Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

FIIT-5220-47893

Bc. Ivan Polko

OPTIMALIZÁCIA PLÁNOVANIA CESTY PRE SKUPINU ROBOTOV

Diplomová práca

Študijný program: Softvérové inžinierstvo

Študijný odbor: 9.2.5 Softvérové inžinierstvo

Miesto vypracovania: Ústav aplikovanej informatiky, FIIT STU Bratislava

Vedúci práce: prof. RNDr. Jiří Pospíchal, DrSc.

máj 2012

ANOTÁCIA

Slovenská technická univerzita v Bratislave

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Študijný program: Softvérové inžinierstvo

Autor: Bc. Ivan Polko

Diplomová práca: Optimalizácia plánovania cesty pre skupinu robotov

Vedúci diplomovej práce: prof. RNDr. Jiří Pospíchal, DrSc.

máj 2012

Práca sa zaoberá problémom plánovania cesty pre skupinu robotov v prostredí modelovanom neorientovaným grafom. Cieľom je premiestniť roboty z počiatočných do cieľových pozícií pomocou čo najmenšieho počtu krokov. Zámerom práce bolo vylepšiť algoritmus BIBOX navrhnutý P. Surynkom, ktorý sa zameriava na 2-súvislé grafy s malým počtom neobsadených uzlov, pričom jeho základom je dekompozícia grafu na cyklus a uchá.

V práci je skúmaný vplyv dekompozície grafu na kvalitu riešení nájdených algoritmom BIBOX. Navrhnutá je metóda prehľadávania priestoru možných dekompozícií, ktorá využíva existujúci algoritmus na dekompozíciu grafu a metódu Monte-Carlo prehľadávania stromu. Boli tiež navrhnuté dve heuristiky, ktoré výrazne zlepšujú priebeh prehľadávania.

S vybraným nastavením bolo prehľadávanie vyskúšané na zvolenej testovacej sade zloženej zo 190 rôznych grafov. V priemere sa prehľadávaním dokázala nájsť lepšia dekompozícia pri 94% grafov, pričom zlepšenie riešenia oproti originálnej dekompozícii bolo až o 38%.

ANNOTATION

Slovak University of Technology Bratislava

FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES

Degree Course: Software Engineering

Author: Bc. Ivan Polko

Master´s Thesis: Optimization of Path Planning for Multiple Robots

Supervisor: prof. RNDr. Jiří Pospíchal, DrSc.

2012, May

Presented thesis deals with problem of path planning for multiple robots in environment modeled by undirected graph. The objective is to move robots from initial positions to goal positions using minimal number of steps. The aim of this thesis is to improve algorithm BIBOX designed by P. Surynek, that is targeted on 2-connected graphs with small number of unoccupied vertices and is based on ear decomposition of graph.

Focus is on influence of ear decomposition on the quality of BIBOX solutions. Our approach for searching in space of possible decompositions is based on existing algorithm for ear decomposition and Monte-Carlo tree search. Two heuristics are also proposed, which significantly improve progress of search.

With selected settings, our method was tested on test set consisting of 190 different graphs. On average, better decomposition was found in 94% of graphs. Solution improvement in comparison with original decomposition was up to 38%.

Čestne prehlasujem, že som diplomovú práca vypracoval samostatne a použil len literatúru, ktorú uvádzam v zozname.

.....

Bratislava 2012 (podpis)

Ďakujem prof. RNDr. Jiřímu Pospíchalovi, DrSc. za cenné rady, pripomienky a odborné vedenie pri vypracovávaní diplomovej práce.

Obsah

1. Úvod

Práca sa venuje problému plánovania cesty pre skupinu robotov, ako bol definovaný v [1]. Roboty sa pohybujú v určitom prostredí, v ktorom majú stanovenú počiatočnú a cieľovú pozíciu. Cieľom je premiestniť všetkých robotov z počiatočných do cieľových pozícií tak, aby navzájom nekolidovali a vyhli sa prekážkam v prostredí [1].

Motiváciou pre riešenie takéhoto problému sú úlohy, pri ktorých je potrebné presúvať entity v prostredí s obmedzeným voľným miestom. Príkladom z reálneho života môže byť presúvanie kontajnerov v prístavisku, koordinácia vozidiel v hustej premávke alebo presúvanie paketov v sieti medzi uzlami s obmedzenou úložnou kapacitou [2].

Prostredie, v ktorom sa roboty nachádzajú, sa modeluje neorientovaným grafom. Každý robot v grafe má určenú začiatočnú a cieľovú pozíciu (uzol). Cieľom je presunutie všetkých robotov zo začiatočných pozícií na cieľové pozície pomocou čo najmenšieho počtu krokov. Roboty sú pri tomto presúvaní riadené centrálne. Algoritmus BIBOX, navrhnutý v [1], a jeho variácie sa zameriavajú na 2-súvislé grafy, v ktorých je málo voľného miesta (minimálne musia byť voľné 1 alebo 2 uzly v závislosti od použitého algoritmu)[1].

V kapitole 2 sú opísané základné pojmy z teórie grafov, ktoré sa používajú v ďalších kapitolách.

Kapitola 3 obsahuje definíciu problému plánovania cesty pre skupinu robotov ako aj definíciu iných grafových problémov, z ktorých tento problém vychádza. Tiež sú tu opísané rôzne príbuzné práce venujúce sa podobným problémom.

Kapitola 4 opisuje existujúce algoritmy na riešenie uvedeného problému (BIBOX, BIBOX-θ). Ďalej sú tu opísané existujúce spôsoby, ktorými je možné zlepšiť riešenia problému, ako aj publikované výsledky týchto algoritmov.

Kapitola 5 opisuje algoritmus, pomocou ktorého je možné vytvoriť dekompozíciu grafu na cyklus a uchá. Je totiž predpoklad, že inou dekompozíciou, ako bola použitá pri publikovaných pokusoch s algoritmom BIBOX, je možné nájsť kratšie riešenia.

Kapitola 6 sa venuje použitiu algoritmu z kapitoly 5 s algoritmom BIBOX. Uvedené sú tiež výsledky pri náhodnom prehľadávaní priestoru možných dekompozícií, ktoré potvrdzujú, že inou dekompozíciou je možné nájsť kratšie riešenia.

Kapitola 7 sa venuje popisu stavového priestoru, ktorý sa prechádza pri hľadaní lepšej dekompozície, a odhadu jeho veľkosti.

Kapitola 8 uvádza opis navrhnutých heuristík, ktoré by mali zrýchliť proces hľadania lepšej dekompozície.

Kapitola 9 opisuje metódu Monte-Carlo prehľadávania stromu, ktorá má vhodné vlastnosti na to, aby sa použila na prehľadávanie stavového priestoru možných dekompozícií.

Kapitola 10 obsahuje návrh použitia Monte-Carlo prehľadávania stromu na hľadanie lepšej dekompozície.

Kapitola 11 obsahuje výsledky experimentov s hľadaním lepšej dekompozície. Experimenty sú rozdelené na dve časti, v prvej sa určuje vhodné nastavenie parametrov prehľadávania na 6 grafoch, následne v druhej časti sa s určeným nastavením vykonajú experimenty s vybranou testovacou sadou zloženou zo 190 rôznych grafov.

Kapitola 12 uvádza možné smery, ktorými by sa problematika tejto práce mohla rozvíjať do budúcnosti. Ide buď o vylepšenie použitých metód, o otestovanie na iných typoch grafov alebo o aplikáciu na varianty algoritmu BIBOX.

Kapitola 13 obsahuje zhodnotenie celej práce.

2. Základné využívané pojmy teórie grafov

Obsahom tejto kapitoly je vysvetlenie základných pojmov z teórie grafov, ktoré budú použité v ďalších kapitolách. Ak nie je uvedené inak, definície pochádzajú z [3].

Graf je dvojica $G = V, E$, kde V je množina uzlov, E je množina hrán a platí, že $E \subseteq V^2$. Stupeň uzla v je počet hrán $E(v)$ v danom uzle. Prienik grafov G a G' je graf $G \cap G' = (V \cap V', E \cap E')$. Ak $G \cap G' = \emptyset$, tak grafy G a G' nazývame nezávislé.

Cesta je neprázdny graf $P = (V, E)$ uvedenej formy:

$$V = \{v_0, v_1, \ldots, v_k\} \qquad E = \{v_0v_1, v_1v_2, \ldots, v_{k-1}v_k\},$$

kde všetky uzly v_i sú rozdielne. Cesta sa často určuje postupnosťou svojich uzlov: $P = v_0v_1 \ldots v_k$. Uzly $v_1 \ldots v_{k-1}$ sa nazývajú vnútorné uzly cesty P. Vzdialenosť dvoch uzlov u, v je dĺžka najkratšej cesty $u - v$ v G. Najväčšia vzdialenosť zo všetkých dvojíc uzlov v grafe sa nazýva priemer grafu G.

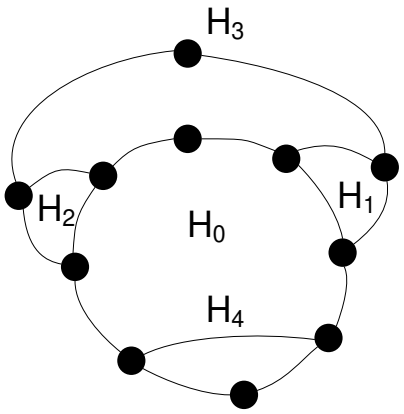
Ak $P = v_0v_1 \ldots v_{k-1}$ je cesta a $k \geq 3$ tak graf $C = P + v_{k-1}v_0$ sa nazýva cyklus.

Neprázdny graf G je súvislý, ak všetky dvojice jeho uzlov sú spojené cestou v G. Graf G sa nazýva 2-súvislý, ak $|V| > 2$ a graf G zostane súvislý pri odobratí ľubovoľného uzla. Graf G je 2-súvislý práve vtedy, keď v ňom existujú dve nezávislé cesty medzi každou dvojicou uzlov.

Nech sú dané dva grafy $G = V, E$ a $G' = V', E'$. Grafy G a G' nazývame izomorfné, ak existuje bijekcia $\varphi: V \to V'$ pre ktorú platí: $uv \in E \Leftrightarrow \varphi\ u\ \varphi\ v \in E'$ pre $\forall u, v \in V$.

Pod netriviálnym 2-súvislým grafom sa bude ďalej rozumieť 2-súvislý graf, ktorý nie je izomorfný s cyklom [2].

Ucho (*angl. ear* [4], *handle* [2], *H-path* [3]) grafu G je najdlhšia taká cesta v G, ktorej vnútorné uzly majú stupeň 2. Dekompozícia grafu G na uchá je dekompozícia v tvare $H_0, H_1, \ldots, H_k$ taká, že H_0 je cyklus a H_i pre $i \geq 1$ je ucho grafu $H_0 \cup \ldots \cup H_{i-1}$. Príklad dekompozície grafu na cyklus a 4 uchá sa nachádza na Obr. 1 [4].



Obr. 1. Dekompozícia grafu na uchá (prebraté z [4]).

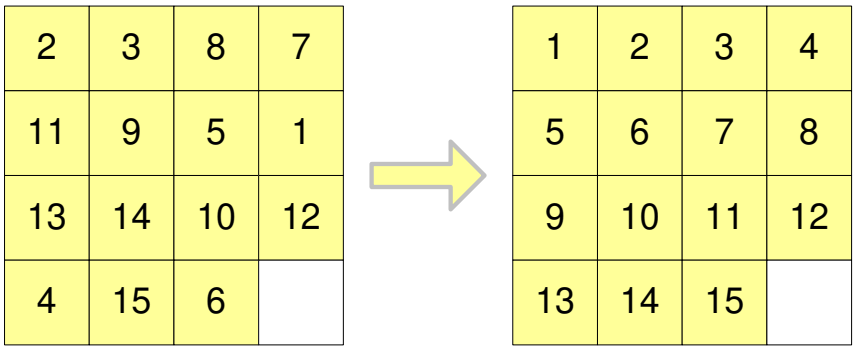
Graf je 2-súvislý práve vtedy, keď má dekompozíciu na uchá. Tiež platí, že každý cyklus v 2-súvislom grafe je počiatočný cyklus v niektorej dekompozícii na uchá [4].

Ak nie je uvedené inak, pod dekompozíciou grafu sa bude ďalej rozumieť práve dekompozícia grafu na cyklus a uchá.

3. Definícia problému plánovania cesty pre skupinu robotov

V tejto kapitole je zadefinovaný problém presúvania kameňov po grafe a následne aj problém plánovania cesty pre skupinu robotov. Sú tu tiež uvedené niektoré ich vlastnosti.

Počiatkom uvedených grafových problémov je puzzle-8, resp. puzzle-15. Puzzle-15 sa skladá z priestoru 4x4 štvorcov, pričom 15 z nich je obsadených očíslovanými kameňmi a jeden štvorec zostáva voľný (Obr. 2). Tento problém sa dá zovšeobecniť na puzzle-(n^2-1). V takomto zovšeobecnenom puzzle je v priestore $n \times n$ štvorcov umiestnených n^2-1 kameňov, pričom jedno miesto zostáva voľné. Cieľom je nájsť takú postupnosť pohybov, ktoré premenia počiatočnú konfiguráciu kameňov na konečnú. Pohyb pozostáva z posunutia kameňa na priľahlé prázdne miesto v horizontálnom alebo vertikálnom smere [5].

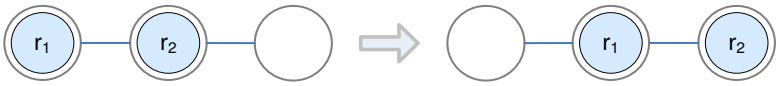


Obr. 2. Príklad počiatočného a koncového rozmiestnenia kameňov v puzzle-15.

Uvedené problémy sa dajú vyjadriť grafom v tvare mriežky, kde kamene sú umiestnené na jednotlivých uzloch. Zovšeobecnením puzzle-(n^2-1) na všeobecné grafy potom získame problém pohybu kameňov po grafe (*angl. pebble motion on graphs*). Tento problém je definovaný nasledovne:[6]

Nech G je graf s n uzlami a $k < n$ kameňmi očíslovanými $1, \ldots, k$ umiestnenými na rôznych uzloch. Pohyb spočíva v premiestnení kameňa na susedný neobsadený uzol. Problémom je určiť, či jedno usporiadanie kameňov je dosiahnuteľné z iného usporiadania a nájsť najkratšiu takú postupnosť pohybov, ktorou uvedenú zmenu usporiadania dosiahneme [6].

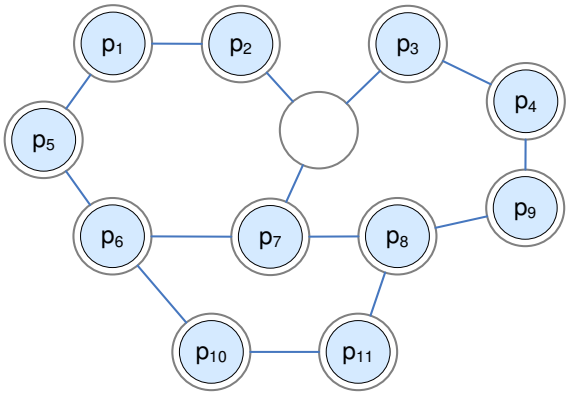
Problém plánovania cesty pre skupinu robotov (*angl. multi-robot path planning*) je podľa [2] definovaný ako zoslabenie problému pohybu kameňov po grafe. Podmienka, že kameň, resp. robot sa môže presunúť len do neobsadeného uzla, je zmiernená. Presunutie robota do susedného uzla, ak iný robot daný uzol práve opúšťa a žiaden iný robot do daného uzla práve nevchádza, je v tomto probléme povolené, čo je vo vzťahu k realite lepšia podmienka. Avšak musí existovať prvý robot, ktorý spustí reťaz takýchto pohybov presunom do aktuálne neobsadeného uzla, do ktorého žiaden iný robot práve nevchádza. Roboty sa teda môžu hýbať „ako vlak“ (Obr. 3), nemôžu sa však otáčať v cykle bez voľného uzla [2].



Obr. 3. Paralelný pohyb dvoch robotov naraz.

Všetky roboty sú riadené centrálne a sú identické. Pohyb robotov z jedného uzla do druhého prebieha okamžite v diskrétnych časových krokoch. Terminologická odlišnosť kameň - robot teda pri takejto definícii nemá žiaden význam [2].

Inštancia problému je pri všetkých spomenutých grafových problémoch určená grafom G, počiatočným a cieľovým rozložením robotov, resp. kameňov na uzloch grafu G. Počet robotov, resp. kameňov musí byť menší ako počet uzlov, musí teda zostať aspoň jeden uzol neobsadený. Príklad rozloženia kameňov na grafe s jedným voľným uzlom je na Obr. 4 [2].



Obr. 4. Rozloženie kameňov na uzloch grafu s jedným voľným miestom (uzlom).

Pri oboch spomenutých problémoch je potrebné rozlišovať medzi počtom krokov riešenia (*angl. makespan*) a počtom pohybov, ktoré roboty, resp. kamene vykonajú. Krok riešenia zodpovedá jednému časovému kroku. V jednom kroku riešenia sa totiž môže pohnúť aj viac robotov, resp. kameňov naraz, ak je pre každý kameň, resp. robota

splnená podmienka pohybu. Problém plánovania cesty pre skupinu robotov však umožňuje väčší paralelizmus, kvôli slabším obmedzeniam na pohyb robotov. Pri tomto probléme stačí na paralelný pohyb jeden neobsadený uzol v grafe, pri pohybe kameňov po grafe sú pre paralelizmus potrebné minimálne 2 neobsadené uzly [2].

Ak kamene v ľubovoľnom riešení problému pohybu kameňov po grafe nahradíme robotmi, získame riešenie problému plánovania cesty pre skupinu robotov. Na riešenie problému plánovania cesty pre skupinu robotov môžeme teda použiť ľubovoľný algoritmus, ktorý rieši pohyb kameňov po grafe [2].

Všetky uvedené problémy (puzzle-(n^2-1), pohyb kameňov po grafe, plánovanie cesty pre skupinu robotov) sú problémy, v ktorých je možné nájsť riešenie v polynomiálnom čase [6,7,1]. Nájdenie optimálnej postupnosti pohybov vzhľadom k počtu krokov riešenia je však NP-úplný problém [5,8].

3.1 Príbuzné problémy

Obsahom tejto kapitoly je opis niektorých príbuzných problémov a ich odlišností od problémov definovaných v predchádzajúcej kapitole. Aj keď je v nasledujúcich opisoch spomenutý robot (podľa terminológie samotnej práce), vo všetkých uvedených prácach je podmienka prechodu do susedného uzla zadefinovaná rovnako ako pri probléme pohybu kameňov po grafe.

Pri probléme plánovania pohybu po grafe zadefinovanom v práci Papadimitriou et al: *Motion Planning on a Graph* [9] sa v grafe okrem jedného robota nachádzajú aj prekážky, ktoré sú tiež rozmiestnené na uzloch grafu. Cieľom je premiestniť robota na cieľové miesto, pričom nie je dôležité, kam sa umiestnia prekážky. Uvedená práca sa zaoberá pohybom na strome. Na túto prácu je nadviazané v [10,11], kde sa uvažuje o pohybe k robotov na strome.

V práci Ryan: *Exploiting Subgraph Structure in Multi-Robot Path Planning* [12] sa pracuje s problémom plánovania cesty pre skupinu robotov, pri ktorom sa využíva dekompozícia grafu na podgrafy špecifickej štruktúry, čím sa zmenší priestor hľadania. Každý typ podgrafu sa potom rieši špeciálnym spôsobom. Podľa výsledkov je tento prístup vhodný iba pre malý počet robotov v grafe.

Práca Wu, Grumbach: *Feasibility of Motion Planning on Directed Graphs* [13] sa venuje plánovaniu pohybu na orientovaných grafoch, čo je náročnejší problém, pretože roboty nemôžu vykonať spätný pohyb. Výsledkom uvedenej práce je algoritmus, ktorý dokáže rozhodnúť o riešiteľnosti takýchto problémov v polynomiálnom čase.

Práca Goraly, Hassin: *Multi-Color Pebble Motion on Graphs* [14] sa venuje odlišnej definícii problému pohybu kameňov po grafe. Graf má n uzlov a $p < n$ kameňov, pričom každý z kameňov má priradenú jednu z m farieb. Kamene s rovnakou farbou sa nerozlišujú. Uvedená práca sa zaoberá otázkou riešiteľnosti takto definovaného problému na stromoch a následne na súvislých grafoch.

4. Algoritmy na riešenie plánovania cesty pre skupinu robotov

Obsahom tejto kapitoly je opis existujúcich algoritmov na riešenie problému plánovania cesty pre skupinu robotov.

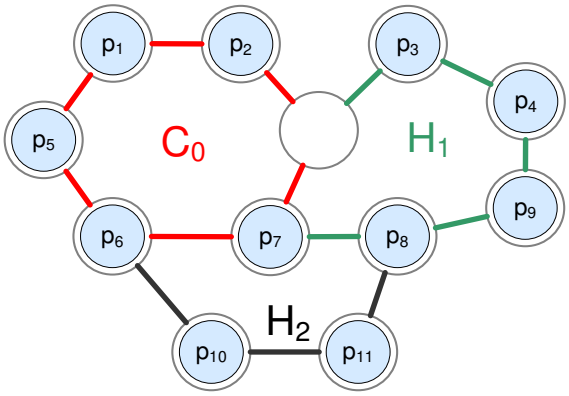
Ako už bolo uvedené v kapitole 3, nájdenie riešenia s optimálnym počtom krokov je NP-úplný problém. Všetky uvedené algoritmy sa preto snažia v polynomiálnom čase nájsť sub-optimálne riešenie, vzhľadom k počtu krokov riešenia. Problém plánovania cesty pre skupinu robotov sa dá riešiť algoritmami pre pohyb kameňov po grafe, takéto riešenie však nebude využívať možnosti paralelných pohybov, ktoré definícia pripúšťa. V tomto prípade je možné výsledné riešenie spracovať spôsobom, ktorý paralelizmus riešenia zvýši [2]. Tento spôsob je opísaný v kapitole 4.3.1.

Existujúce algoritmy [6] pre pohyb kameňov po grafe však vytvárajú riešenia s veľmi veľkým počtom krokov a teda sú pre praktické použitie nevhodné, čo podnietilo vytvorenie nových algoritmov, opísaných v nasledujúcich podkapitolách. Obidva ďalej opísané algoritmy riešia problém pohybu kameňov po grafe, pri riešení nevyužívajú možnosti paralelizmu pohybov. Paralelizmus pohybov sa využije až pri spracovaní nájdeného riešenia [2].

4.1 Algoritmus BIBOX

Opis algoritmu BIBOX v tejto kapitole pochádza z [1,2]. Tento algoritmus rieši špecifické inštancie problému pohybu kameňov po grafe. Graf musí byť netriviálny 2-súvislý s presne 2 neobsadenými uzlami.

Algoritmus využíva vlastnosť konštrukcie 2-súvislého grafu podľa jeho dekompozície na cyklus a uchá, že graf je 2-súvislý v ľubovoľnom štádiu jeho konštrukcie. Príklad dekompozície grafu je na Obr. 5.



Obr. 5. Dekompozícia grafu na cyklus C_0 a dve uchá H_1 a H_2.

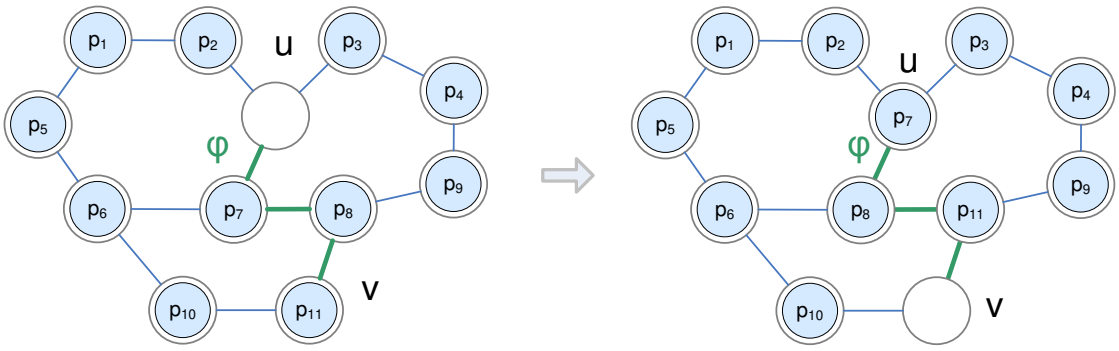
Algoritmus BIBOX začína s posledným uchom dekompozície grafu. Tie kamene, ktoré majú cieľové pozície v danom uchu, sú na tieto pozície presunuté. Problém sa tým zredukuje na menší 2-súvislý graf, ktorý sa rieši rovnakým spôsobom. Tie uchá, v ktorých sú už kamene umiestnené, sa ďalej neuvažujú. Tento postup sa opakuje, kým nezostane začiatočný cyklus dekompozície, ktorý sa rieši odlišným spôsobom.

4.1.1 Operácie algoritmu

Na opis samotného algoritmu je potrebné zadefinovať základné operácie, ktoré sa v algoritme využívajú.

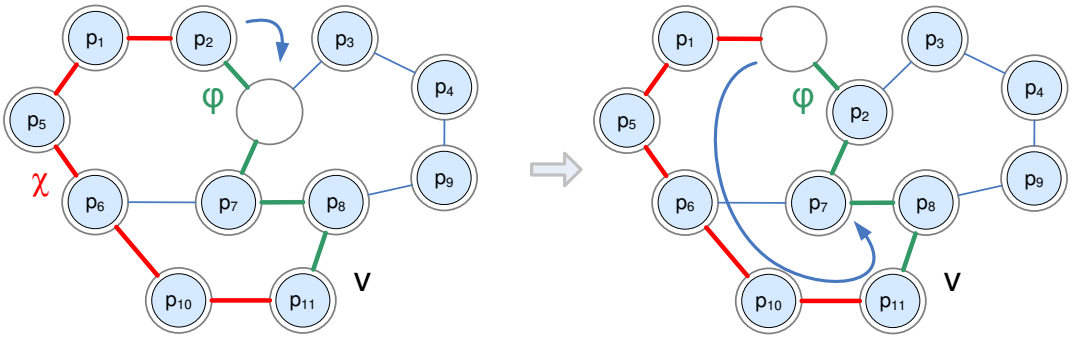
Operácie *Zamkni(X)* a *Odomkni(X)* zamknú alebo odomknú množinu uzlov X. Na začiatku je každý uzol v grafe odomknutý. Keď je uzol zamknutý, žiadny kameň sa na jeho miesto nemôže premiestniť.

