množství bloků od jednotek až po tisíce.



Obrázek 6.1: Rozmístění bloků a robotů v počáteční pozici



Obrázek 6.2: Stavba pyramidy

6.1.1 Euklidovská vzdálenost

V této části jsou porovnávány výsledky, viz Tabulka 6.1, pro různé počty bloků a robotů za použití Euklidovské vzdálenosti pro výpočet heuristiky. Přičemž testovaná implementace je omezena na použití maximálně 5000 bloků a 100 robotů. Při dostatečném počtu robotů s ohledem na velikost stavby lze po optimalizaci zkrátit čas stavění o více než polovinu.

N_F	N_c	N_r	N_s^c	N_s^r	N_s	N_s^{opt}	$t[s]$	$N_s^{opt}/N_s[\%]$
1	1	-	-	-	-	-	-	-
2	10	2	2	4	6	5	0,068	83,3
3	35	2	67	158	225	206	0,11	91,6
		4	6	148	215	179	0,12	83,3
4	84	2	182	602	784	737	0,21	94
		4		517	699	358	0,23	51,2
		10		497	679	321	0,21	47, 3
5	165	2	546	2031	2577	2486	0,56	96,5
		4		1739	2285	1260	0,62	55,1
		10		1698	2244	1101	0,61	49,1
8	680	2	4945	18801	23746	23180	6,93	97,6
		10		17394	22339	11283	23,42	50,5
		50		15768	20713	9013	22,81	43,5
10	1330	2	11064	42152	53216	52278	21,57	98,2
		4		42152	53216	52278	21,57	54,2
		10		39779	50843	23073	166,08	45,4
		20		35526	46590	18318	151,21	43
		50		35526	46590	18318	151,21	39,3
15	4495	2	68298	262479	330777	327071	810,33	98,9
		10		253566	321864	147111	12896,3	45,7
		100		215665	283963	-	(66,22)	-

Tabulka 6.1: Tabulka počtu kroků při použití Euklidovské vzdálenosti jako heuristické funkce

6.1.2 Manhattanská vzdálenost

V této části jsou porovnávány výsledky, viz Tabulka 6.2, pro různé počty bloků a robotů za použití Manhattanské vzdálenosti pro výpočet heuristiky. Pro testovanou implementaci bylo dosaženo horších maximálních výsledků než v případě použití Euklidovské vzdálenosti pro výpočet heuristické funkce. Maximální počet bloků je shodný 5000, ale maximální počet robotů byl 40. Nejspíše se jedná o implementační chybu, kterou neodhalilo předchozí testování.

6.1.3 Porovnání heuristik

Tato část porovnává výsledky z Kapitol 6.1.1 a 6.1.2. V následující Tabulce 6.3, kde N_M je optimalizovaný počet kroků při použití Manhattanské vzdálenosti pro výpočet heuristické funkce, N_E je optimalizovaný počet kroků při použití Euklidovské vzdálenosti pro výpočet heuristické funkce a sloupec $\delta(N_E, N_M)$

N_F	N_c	N_r	N_s^c	N_s^r	N_s	N_s^{opt}	$t[s]$	$N_s^{opt}/N_s[\%]$
1	1	-	-	-	-	-	-	-
2	10	2	2	4	6	5	0,07	83,3
3	35	2	68	165	233	209	0,11	89,7
		4		159	227	186	0,12	81,9
		2		599	782	732	0.21	93,6
4	84	4	183	523	706	377	0,24	53,4
		10		523	706	338	0,23	47,9
		2		1968	2520	2425	0,57	96,2
5	165	10	552	1701	2253	1248	0,62	55,4
		20		1650	2202	1056	0,63	48
		2		18216	23184	22597	7,53	97,5
8	680	20	4968	16428	21396	12135	23,42	56,7
		50			16920	21888	23,5	52
		2		40783	51894	50930	27,48	98,1
		4		36098	47209	25931	116,44	54,9
10	1330	10	11111	38700	49811	24199	143,25	48,6
		20		34633	45744	18167	141,45	39,7
		40		34354	45456	17334	139,29	38,1
		2		252813	321406	-	(190,2)	-
15	4495	10	68593	241365	309958	-	(176,6)	-
		20		220101	288694	-	(96,2)	-

Tabulka 6.2: Tabulka počtu kroků při použití Manhattanské vzdálenosti jako heuristické funkce

ukazuje v procentech o kolik se liší výsledky plánování v dle použitých heuristik.