Každý uzol v súvislom grafe je možné uvoľniť operáciou *Uvoľni-uzol*. Ak má byť uvoľnený uzol v, nájde sa najkratšia taká cesta φ, ktorá spája v a niektoré voľné miesto tak, že sa vyhne uzamknutým uzlom. Kamene pozdĺž cesty sú vzájomnými výmenami posúvané smerom k voľnému miestu. Táto operácia je znázornená na Obr. 6.



Obr. 6. Uvoľnenie uzla v.

Operácia *Presuň-kameň* presunie zvolený kameň do konkrétneho uzla. Nech kameň p sa ma presunúť do uzla v. Najprv sa nájde taká cesta φ, ktorá spája p a v. Podľa tvrdenia uvedeného v kapitole 2 platí, že existuje alternatívna cesta χ, ktorá tiež spája p a v.

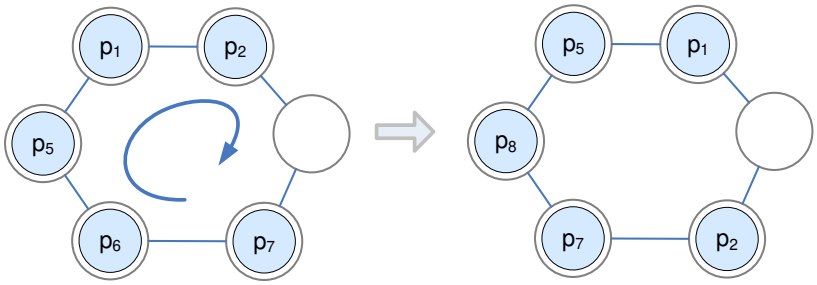


Obr. 7. Presúvanie kameňa do konkrétneho uzla.

Následne sú prechádzané hrany cesty φ nasledovným spôsobom. Prvý uzol hrany je uzamknutý (nachádza sa v ňom presúvaný kameň). Druhý uzol hrany sa uvoľní operáciou *Uvoľni-uzol* (využije sa pritom cesta χ, ktorá sa používa práve na to, aby sa voľné miesto presunulo pred posúvaný kameň). Prvý uzol s kameňom sa odomkne a daný kameň sa presunie do druhého uzla cesty φ. Tento postup sa ďalej opakuje. Jeden krok presúvania kameňa je znázornený na Obr. 7. Kameň p_2 sa má presunúť do uzla v. Najprv sa teda presunie do susedného voľného uzla. Voľné miesto sa následne znovu presunie pred kameň p_2 a celý postup sa môže zopakovať.

Algoritmus BIBOX predpokladá, že každý neorientovaný cyklus objavujúci sa v dekompozícii grafu má priradenú fixnú orientáciu (nezáleží na tom, akú). Orientácia je určená dvoma funkciami *Nasledujúci-uzol(C,v)* a *Predchádzajúci-uzol(C,v)*, ktoré pre cyklus C a uzol v vrátia nasledujúci, resp. predchádzajúci uzol v cykle vzhľadom na pozitívnu orientáciu.

Ďalšími operáciami sú operácie *Otoč-cyklus+* a *Otoč-cyklus-*, ktoré otáčajú kamene umiestnené v cykle. Prvá operácia posúva kamene v cykle v pozitívnom smere (rovnakom ako má daný cyklus priradený), druhá v negatívnom (opačnom). Predpokladá sa, že aspoň jeden uzol v danom cykle je voľný. Otáčanie 5 kameňov v cykle je znázornené na Obr. 8. Pod otáčaním cyklu sa bude ďalej rozumieť práve otáčanie kameňov v cykle uvedenými 2 operáciami.



Obr. 8. Otáčanie kameňov v cykle.

Ďalej sa predpokladá, že v grafe sú presne dva voľné uzly. Algoritmus ďalej vyžaduje, aby tieto dve voľné miesta boli v cieľovom rozostavení na prvých dvoch uzloch počiatočného cyklu, ktorý vznikne pri dekompozícii grafu. Túto podmienku zabezpečia operácie *Uprav-cieľové-rozloženie* a *Dokonči-riešenie*.

Operácia *Uprav-cieľové-rozloženie* najprv určí dve nepretínajúce sa cesty v grafe, ktoré vedú z neobsadených uzlov v cieľovom rozmiestnení do prvých dvoch uzlov počiatočného cyklu dekompozície. Neobsadené miesta sú potom určenými cestami premiestnené na požadované miesto v cieľovom rozmiestnení. Takto upravenú inštanciu dokáže algoritmus BIBOX vyriešiť. Po nájdení riešenia pre takúto inštanciu problému, operácia *Dokonči-riešenie* premiestni neobsadené miesta v grafe späť na ich pôvodné uzly (s využitím ciest, ktoré predtým určila operácia *Uprav-cieľové-rozloženie*).

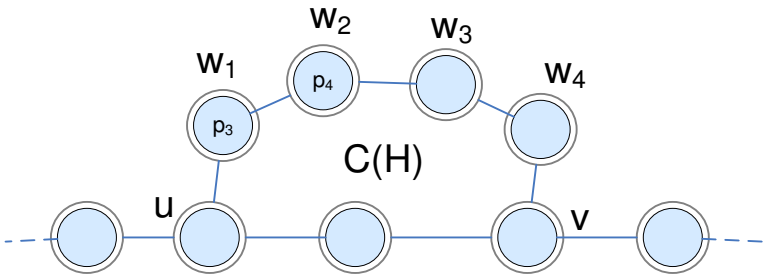
4.1.2 Priebeh algoritmu

Samotný algoritmus sa dá rozdeliť na dve väčšie časti: presúvanie kameňov na cieľové pozície v uchách, ktoré vznikli dekompozíciou, a nakoniec správne umiestnenie kameňov v počiatočnom cykle dekompozície.

Kamene, ktoré majú cieľovú pozíciu v práve riešenom uchu, sa do neho umiestňujú ako do zásobníka. Pre ďalší opis predpokladajme nasledovnú situáciu:

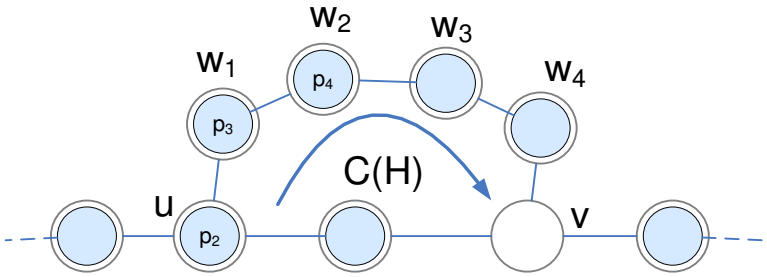
Nech aktuálne spracovávané ucho H sa skladá z uzlov $[u, w_1, w_2, \dots, w_h, v]$ a uzly $w_1, \dots, w_h$ nech sú cieľovými pozíciami pre kamene $p_1, \dots, p_h$. Práve spracovávaný kameň označme p_i a predpokladajme, že cieľovú pozíciu má v uzle w_i. Ďalej predpokladajme, že kamene $p_{i+1}, p_{i+2}, \dots, p_h$ boli už spracované a sú umiestnené v uzloch $w_1, w_2, \dots, w_{h-i}$ Rovnaká situácia sa teraz musí zabezpečiť aj pre kameň p_i. Najprv sa všetky vnútorné uzly ucha H zamknú. Príklad počiatočnej situácie je

znázornený na Obr. 9 (voľné miesto je v nezobrazenej časti grafu). Potrebné je teraz rozlíšiť dva prípady:



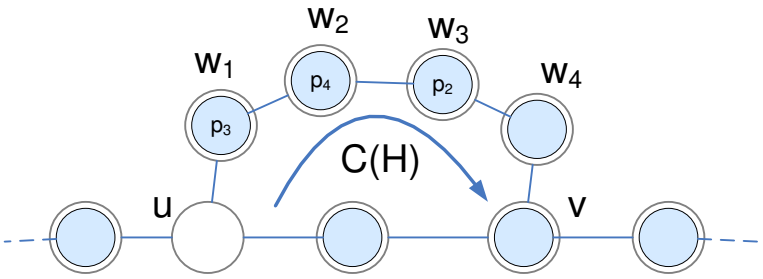
Obr. 9. Počiatočná situácia pri umiestňovaní kameňov do ucha *H*.

Ak sa kameň p_i nenachádza na vnútorných uzloch ucha H, je tento kameň presunutý do uzla u procedúrou *Presuň-kameň*. Kameň p_i sa teraz v uzle u zamkne. Následne je voľné miesto operáciou *Uvoľni-uzol* presunuté do uzla v. Otočením cyklu C H , ktorý je tvorený uchom H a časťou zvyšného grafu, v pozitívnom smere sa iterácia algoritmu skončí. Takáto situácia je pre uzol p_2 znázornená na Obr. 10.



Obr. 10. Umiestnenie ďalšieho kameňa do ucha *H*.

Druhý prípad nastáva vtedy, keď sa kameň p_i nachádza na niektorom z vnútorných uzlov ucha H. Postup riešenia bude rovnaký ako v prvom prípade, je však nutné najprv dostať kameň p_i preč z vnútorných uzlov ucha H. Uzol u sa uvoľní a cyklus $C(H)$ sa otáča v pozitívnom smere, kým sa kameň p_i nedostane do uzla v. Teraz sa vyberie uzol w v nevyriešenej časti grafu (mimo $C(H)$), do ktorého sa kameň p_i presunie a uzamkne. Uzol u sa uvoľní, ak uvoľnený nie je. Cyklus $C(H)$ sa potom otočí rovnaký počet krát v negatívnom smere tak, aby sme dostali počiatočnú situáciu, čo sa týka umiestnenia kameňov $p_{i+1}, p_{i+2}, \dots, p_h$. Riešenie potom môže pokračovať predchádzajúcim spôsobom. Takáto situácia je pre uzol p_2 znázornená na Obr. 11.



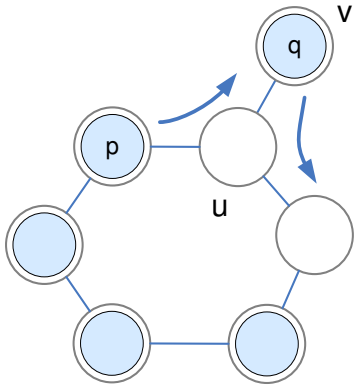
Obr. 11. Počiatočná situácia, ak sa ďalší kameň nachádza vnútri ucha *H*.

Iterácie pokračujú dovtedy, kým kamene $p_1, \ldots, p_h$ nebudú na svojich cieľových pozíciách v uzloch $w_1, .., w_h$. Na všetky uvedené operácie je pritom potrebné len jedno voľné miesto. Takýmto spôsobom sa naplnia všetky uchá, ktoré vznikli pri dekompozícii grafu.

Pri riešení počiatočného cyklu C_0 sa využije druhé voľné miesto. Usporiadanie kameňov v cykle sa chápe ako permutácia. Cieľom je získať správnu permutáciu, ktorá zodpovedá cieľovému rozmiestneniu. Takúto permutáciu je možné získať výmenami dvojíc kameňov. Ak sa kameň p v konkrétnom uzle líši od kameňa q, ktorý má v danom uzle byť (v cieľovom usporiadaní), tieto dva kamene sa vymenia. Výmena kameňov je implementovaná procedúrou *Vymeň-kamene*, ktorú využíva procedúra *Vyrieš-počiatočný-cyklus*, riešiaca počiatočný cyklus dekompozície.

Procedúra *Vymeň-kamene* očakáva, že prvé dva uzly cyklu sú neobsadené. Túto vlastnosť však procedúra nezachováva a teda musí byť zakaždým obnovená. Pri výmene dvojice kameňov sa ďalej využijú dva uzly u a v, ktoré sú spojené hranou, pričom uzol u patrí počiatočnému cyklu dekompozície a uzol v nepatrí.

Pri výmene kameňov p a q je nutné zachovať zoradenie zvyšných kameňov. Najprv kameň, ktorý sa nachádza v uzle v, sa presunie do cyklu, aby sa uzol v uvoľnil. Kamene v cykle sa budú otáčať, kým sa kameň p nedostane do uzla u. Potom sa kameň p presunie do uzla v. Následne sa kamene v kružnici začnú otáčať tak, že kameň q sa dostane na nasledujúci uzol k uzlu u (vzhľadom k pozitívnej orientácii cyklu). Druhé voľné miesto sa dostane na predchádzajúci uzol k uzlu u. Takáto situácia je znázornená na Obr. 12. Teraz sa kamene p a q navzájom vymenia. Cyklus sa teraz otočí v negatívnom smere rovnaký počet krát tak, aby miesto, v ktorom sa nachádzal kameň p, sa nachádzalo v uzle u. Na toto miesto sa umiestni kameň q. Následne sa cyklus otáča v negatívnom smere tak, aby kamene p a q sa nachádzali na pôvodných miestach.



Obr. 12. Výmena dvoch kameňov v počiatočnom cykle dekompozície.

Pomocou opísaných operácií vyzerá hlavná funkcia algoritmu BIBOX nasledovne:

```
BIBOX(graf G, množina kameňov, počiatočné rozloženie kameňov,
konečné rozloženie kameňov)
    D = dekompozícia grafu G
    Uprav-cieľové-rozloženie
    Pre všetky uchá dekompozície D:
        ak je počet uzlov ucha viac ako 2:
            Vyrieš-ucho(H)
    Vyrieš-počiatočný-cyklus
    Dokonči-riešenie
```

Pre inštanciu problému na grafe $G = V, E$ platí, že časová zložitosť uvedeného algoritmu je $O(V^3)$. Počet krokov riešenia je tiež $O(V^3)$. Pamäťová zložitosť je $O(V + E)$.

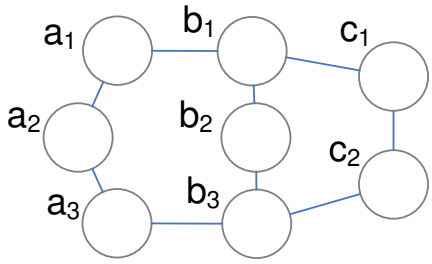
Algoritmus vyžaduje presne dva voľné uzly v grafe. Rozšírenie na viacero voľných miest je možné vykonať tak, že sa na nadbytočné voľné miesta pridajú ďalšie kamene, ktorých pohyby sa však vo finálnom riešení odfiltrujú.

4.2 Algoritmus BIBOX-θ

Nevýhodou algoritmu BIBOX je fakt, že potrebuje najmenej dva neobsadené uzly. Druhé voľné miesto je pritom použité len v poslednej fáze riešenia, v ktorej sú kamene umiestňované do počiatočného cyklu dekompozície. Pri jednom neobsadenom mieste teda algoritmus dokáže umiestniť všetky kamene správne, až na tie, ktoré majú konečnú pozíciu v počiatočnom cykle dekompozície [2,15]. Nasledujúci opis algoritmu, ktorý dokáže riešiť inštancie problému s jedným voľným uzlom je spracovaný z [2,15].

Na vyriešenie počiatočného cyklu je možné použiť existujúci algoritmus MIT [6](pomenovanie algoritmu ako MIT je prevzaté z [1]). Algoritmus MIT dokáže vyriešiť inštancie problému pohybu kameňov po grafe na netriviálnych 2-súvislých grafoch iba s jedným neobsadeným uzlom. Takéto inštancie môžu byť neriešiteľné, čo však algoritmus dokáže určiť. V kombinácii s algoritmom BIBOX môže kombinovaný algoritmus postupovať tak, že všetky uchá zaplní kameňmi až po prvé, ktoré nechá nevyplnené. MIT algoritmus potom vyrieši inštanciu problému na cykle spojenom s prvým uchom. Algoritmus MIT však vytvára veľmi dlhé riešenia.

Algoritmus BIBOX-θ opísaný v [2,15] nadväzuje na túto myšlienku kombinovaného algoritmu. Počiatočný cyklus a prvé ucho tvoria štruktúrou jednoduchý graf (nazývaný θ-graf, znázornený je na Obr. 13), preto je možné pokúsiť sa vyriešiť niektoré inštancie problému na takomto grafe optimálne, čo sa týka počtu krokov riešenia. Vyberať by sa mali také inštancie, z ktorých je možné poskladať celkové riešenie. Riešenie na θ-grafe sa potom poskladá z optimálnych riešení uložených v databáze. Ak potrebné riešenie nie je vypočítané, použije sa napríklad MIT algoritmus.



Obr. 13. θ-graf (prebraté z [2]).

Pre inštanciu problému na grafe $G = V, E$ platí, že časová zložitosť daného algoritmu je $O(V^4)$, počet krokov riešenia je tiež $O(V^4)$. Pamäťová zložitosť je $O(V + E)$ ak sa neuvažuje s databázou vypočítaných riešení.

4.3 Zmenšenie počtu krokov riešenia

V tejto kapitole sú opísané rôzne existujúce prístupy, ktoré umožnia zmenšiť počet krokov riešení vytváraných algoritmami opísanými v kapitolách 4.1 a 4.2.

4.3.1 Zlepšovanie paralelizmu

Oba algoritmy BIBOX aj BIBOX-θ nevyužívajú paralelné pohyby umožnené definíciou problému plánovania cesty pre skupinu robotov. Uvedené algoritmy riešia všeobecnejší

problém pohybu kameňov po grafe. Paralelizmus sa využije až vo fáze spracovania nájdeného riešenia pre pohyb kameňov po grafe. Nasledujúci opis paralelizácie riešenia je spracovaný podľa [16]. Keďže sa využíva paralelizmus umožnený definíciou problému plánovania cesty pre skupinu robotov, v tejto kapitole sa hovorí o robotoch a nie o kameňoch.

Výstupom všetkých opísaných algoritmov je sekvenčné riešenie. Je to také riešenie, v ktorom sa v každom časovom kroku pohne iba jeden robot, ostatné roboty zostávajú na svojich miestach. Takéto riešenie sa teda skladá z množiny pohybov robotov $S = [r_1: u_1 \rightarrow v_1, r_2: u_2 \rightarrow v_2, \dots, r_n: u_n \rightarrow v_n]$, kde r_i je premenná predstavujúca robota, ktorý sa hýbe a u_i a v_i sú premenné predstavujúce začiatočný a konečný uzol pohybu. Niektoré dvojice pohybov je možné vykonať paralelne, iné nie, čo je vyjadrené nasledujúcimi definíciami.

Prekážajúce si pohyby sú definované tak, že pohyb $r_h: u_h \rightarrow v_h$ si prekáža s pohybom $r_k: u_k \rightarrow v_k$ ak prienik množín u_k, v_k a u_h, v_h je neprázdna množina.

$r_k: u_k \rightarrow v_k$ je potenciálne paralelný pohyb so skorším pohybom $r_h: u_h \rightarrow v_h$, vtedy, ak ide o pohyby rozdielnych robotov, $u_h = v_k$ a $v_h \neq u_k$ (a neexistuje iný taký pohyb v čase medzi r_h a r_k, ktorý by si s nimi prekážal). Táto definícia hovorí, že je možné v jednom časovom okamihu posunúť dva roboty „ako vlak“ – zároveň sa posunie jeden robot z u_h do v_h a následne druhý robot z u_k do v_k. Tento vzťah sa zapisuje nasledovne: $r_h: u_h \rightarrow v_h \preccurlyeq r_k: u_k \rightarrow v_k$.

$r_k: u_k \rightarrow v_k$ je triviálne závislý na skoršom pohybe $r_h: u_h \rightarrow v_h$ ak si tieto pohyby prekážajú, ide o pohyby rovnakého robota, $u_h \neq v_k$ alebo $v_h = u_k$ (negácia podmienky z definície potenciálne paralelného pohybu) a neexistuje žiadny iný prekážajúci pohyb v čase medzi nimi, ktorý by si prekážal s r_k alebo r_h. Takéto pohyby nemôžu byť vykonané paralelne. Zapisuje sa to $r_h: u_h \rightarrow v_h \prec r_k: u_k \rightarrow v_k$.

Na vytvorenie riešenia s využitím paralelizmu sa použije funkcia t, ktorá množinu krokov sekvenčného riešenia zobrazí na množinu časových krokov. Funkcia t musí spĺňať podmienky triviálnej závislosti a potenciálne paralelného pohybu, teda ak je pohyb r_h triviálne závislý na pohybe r_k, tak r_k prebehne v skoršom kroku ako r_h. Analogicky, ak r_h je potenciálne paralelný s r_k, tak r_h prebehne v skoršom alebo tom istom kroku.

Paralelné riešenie sa vytvára metódou kritickej cesty. Začína sa prvým pohybom, ktorý sa vykoná v čase 0 a iteratívne sa zo sekvenčného riešenia spracovávajú ďalšie pohyby. Pre každý ďalší pohyb sa určí jeho najskorší možný čas vykonania ako $max\{t'_{\prec} + 1, t'_{\preccurlyeq}\}$, kde $t'_{\prec}$ je čas najneskoršieho pohybu, ktorý je s pridávaným pohybom závislý. $t'_{\preccurlyeq}$ je čas najneskoršieho pohybu, ktorý je s daným pohybom potenciálne paralelný.

4.3.2 Odstraňovanie nadbytočných pohybov

Riešenie tiež môže obsahovať nadbytočné pohyby, ktorých elimináciou sa zlepší počet krokov riešenia. Na odhaľovanie nadbytočných pohybov bol vytvorený grafický nástroj GraphRec [17]. Pomocou tohto nástroja sa našli ďalej opísané nadbytočné pohyby: [18]

Inverzný pohyb – nasledujúci pohyb je opakom predchádzajúceho. Takáto dvojica pohybov sa môže z riešenia vynechať. Vynechaním môže vzniknúť ďalšia dvojica inverzných pohybov, tento proces teda treba opakovať dovtedy, kým sa žiadna taká dvojica nájsť nedá.

Nadbytočné pohyby – zovšeobecnenie inverzných pohybov. Ide o sekvenciu pohybov, pri ktorej sa konkrétny kameň dostane druhýkrát do toho istého uzla, pričom do tohto uzla medzitým žiaden iný kameň nevstúpil.

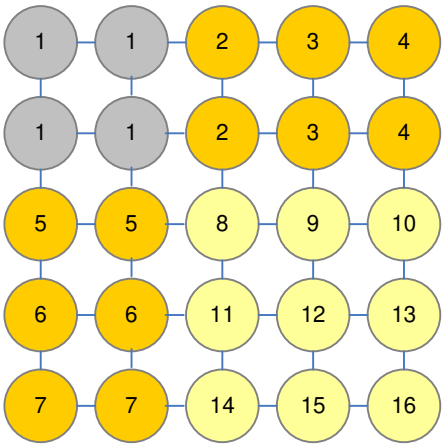
Dlhé sekvencie – zovšeobecnenie nadbytočných pohybov. Ak sa kameň presúva z uzla u do uzla v po určitej ceste, môže existovať kratšia cesta, ktorá sa nevyužíva (žiaden kameň do nej nevstupuje, ani nevystupuje). Pôvodná dlhšia cesta sa potom dá nahradiť kratšou.

Je odporúčané postupovať od špecifických typov nadbytočných pohybov k všeobecnejším, vzhľadom na zvyšujúcu sa výpočtovú náročnosť [18].

4.4 Publikované výsledky

Obsahom tejto kapitoly sú výsledky a zistenia publikované v [2] týkajúce sa porovnávania opísaných algoritmov BIBOX, BIBOX-θ a algoritmu MIT.

Grafy boli generované náhodne, a to dvoch typov: 2-súvislé náhodné grafy a mriežky. Generovanie 2-súvislého grafu prebiehalo tak, že sa najprv vygeneroval cyklus náhodnej dĺžky a k nemu sa na náhodne určené uzly pripájali uchá. Pri grafe v tvare mriežky bol proces opačný, z grafu sa určila dekompozícia (Obr. 14). Pri oboch typoch grafov sa náhodne určilo počiatočné rozloženie aj finálne rozloženie robotov. Riešenie nájdené každým z algoritmov bolo spracované kvôli zvýšeniu paralelizmu.



Obr. 14. Dekompozícia grafu v tvare mriežky s očíslovanými uchami (prebraté z [2]).

V [2] sú opísané výsledky experimentov rôznych typov:

1. Dĺžka riešenia v závislosti od počtu neobsadených uzlov v grafe.
2. Paralelizmus riešenia (pomer počtu pohybov a počtu krokov riešenia) v závislosti od počtu neobsadených uzlov v grafe.
3. Dĺžka riešenia v závislosti od počtu uzlov grafu.
4. Čas behu algoritmu v závislosti od počtu uzlov grafu.

Čo sa týka dĺžky riešenia, algoritmus BIBOX generuje riešenia približne 10 až 100 krát kratšie ako MIT. Pri náhodných 2-súvislých grafoch sa rozdiel zmenšuje, ako sa veľkosť úch zvyšuje. Pri grafoch v tvare mriežky generuje algoritmus BIBOX približne 10-krát kratšie riešenia. Pri dostatočne malom množstve obsadených uzlov bol pozorovaný prudký pokles dĺžky riešenia, čo bolo prisúdené tomu, že vtedy už skoro nedochádza k vzájomnej interakcii medzi robotmi.

Algoritmus BIBOX-θ podľa zistení významne nezlepšuje počet krokov riešenia v porovnaní s algoritmom BIBOX. Teda je vhodný hlavne pre tie inštancie problému, kde je len jeden voľný uzol a algoritmus BIBOX sa tak nedá použiť.

Čo sa týka paralelizmu, tak algoritmy BIBOX poskytujú väčšie možnosti paralelizmu ako algoritmus MIT. Z výsledkov tiež vyplynulo, že paralelizmus koreluje s priemernou dĺžkou ucha, pretože v uchu sa všetky roboty posúvajú naraz (paralelne). Ďalšia pozorovaná korelácia je medzi paralelizmom a priemerom grafu. Je to spôsobené tým, že roboty na ceste sa pohybujú všetky naraz pri premiestňovaní voľného miesta. Priemerná dĺžka takejto cesty koreluje s priemerom grafu.

Pri zisťovaní závislosti dĺžky riešenia od veľkosti grafu, algoritmus MIT obstál najhoršie a BIBOX najlepšie, algoritmus BIBOX-θ skončil medzi nimi.

Čas behu algoritmu sa počítal ako súčet celkového času potrebného na vytvorenie sekvenčného riešenia a času potrebného na spracovanie riešenia za účelom využitia paralelizmu. Čas behu algoritmov BIBOX a MIT bol veľmi podobný, pričom BIBOX bol o niečo rýchlejší.

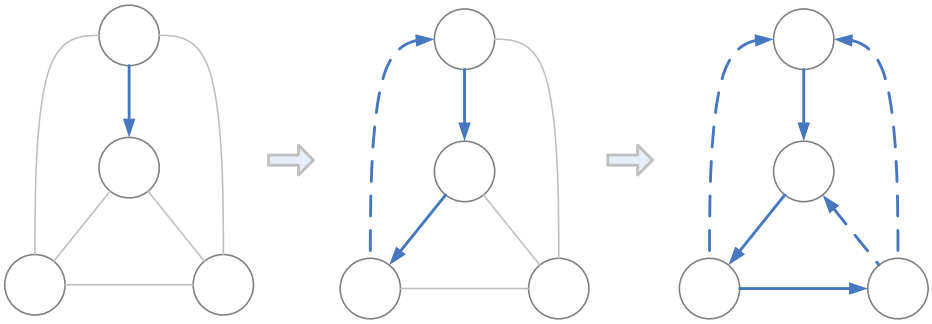
V [1] bol algoritmus BIBOX porovnaný s dvoma zvolenými doménovo nezávislými plánovačmi. Z hľadiska dĺžky riešenia algoritmus BIBOX generoval porovnateľne dobré riešenia ako doménovo nezávislé plánovače. Z hľadiska dĺžky behu však bol algoritmus BIBOX rádovo rýchlejší. Problémom pri doménovo nezávislých plánovačoch je, že v prijateľnom čase dokážu vyriešiť iba inštancie problému na malých grafoch, pričom BIBOX dokáže nájsť riešenia, ktoré pozostávajú z rádovo 1000 pohybov, čo je mimo schopností doménovo nezávislých plánovačov.

Podľa experimentálnych výsledkov v [18] sa pri odstránení nadbytočných pohybov z riešenia zmenší počet krokov až na 50% pri náhodných 2-súvislých grafoch. Pri grafoch v tvare mriežky sa dosiahlo zlepšenie až o 10%. V oboch druhoch grafov sa lepšie výsledky dosiahli pri väčšom počte neobsadených uzlov. Problémom takýchto metód na odstraňovanie nadbytočných pohybov je dlhý čas behu.

5. Štandardný algoritmus dekompozície

Autor algoritmu BIBOX, P. Surynek, v [1] uvádza, že otvoreným problémom zostáva vplyv dekompozície grafu na kvalitu výsledného riešenia. Algoritmus BIBOX a jeho modifikácie sú založené práve na tejto dekompozícii, teda je predpoklad, že by malo byť možné nájsť lepšie riešenie, pomocou vhodnejšej dekompozície grafu. Súčasťou algoritmu BIBOX však nie je algoritmus na dekompozíciu grafu. Ako bolo opísané v kapitole 4.4, pri publikovaných experimentoch bola dekompozícia grafu určená jeho konštrukciou, teda takýto algoritmus nebol potrebný. Obsahom tejto kapitoly je preto opis existujúceho algoritmu na dekompozíciu grafu, spracovaný podľa [19].

Dekompozíciu grafu na cyklus a uchá je možné vo všeobecnosti vykonať pomocou prehľadávania grafu do hĺbky. Pri takomto prehľadávaní začíname v určitom začiatočnom uzle a prechádzame postupne všetky hrany grafu tak, že ako nasledujúcu spracujeme tú hranu, ktorá ešte nebola navštívená a je priľahlá k naposledy navštívenému uzlu. Hranu v,w , prechádzanú v smere od uzla v k uzlu w, nazývame hranou stromu vtedy, ak uzol w je prvýkrát navštívený pri prechádzaní hrany v,w . Označenie takejto hrany je $v \rightarrow w$. Inak sa hrana nazýva spätnou hranou a označuje sa $v \hookrightarrow w$. Výsledkom prehľadávania do hĺbky, ak uvažujeme len hrany stromu, je tzv. DFS strom (z angl. *depth first search*). Príklad prehľadávania do hĺbky je na Obr. 15 (hrana stromu je znázornená plnou šípkou, spätná hrana je znázornená prerušovanou šípkou) [19].



Obr. 15. Prehľadávanie do hĺbky v grafe.

Dekompozícia grafu na uchá je problém, ktorý je v podstate rovnaký ako hľadanie tzv. st-usporiadania grafu. Medzi st-usporiadaním a dekompozíciou na uchá totiž existuje jednoduchý prevod [19]. St-usporiadanie grafu je definované nasledovne:

Nech $G = V, E$ je 2-súvislý graf a $s \neq t \in V$. Usporiadanie $s = v_1, v_2, \ldots, v_n = t$ uzlov grafu G sa nazýva st-usporiadaním, ak pre všetky uzly v_j, $1 < j < n$ existujú $1 \leq i < j < k \leq n$ také, že v_i, v_j , $v_j, v_k \in E$ [19].

Na určenie st-usporiadania grafu a teda aj na dekompozíciu grafu na uchá existuje niekoľko algoritmov, využívajúcich prehľadávanie grafu do hĺbky [19,20]. Pre ďalšie použitie bol však vybratý práve algoritmus [19], ktorý explicitne určuje dekompozíciu grafu na uchá a postupuje inkrementálne, teda v každom kroku algoritmu vráti dekompozíciu 2-súvislej časti dosiaľ spracovaného grafu (tej časti, ktorá obsahuje počiatočnú hranu s, t). Tento algoritmus pracuje v lineárnom čase.

Pseudokód algoritmu podľa [19] vyzerá nasledovne, pričom sú vynechané záležitosti týkajúce sa orientácie hrán, kvôli vytvoreniu st-usporiadania (pri dekompozícii pre účely algoritmu BIBOX nás orientácia hrán nezaujíma):

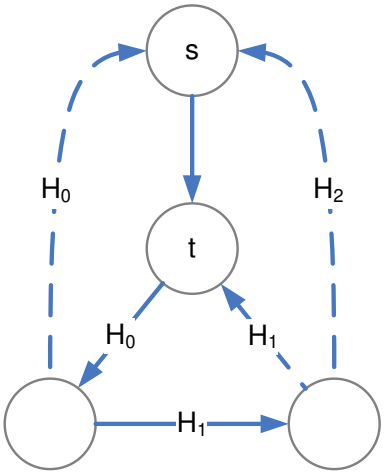
```
Dekomponuj-graf ( graf G = (V,E), hrana  s,t ∈ E )
      inicializuj zoznam uzlov L s uzlami s,t
      Prehľadávaj-do-hĺbky (t)


Prehľadávaj-do-hĺbky ( uzol v )
      pre každý susedný uzol w uzlu v
            ak v ⟶ w (hrana stromu)
                  Prehľadávaj-do-hĺbky(w)
            ak v ↪ w (spätná hrana)
                  nech x je aktuálnym potomkom w                  *
                  označ hranu v ↪ w ako závislú na hrane w ⟶ x
                  ak x ∈ L tak
                        Spracuj-uchá(w ⟶ x)
```

```
Spracuj-uchá ( hrana stromu w ⟶ x )
      pre každú hranu v ↪ w závislú na hrane w ⟶ x
            urči u ∈ L  na ceste z x ⟹ v najbližšie k v          **
            ucho H je u ⇒ v ↪ w
            pridaj vnútorné uzly H do zoznamu L
            pre každú hranu w′ → x′ ucha H
                  Spracuj-uchá(w′ → x′)
      zmaž všetky závislosti na hrane w ⟶ x
```

* Uzol w sa nachádza v už spracovanej časti grafu (keďže $v \hookrightarrow w$ je spätná hrana). Aktuálnym potomkom sa myslí ten uzol, ktorý sa naposledy zvolil v uzle w ako nasledujúci.

** Označenie $x \Longrightarrow v$ predstavuje cestu, ktorá smeruje z uzlu x k uzlu w a skladá sa iba z hrán stromu. Táto cesta môže byť aj prázdna.

Po aplikácii algoritmu na graf z Obr. 15 dostaneme dekompozíciu na uchá, znázornenú na Obr. 16. Treba zdôrazniť, že tento algoritmus nevráti dekompozíciu na počiatočný cyklus a uchá v zmysle definície uvedenej v kapitole 2, ale iba množinu úch. Počiatočný cyklus je tak tvorený počiatočnou hranou s, t a uchom H_0. Ďalej boli v uvedenom príklade identifikované uchá H_1 a H_2.



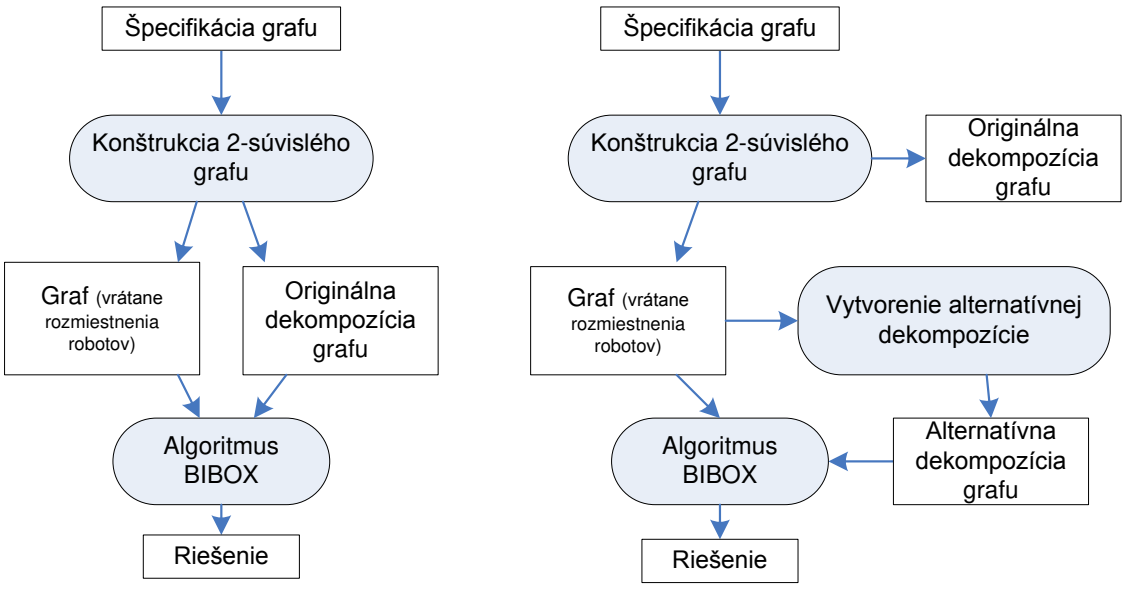
Obr. 16. Dekompozícia grafu algoritmom z [19].

6. Nový návrh a analýza dekompozície grafu pre algoritmus BIBOX

V tejto kapitole je opísaný návrh použitia algoritmu dekompozície z kapitoly 5 s algoritmom BIBOX. Ďalej sú tu uvedené výsledky pri použití dekompozícií vytvorených uvedeným algoritmom. Tieto výsledky sú následne analyzované.

V publikovaných pokusoch opísaných v kapitole 4.4 sa 2-súvislý graf konštruoval postupným pripájaním úch k počiatočnému cyklu. Dekompozícia bola v tomto prípade určená samotnou konštrukciou grafu. Takto vzniknutú dekompozíciu budeme ďalej nazývať originálnou dekompozíciou. Pri hľadaní lepšej, alternatívnej, dekompozície však budeme musieť postupovať opačne, teda 2-súvislý graf bude potrebné rozložiť na počiatočný cyklus a uchá (podľa vety uvedenej v kapitole 2 je to vždy možné).

Na Obr. 17 je znázornený pôvodný postup (podľa kapitoly 4.4) pri pokusoch s algoritmom BIBOX na riešenie plánovania cesty pre skupinu robotov. Na vstupe je špecifikácia grafu (počet úch, dĺžka počiatočného cyklu a pod.), podľa ktorej sa skonštruuje 2-súvislý graf. Výstupom je okrem grafu aj originálna dekompozícia, čo sú dva vstupy pre algoritmus BIBOX. Pri hľadaní lepšej dekompozície, originálnu dekompozíciu nepoužijeme, ale priamo z grafu vytvoríme alternatívnu dekompozíciu, ktorú dáme na vstup algoritmu BIBOX.



Obr. 17. Postup pokusu pri využití originálnej (vľavo) a alternatívnej (vpravo) dekompozície.

Pod lepšou dekompozíciou rozumieme takú dekompozíciu, pri ktorej dosiahne algoritmus plánovania cesty pre skupinu robotov (BIBOX, resp. jeho modifikácie) lepší výsledok ako pri originálnej dekompozícii, konkrétne teda ide o hľadanie kratšieho riešenia.

Otvorenou otázkou je tiež vplyv rozmiestnenia robotov a ich cieľových pozícií na hľadanie lepšej dekompozície. Je totiž možné, že v rovnakom grafe pri rôznych rozloženiach robotov (počiatočnom aj koncovom) dosiahneme lepšie riešenie pri rôznych dekompozíciách. Túto skutočnosť budeme musieť zahrnúť do úvahy pri experimentoch a interpretácii výsledkov.

Algoritmy BIBOX pracujú s uchami v určitom poradí, teda dekompozícia je určená aj poradím jednotlivých úch. Rôzne dekompozície sa tak môžu od seba líšiť napr. iba poradím dvoch úch.

6.1 Návrh použitia

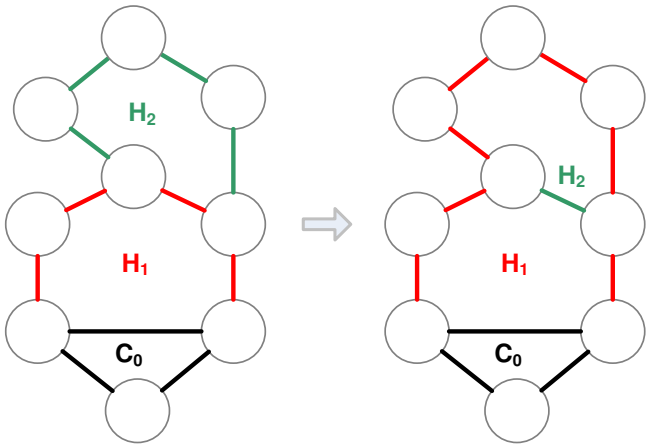
V tejto kapitole je opísaný návrh použitia algoritmu z kapitoly 5 na problém plánovania cesty pre skupinu robotov.

Na vstupe teda máme 2-súvislý graf vygenerovaný postupným pripájaním úch k počiatočnému cyklu. Algoritmus z kapitoly 5 potrebuje na vstup prvú hranu DFS stromu - hranu s, t . Uzol s má zmysel vyberať len z uzlov grafu, ktoré majú stupeň 3 a viac. Uzol t sa potom vyberie zo susedov uzla s. Uzly stupňa 2 nie sú vhodné ako začiatočné uzly, pretože pri nich nemôžeme dostať žiadne nové počiatočné cykly, v porovnaní s tým, keď začneme v najbližšom uzle stupňa 3 a viac. Keby sme vyberali štartovacie uzly aj z uzlov stupňa 2, len by sme zbytočne zvyšovali počet možností pri výbere štartovacieho uzla.

Keďže chceme hľadať rôzne dekompozície, musíme si stanoviť, čím je určená dekompozícia pri konkrétnom grafe. V prvom rade je to výber uzla s, ďalej výber uzla t zo susedných uzlov uzla s. V procedúre *Prehľadávanie-do-hĺbky* je potom dekompozícia určená poradím výberu susedných uzlov. Rôznym výberom uzlov v spomenutých situáciách však nemusíme vždy dôjsť k rôznej dekompozícii.

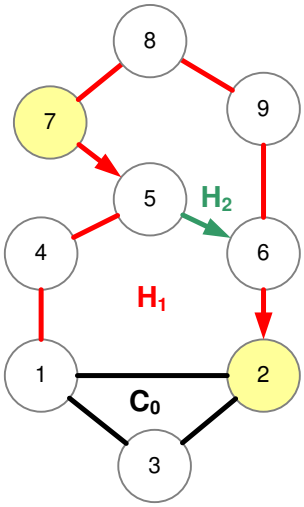
Pri vytváraní alternatívnej dekompozície vzniká otázka, čo s prázdnymi uchami, t.j. takými uchami, ktoré nemajú žiaden vnútorný uzol. Pri konštrukcii 2-súvislého grafu

sa v implementácii algoritmu, z ktorej sme vychádzali (bližšie v prílohe A), uchá vytvárali vždy s aspoň jedným vnútorným uzlom. Pri alternatívnej dekompozícii však prázdne uchá vzniknúť môžu. Uvedená situácia je znázornená na Obr. 18.



Obr. 18. Alternatívna dekompozícia (vpravo) s prázdnym uchom H_2.

Podľa algoritmu BIBOX sa postupne umiestňujú roboty do jednotlivých úch, a tie uchá, ktoré sú už spracované (roboty sú na svojich cieľových pozíciách v danom uchu), sa ďalej pri postupe algoritmu neuvažujú. Prázdne ucho nemá žiadne vnútorné uzly, teda sa do neho ani neumiestňujú roboty. Takéto ucho však práve preto môžeme uvažovať aj ďalej v postupe algoritmu, pretože nám môže skrátiť cestu pri presúvaní robotov do úch s menším poradovým číslom (uchá, vznikajúce pri dekompozícii, sú očíslované vzostupne). Príklad takejto situácie je na Obr. 19. Využitím prázdneho ucha H_2 pri presúvaní robota z uzla 7 do uzla 2 tak môžeme použiť cestu 7-5-6-2. Keby sme však prázdne ucho nevyužili, na presun by sa použili dlhšie cesty 7-5-4-1-2 alebo 7-8-9-6-2.



Obr. 19. Skrátenie cesty využitím prázdneho ucha.

6.2 Výsledky

V tejto kapitole sú uvedené výsledky implementácie algoritmu pre alternatívnu dekompozíciu grafu. Dekompozície sa určovali náhodne, t.j. počiatočný uzol sa určil náhodne, a takisto susedný uzol sa určil vždy náhodne. Pokus prebiehal na zvolených grafoch menšieho rozsahu.

Alternatívna dekompozícia sa vytvárala vždy 1000 krát pre každý z grafov. Nastavenia aj výsledky experimentov sú zaznamenané v Tab. 1. Použitý algoritmus bol BIBOX a nebola použitá žiadna metóda skracovania riešenia z kapitoly 4.3.2. Pod dĺžkou riešenia sa myslí dĺžka riešenia po paralelizácii algoritmom opísaným v kapitole 4.3.1.

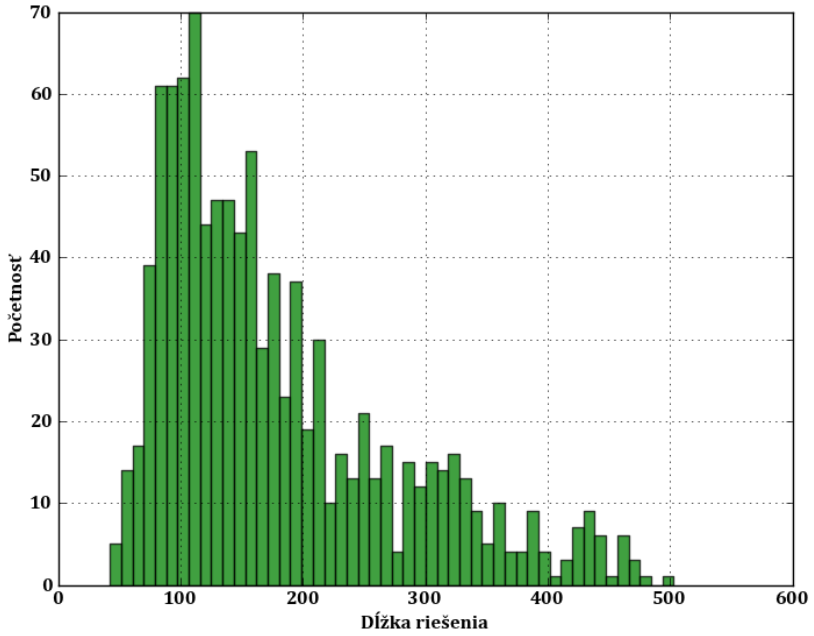
Tab. 1. Výsledky hľadania lepšej alternatívnej dekompozície.

Číslo grafu	1	2	3	4	5
Počet uzlov grafu	20	20	20	23	39
Dĺžka počiatočného cyklu	5	5	4	7	6
Počet voľných uzlov	5	2	2	3	4
Počet úch	5	5	8	7	12
Maximálna dĺžka ucha	5	5	3	4	5
Dĺžka riešenia pri originálnej dekompozícii (podľa P. Suryneka)	47	83	75	132	157
Najlepšie nájdené riešenie pri alternatívnej dekompozícii	**40**	**42**	**47**	**59**	**110**

Z výsledkov v Tab. 1 vidieť, že alternatívnou dekompozíciou je možné nájsť riešenie s výrazne menším počtom krokov, a to pre všetky skúmané grafy. Ďalej bude uvedená analýza riešení, ktoré sa vygenerovali v priebehu pokusu, z hľadiska rôznych parametrov.

6.3 Analýza výsledkov

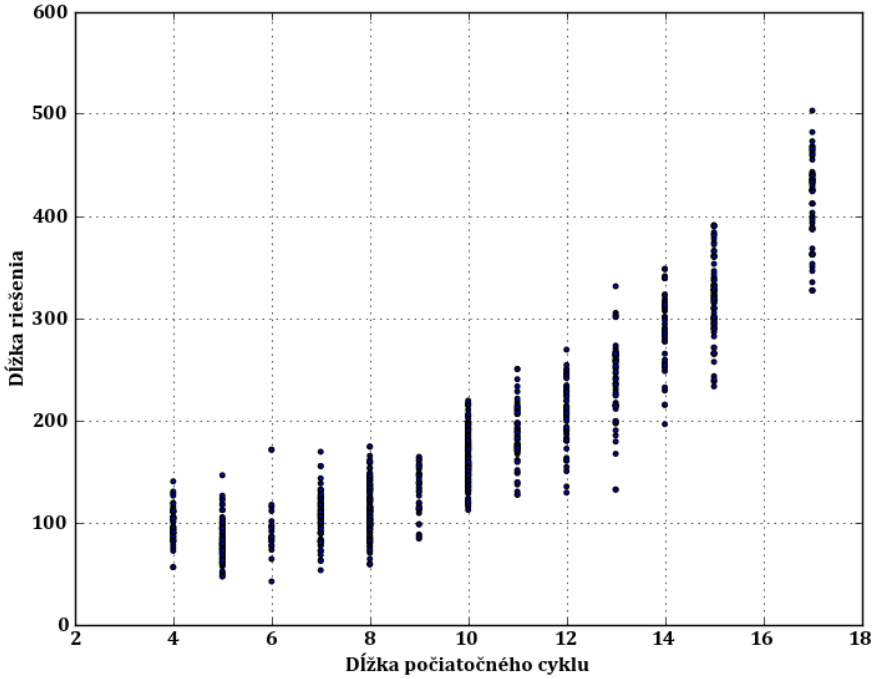
Pre graf č.2 z Tab. 1 je na Obr. 20 uvedený histogram, ktorý znázorňuje rozdelenie dĺžok riešení pri 1000 alternatívnych dekompozíciách, ktoré boli vygenerované v priebehu pokusu. Dĺžky riešení sú rozdelené do 50 rovnako veľkých intervalov a pre každý interval je vypočítaná početnosť riešení, ktoré sa nachádzajú v danom intervale.



Obr. 20. Histogram pre dĺžku riešení pri alternatívnej dekompozícii.

Z Obr. 20 vidíme, že v dĺžkach riešení môže byť rozdiel až jedného rádu (50 – 500). Tiež môžeme vidieť, že rozdelenie nie je symetrické, väčšina dekompozícií má dĺžku riešenia z prvej polovice celkového rozsahu.

Na grafe na Obr. 21 sú znázornené body, ktoré predstavujú závislosť dĺžky riešenia od dĺžky počiatočného cyklu dekompozície, znovu pre graf č.2. Vidieť, že s narastajúcou dĺžkou počiatočného cyklu sa zvyšuje aj dĺžka riešenia. Pre čo najkratšie riešenie je teda dobré vyberať čo najkratšie počiatočné cykly. Takéto zistenie by malo byť základom pre heuristiku, ktorá by vyhľadávanie usmerňovala k lepším riešeniam.

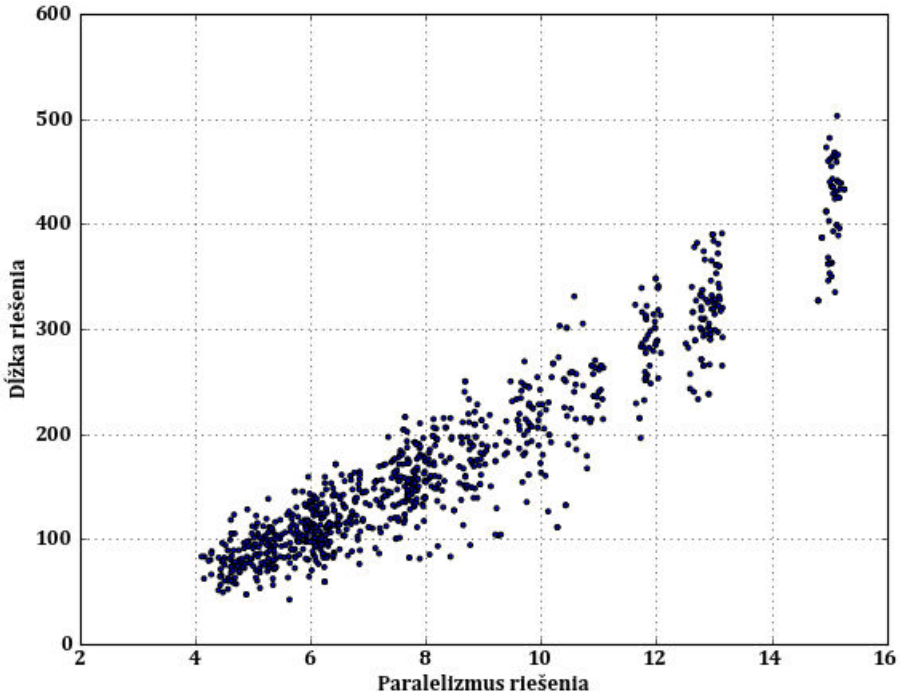


Obr. 21. Závislosť dĺžky riešenia od dĺžky počiatočného cyklu.