N_c	N_r	N_E	N_M	$\delta(N_E, N_M)[\%]$
35	4	179	186	-3,76
	2	2486	2425	2,52
165	4	1260	1248	0,96
	10	1101	1056	4,26
	2	23180	22597	2,58
680	10	11283	12135	-7,02
	20	9013	11382	-20,81
	2	52278	50930	2,65
	4	26171	25931	0,93
1330	10	23073	24199	-4,66
	20	18318	18167	0,83
	40	17778	17334	2,56

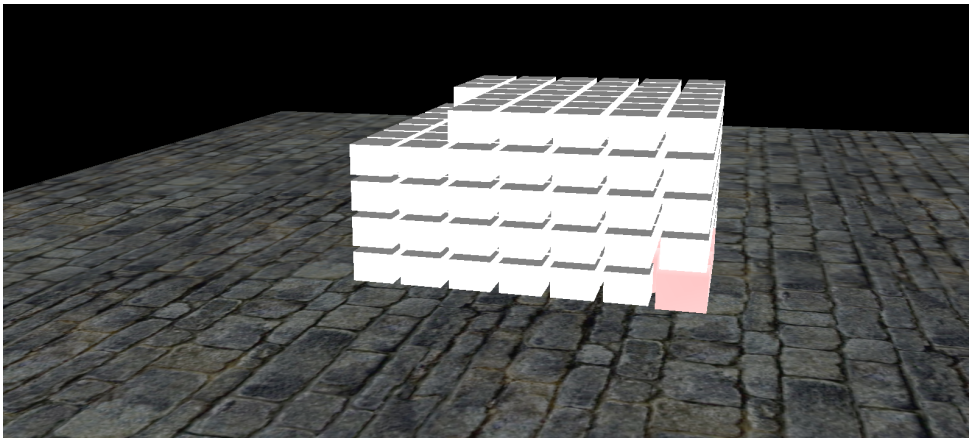
Tabulka 6.3: Tabulka porovnávající počty kroků algoritmu obou heuristických funkcí

Při porovnání výsledků simulací, nelze jednoznačně říci, která z heuristik

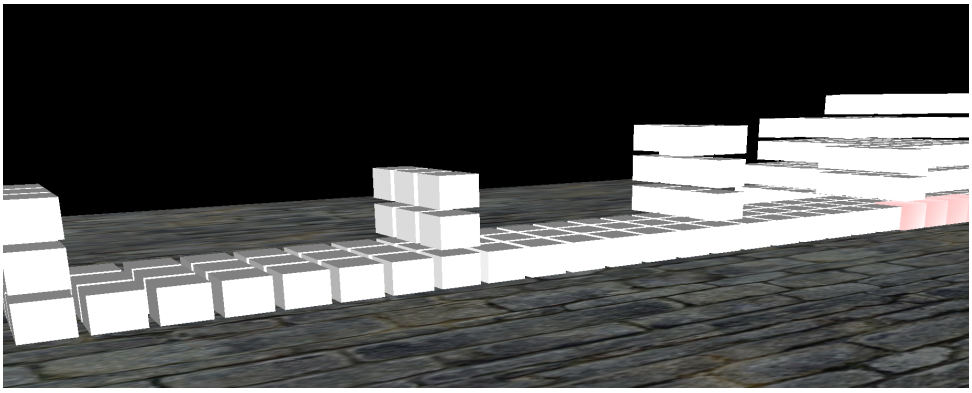
je v případě použité implementace lepší. Viz Tabulka 6.3 je rozdíl počtu v kroků algoritmu menší než 5 % pro většinu případů. Toto platí i pro ostatní parametry, viz Tabulky 6.1 a 6.2. Z výsledků je patrné, že při zlepšování algoritmu je třeba věnovat zvýšenou pozornost optimalizační části, protože doba běhu předchozích částí je zvláště pro velké stavby zanedbatelná v porovnání s dobou optimalizace.