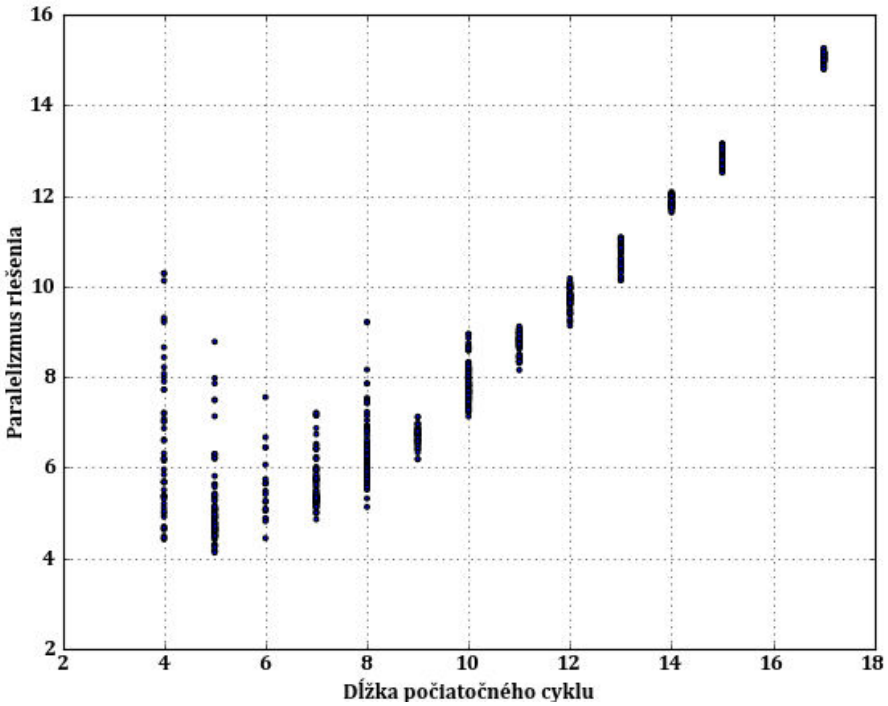
Obr. 22 a Obr. 23 znázorňujú vzťah paralelizmu (pomer počtu pohybov robotov a dĺžky riešenia) s dĺžkou riešenia, resp. dĺžkou počiatočného cyklu. Z Obr. 22 vidieť približnú priamu úmernosť medzi paralelizmom a dĺžkou riešenia, čiže môžeme povedať, že lepšie riešenia majú menší paralelizmus. Na Obr. 23 jednoznačne vidieť, že vysokú hodnotu paralelizmu dosiahneme len s veľkým počiatočným cyklom.

Závislosť medzi paralelizmom a dĺžkou počiatočného cyklu sa dá vysvetliť fungovaním algoritmu BIBOX. Pri riešení počiatočného cyklu dekompozície dochádza k otáčaniu robotov v tomto cykle (táto operácia algoritmu BIBOX je opísaná v kapitole 4.1.2). Pri otáčaní robotov v cykle sa využíva možnosť paralelného posunu, teda v jednom kroku riešenia nastáva viac pohybov robotov. Čím dlhší je počiatočný cyklus, tým väčší je tento pomer a teda aj paralelizmus.

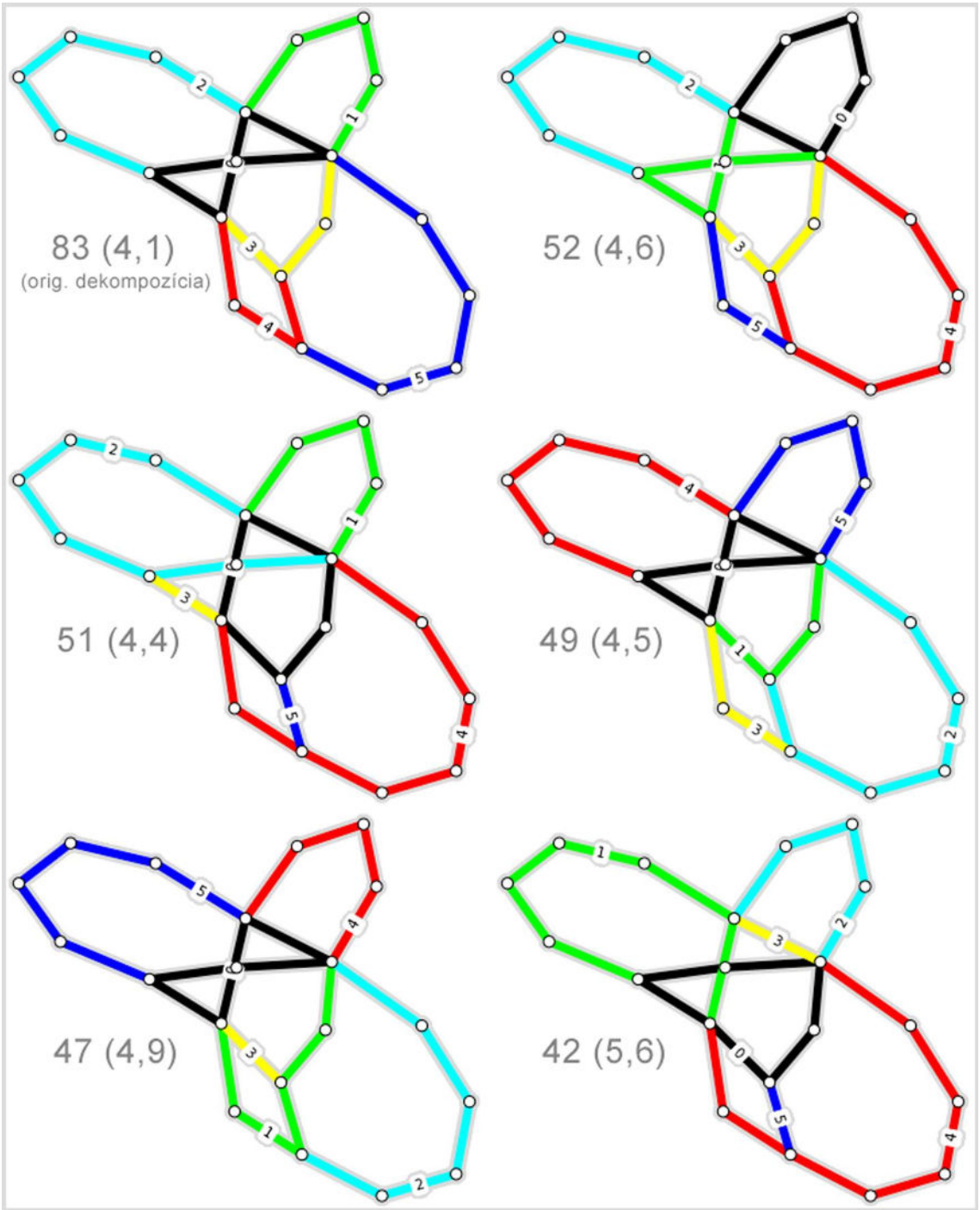
Znovu pre ten istý graf č. 2 je na Obr. 24 porovnanie originálnej dekompozície a 5 najlepších alternatívnych dekompozícií, pričom ku každej dekompozícii je uvedená aj dĺžka riešenia a v zátvorke paralelizmus. Jednotlivé uchá sú očíslované a farebne odlíšené, pričom počiatočný cyklus je vždy čiernej farby a má priradené číslo 0. Dekompozície na Obr. 24 sú rôzne, vyskytujú sa tam aj prázdne uchá, ale nie je to pravidlom. Čo však nevidíme, je rozmiestnenie robotov, teda ich počiatočné a cieľové pozície. Vhodnosť dekompozície určite závisí aj od rozmiestnenia robotov, preto vizuálna analýza dekompozície nemá až taký význam.



Obr. 22. Závislosť dĺžky riešenia od paralelizmu riešenia.



Obr. 23. Závislosť paralelizmu od dĺžky počiatočného cyklu.



Obr. 24. Originálna a najlepšie alternatívne dekompozície grafu č. 2 s dĺžkou riešenia (v zátvorke je hodnota paralelizmu).

7. Analýza prehľadávania stavového priestoru

V predchádzajúcej kapitole sú uvedené výsledky pri náhodnom prehľadávaní možných alternatívnych dekompozícií. Náhodné prehľadávanie je však neefektívne, pretože nevyužíva žiadne heuristické znalosti pre hľadanie lepšieho riešenia. Z rozloženia dĺžok riešení na Obr. 20 vidieť, že na nájdenie čo najlepšej dekompozície je potrebných veľa iterácií. Na druhej strane nám však takéto náhodné prehľadávanie poskytlo vedomosti, na základe ktorých môžeme navrhnúť vhodné heuristiky, a tak proces hľadania zefektívniť. Na návrh lepšieho spôsobu prehľadávania však najprv potrebujeme opísať stavový priestor, čo je predmetom tejto kapitoly.

7.1 Stavový priestor

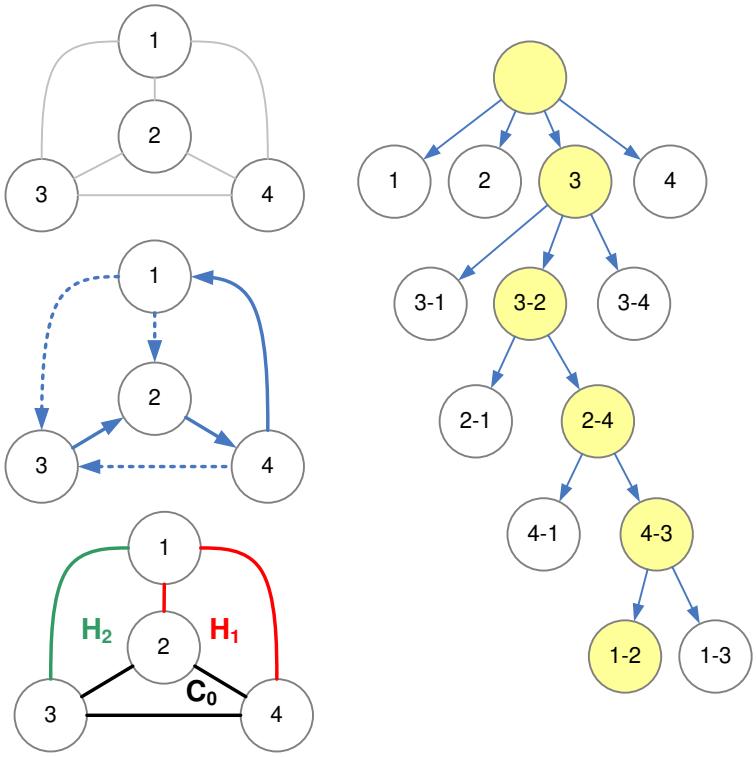
Vo všeobecnosti ide pri vytváraní alternatívnych dekompozícií o prehľadávanie stavového priestoru [21]. Každý stav v stavovom priestore predstavuje úplnú dekompozíciu, resp. čiastočnú dekompozíciu. Podstatou algoritmu na dekompozíciu grafu z kapitoly 5 je prehľadávanie do hĺbky. Stav, ktorý predstavuje čiastočnú dekompozíciu, sa potom dá chápať ako stav algoritmu prehľadávania do hĺbky, v ktorom je potrebné vybrať si jednu z priľahlých nenavštívených hrán. Stav predstavujúci úplnú dekompozíciu potom reprezentuje výsledok algoritmu prehľadávania do hĺbky.

Operátorom pri prehľadávaní je teda výber jednej hrany z množiny priľahlých ešte nenavštívených hrán v algoritme prehľadávania do hĺbky. Špecifickým operátorom je výber začiatočného uzlu, z ktorého sa prehľadávanie do hĺbky začne. V tomto prípade sa vyberá uzol z množiny uzlov grafu so stupňom aspoň 3 (zdôvodnenie je uvedené v kapitole 6.1).

Príklad prehľadávania stavového priestoru pre malý graf so 4 uzlami je na Obr. 25. V ľavej časti tohto obrázku je zhora nadol znázornený samotný graf, graf so znázorneným DFS stromom (plná šípka je hrana stromu, prerušovaná šípka znázorňuje spätnú hranu) a nakoniec samotná dekompozícia grafu na počiatočný cyklus C_0 a uchá H_1,H_2.

V pravej časti Obr. 25 vidieť strom prehľadávania pre uvedený graf. Koreňový uzol tohto stromu predstavuje začiatočný stav, kedy sa prehľadávanie do hĺbky ešte

nezačalo. Potomok koreňového uzla určuje počiatočný uzol, z ktorého sa začne prehľadávanie do hĺbky. Ďalšie uzly reprezentujú výber nenavštívenej hrany, napr. v uzle 3 sa rozhodujeme, či budeme pokračovať hranou 3-1 (hranou medzi uzlami 3 a 1), 3-2, alebo 3-4. Hrany, ktoré si vyberáme, nemusia za sebou nasledovať, ako to vidieť v prípade uzla 4-3. Po výbere tohto uzla prehľadávanie do hĺbky pokračuje jedinou možnou hranou 4-1. Keďže v tomto okamihu nedochádza k žiadnemu výberu hrany, hrana 4-1 sa v zvýraznenej postupnosti uzlov vôbec nenachádza. Až v uzle 1 sa vyberá, či sa bude pokračovať hranou 1-2 alebo 1-3. Po výbere hrany 1-2 prehľadávanie do hĺbky ešte navštívi zostávajúcu hranu 1-3. Prehľadávanie stavového priestoru potom končí. V tomto okamihu je dekompozícia grafu kompletná, a je možné spustiť samotný algoritmus BIBOX.



Obr. 25. Príklad stromu prehľadávania (vpravo), zodpovedajúci DFS strom (vľavo v strede) a dekompozícia (vľavo dole).

Pri takto definovanom stavovom priestore vieme ohodnotiť stavy zodpovedajúce úplnej dekompozícii, a to tak, že s takouto dekompozíciou spustíme algoritmus BIBOX (resp. jeho modifikácie). Ohodnotením stavu potom bude výsledná dĺžka riešenia. V stavoch zodpovedajúcich čiastočnej dekompozícii zatiaľ nevieme spraviť žiadny odhad kvality úplnej dekompozície, ktorú môžeme z daného stavu dosiahnuť.

Keďže nevieme žiadnu prípustnú heuristiku, nemôžeme na hľadanie použiť algoritmus A*, resp. jeho modifikácie. Pri genetických algoritmoch by sa ťažko reprezentovalo riešenie, pretože každá dekompozícia môže byť určená inou postupnosťou výberu hrán v algoritme prehľadávania do hĺbky. Nejde len o usporiadanie hrán v postupnosti, ale aj o to, že v každej postupnosti niektoré hrany chýbajú (napr. v postupnosti uzlov zvýraznenej na Obr. 25, chýba hrana 4-1). Krížením a mutáciou by tak vznikali gény, ktoré nereprezentujú žiadnu dekompozíciu.

7.2 Odhad veľkosti stavového priestoru

Pre predstavu o veľkosti stavového priestoru pre určitý 2-súvislý graf nás zaujíma odhad počtu konečných stavov v stavovom priestore, teda počet potenciálne rôznych dekompozícií. Pri odhade veľkosti budeme vychádzať z príkladu na Obr. 25. Predpokladajme, že vytvárame graf nasledovným spôsobom (mierne modifikovaná konštrukcia grafu z kapitoly 4.4):

1. Graf vytvárame z počiatočného cyklu.
2. Opakovane pripájame k dosiaľ skonštruovanému grafu H úch tak, že nikdy nevznikne uzol väčšieho stupňa ako 3. Pre účel odhadu predpokladajme, že je to vždy možné. Pripojením ucha tak vždy vzniknú 2 nové uzly stupňa 3. Celkový počet uzlov stupňa 3 v grafe je potom $2H$.

V strome predstavujúcom stavový priestor je na začiatku potrebné vybrať štartovací uzol z množiny uzlov stupňa 3. Ďalej je potrebné vybrať jednu z 3 hrán, ktorou sa bude pokračovať. V každom ďalšom doposiaľ nenavštívenom uzle stupňa 3, do ktorého sa prehľadávanie dostane, si bude treba zvoliť jednu z dvoch hrán, ktorou sa bude pokračovať. Ďalšie možnosti výberu už neexistujú. Celkový odhad počtu konečných stavov stavového priestoru S pre graf s H uchami teda je:

$$S = (2H) \times 3 \times 2^{2H-1}$$

Stále však ide len o odhad, pretože predpokladáme, že nevznikne uzol stupňa 4 alebo viac. Treba tiež zdôrazniť, že počet dekompozícií je menší ako počet stavov, pretože jedna dekompozícia môže vzniknúť rôznymi spôsobmi, teda konečné stavy nebudú vždy jedinečné. Pre graf na Obr. 25 s 2 uchami uvedený vzorec poskytuje číslo 96. Pre graf, ktorý má napr. 10 úch, dostávame ako odhad počtu konečných stavov číslo 3×10^7.

8. Návrh nových heuristík

V kapitole 7.1 bol spomenutý operátor pri prehľadávaní stavového priestoru, ktorý spočíval vo výbere jednej hrany z množiny priľahlých ešte nenavštívených hrán. Tento operátor však môže pri výbere hrany zohľadniť heuristické znalosti a tak smerovať prehľadávanie stavového priestoru k lepším riešeniam. Obsahom tejto kapitoly bude práve opis takýchto heuristík.

8.1 Heuristika krátkeho počiatočného cyklu

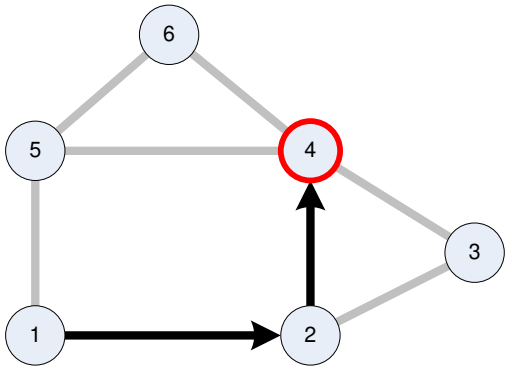
Ako je uvedené v analýze výsledkov v kapitole 6.3, lepšie riešenia dosiahneme vtedy, keď je počiatočný cyklus dekompozície krátky. Príslušná heuristika by teda pri výbere nasledujúcej hrany pri prehľadávaní stavového priestoru mala preferovať také hrany, ktoré vedú ku kratšiemu počiatočnému cyklu. Táto heuristika je samozrejme aktívna len dovtedy, kým sa počiatočný cyklus nevytvorí.

Zadefinujme si množinu susedných uzlov, ktoré patria nenavštíveným susedným hranám, ako množinu nasledujúcich uzlov. V okamihu, keď je potrebné vybrať si ďalšiu hranu z množiny nenavštívených susedných hrán a ešte nebol vytvorený počiatočný cyklus, táto heuristika postupuje nasledovne:

1. Pre každý z nasledujúcich uzlov sa vypočíta jeho najkratšia vzdialenosť od prvého uzla dekompozície (je to uzol, v ktorom počiatočný cyklus začína a musí v ňom aj končiť). Pri určovaní najkratšej cesty, sa však musia brať do úvahy iba tie cesty, ktorých vnútorné uzly sú zatiaľ nenavštívené. Iba takýmito uzlami bude môcť prehľadávanie pokračovať. Medzi navštívené uzly patrí aj aktuálny uzol. Vzdialenosť sa nemusí dať vždy určiť v prípade, že neexistuje vhodná cesta medzi daným uzlom a prvým uzlom dekompozície.
2. Z dvojíc uzol – vzdialenosť si vyberieme tú dvojicu, ktorá má vzdialenosť najmenšiu. Keď je takých dvojíc viac, výber môže prebiehať napr. náhodne.

Rozhodovanie na základe tejto heuristiky je znázornené na Obr. 26. Predpokladajme, že dekompozícia sa začala v uzle 1 a prehľadávanie do hĺbky pokračovalo uzlami 2 a 4. Teraz sa v uzle 4 rozhodujeme medzi nasledujúcimi uzlami: 5, 6, 3. Cez uzol 3 nevedie cesta do počiatočného uzla dekompozície, tak aby využívala

iba nenavštívené uzly. Cez uzol 6 vedie k počiatočnému uzlu dekompozície cesta dĺžky 3 a ak by sa zvolil uzol 5, bola by to cesta dĺžky 2. Ako nasledujúci uzol sa teda vyberie uzol 5, čím sa dosiahne pri danom počiatočnom uzle dekompozície najkratší možný cyklus.



Obr. 26. Rozhodovanie pri heuristike krátkeho počiatočného cyklu.

8.2 Heuristika minimálneho počtu kolízií

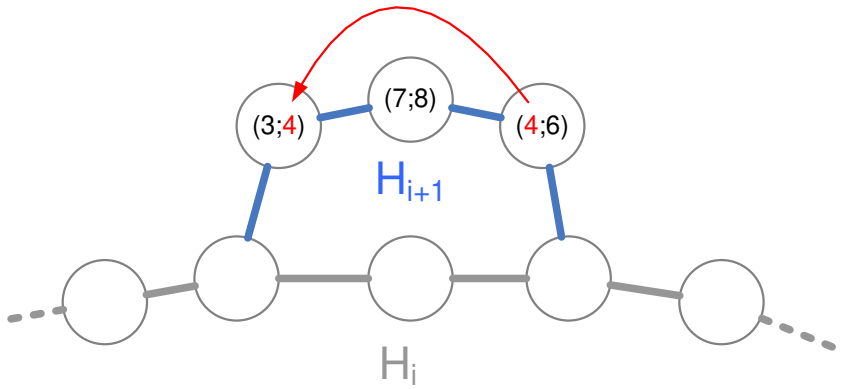
Ďalšia heuristika vychádza z analýzy samotného algoritmu BIBOX. V kapitole 4.1.2 je opísaný postup, akým sa roboty umiestňujú na cieľové pozície vo vnútorných uzloch ucha. Rozlišujú sa vtedy nasledujúce dva prípady, ktoré sa týkajú aktuálnej pozície robota, ktorý ide byť presunutý na svoju cieľovú pozíciu:

1. Daný robot sa aktuálne nachádza vo vnútornom uzle iného, ako práve spracovávaného ucha.
2. Daný robot sa aktuálne nachádza vo vnútornom uzle práve spracovávaného ucha.

Ako je uvedené v opise algoritmu BIBOX, tak spomenutý druhý prípad sa rieši tým spôsobom, že daný robot sa premiestni do iného uzla tak, aby sme dostali situáciu opísanú v prvom prípade. Riešenie druhého prípadu, v porovnaní s prvým prípadom, teda zahŕňa aj pohyby robotov potrebné na premiestnenie daného robota na uzol mimo ucha. Z hľadiska počtu krokov riešenia teda ide o náročnejšiu alternatívu. Ďalej opísaná heuristika sa snaží práve o to, aby takáto situácia nastávala čo najmenej krát, čím by mala prispieť ku kratšiemu riešeniu. Pre jednoduchší opis heuristiky je však vhodné zadefinovať si najprv niektoré pojmy.

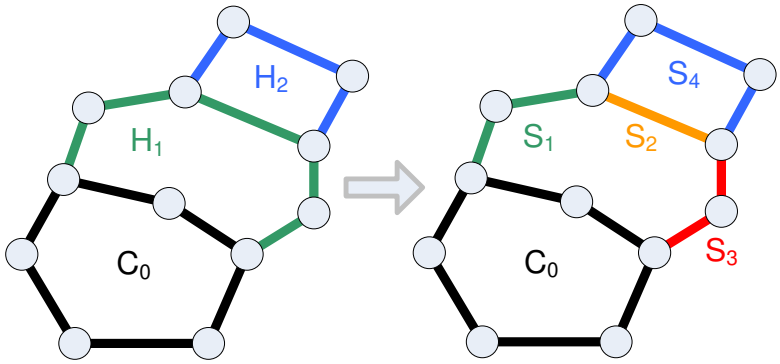
Kolízia – situácia, pri ktorej sa robot nachádza vo vnútornom uzle ucha a zároveň má v danom uchu aj cieľovú pozíciu (uvedený prípad č. 2). Príklad kolízie je

znázornený na Obr. 27, kde je znázornené ucho H_{i+1}, v ktorého uzloch sú dvojice čísel, ktoré predstavujú číslo robota, ktorý sa v danom uzle aktuálne nachádza a číslo robota, ktorý má v danom uzle cieľovú pozíciu. Ku kolízii dochádza pri robotovi číslo 4, ktorý sa nachádza vo vnútornom uzle ucha a zároveň má vo vnútornom uzle ucha cieľovú pozíciu.



Obr. 27. Znázornenie kolízie - robot č. 4 má cieľovú pozíciu v uchu, v ktorom sa nachádza.

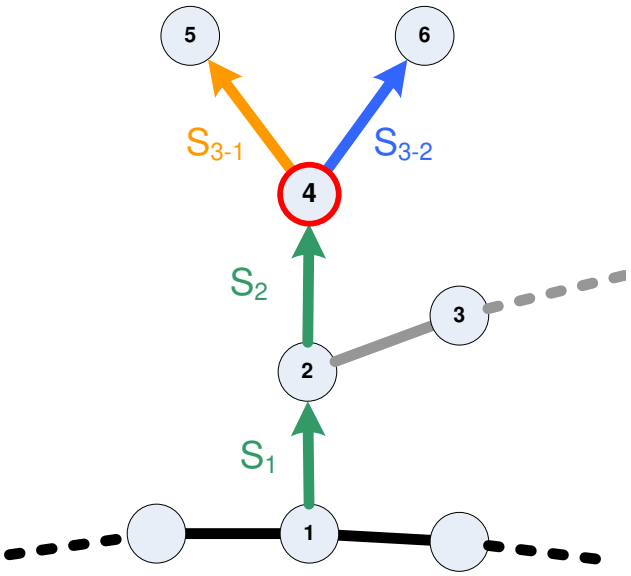
Segment – najdlhšia taká cesta v grafe, ktorej vnútorné uzly sú stupňa 2. Ide v podstate o rovnakú definíciu, akou je definícia ucha v kapitole 2. Ucho však musíme chápať ako samostatnú časť dekompozície grafu, pretože iba vtedy platí, že každé také ucho má vnútorné uzly stupňa 2. V samotnom grafe, ktorý sa skladá z viacerých úch, však ucho môže mať vnútorné uzly aj stupňa 3 a viac, pretože sa na tieto uzly môžu pripájať ďalšie uchá. Preto bol zavedený nový pojem segment. Platí pritom, že ucho sa skladá z jedného alebo viac segmentov. Rozdiel medzi segmentom a uchom znázorňuje Obr. 28. Vidieť, že ucho H_1 sa skladá z 3 segmentov: S_1, S_2, S_3, zatiaľ čo ucho H_2 sa skladá iba zo segmentu S_4.



Obr. 28. Rozdiel medzi uchami (vľavo) a segmentmi (vpravo).

Ďalej opisovaná heuristika sa používa vtedy, keď je potrebné si vybrať, ktorou hranou bude prehľadávanie do hĺbky pokračovať. Výsledné rozhodnutie heuristiky by malo

zohľadňovať to, aby ucho, ktoré sa práve vytvára, malo čo najmenší počet kolízií. Aktuálna situácia, v ktorej je potrebné použiť heuristiku, môže vyzerať ako na Obr. 29. Prostredníctvom prehľadávania do hĺbky je už nájdené nejaké ucho, resp. počiatočný cyklus, na obrázku je znázornené čiernou farbou. Predpokladajme, že z uzla č.1 prehľadávanie do hĺbky pokračuje a hľadá sa ďalšie ucho. Prehľadávanie pokračuje segmentom S_1. V uzle č. 2 je potrebné sa rozhodnúť, či pokračovať smerom k uzlu 3 alebo 4. Predpokladajme, že sa zvolil uzol č. 4 a segment medzi uzlami 2 a 4 označme S_2. V uzle č. 4 je potrebné sa rozhodnúť, či prehľadávanie do hĺbky bude pokračovať smerom k uzlu č.5 alebo k uzlu č. 6. Z hľadiska segmentov tak máme na výber segment S_{3-1} alebo S_{3-2}. Na Obr. 29 nie sú znázornené vnútorné uzly segmentov, iba ich začiatočné a koncové uzly.



Obr. 29. Príklad situácie pre heuristiku minimálneho počtu kolízií.

Každý segment je charakterizovaný robotmi, ktorí sa v ňom nachádzajú, a robotmi, ktorí majú v danom segmente cieľové pozície. Chceme teda vybrať taký segment, ktorý by priniesol do práve vytváraného ucha čo najmenej kolízií. Kolízie môžu nastať niekoľkými spôsobmi:

1. Robot v nasledujúcom segmente (S_{3-1} alebo S_{3-2}) môže mať cieľovú pozíciu v rovnakom segmente.
2. Robot v nasledujúcom segmente (S_{3-1} alebo S_{3-2}) môže mať cieľovú pozíciu v predchádzajúcich segmentoch (S_1 alebo S_2).
3. Robot v predchádzajúcich segmentoch (S_1 alebo S_2) môže mať cieľovú pozíciu v nasledujúcom segmente (S_{3-1} alebo S_{3-2}).

Ďalej je potrebné uviesť, či sa pri počítaní kolízií uvažuje s robotmi, ktorí sa nachádzajú, resp. ktorí majú cieľovú pozíciu, v začiatočnom alebo koncovom uzle segmentu. Začiatok a koniec segmentu sa chápe v smere prehľadávania do hĺbky (na Obr. 29 je to smer šípky). Začiatočný uzol segmentu sa nikdy neuvažuje pri počítaní kolízií, pretože vždy ide o jednu z nasledujúcich možností:

1. Začiatočný uzol je uzlom už existujúceho ucha, resp. počiatočného cyklu. V takom prípade ku kolízii nedochádza, keďže roboty sa umiestňujú len do vnútorných uzlov ucha (na Obr. 29 ide o uzol č. 1).
2. Začiatočný uzol segmentu bol zarátaný ako konečný uzol predchádzajúceho segmentu (na Obr. 29 ide napr. o uzol č. 4, ktorý je súčasťou segmentu S_2, teda pri segmentoch S_{3-1} a S_{3-2} sa už neuvažuje).

Koncový uzol segmentu sa uvažuje vždy, okrem prípadu, že daný uzol je súčasťou existujúceho ucha, resp. počiatočného cyklu. V takom prípade ku kolízii nedochádza, keďže roboty sa umiestňujú len do vnútorných uzlov ucha.

Počet pohybov robotov, ktoré je potrebné vykonať na ich umiestnenie mimo ucha, však závisí aj od dĺžky ucha, pretože daného robota je potrebné najprv premiestniť do prípojného uzla (uzla, v ktorom sa ucho pripája na zvyšok grafu). Bližšie je tento proces opísaný v kapitole 4.1.2. Pri rozhodovaní by sme teda mali brať do úvahy nielen počet kolízií, ale aj dĺžku potenciálneho ucha. Pre každý z nasledujúcich segmentov teda vypočítame nasledovný výraz:

$$n_i = \ k_{1,i} + k_{2,i} + k_{3,i} \ * (s + s_i),$$

kde $k_{1,i}$, $k_{2,i}$, $k_{3,i}$ sú počty jednotlivých typov kolízií pre segment i, s je dĺžka všetkých predchádzajúcich segmentov a s_i je dĺžka segmentu i, pre ktorý daný výraz počítame. Výrazom $(s + s_i)$ teda odhadujeme dĺžku ucha, ktoré sa práve vytvára.

Následne sa vyberie ten segment, ktorý má najmenšiu hodnotu n_i, čím by sa mal minimalizovať počet kolízií v práve vytváranom uchu. Ide v podstate o lačný (angl. *greedy*) algoritmus, teda v aktuálnej situácii sa vyberá to riešenie, ktoré sa zdá byť najvýhodnejšie.

Táto heuristika sa prirodzene dopĺňa s heuristikou krátkeho počiatočného cyklu, opísanou v predchádzajúcej kapitole. Heuristika krátkeho počiatočného cyklu je totiž

aktívna do okamihu nájdenia počiatočného cyklu, potom nastupuje heuristika opisovaná v tejto kapitole. Nemá zmysel použiť opisovanú heuristiku na hľadanie počiatočného cyklu, pretože pri riešení počiatočného cyklu sa roboty presúvajú iným spôsobom ako pri riešení jednotlivých úch (detailnejší opis je v kapitole 4.1.2).

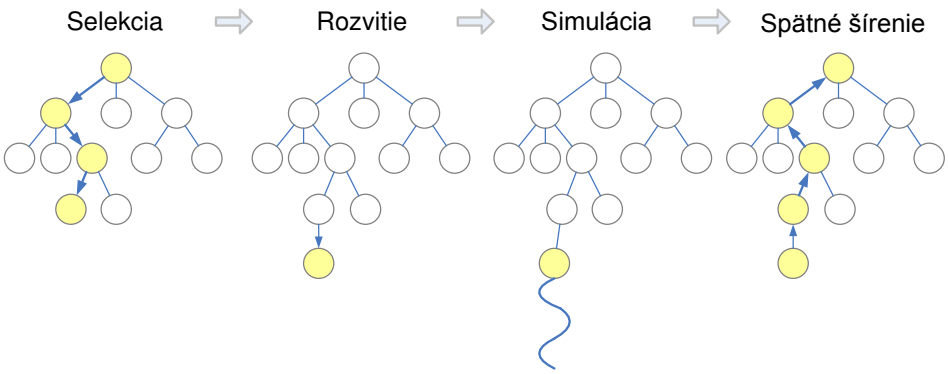
9. Monte-Carlo prehľadávanie stromu

Hľadanie vhodnej dekompozície, teda aj prehľadávanie príslušného stavového priestoru, sa dá chápať aj ako hra pre jedného hráča. Tento hráč si musí vybrať počiatočný uzol a následne vyberá hrany, ktorými bude pokračovať algoritmus prehľadávania do hĺbky. Výsledné skóre takejto hry je dĺžka riešenia, získaná spustením algoritmu BIBOX nad takouto dekompozíciou. Hráč sa snaží nahrať čo najmenšie skóre. Pri takomto chápaní problému je možné použiť aj metódu Monte-Carlo prehľadávania stromu, ktorá je ďalej opísaná.

Monte-Carlo metódy sú populárne v oblasti hier, pri ktorých sa najprv používali ako spôsob vyhodnotenia v klasickom strome hľadania. Monte-Carlo prehľadávanie stromu (ďalej len MCPS) je metóda, pri ktorej je samotné prehľadávanie stromu riadené Monte-Carlo simuláciou. Je to hľadanie typu „najprv najlepší“. Úspešne bola táto metóda aplikovaná na hru Go, ktorá má veľký priestor prehľadávania, ale aj na iné doskové, či počítačové hry. Vhodná je tiež pri tých problémoch, pri ktorých nevieme stanoviť, resp. nepoznáme prípustnú heuristiku [22,23].

MCPS bolo úspešne aplikované aj na hry pre jedného hráča [23]. Keďže problém hľadania vhodnej dekompozície sa dá chápať ako hra pre jedného hráča, ďalší opis MCPS bude preto spracovaný práve podľa [23].

MCPS buduje strom hľadania s využitím Monte-Carlo simulácie v listoch stromu. Každý uzol v strome reprezentuje aktuálny stav hry a typicky uchováva priemerné skóre dosiahnuté v danom podstrome a počet návštev uzla. Vo všeobecnosti MCPS pozostáva zo 4 krokov, ktoré sa opakujú, a to: selekcia, rozvitie, simulácia a spätné šírenie. Tieto kroky sú znázornené na Obr. 30 a ďalej opísané.



Obr. 30. Základné operácie Monte-Carlo prehľadávania stromu (prebraté z [22]).

Úlohou selekčnej stratégie je vybrať potomka konkrétneho uzla v strome. Mala by zohľadňovať dve protichodné požiadavky. Na jednej strane sa chceme zamerať na najlepšie uzly, ktoré sme dosiaľ našli, na druhej strane chceme preskúmať aj iné uzly, ktoré môžu viesť k lepším výsledkom. Podľa [24] je ešte vhodné používať selekčnú stratégiu len vtedy, ak počet návštev uzlu bol vyšší ako stanovená hranica. Predtým, ako je táto hranica prekročená, sa používa simulačná stratégia.

V prípade dosiahnutia listu, stratégia rozvitia určí, ktoré uzly sa pridajú do stromu. V [23] bol použitý prístup, pri ktorom sa pridá iba jeden uzol, ktorý tak zodpovedá prvému stavu hry, ktorý sa v strome nenachádzal.

Vo fáze simulácie pokračuje ďalej hra od listu stromu náhodným spôsobom. Na zlepšenie kvality výsledného riešenia sa môže pri simulácii zohľadniť heuristika, ak je nám známa.

Počas fázy spätného šírenia sa výsledky simulácie šíria z listu až do koreňového uzla. Zvykne sa používať priemer z dosiahnutého skóre, taktiež sa aktualizuje počet návštev každého uzla.

10. Hľadanie lepšej dekompozície Monte-Carlo prehľadávaním stromu

Monte-Carlo prehľadávanie stromu (MCPS) sa teda hodí pre problém hľadania vhodnej dekompozície, pretože nie je známa prípustná heuristika a stavový priestor je pomerne rozsiahly. Do algoritmu MCPS tiež môžeme zapojiť heuristiky opísané v kapitole 8. MCPS sa dá tiež použiť ako náhodné prehľadávanie, ktoré bolo predmetom pokusov v kapitole 6.2, vtedy, ak nepoužijeme pri simulácii žiadnu heuristiku a nepoužijeme ani selekčnú stratégiu.

MCPS prehľadáva strom hľadania, ktorého štruktúra bola opísaná v kapitole 7.1. Uzol takéhoto stromu hľadania uchováva:

- Číslo uzla grafu, v ktorom sa rozhodovalo o nasledujúcom uzle.
- Číslo zvoleného nasledujúceho uzla grafu.
- Odkaz na rodičovský uzol stromu (pre spätné šírenie informácií o výsledkoch).
- Počet návštev uzla stromu (pri spätnom šírení, od rozvitého uzla po koreňový uzol, sa počet návštev každému uzlu na ceste zvýši o 1).
- Priemerný dosiahnutý výsledok v podstrome konkrétneho uzla stromu.
- Najlepší výsledok dosiahnutý v podstrome konkrétneho uzla stromu.
- Podľa potreby selekčnej stratégie ďalšie štatistické informácie.

Výsledkom behu algoritmu BIBOX (resp. jeho modifikácií) s konkrétnou dekompozíciou je dĺžka riešenia, pričom čím kratšie riešenie, tým je lepšie. Ide o odlišnosť od hry použitej v [23], kde išlo o to, nahrať čo najvyššie skóre.

Pri rozvíjaní sme sa rozhodli použiť prístup z [23] opísaný v kapitole 9, teda rozvíjať sa vždy bude iba jeden uzol. Pri spätnom šírení sa budú aktualizovať opísané hodnoty, ktoré sa v uzle ukladajú. Zo súčastí MCPS zostáva teda opísať možnosti selekčnej a simulačnej stratégie, čo je predmetom nasledujúcich kapitol.

10.1 Selekčná stratégia

Pri selekcii si musíme zvoliť spôsob ako vyberieme nasledujúci uzol v strome hľadania tak, aby sa z dlhodobého hľadiska preferovali výhodnejšie uzly (teda uzly, ktoré vedú k lepším výsledkom). Selekčná stratégia by mala mať možnosť nastavenia

koeficientom, aby bolo možné vybalansovať preskúmavanie najsľubnejších uzlov s preskúmavaním horších uzlov, pretože v ich podstrome sa môže nachádzať aj oveľa lepšie riešenie.

V [23] sa uvádza stratégia selekcie, ktorá maximalizuje hodnotu nasledujúceho vzťahu:

$$X + C \times \frac{\ln t(N)}{t(N_i)} + \frac{X^2 - t\ N_i\ \times X^2 + D}{t(N_i)}$$

kde N predstavuje uzol stromu, v ktorom sa vyberá niektorý potomok – uzol N_i. Počet návštev uzla N, resp. N_i, je reprezentovaný výrazom $t\ N$, resp. $t\ N_i$. X predstavuje priemerný výsledok v podstrome uzla N. Druhý výraz v tomto vzťahu predstavuje horný odhad spoľahlivosti pre priemerný výsledok (X). Tretí výraz reprezentuje možnú odchýlku potomka – uzla N_i. Nastavením koeficientov C a D môžeme určovať, do akej miery sa prehľadávanie zameria na najlepšie uzly [23].

Výhradou pre použitie takéhoto vzťahu je faktor vetvenia stromu prehľadávania. V práci [23] autori odhadli priemerný faktor vetvenia na $20{,}7$, zatiaľ čo pri strome hľadania pre dekompozície má väčšina uzlov dvoch potomkov, teda priemerný faktor vetvenia je rádovo menší. Pre výber medzi niekoľkými potomkami je potom takýto vzťah príliš zložitý.

Pre účely selekcie sme namiesto toho chceli navrhnúť jednoduchší systém. Vlastnosti takéhoto systému by mali byť nasledovné, pričom predpokladáme, že každý potomok uzla má priradenú určitú hodnotu, nazvime ju kvalita, ktorá určuje, aké dobré riešenia očakávame v danom podstrome:

- Potomkovia s rovnakou kvalitou by mali mať rovnakú pravdepodobnosť výberu.
- Potomok s horšou kvalitou by mal mať menšiu pravdepodobnosť výberu ako potomok s lepšou kvalitou.
- Aj potomok, ktorý nebol navštívený, by mal mať nenulovú pravdepodobnosť výberu.
- Možnosť nastavenia koeficientom, ktorý by určoval preferenciu sľubnejších riešení.

Takéto podmienky spĺňa nasledovný systém. Pre každého potomka sa určí hodnota kvality. Môže to byť buď priemerná dĺžka riešenia, najlepšie nájdené riešenie v danom podstrome alebo iná veličina. Následne zoradíme uzly od najlepšej hodnoty po najhoršiu zostupne. Prvému uzlu priradíme pomocnú hodnotu, napr. 100. Ďalšiemu uzlu priradíme rovnakú hodnotu, ak je kvalita daného uzla rovnaká ako uzla predchádzajúceho. V opačnom prípade danému uzlu priradíme K-krát menšiu hodnotu, kde K je nastaviteľný koeficient. Takto postupne spracujeme všetky uzly. Z pomocných hodnôt vypočítame pravdepodobnosti tak, aby pomery pravdepodobností medzi potomkami boli rovnaké ako pomery pomocných hodnôt, a aby súčet pravdepodobností pre všetkých potomkov bol rovný 1. Príklad pre 3 potomkov, kde hodnota kvality predstavuje napr. najlepšie dosiahnuté riešenie v príslušnom podstrome, je v Tab. 2, pričom koeficient K má hodnotu 2.

Tab. 2. Príklad navrhovanej selekčnej stratégie (koeficient K=2).

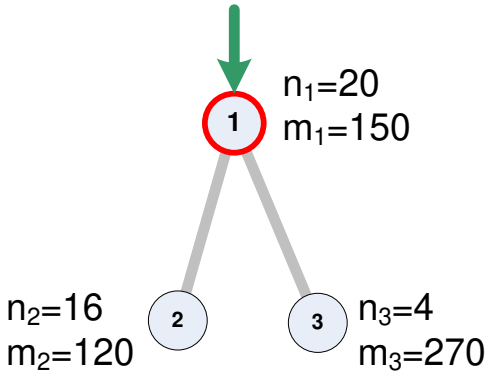
Potomok	Kvalita (najlepšie riešenie)	Pomocná hodnota	Pravdepodobnosť výberu
1	200	25	0,14
2	100	100	0,57
3	150	50	0,29

Takýto jednoduchý systém spĺňa požadované vlastnosti. Aj uzol, ktorý ešte nebol navštívený (a teda mu nevieme priradiť hodnotu kvality), môže byť vybratý, keď ho zaradíme na koniec zoznamu. Čím väčší zvolíme koeficient K, tým viac sa budú preferovať potomkovia s lepšou hodnotou kvality. V prípade, že $K = 1$, ide o náhodný výber.

Nastavenie opísanej selekčnej stratégie zahŕňa zvolenie:

- Veličiny kvality – priemerná dĺžka riešenia dosiahnutá v danom podstrome alebo najlepšie riešenie dosiahnuté v danom podstrome.
- Koeficientu K pre selekčnú stratégiu.
- Hranice pre počet návštev uzla, od ktorej sa začne selekčná stratégia používať (táto hranica je spomínaná v kapitole 9). Ďalej sa bude nazývať limit pre aktiváciu selekčnej stratégie.

Situácia pre rozhodnutie selekčnej stratégie môže vyzerať ako na Obr. 31. Nachádzame sa v uzle č. 1 stromu prehľadávania, ktorý má dvoch potomkov. Tento uzol bol navštívený 20 krát (hodnota n_1) a priemerné riešenie v podstrome s koreňom v tomto uzle dosiahlo hodnotu 150 (hodnota m_1). Rozhodnutie pozostáva z toho, či si vybrať uzol č. 2 alebo 3. Vidíme, že uzol č. 2 bol navštívený viackrát, pretože v jeho podstrome sa dosiahlo lepšie priemerné riešenie. Podľa nastavenia koeficientu K sa vypočítajú pravdepodobnosti výberu pre oba uzly, a na základe toho sa vyberie nasledujúci uzol. Ak by sa vybral napr. uzol č.3, a limit pre aktiváciu selekčnej stratégie je 10, tak selekčná stratégia sa už v tomto uzle používať nebude, pretože vtedy bude mať tento uzol iba 4 návštevy. V uzle č. 2 by sa však selekčná stratégia použila znova.



Obr. 31. Situácia pre rozhodovanie selekčnej stratégie.

10.2 Simulačná stratégia

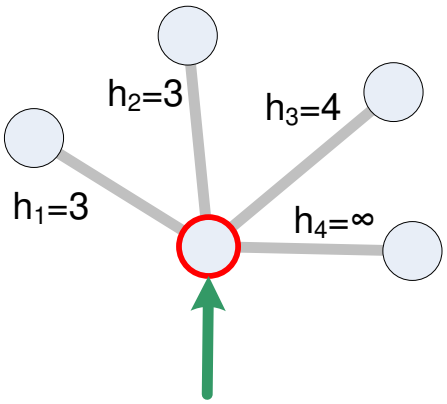
Základnou simulačnou stratégiou je náhodný výber, čo je prístup, ktorý bol už použitý pri pokusoch v kapitole 6.2. Ďalej vo fáze simulácie môžeme použiť heuristiky opísané v kapitole 8.

Pri oboch heuristikách chceme na jednej strane využiť smerovanie k lepšiemu riešeniu, ktoré nám heuristika ponúka, na druhej strane chceme mať možnosť preskúmať aj iné možnosti. Dá sa na to použiť systém výberu opísaný v kapitole 10.1 o selekčnej stratégii, musíme však určiť akú hodnotu kvality priradíme jednotlivým uzlom pri konkrétnej heuristike (pri simulačnej stratégii riešime prehľadávanie grafu, teda hovoriť sa bude o výbere uzlov, nie potomkov ako to bolo pri stratégii selekcie).

Pre heuristiku krátkeho počiatočného cyklu je mierou kvality vzdialenosť k počiatočnému cyklu dekompozície. Uzly sa teda zoradia podľa tejto vzdialenosti

vzostupne. Uzol, z ktorého sa nedá dostať k počiatočnému uzlu dekompozície, dostane priradené veľké číslo tak, aby sa dostal na koniec zoradeného zoznamu uzlov. Následne sa vyberá spôsobom opísaným v kapitole 10.1.

Situácia, pri ktorej sa používa táto heuristika, môže vyzerať ako na Obr. 32. V aktuálnom uzle grafu sa potrebujeme rozhodnúť, ktorým uzlom bude prehľadávanie pokračovať, pričom každá možnosť má priradenú hodnotu kvality h_i, ktorá reprezentuje vzdialenosť k počiatočnému uzlu dekompozície (nekonečno reprezentuje uzol, cez ktorý nie je možné vytvoriť počiatočný cyklus). Na základe týchto hodnôt sa potom vyberá nasledujúci uzol.



Obr. 32. Situácia pre použitie heuristiky krátkeho počiatočného cyklu.

Heuristika minimálneho počtu kolízií určuje mieru kvality výrazom, definovaným v kapitole 8.2, ktorý berie do úvahy počet kolízií a dĺžku vytváraného ucha. Uzly sa zotriedia podľa hodnoty daného výrazu vzostupne. Následne sa spomedzi nich znovu vyberá spôsobom opísaným v kapitole 10.1.

Pri každej z týchto heuristík je teda možné koeficientom K určiť, ako veľmi budú preferované lepšie hodnoty. Čím väčšia hodnota koeficientu K, tým viac sú lepšie hodnoty preferované. Pri hodnote $K = 1$ ide o náhodné prehľadávanie.

11. Výsledky

V tejto kapitole sú uvedené jednotlivé experimenty s hľadaním lepšej dekompozície prostredníctvom MCPS. Výsledky jednotlivých experimentov budú porovnávané medzi sebou, s náhodným prehľadávaním a taktiež s riešením získaným pri originálnej dekompozícii. Pod originálnou dekompozíciou sa rozumie dekompozícia určená konštrukciou grafu (detailnejší opis je v kapitole 6).

Experimenty sú rozdelené na dve fázy. V prvej fáze sú experimenty, ktorých cieľom je nájsť vhodné nastavenie parametrov pre MCPS, ako sú:

- Použitá heuristika.
- Koeficient pre použitú heuristiku.
- Koeficient pre selekčnú stratégiu.
- Hranica aktivácie selekčnej stratégie.

V druhej fáze sú experimenty, ktorých cieľom je otestovať MCPS s pevne stanovenými parametrami, určenými v prvej fáze, na väčšom počte rôznych grafov. Grafy sa môžu líšiť nasledujúcimi parametrami:

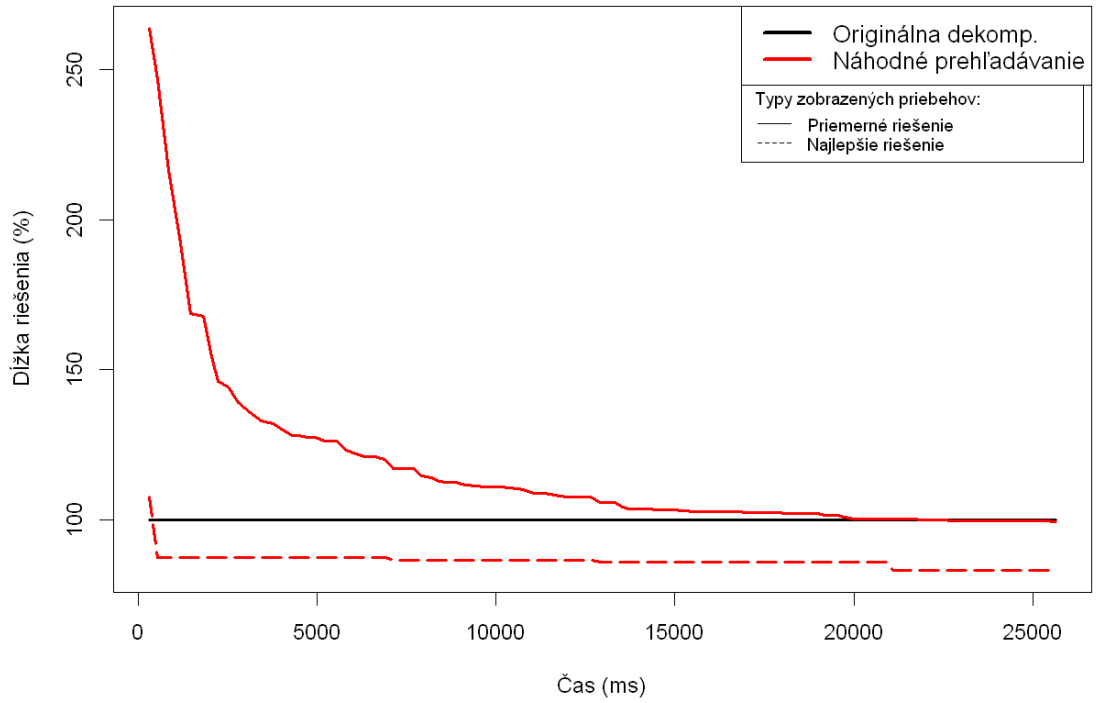
- Počet úch.
- Veľkosť počiatočného cyklu.
- Maximálna dĺžka ucha.
- Počet voľných uzlov.
- Inicializáciou generátora náhodných čísel (*seed*), ktorý jednoznačne určuje graf a rozmiestnenie robotov v ňom.

Pri experimente je vždy istá miera náhody. Pre získanie štatisticky významných údajov bude MCPS vždy prebiehať 30 behov s rôznou inicializáciou generátora náhodných čísel (*seed*). Ide o odlišný parameter ako *seed*, ktorý je uvedený medzi parametrami grafu. Konkrétna hodnota, na ktorú bol inicializovaný generátor náhodných čísel, v tomto prípade určuje jednoznačne priebeh MCPS. Všetky pokusy boli vykonávané na počítači s operačným systémom Windows 7 64-bit, procesorom Intel Core i7 860, 4 GB RAM, použitý kompilátor bol GCC verzie 4.5.2 (z projektu MinGW).

11.1 Grafy s priebehmi prehľadávania

Na znázornenie priebehu prehľadávania budú použité grafy, ktorých opis bude predmetom tejto kapitoly. Na začiatok je však potrebné zdôrazniť, že špeciálne v tejto kapitole sa bude pod pojmom graf rozumieť grafické znázornenie priebehu určitých veličín. V ostatných kapitolách sa pod pojmom graf rozumie matematická štruktúra s uzlami a hranami.

Ako príklad bude použité náhodné prehľadávanie. Ide vlastne o MCPS bez selekčnej stratégie s tým, že simulačná stratégia nepoužíva žiadnu heuristiku. Výsledok takéhoto prehľadávania vidieť na Obr. 33.