6.2 Stavba mostu

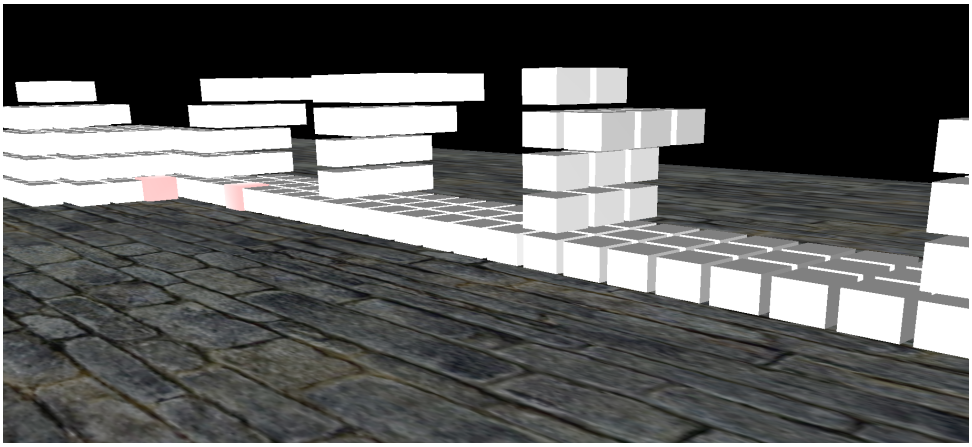
Pyramida není jedinou stavbou, kterou lze s využitím navrženého algoritmu postavit. Obecně lze s jeho pomocí postavit jakoukoliv stavbu, která bude splňovat pravidla a omezení, viz Kapitola 3.2. Pro příklad zde uvedeme jednu z aplikací zmíněnou v Kapitole 2. Jedná se o stavbu mostu. Postup stavby je včetně dílčích částí zdokumentován na Obrázku 6.3. Most je složen z 234 bloků. Pro tuto stavu byli použiti 4 roboti. Počet kroků bloků byl 2653 a počet kroků robotů byl 11527. Celkový počet kroků po optimalizaci byl 7941.



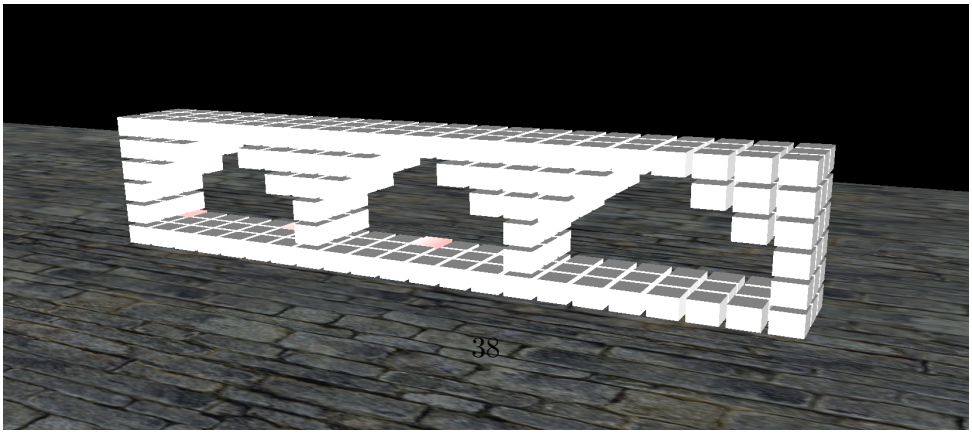
(a) : Počáteční stav - bloky jsou poskládány do krychle



(b) : Nejdříve se postaví spodní vrstva a následně se začnou stavět sloupy



(c) : Pokračuje se od spodní vrstvy stavbou oblouků



(d) : Následně se postaví horní deska a stavba je hotová

Obrázek 6.3: Průběh stavby mostu

Kapitola 7

Závěr

Cílem této práce bylo navrhnout plánovací algoritmus pro systém MoleMOD 2, který vzniká na Fakultě architektury ČVUT v Praze. Konkrétně se tato práce zabývala situací, kdy jsou na staveniště přivezeny stavební moduly (bloky) složené v kompaktním tvaru (např. krychle) a ty musí být postupným přemisťováním pomocí robotů poskládány do požadovaného finálního tvaru. Jedná se tedy o plánovací úlohu ve 3D prostoru.

Pro řešení plánovací úlohy byl nejprve vytvořen matematický popis, který definuje stavový prostor systému MoleMOD jako množiny stavů bloků a robotů. Dále byly popsány podmínky nutné pro přechody mezi stavy v rámci jednotlivých množin. Tento popis lze dále využít pro vylepšování algoritmu navrženého v rámci této práce nebo pro další plánovací úlohy týkající se systému MoleMOD.