Obr. 33. Priebeh náhodného prehľadávania.

Na osi X je zaznačený čas v milisekundách od spustenia behu. Na osi Y je uvedená dĺžka riešenia v percentách, pričom platí, že 100% zodpovedá dĺžke riešenia pri originálnej dekompozícii. V každom grafe sú zaznačené minimálne 3 priebehy:

- Dĺžka riešenia pri originálnej dekompozícii (vždy čiernou farbou) zodpovedá hodnote 100%.
- Priemerné najlepšie riešenie dosiahnuté prehľadávaním v konkrétnom čase. Hodnoty priebehu vznikajú tak, že v každom behu sa určí dosiaľ najlepšie nájdené riešenie, a následne sa tieto hodnoty spriemerujú. Tento priebeh je vždy zaznačený súvislou čiarou.

- Najlepšie dosiahnuté riešenie v danom čase. Pre každý beh sa určí najlepšie dosiahnuté riešenie, hodnota v grafe potom bude najmenšou z týchto hodnôt. Tento priebeh je vždy zaznačený prerušovanou čiarou.

Pre každý ďalší experiment sa teda do grafu pridávajú ďalšie dva priebehy – pre priemerné najlepšie riešenie a pre celkové najlepšie riešenie. Jednotlivé priebehy, predstavujúce rôzne nastavenia prehľadávania, nemusia končiť v rovnakom čase, aj keď dané prehľadávania pozostávajú z rovnakého počtu iterácií. Pri niektorých nastaveniach prehľadávania totiž môže trvať vykonanie stanoveného počtu iterácií dlhšie ako pri iných.

11.2 Vyhodnocovanie

Pri vyhodnocovaní výsledkov porovnávame viaceré priebehy medzi sebou (kde každý priebeh predstavuje jedno nastavenie parametrov prehľadávania), alebo porovnávame niektorý priebeh s riešením nájdeným pri originálnej dekompozícii.

Ak však máme viac priebehov prehľadávania, ktoré môžu trvať rôzne dlhý čas, je potrebné sa zjednotiť na čase, pre ktorý sa budú ich hodnoty uvádzať a porovnávať. Pre tieto účely bol zvolený čas, v ktorom skončilo najkratšie prehľadávanie. Napr. ak prehľadávanie P_1 trvalo 60 sekúnd a prehľadávanie P_2 trvalo 30 sekúnd (obidve prehľadávania mali zhodne po 100 iterácií), tak porovnávať sa budú hodnoty v čase $t = 30s$. Tento čas sa bude ďalej označovať ako čas porovnávania.

11.2.1 Spracovanie dlhých riešení

Algoritmus BIBOX môžeme rozdeliť na 2 fázy, najprv sa nájde riešenie, ktoré nevyužíva možnosti paralelného pohybu, nazvime ho neparalelizované riešenie. V druhej fáze dochádza k jeho paralelizácii, podľa algoritmu opísanom v kapitole 4.3.1. Podľa použitého zdrojového kódu algoritmu BIBOX má výpočet paralelizácie kvadratickú zložitosť vo vzťahu k počtu krokov neparalelizovaného riešenia. Paralelizovanie veľmi dlhých riešení tak potom značne predlžuje čas prehľadávania. Preto bol zavedený limit pre počet krokov neparalelizovaného riešenia. Od tohto limitu vyššie sa riešenia neparalelizujú. Hodnota limitu bola nastavená na 10000 krokov. Nestratíme tým žiadne dobré riešenia, pretože ak neparalelizované riešenia majú takýto vysoký počet krokov, paralelizované riešenie nebude patriť medzi najlepšie. Môžeme to tvrdiť na základe toho, že na dosiahnutie malého počtu krokov, by bola potrebná vysoká

hodnota paralelizmu, ktorá však podľa analýzy výsledkov v kapitole 6.3 pozitívne koreluje s dĺžkou riešenia.

Iterácie prehľadávania, v ktorých riešenia nebudú paralelizované, však skončia skôr, keďže nebude treba vykonávať časovo náročnú paralelizáciu. Meranie času potrebného na dokončenie prehľadávania je teda vhodné korigovať. Korekcia bude vykonaná tak, že pre iterácie, v ktorých sa riešenie neparalelizovalo, sa za čas vykonania určí priemerný čas, ktorý trvalo vykonanie iterácie v danom behu. Do výpočtu sa však budú uvažovať iba tie iterácie, v ktorých sa riešenie paralelizovalo. Tým sa priblíži čas vykonania celého behu k hodnotám, ktoré by sme dosiahli, ak by sa riešenia vo všetkých iteráciách paralelizovali.

11.2.2 Testovanie štatistickej významnosti

Na zistenie toho, či rozdiel medzi priemerom z najlepších nájdených riešení a riešením nájdeným pri originálnej dekompozícii je štatisticky významný, bude použitý t-test. Predpokladá sa teda normálne rozloženie týchto hodnôt v rámci behov. Nulovou hypotézou (H_0) bude hypotéza, že skúmaný priemer (m) je väčší alebo rovný ako riešenie pri originálnej dekompozícii (m_{orig}), alternatívnou hypotézou (H_1) bude opak:

$$H_0: m \geq m_{orig} \qquad H_1: m < m_{orig}$$

Vo výsledkoch bude uvádzaná tzv. p-hodnota, ktorá predstavuje najmenšiu takú hladinu významnosti, pri ktorej by dané údaje viedli k zamietnutiu nulovej hypotézy. Udáva pravdepodobnosť, s akou je možné pozorovať dané údaje, keď H_0 je pravdivá. Čím menšia p-hodnota, tým istejšie môžeme tvrdiť, že nulová hypotéza nie je pravdivá [25].

Všetky testy budú vyhodnocované na hladine významnosti $\alpha = 5\%$, ktorá reprezentuje pravdepodobnosť chyby prvého druhu, t.j. že hypotézu H_0 zamietneme, aj keď je pravdivá. Rozdiel medzi priemermi teda bude považovaný za štatisticky signifikantný vtedy, ak p-hodnota bude menšia ako zvolená hladina významnosti $\alpha = 5\%$, teda nulovú hypotézu zamietneme.

Na porovnanie hodnôt rôznych priebehov medzi sebou bude znovu použitý t-test. Nulovou hypotézou (H_0) bude hypotéza, že číselne menší priemer (m_1) je väčší alebo rovný ako druhý priemer (m_2), a alternatívnou hypotézou (H_1) bude opak:

$$H_0: m_1 \geq m_2 \qquad H_1: m_1 < m_2$$

Rozdiel medzi hodnotami priemerov bude štatisticky významný, ak p-hodnota bude menšia ako hladina významnosti $\alpha = 5\%$. Nulovú hypotézu vtedy zamietneme.

11.2.3 Opis štruktúry výsledkov

Príkladom tabuľky s výsledkami je Tab. 3. V danej tabuľke je uvedený čas porovnávania, ku ktorému sú uvádzané príslušné hodnoty priemerného riešenia a minimálneho riešenia. Popis jednotlivých stĺpcov je nasledovný:

1. Nastavenie – predstavuje nastavenie parametrov prehľadávania.
2. Celkový čas – priemerný čas, za ktorý prehľadávanie s daným nastavením dokončilo stanovený počet iterácií MCPS.
3. Priemer (%) – priemerné riešenie v čase porovnávania ako percento z dĺžky riešenia pri originálnej dekompozícii. Na základe tejto hodnoty budeme porovnávať jednotlivé nastavenia medzi sebou.
4. Minimum (%) – najkratšie riešenie v čase porovnávania ako percento z dĺžky riešenia pri originálnej dekompozícii. Na základe tejto hodnoty sa nebudeme rozhodovať medzi nastaveniami, pretože táto hodnota dosť závisí od náhody.
5. p_{orig} – p-hodnota pre test, či rozdiel medzi priemerným riešením a riešením pri originálnej dekompozícii je štatisticky významný. Pri kladnej odpovedi ($p_{orig} < 0{,}05$) bude daná bunka tabuľky farebne zvýraznená. Pri niektorých experimentoch bude namiesto porovnania s riešením nájdeným pri originálnej dekompozícii použité porovnanie so zvoleným nastavením prehľadávania.
6. p_{nasl} – p-hodnota pre test, či rozdiel medzi priemerným riešením pri danom nastavení a nasledujúcom nastavení je štatisticky významný. Pri kladnej odpovedi ($p_{nasl} < 0{,}05$) bude daná bunka tabuľky farebne zvýraznená. Posledný záznam v tomto stĺpci vyplnený nie je, pretože nasledujúce nastavenie sa už v tabuľke nenachádza.

Aby bolo možné vypočítať poslednú hodnotu p_{nasl}, sú záznamy v tabuľke zoradené podľa priemerného riešenia zostupne, teda najlepší výsledok je na konci. Výsledky sú zaokrúhlené na 2 desatinné miesta, pri p-hodnotách na 3 desatinné miesta.

Tab. 3. Príklad tabuľky s výsledkami experimentov.

Čas porovnávania (s):					24,18
Nastavenie	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{orig}	p_{nasl}
K = 1,2	31,46	103,54	74,13	0,921	0,030
K = 1,5	28,13	97,83	77,62	0,133	0,064
K = 2	24,18	93,75	71,33	0,001	-

11.3 Porovnanie nastavení prehľadávania

V tejto časti sú uvedené experimenty s rôznym nastavením prehľadávania. Najprv sa porovnávali iba heuristiky medzi sebou, bez použitia selekčnej stratégie. Následne sa s najlepšou nájdenou heuristikou vyskúšajú rôzne nastavenia selekčnej stratégie.

Je potrebné zdôrazniť, že cieľom experimentov v tejto časti nie je nájsť optimálne nastavenie. Priebeh prehľadávania totiž závisí od grafu, na ktorom sa prehľadávanie vykonáva. Optimálne nastavenie by tak mohlo byť pre každý graf a každé rozmiestnenie robotov iné. Cieľom je skôr na niekoľkých grafoch s rôznymi parametrami zistiť, ako sa správajú rôzne nastavenia heuristík a selekčných stratégií. Výsledkom by malo byť určenie nastavení, s ktorými prehľadávanie dosahovalo v priemere najlepšie výsledky.

Takéto skúšanie rôznych nastavení je aj výpočtovo náročné, keďže pre každý graf sa skúša 40 rôznych nastavení, kde pre každé nastavenie sa 30-krát spustí MCPS po 100 iterácií. Ak teda máme 6 grafov, celkovo to vychádza na 720000 spustení algoritmu BIBOX. Takto sme obmedzení pri veľkosti a množstve grafov, pre ktoré chceme podrobne preskúmať rôzne nastavenia.

Pre tieto experimenty bolo zvolených 6 grafov, označených G1 až G6, s parametrami uvedenými v Tab. 4. Farebne zvýraznené sú tie nastavenia, ktoré sa pri grafoch G2 až G6 líšia od nastavení grafu G1. MCPS bolo spustené v 30 behoch po 100 iterácií, zároveň bol nastavený limit na paralelizáciu riešení (kapitola 11.2.1). V tejto fáze experimentov sa však nebudeme detailne zaoberať rozdielmi vo výsledkoch pri rôznych typoch grafov, toto zisťovanie bude predmetom experimentov v ďalšej časti.

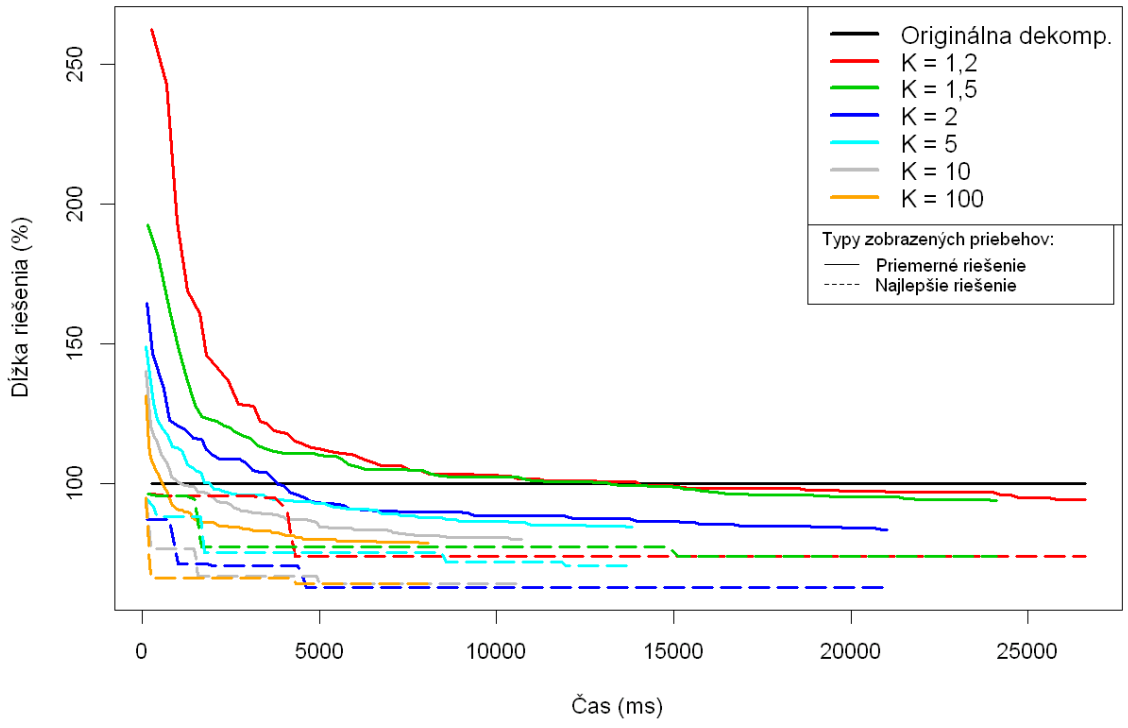
Tab. 4. Parametre grafov pre pokusy s heuristikami a parametrami MCPS.

Parameter	G1	G2	G3	G4	G5	G6
Počet uzlov	33	33	21	43	25	46
Počet úch	10	10	5	15	10	10
Maximálna dĺžka ucha	5	5	5	5	3	7
Dĺžka počiatočného cyklu	5	5	5	5	5	5
Počet voľných uzlov	2	5	2	2	2	2
Dĺžka riešenia pri originálnej dekompozícii	143	112	76	170	95	218

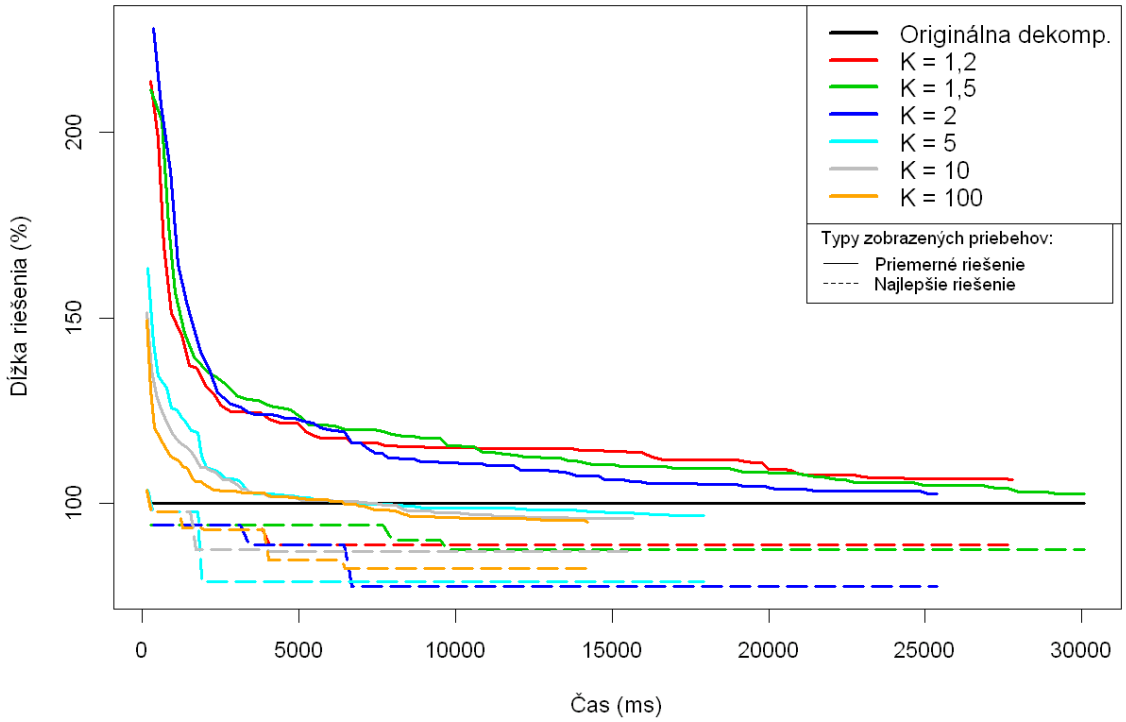
11.3.1 Heuristika krátkeho počiatočného cyklu

V tomto prípade sa používala iba heuristika krátkeho počiatočného cyklu a nastavoval sa jej koeficient K. Čím väčšia hodnota koeficientu, tým viac sa preferovali najkratšie počiatočné cykly. Pre koeficient sa použili hodnoty z množiny $1{,}2; 1{,}5; 2; 5; 10; 100$.

Pri grafe G1, ktorého priebehy prehľadávania sú zaznačené na Obr. 34, vidieť, že so zväčšujúcim sa koeficientom, má krivka prehľadávania strmší priebeh, ako aj lepšie priemerné riešenie nájdené v prvej iterácii (začiatok daného priebehu je nižšie).



Obr. 34. Graf G1 - Rôzne koeficienty heuristiky krátkeho počiatočného cyklu.



Obr. 35. Graf G4 - Rôzne koeficienty heuristiky krátkeho počiatočného cyklu.

Pri grafe G4, zaznačenom na Obr. 35, vidieť odlišné odstupňovanie jednotlivých priebehov. Pri malej hodnote koeficientu ($K \in 1{,}2; 1{,}5; 2$) majú prehľadávania podobný priebeh, pri väčších hodnotách sú to znova porovnateľné priebehy. Pri ostatných grafoch sú priebehy podobné tým opísaným. Vo všetkých grafoch však platí, že najlepšie číselné výsledky dosiahol koeficient $K = 100$, aj keď pri 5 grafoch rozdiel od výsledkov pri $K = 10$ nie je štatisticky významný ($p_{nasl} > 0{,}05$). Taktiež pri nastavení $K = 100$ prebehlo 100 iterácií v 5 zo 6 grafov najrýchlejšie (v porovnaní s ostatnými nastaveniami). Výsledky pre graf G1, resp. G4 sú zaznačené v Tab. 5, resp. v Tab. 6. Zvyšné výsledky sú uvedené v prílohe B.

Tab. 5. Graf G1 - Rôzne koeficienty heuristiky krátkeho počiatočného cyklu.

Čas porovnávania (s):					8,07
	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{orig}	p_{nasl}
K = 1,5	24,11	104,99	77,62	0,983	0,486
K = 1,2	26,6	104,87	74,13	0,969	0
K = 2	21,03	90,02	62,94	0	0,239
K = 5	13,83	87,97	75,52	0	0
K = 10	10,71	81,35	64,34	0	0,08
K = 100	8,07	78,86	64,34	0	-

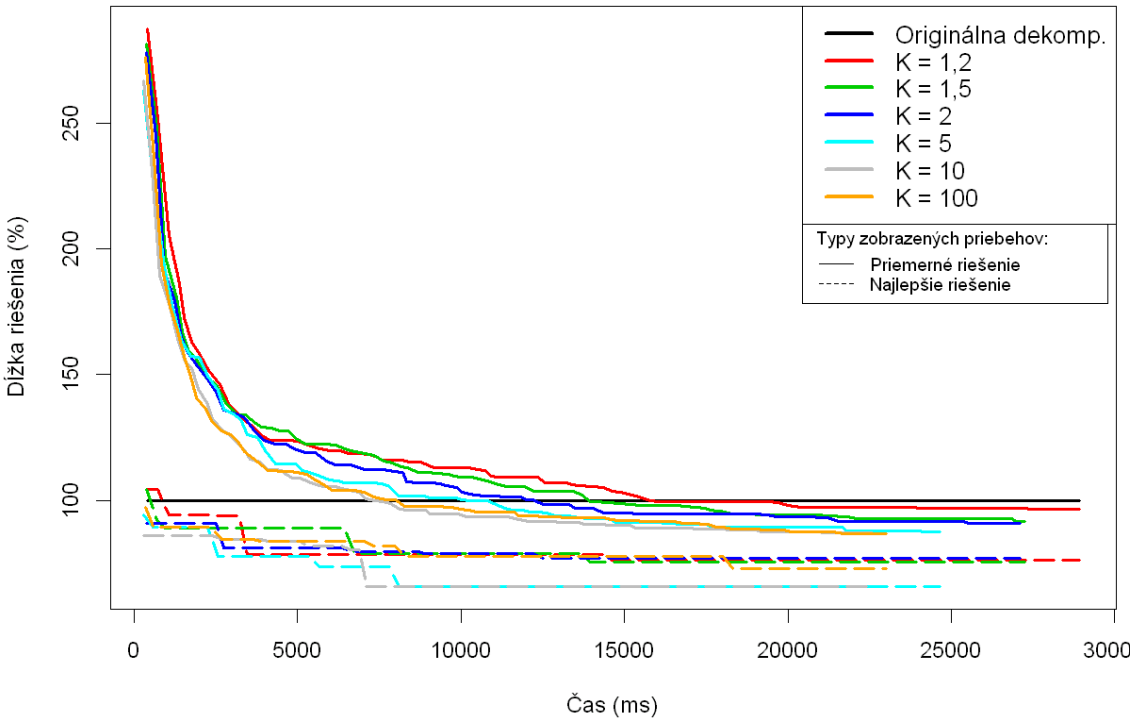
Tab. 6. Graf G4 - Rôzne koeficienty heuristiky krátkeho počiatočného cyklu.

Čas porovnávania (s):					14,24
	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{orig}	p_{nasl}
K = 1,2	27,79	114,35	88,82	1	0,15
K = 1,5	30,08	111,06	87,65	1	0,137
K = 2	25,37	107,29	77,65	0,995	0,001
K = 5	17,93	98	78,82	0,046	0,113
K = 10	15,66	96,02	87,06	0,001	0,272
K = 100	14,24	95,06	82,35	0	-

11.3.2 Heuristika minimálneho počtu kolízií

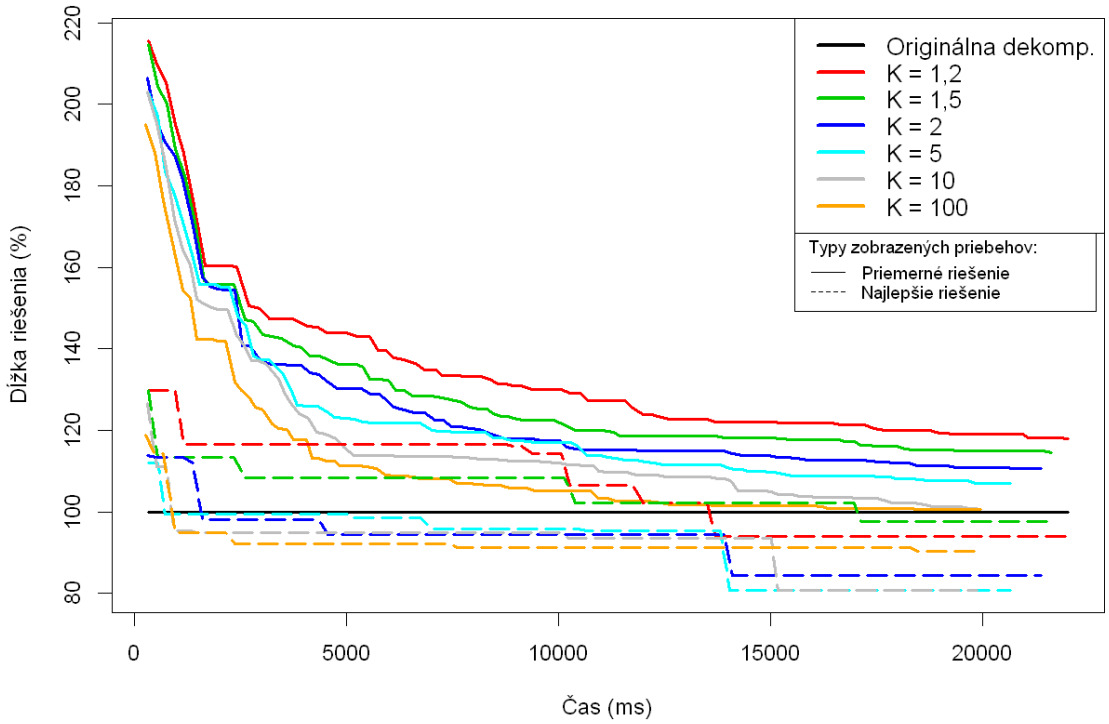
V tomto prípade sa používala iba heuristika minimálneho počtu kolízií a nastavovala sa hodnota koeficientu K. Čím vyššia hodnota koeficientu, tým viac sa preferoval pri prehľadávaní ten segment, ktorý prinesie do práve vytváraného ucha najmenej kolízií. Pre koeficient sa použili hodnoty z množiny $1{,}2; 1{,}5; 2; 5; 10; 100$.

Pri týchto pokusoch sú rozdiely medzi nastaveniami menšie, väčšina priebehov vyzerá tak ako na Obr. 36 (priebehy pre graf G1). Vidieť tam znovu odstupňovanie medzi priebehmi so zvyšujúcim sa koeficientom, ale nie je to už také výrazné, ako pri heuristike krátkeho počiatočného cyklu.



Obr. 36. Graf G1 - Rôzne koeficienty heuristiky minimálneho počtu kolízií.

Rozdielne vyzerajú priebehy pri grafe G6, ktoré sú znázornené na Obr. 37. Vidieť, že rozdiely medzi priebehmi pre rôzne koeficienty sú výraznejšie. Graf G6 sa vyznačuje tým, že má dlhšie uchá. Čím dlhšie uchá, tým viac možností existuje pre kolízie medzi robotmi. A keďže existuje viac možností pre kolízie, heuristika snažiaca sa o ich minimalizáciu, môže byť tak účinnejšia ako pri kratších uchách.



Obr. 37. Graf G6 - Rôzne koeficienty heuristiky minimálneho počtu kolízií.

Priebehy pre ostatné grafy sú podobné tým pre graf G1. Číselne najlepšie hodnoty dosiahlo v 5 zo 6 grafov nastavenie s koeficientom $K = 100$. V porovnaní s koeficientom $K = 10$ však hodnoty nie sú signifikantne lepšie ($p_{nasl} > 0{,}05$). Číselné výsledky pre graf G1 sú zaznačené v Tab. 7, pre graf G6 sú zaznačené v Tab. 8. Ostatné výsledky sú uvedené v prílohe B.

Tab. 7. Graf G1 - Rôzne koeficienty heuristiky minimálneho počtu kolízií.

Čas porovnávania (s):					22,59
	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{orig}	p_{nasl}
K = 1,2	28,91	97,34	76,22	0,116	0,049
K = 1,5	27,25	92,68	75,52	0	0,351
K = 2	27,11	91,77	76,92	0	0,061
K = 5	24,65	88,02	65,73	0	0,374
K = 10	22,59	87,23	65,73	0	0,407
K = 100	23,01	86,69	72,73	0	-

Tab. 8. Graf G6 - Rôzne koeficienty heuristiky minimálneho počtu kolízií.

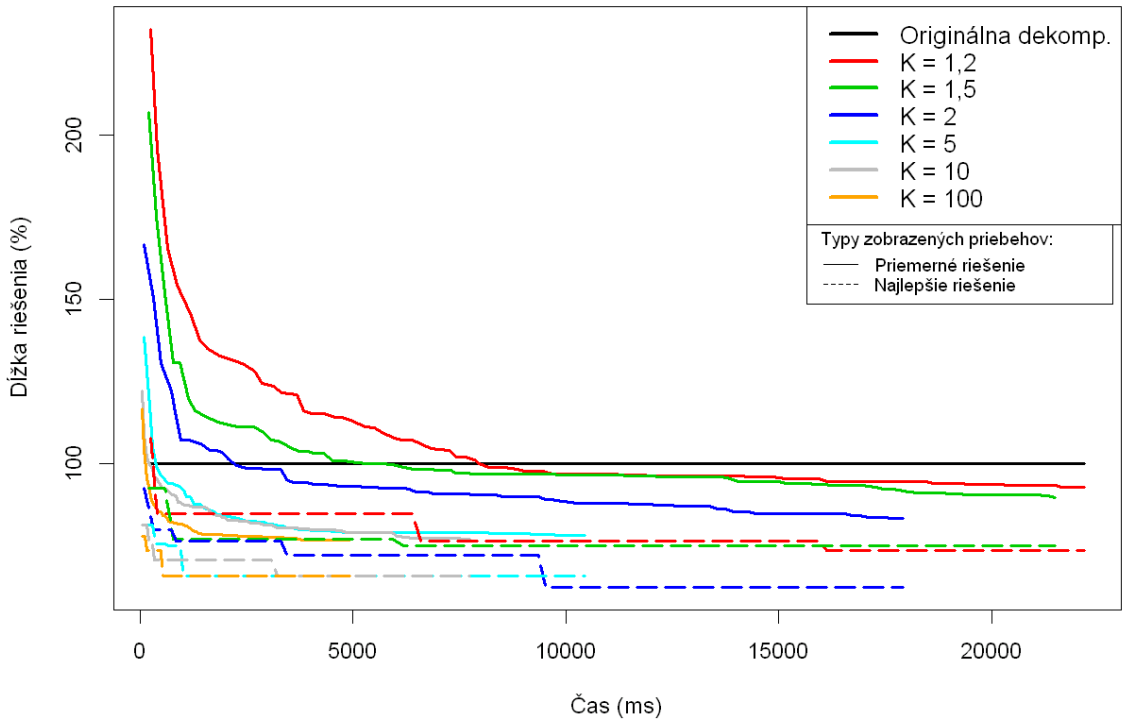
Čas porovnávania (s):					19,87
	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{orig}	p_{nasl}
K = 1,2	22,01	119,13	94,04	1	0,05
K = 1,5	21,63	115,06	97,71	1	0,03
K = 2	21,39	110,93	84,4	1	0,053
K = 5	20,66	106,91	80,73	1	0,006
K = 10	19,87	100,92	80,73	0,716	0,437
K = 100	19,97	100,6	90,37	0,682	-

11.3.3 Kombinácia heuristík

V tomto prípade sa použili obe heuristiky a nastavoval sa ich koeficient K, pričom jeho hodnota bola rovnaká pre každú z nich. Pre koeficient sa znovu použili hodnoty z množiny $1{,}2; 1{,}5; 2; 5; 10; 100$.

Priebehy pri kombinácii heuristík vyzerajú ako na Obr. 38, kde sú zachytené priebehy pre graf G1. Vidieť, že so zväčšujúcim koeficientom prehľadávanie trvá kratšie a dosahuje priemerne nižšie hodnoty. Číselné výsledky pre graf G1 sú zaznačené v Tab. 9. Výsledky pre ostatné grafy sú uvedené v prílohe B.

Pre všetky grafy platí, že prehľadávanie s koeficientom $K = 100$ dosiahlo číselne najnižšie hodnoty. Pri polovici grafov sú výsledky signifikantne lepšie ($p_{nasl} < 0{,}05$) ako pri $K = 10$. Pri všetkých grafoch tiež platí, že prehľadávanie pri koeficiente $K = 100$ trvalo najkratšie.



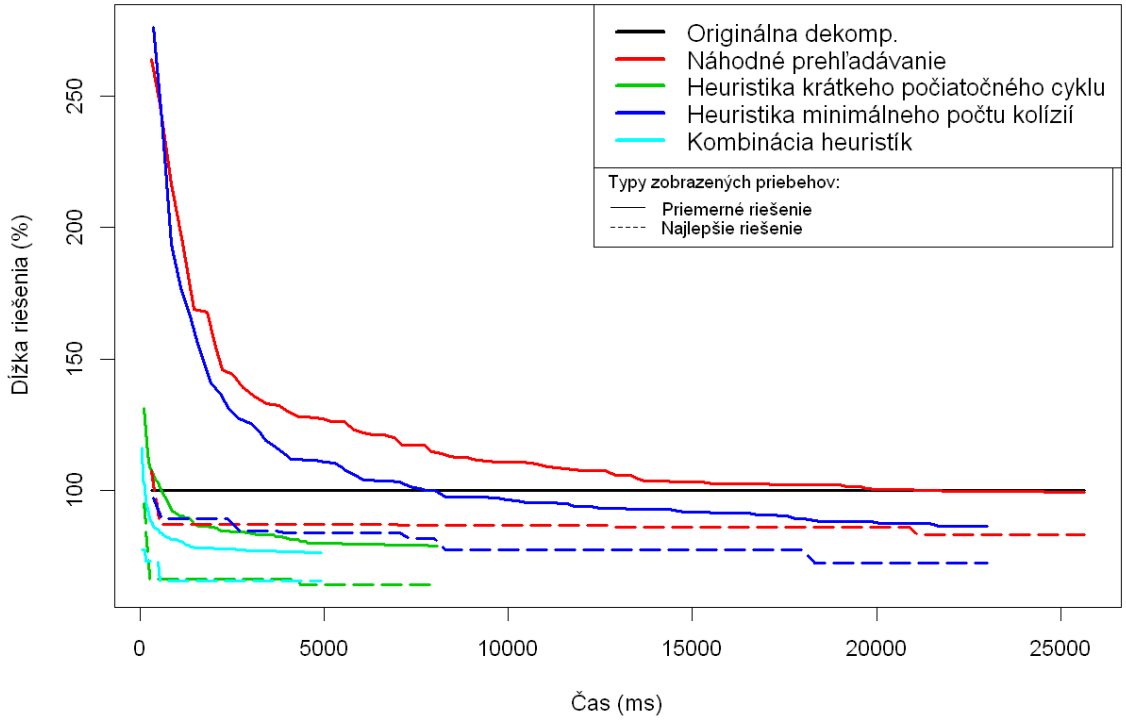
Obr. 38. Graf G1 - Rôzne koeficienty pre kombináciu heuristík.

Tab. 9. Graf G1 - Rôzne koeficienty pre kombináciu heuristík.

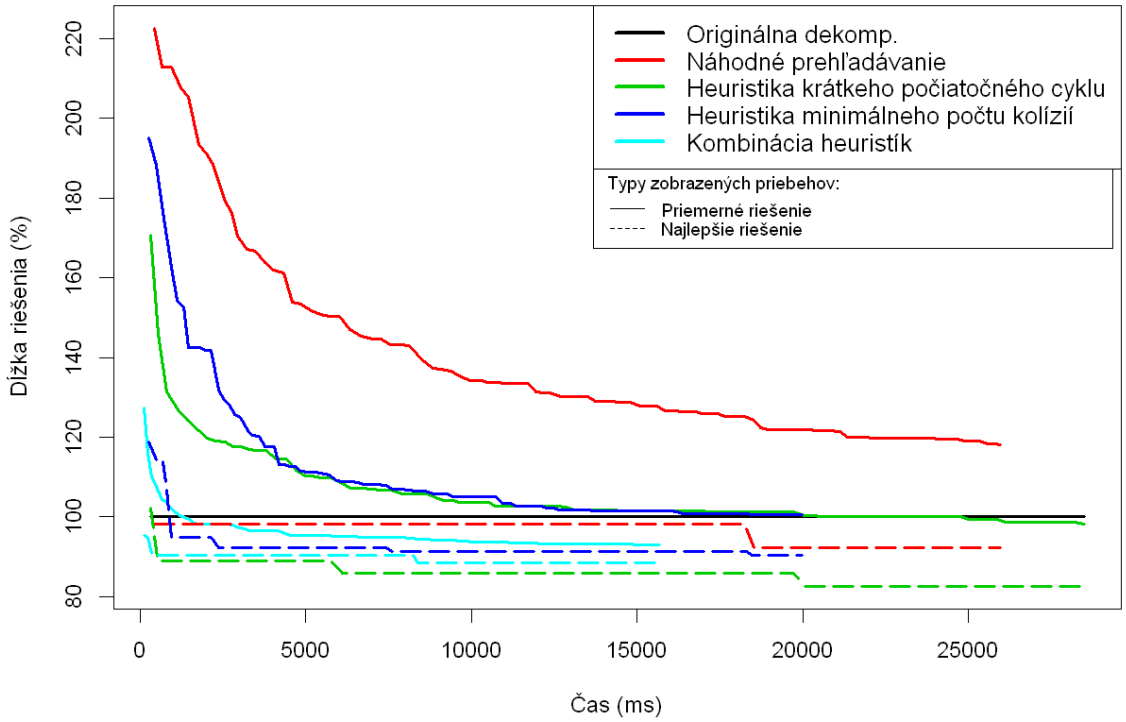
Čas porovnávania (s):					4,92
	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{orig}	p_{nasl}
K = 1,2	22,16	113,96	84,62	1	0,001
K = 1,5	21,48	100,75	76,92	0,652	0,003
K = 2	17,91	93,05	72,03	0,001	0
K = 5	10,44	79,02	65,73	0	0,494
K = 10	7,73	79	65,73	0	0,04
K = 100	4,92	76,64	65,73	0	-

11.3.4 Vyhodnotenie heuristík

V tejto časti budú porovnané výsledky samostatných heuristík, kombinácie heuristík ako aj náhodného prehľadávania. Pre každú heuristiku bude použitý koeficient $K = 100$, ktorý dosiahol číselne najlepšie výsledky. Na ilustráciu sú priebehy pre graf G1, resp. G6, zaznačené na Obr. 39, resp. Obr. 40. Číselné výsledky sú zaznačené v Tab. 10, resp. Tab. 11.



Obr. 39. Graf G1 - porovnanie heuristík.



Obr. 40. Graf G6 - porovnanie heuristík.

Pri všetkých grafoch dosiahla kombinácia heuristík najlepšie výsledky, čo sa týka priemerného riešenia, rozdiel oproti ostatným heuristikám je štatisticky významný ($p_{nasl} < 0{,}05$). Taktiež kombinácia heuristík potrebovala na vykonanie 100 iterácií výrazne menší čas oproti ostatným heuristikám.

Tab. 10. Graf G1 - porovnanie heuristík.

Čas porovnávania (s):					4,92
	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{orig}	p_{nasl}
Bez heuristiky	25,64	127,65	87,41	1	0,005
Heuristika min. počtu kolízií	23,01	111,52	83,92	0,998	0
Heuristika krátkeho počiatočného cyklu	8,07	80,12	64,34	0	0,013
Kombinácia heuristík	4,92	76,64	65,73	0	-

Tab. 11. Graf G6 - porovnanie heuristík.

Čas porovnávania (s):					15,69
	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{orig}	p_{nasl}
Bez heuristiky	25,94	127,77	98,17	1	0
Heuristika min. počtu kolízií	19,97	101,51	91,28	0,865	0,484
Heuristika krátkeho počiatočného cyklu	28,48	101,44	85,78	0,849	0
Kombinácia heuristík	15,69	92,97	88,53	0	-

Pre ďalšie použitie budú teda používané obe heuristiky spoločne, s koeficientom $K = 100$. To, že ako koeficient je zvolená hodnota $K = 100$, znamená, že najlepšie je v podstate vždy sa riadiť odporúčaním heuristiky a neskúšať iné riešenia. Neznamená to však, že priebeh prehľadávania je deterministický. Aj pri vysokom koeficiente majú pri prehľadávaní rovnocenné možnosti rovnakú pravdepodobnosť výberu. Napr. ak sa pri prehľadávaní rozhodujeme medzi 3 uzlami a cez každý sa dá dostať do počiatočného uzla dekompozície rovnako dlhou cestou, tak pravdepodobnosť výberu má každý uzol rovnakú. Toto rozhodnutie a tým pádom aj celé prehľadávanie závisí teda stále aj od náhody. Zavedenie selekčnej stratégie by tak mohlo prehľadávanie ešte vylepšiť, čo bude predmetom ďalších kapitol.

11.3.5 Parametre selekčnej stratégie

Po zvolení heuristiky a jej koeficientu budú ďalej uvedené experimenty, ktoré majú zistiť, či zavedenie selekčnej stratégie dokáže ešte zlepšiť výsledky prehľadávania s najlepšími parametrami zistenými v predchádzajúcich kapitolách. Použité sú teda obe heuristiky s koeficientom $K = 100$.

Všetky priebehy sa porovnávajú s priebehom bez použitia selekčnej stratégie. V nasledujúcich pokusoch teda uvádzanie p-hodnoty pre porovnanie s riešením pri originálnej dekompozícii (stĺpec p_{orig}) nemá až taký význam. Namiesto danej hodnoty budeme uvádzať p-hodnotu pre porovnanie s výsledkom prehľadávania bez selekčnej

stratégie, s označením stĺpca p_{bss}. Zlepšenie bude štatisticky významné, ak $p_{bss} < 0{,}05$.

Meniť sa budú nasledovné parametre selekčnej stratégie (bližšie opísané sú v kapitole 10.1):

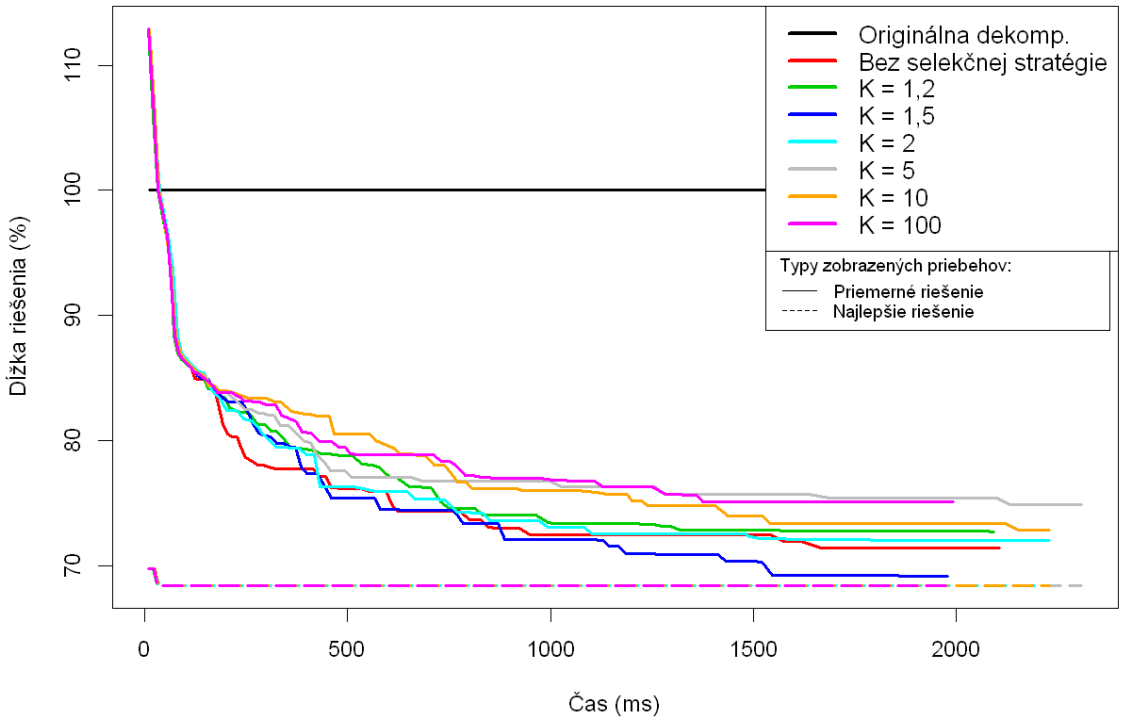
- Koeficient K, ktorý určuje preferenciu lepších riešení.
- Spôsob rozhodovania selekčnej stratégie – môže sa riadiť buď podľa priemerného alebo najlepšieho riešenia nájdeného v danom podstrome.
- Limit pre aktiváciu selekčnej stratégie, čo je minimálny počet návštev uzla stromu prehľadávania, pri ktorom sa začne používať selekčná stratégia. Pri uzloch, ktorý tento limit nedosiahli, sa používa simulačná stratégia.

11.3.6 Koeficient selekčnej stratégie

Najprv experimenty prebiehali s limitom pre aktiváciu selekčnej stratégie nastaveným na 10 (hodnota akú použili v práci [23]). Menila sa hodnota koeficientu pre selekčnú stratégiu. Použité hodnoty boli $1{,}2; 1{,}5; 2; 5; 10; 100$.

Pri vyhodnocovaní na základe priemerného riešenia sa iba pri grafoch G3 a G5 podarilo získať štatisticky významné zlepšenie ($p_{bss} < 0{,}05$) oproti tomu, keď sa selekčná stratégia nepoužíva.

Ako príklad je teda uvedený priebeh pre graf G3 na Obr. 41, kde je možné vidieť, že priebehy postupujú rovnako do okamihu dosiahnutia limitu pre aktiváciu selekčnej stratégie. Od tohto momentu sa priebehy pre rôzne koeficienty rozchádzajú. Výsledky pre tento priebeh sú uvedené v Tab. 12.



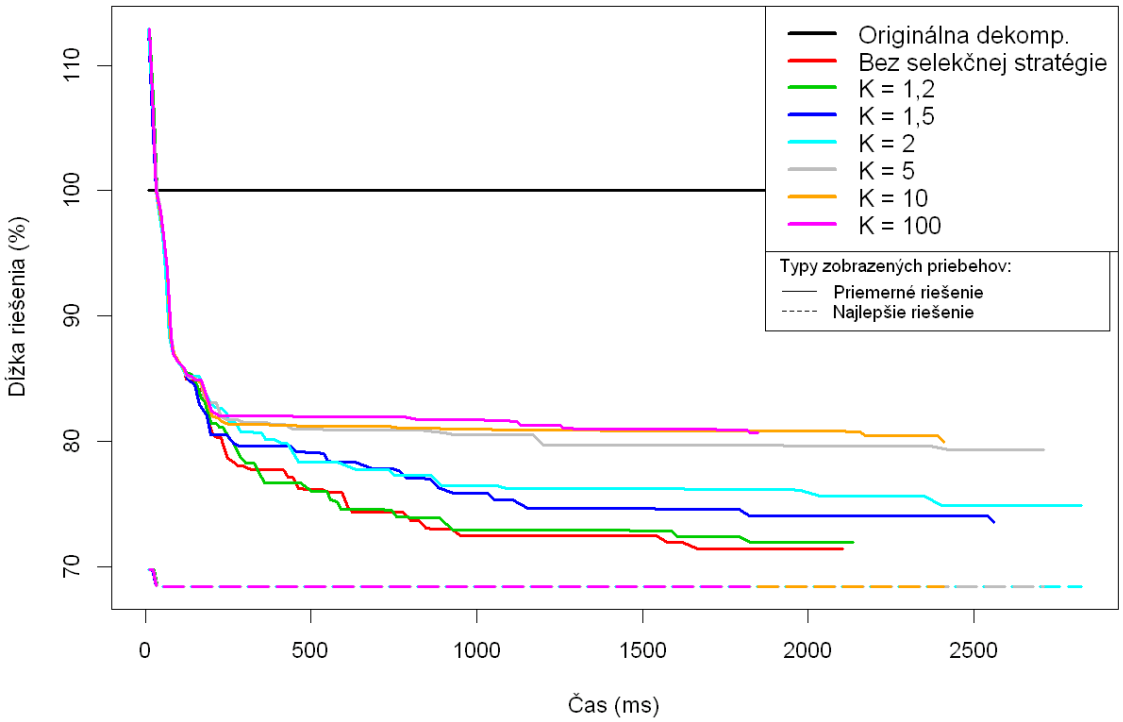
Obr. 41. Graf G3 - Rôzny koeficient selekčnej stratégie, rozhodovanie podľa priemerného riešenia.

Tab. 12. Graf G3 - Rôzny koeficient selekčnej stratégie, rozhodovanie podľa priemerného riešenia.

Čas porovnávania (s):					1,98
	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{bss}	p_{nasl}
K = 5	2,31	75,44	68,42	0,983	0,434
K = 100	1,99	75,09	68,42	0,972	0,2
K = 10	2,23	73,42	68,42	0,877	0,356
K = 1,2	2,09	72,76	68,42	0,783	0,335
K = 2	2,23	72,02	68,42	0,634	0,366
Bez sel. stratégie	2,11	71,45	68,42	0,500	0,035
K = 1,5	1,98	69,17	68,42	0,035	-

Pri vyhodnocovaní na základe najlepšieho riešenia sa iba pri grafoch G2 a G4 podarilo získať štatisticky významné zlepšenie ($p_{bss} < 0{,}05$) oproti tomu, keď sa selekčná stratégia nepoužíva. Priebehy vyzerajú podobne ako pri vyhodnocovaní na základe priemerného riešenia. Jediné zaujímavejšie priebehy sú pri grafe G3, zaznačené sú na Obr. 42. Vidieť, že pri vysokých koeficientoch sa selekčná stratégia zameria na

dosiaľ najlepšie nájdené riešenia, a nedovolí tak v priemere nájsť lepšie riešenie. Tento graf G3 však má málo úch, a pri ostatných grafoch takýto jav nebol pozorovaný, teda nie je možné zovšeobecňovať. Číselné výsledky pre tento graf sú uvedené v Tab. 13. Výsledky pre ostatné grafy sú uvedené v prílohe B.



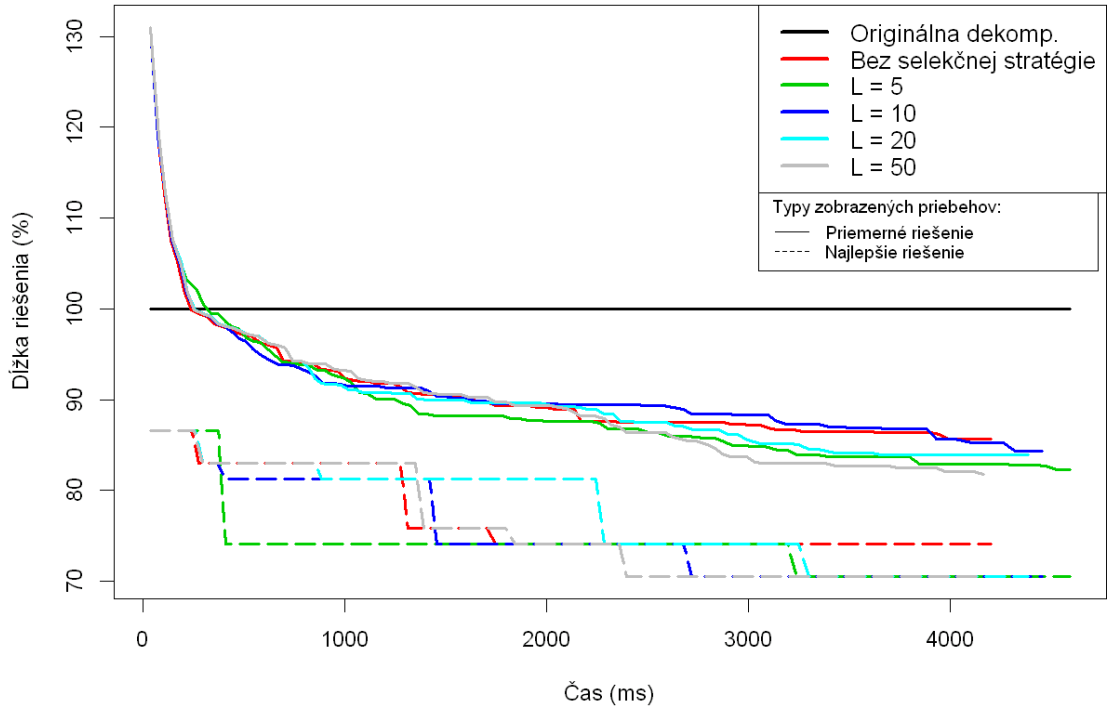
Obr. 42. Graf G3 - Rôzny koeficient selekčnej stratégie, rozhodovanie podľa najlepšieho riešenia.

Tab. 13. Graf G3 - Rôzny koeficient selekčnej stratégie, rozhodovanie podľa najlepšieho riešenia.

Čas porovnávania (s):					1,85
	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{bss}	p_{nasl}
K = 10	2,41	80,83	68,42	1,000	0,476
K = 100	1,85	80,7	68,42	1,000	0,324
K = 5	2,71	79,74	68,42	1,000	0,052
K = 2	2,82	76,18	68,42	0,991	0,159
K = 1,5	2,56	74,08	68,42	0,929	0,117
K = 1,2	2,14	71,93	68,42	0,618	0,382
Bez sel. stratégie	2,11	71,45	68,42	0,500	-

11.3.7 Limit pre aktiváciu selekčnej stratégie

V tomto prípade sa použila selekčná stratégia so zvoleným koeficientom $K = 2$. Menil sa limit pre aktiváciu selekčnej stratégie (označený L), vyskúšané hodnoty boli $\{5; 10; 20; 50\}$. Najprv prebiehali pokusy pre vyhodnocovanie na základe priemerného riešenia. Iba pri grafoch G2 a G5 sa podarilo dosiahnuť štatisticky významné zlepšenie ($p_{bss} < 0{,}05$) oproti výsledkom získaným bez použitia selekčnej stratégie. Priebehy pre graf G2 sú zaznačené na Obr. 43. Číselné výsledky pre tento graf sú v Tab. 14.