Na základě tohoto popisu byl vytvořen návrh algoritmu, který lze rozdělit do tří částí (třídění, plánování a optimalizace). Jádrem celého algoritmu je plánovací algoritmus A*, pro který byly navrženy vhodné heuristické funkce vzhledem k charakteru úlohy. Vstupní data jsou před samotnou fází plánování setříděna podle předem určených pravidel. Účelem třídění je zrychlení běhu algoritmu a předejití možným kolizním stavům během plánování. Výsledná naplánovaná sekvence je následně optimalizována. Pomocí optimalizace bylo docíleno paralelizace činnosti robotů.

Navržený algoritmus byl implementován v jazyce C++ a bylo k němu vytvořeno grafické rozhraní, které slouží pro demonstraci výsledků plánování.

Navržený algoritmus je možné dále zlepšovat. Zde můžeme s výhodou využít rozdělení algoritmu na tři části, které lze zlepšovat nezávisle na sobě. Například v rámci třídění lze brát ohled na počet robotů a podle toho volit cílové pozice bloků tak, aby mohlo pracovat současně co nejvíce robotů. Jinak řečeno, aby se roboti pohybovali v různých částech stavby a jejich cesty se nepřekrývaly.

V rámci plánování by bylo možné vylepšit způsob, jakým jsou přidělováni roboti pro přesouvání jednotlivých bloků. Zde lze například brát v potaz

různý počet stavů, které musí blok projít než dosáhne cílového stavu.

Návrh na zlepšení optimalizace byl uveden již v Kapitole 4.2.1. Jedná se o efektivnější využití prostoru uvnitř bloků, kde procházejí roboti. Jde o to, že by se roboti mohli přesouvat okamžitě, jakmile se jim uvolní cesta do požadovaného stavu, ve kterém budou pohybovat s blokem.

Literatura

[1] PETRŠ, Jan. MoleMOD [online]. [cit. 2020-01-02]. Dostupné z: http://www.studioflorian.com/projekty/347-jan-petrs-molemod

[2] PETRŠ J., HAVELKA J., FLORIÁN M., NOVÁK, J., 2017. MoleMOD - on design specification and applications of a self-reconfigurable constructional robotic system. In: ShoCK! - Sharing Computational Knowledge! - Proceedings of the 35th eCAADe Conference. vol. 2, pp. 159–166.

[3] BREJCHOVÁ M., KULICH M., PETRŠ J. a PŘEUČIL L. , 2018. Modelling, simulation, and planning for the MoleMOD system, Proceedings of Modelling and Simulation for Autonomous Systems (MESAS) Conference 2018

[4] Algoritmus A-Star [online]. [cit. 2020-01-02]. Dostupné z: http://voho.eu/wiki/algoritmus-a-star/

[5] A* Algorithm pseudocode [online]. [cit. 2020-01-02]. Dostupné z: http://mat.uab.cat/~alseda/MasterOpt/AStar-Algorithm.pdf

[6] Heuristics From Amit's Thoughts on Pathfinding [online]. [cit. 2020-01-02]. Dostupné z: http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html

[7] BARTÁK, Roman. Umělá inteligence I [online]. [cit. 2020-01-02]. Dostupné z: https://ktiml.mff.cuni.cz/~bartak/ui/lectures/lecture04.pdf

[8] Metoda A*: analýza a hodnocení biologických dat umělá inteligence prohledavání stavového prostoru ...[online]. [cit. 2020-01-02]. Dostupné z: https://portal.matematickabiologie.cz

[9] Ogre3D dokumentace [online]. [cit. 2020-01-02]. Dostupné z: http://wiki.ogre3d.org/Home

[10] Manhattan distance [online]. [cit. 2020-01-02]. Dostupné z: https://xlinux.nist.gov/dads/HTML/manhattanDistance.html

[11] WRIGHT, Cornell, Zachary MCCORD, Allison NAAKTGEBOREN, et al. Design of a modular snake robot [online]. [cit. 2020-01-02]. Dostupné z: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=$&$arnumber=4399617

[12] HVĚZDA, Jakub. Comparison of path planning methods for amulti-robot team. Praha, 2017. Diplomová práce. CVUT FEL.

[13] JARKOVSKÝ, Jiří a Simona LITTNEROVÁ. Vícerozměrné statistické metody [online]. [cit. 2020-01-02]. Dostupné z: http://www.iba.muni.cz/esf/res/file/bimat-prednasky/vicerozmerne-statisticke-metody/VSM-03.pdf

Příloha A

Obsah CD

- Soubor DP-Urvalek.pdf je elektronická verze tohoto dokumentu.
- Soubor data.txt obsahující odkaz na videa demonstrující činnost algoritmu a odkaz na repozitář se zdrojovými kódy.