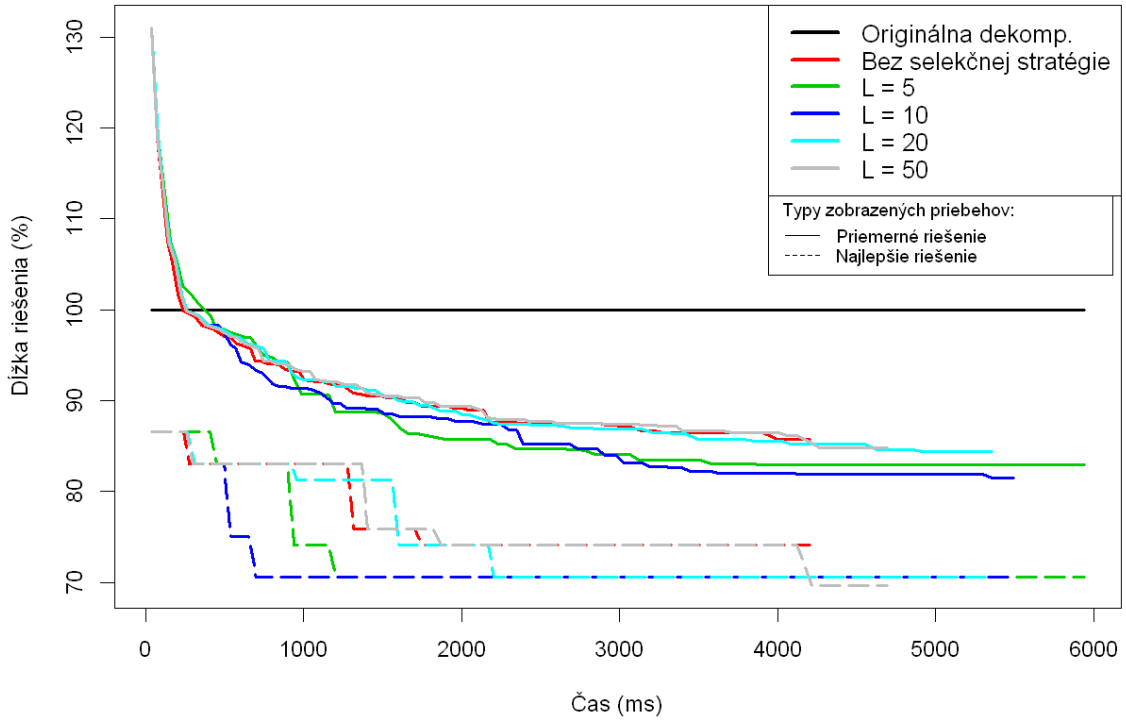


Obr. 43. Graf G2 rôzne hodnoty limitu aktivácie selekčnej stratégie, rozhodovanie podľa priemerného riešenia.

Tab. 14. Graf G2 rôzne hodnoty limitu aktivácie selekčnej stratégie, rozhodovanie podľa priemerného riešenia.

Čas porovnávania (s):					4,17
	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{bss}	p_{nasl}
Bez sel. stratégie	4,21	85,68	74,11	0,500	0,375
L = 10	4,46	85,27	70,54	0,375	0,212
L = 20	4,39	83,93	70,54	0,131	0,298
L = 5	4,59	82,92	70,54	0,045	0,256
L = 50	4,17	81,76	70,54	0,003	-

Pri vyhodnocovaní na základe najlepšieho riešenia sa podarilo dosiahnuť štatisticky významné zlepšenie ($p_{bss} < 0{,}05$) iba pri grafe G2. Priebehy pre tento graf sú zachytené na Obr. 44. Číselné výsledky sú v Tab. 15. Výsledky pre ostatné grafy sú uvedené v prílohe B.



Obr. 44. Graf G2 rôzne hodnoty limitu aktivácie selekčnej stratégie, rozhodovanie podľa najlepšieho riešenia.

Tab. 15. Graf G2 rôzne hodnoty limitu aktivácie selekčnej stratégie, rozhodovanie podľa najlepšieho riešenia.

Čas porovnávania (s):					4,69
	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{bss}	p_{nasl}
L = 50	4,69	86,19	74,11	0,646	0,355
Bez sel. stratégie	4,21	85,68	74,11	0,500	0,373
L = 20	5,35	85,18	70,54	0,373	0,116
L = 5	5,94	82,92	70,54	0,041	0,297
L = 10	5,49	81,88	70,54	0,012	-

11.3.8 Vyhodnotenie pre selekčnú stratégiu

Štatisticky významné zlepšenie oproti prehľadávaniu bez použitia selekčnej stratégie, sa dosiahlo maximálne pri 2 zo 6 grafov (v každom z experimentov). Taktiež žiadne nastavenie selekčnej stratégie nedosahovalo konzistentne lepšie výsledky. Preto v ďalších experimentoch s rôznymi grafmi nepoužijeme selekčnú stratégiu, namiesto nej bude v súlade s MCPS použitá iba simulačná stratégia. Je však možné, že pri iných grafoch, resp. inej logike selekcie, by výsledky boli iné. Preskúmanie iných možností však už zostáva na budúce práce.

Dôvodom, prečo sa selekčná stratégia neosvedčila, môže byť fakt, že pri vysokých koeficientoch heuristík sa celý strom prehľadávania zredukoval na preskúmavanie iba tých možností, ktoré zodpovedajú odporúčaniu podľa heuristík. Stavový priestor sa tak môže výrazne zredukovať, a v takto vzniknutom menšom stavovom priestore už selekčná stratégia so skúmanými nastaveniami nevie prispieť k nájdeniu lepších riešení. Použitie selekčnej stratégie preto môže byť vhodnejšie na ešte zložitejších grafoch, kde by aj takto zredukovaný stavový priestor bol dosť rozsiahly.

11.4 Experimenty s rôznymi grafmi

Na základe predchádzajúcich experimentov bude pri pokusoch s rôznymi grafmi použité nasledujúce nastavenie, ktoré dosiahlo v priemere najlepšie výsledky a najkratší čas behu:

- Kombinácia heuristík krátkeho počiatočného cyklu a minimálneho počtu kolízií, s koeficientom $K = 100$.
- Žiadna selekčná stratégia.
- MCPS bude prebiehať 30 behov po 50 iteráciách (počet iterácií sa zmenšil, keďže je vybraté kvalitné nastavenie).
- Nebolo použité obmedzenie paralelizácie dlhých riešení opisované v kapitole 11.2.1, keďže pri týchto pokusoch sa pracovalo s najlepším nájdeným nastavením prehľadávania. Nebolo tak treba špeciálne ošetrovať prípady, keď nastavenie parametrov prehľadávania bolo zlé a celé prehľadávanie trvalo veľmi dlho.

Na experimenty bude použitá testovacia sada zložená z rôznych grafov, ktoré vychádzajú z grafu G1 z pokusov v kapitole 11.3. Jednotlivé skupiny grafov sú zaznačené v Tab. 16, pričom v každej zo skupín sa mení niektorý parameter grafu. V skupine grafov G100 až G139 sa mení maximálna dĺžka ucha, v skupine grafov G200-G249 sa mení počet úch, v skupine G300-349 sa mení počet voľných uzlov, v skupine G400-G449 sa mení dĺžka počiatočného cyklu. Pre každé z nastavení sa zároveň vyskúša 10 rôznych hodnôt „*seed*“, čo je inicializácia generátora náhodných čísel. V tomto prípade to jednoznačne určuje graf a rozloženie robotov v ňom (samozrejme pre konkrétne hodnoty ostatných parametrov). Celkovo teda prebehne prehľadávanie na 190 rôznych grafoch, s počtom uzlov od 25 po 94.

Tab. 16. Nastavenia grafov pre experimenty v 2. časti.

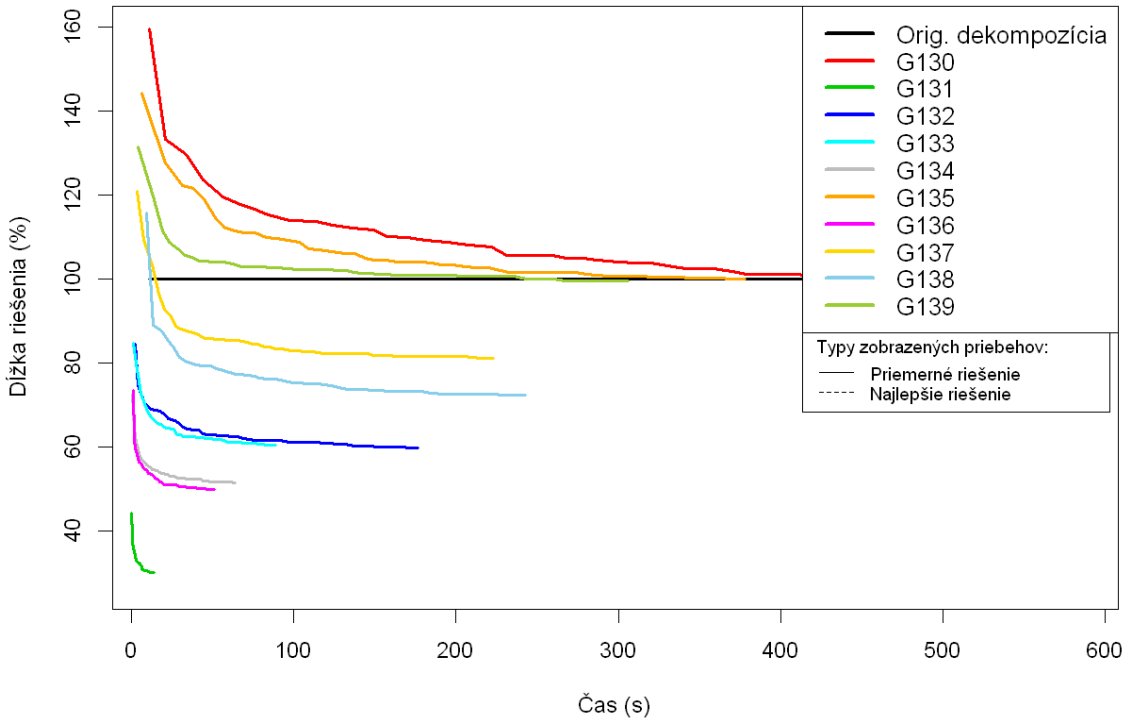
Parameter	G100-139	G200-249	G300-349	G400-449
Počet úch	10	10, 15, 20, 25, 30	10	10
Maximálna dĺžka ucha	3, 5, 8, 16	5	5	5
Dĺžka počiatočného cyklu	5	5	5	6, 8, 10, 12, 14
Počet voľných uzlov	2	2	4, 6, 8, 10, 12	2

Rôzne druhy grafov, tak ako sú uvedené v Tab. 16, vychádzajú z grafov, ktoré boli použité pre vyhodnocovanie algoritmu BIBOX v práci [2]. V uvedenej práci sa pri rôznych experimentoch menil buď počet voľných uzlov v grafe (a ostatné parametre zostali rovnaké), alebo sa menil počet uzlov grafu a maximálna dĺžka ucha zároveň. V danom prípade sa do grafu pridávali uchá s určenou maximálnou dĺžkou pokiaľ počet uzlov grafu nepresiahol stanovenú hranicu. Počet úch bol teda nepriamo určený týmito dvoma parametrami.

Kvôli časovej náročnosti prehľadávania sa pri pokusoch v tejto kapitole mení iba jeden parameter – teda buď maximálna dĺžka ucha alebo počet úch. Ďalšia odlišnosť spočíva v tom, že pri publikovaných pokusoch sa dĺžka počiatočného cyklu určovala rovnako ako pri uchách, teda náhodne z určitého intervalu (hornou hranicou intervalu je maximálna dĺžka ucha). V pokusoch v tejto kapitole sa však dĺžka počiatočného cyklu

nemenila pri zväčšovaní maximálnej dĺžky ucha. Je to kvôli tomu, že ak by bola dĺžka počiatočného cyklu určovaná náhodne, kvalita riešenia pri originálnej dekompozícii by sa mohla meniť veľmi výrazne, keďže je veľmi závislá od dĺžky počiatočného cyklu, ako ukazujú výsledky v kapitole 6.3. Preto sa dĺžka počiatočného cyklu menila iba v jednej skupine grafov.

Na ukážku toho, ako sa môžu líšiť priebehy prehľadávania pri jednom nastavení a rôznych grafoch, môže slúžiť Obr. 45. Sú tam znázornené priebehy prehľadávania pre grafy G130 až G139 (menila sa iba inicializácia generátora náhodných čísel, ktorá jednoznačne určuje štruktúru grafu a rozloženie robotov v ňom). Vidieť, že analýza na základe priebehov pri jednotlivých nastaveniach teda nemá veľký význam, keďže jednotlivé priebehy sú veľmi odlišné.



Obr. 45. Priebehy prehľadávania pre grafy G130 až G139.

Namiesto porovnávania priebehov preto budeme porovnávať výsledky dosiahnuté prehľadávaním pri jednotlivých grafoch. Zaznamenávané údaje pre každý graf sú nasledovné:

- Počet uzlov grafu.
- Priemerné riešenie dosiahnuté za 50 iterácií (ako percento z dĺžky riešenia pri originálnej dekompozícii). Priemer sa počíta z výsledkov dosiahnutých pri jednotlivých behoch.

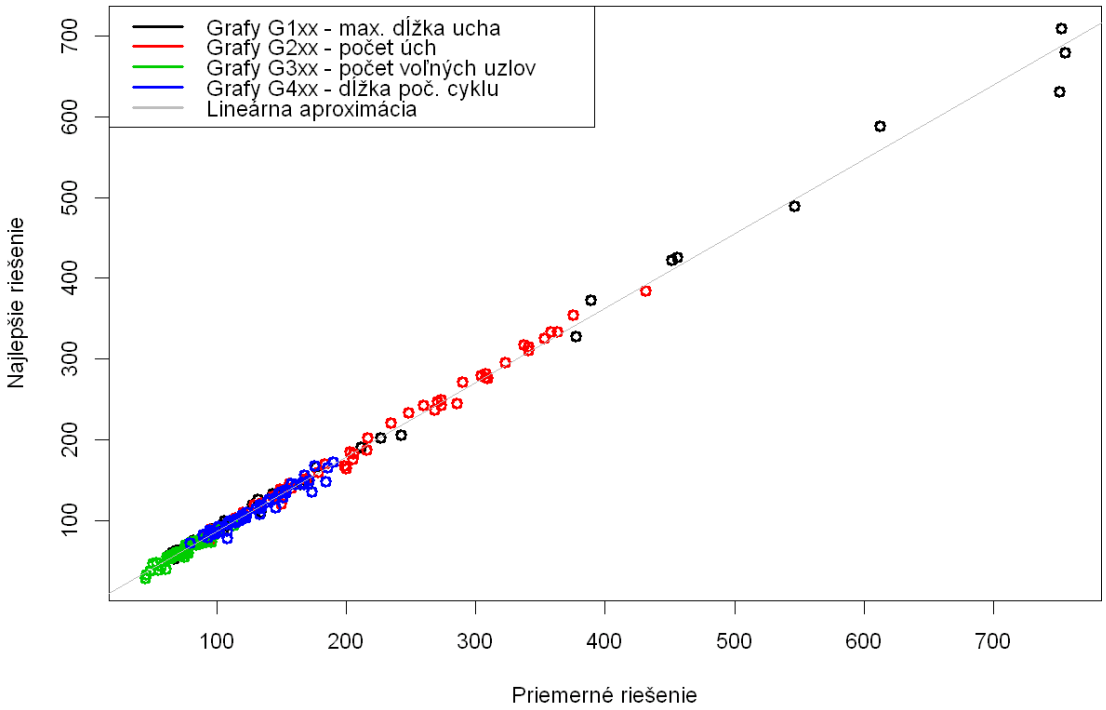
- Najlepšie riešenie dosiahnuté za 50 iterácií (ako percento z dĺžky riešenia pri originálnej dekompozícii).
- Čas potrebný na vykonanie 50 iterácií.

Na Obr. 46 vidieť zaznačené body, ktoré zodpovedajú jednotlivým grafom. Zobrazená je závislosť najlepšieho riešenia od priemerného dosiahnutého riešenia. Farbami sú odlíšené skupiny grafov:

- Zelená: grafy, kde sa menil počet voľných uzlov
- Modrá: grafy, kde sa menila dĺžka počiatočného cyklu
- Červená: grafy, kde sa menil počet úch
- Čierna: grafy, kde sa menila maximálna dĺžka ucha

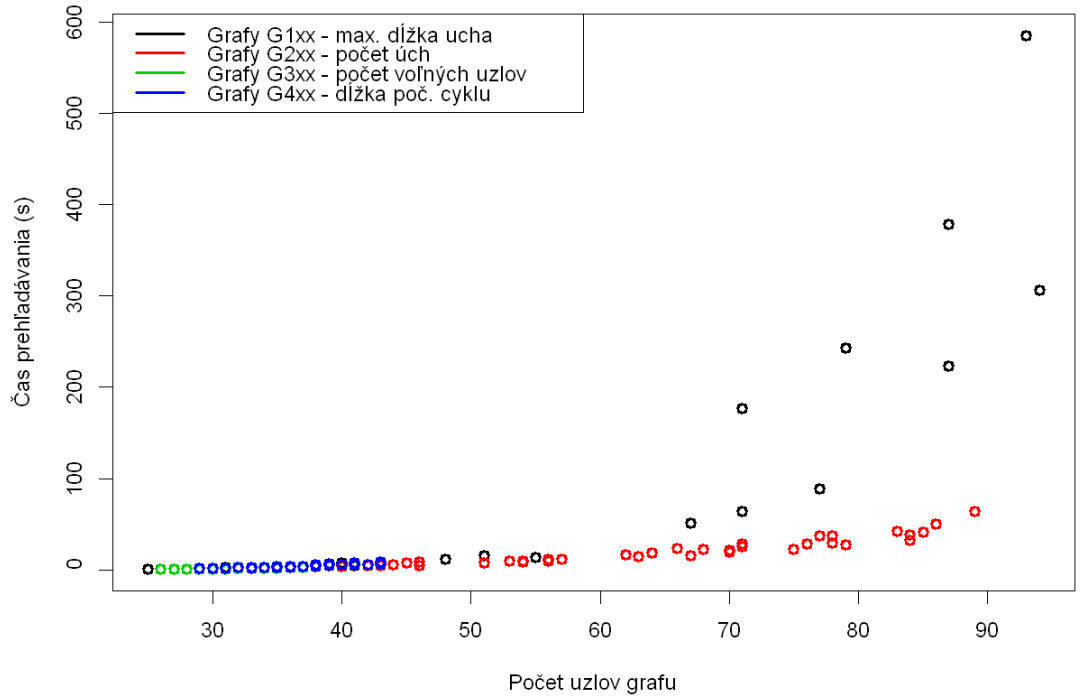
Pre niektoré z nasledovných výsledkov bude použitá regresia na opísanie závislosti medzi veličinami. Ako miera toho, ako dobre opisuje regresný model zvolené dáta, bude použitý koeficient determinovanosti označovaný R^2. Tento koeficient nadobúda hodnoty od 0 po 1. Čím bližšie je hodnota k 1, tak tým lepšie sú údaje opísané regresným modelom [25].

Na Obr. 46 vidíme, že závislosť medzi minimálnym a priemerným riešením je v podstate lineárna. Zaznačená je aj lineárna aproximácia získaná lineárnou regresiou. Rovnica pre danú priamku má tvar $Y = 0{,}92 * X - 5{,}30$, koeficient determinovanosti má hodnotu $0{,}995$. Vysoká hodnota lineárneho koeficientu pri premennej X je najskôr spôsobená vysokým nastavením koeficientu K pri použitej heuristike, ktorá tak výrazne usmerňuje prehľadávanie k najlepším riešeniam.



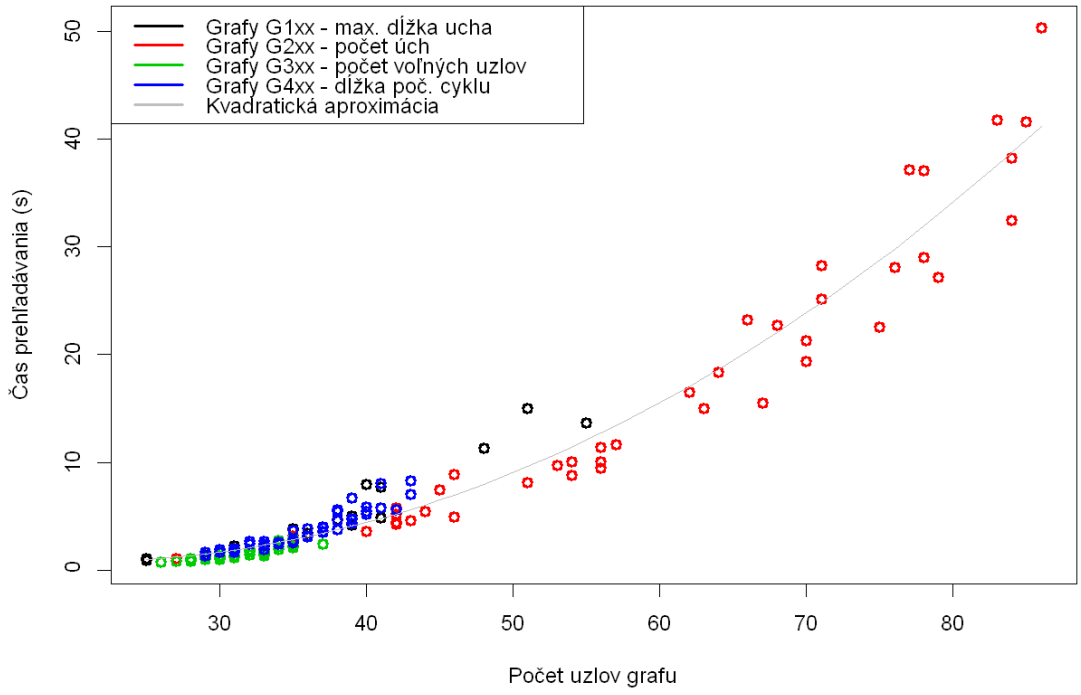
Obr. 46. Závislosť najlepšieho od priemerného riešenia.

Na Obr. 47 je zachytený vzťah medzi časom prehľadávania (priemerným časom potrebným na dokončenie 1 behu, t.j. 50 iterácií) a počtom uzlov grafu. Vidieť, že prehľadávanie pri niektorých grafoch trvalo výrazne dlhšie ako pri iných. Ak však vylúčime 10 grafov, ktoré majú najdlhší čas prehľadávania, dostaneme výsledky ako na Obr. 48. S výnimkou jedného grafu, patrí vylúčených 10 grafov medzi grafy, ktoré mali najdlhšiu max. dĺžku ucha (16). Z Obr. 47 však môžeme vidieť, že pri zväčšujúcej sa maximálnej dĺžke ucha a teda aj zväčšujúcom sa počte uzlov sa veľmi výrazne začne zvyšovať čas prehľadávania. Na lepšie overenie by bolo potrebné vykonať viac pokusov práve na grafoch s väčším počtom uzlov.



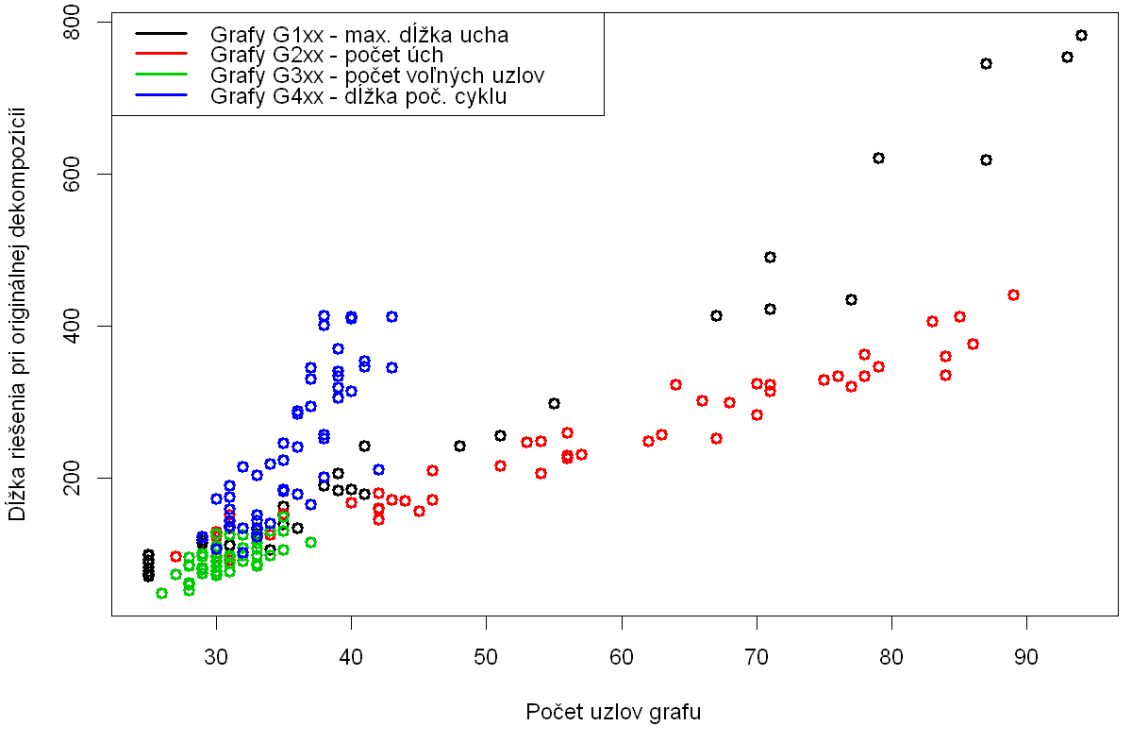
Obr. 47. Závislosť medzi časom prehľadávania a počtom uzlov grafu.

Pri grafoch zachytených na Obr. 48 vidieť, že podobná závislosť je aj pri zvyšovaní počtu úch, ktoré tiež zvyšuje počet uzlov grafu. Čas prehľadávania sa však nezvyšuje tak dramaticky ako pri dlhých uchách. V danom obrázku je uvedená aj kvadratická aproximácia, získaná kvadratickou regresiou. Uvedená kvadratická krivka je určená vzťahom $Y = 0{,}009 * X^2 - 0{,}38 * X + 4{,}713$. Koeficient determinovanosti je 0,962.



Obr. 48. Závislosť medzi časom prehľadávania a počtom uzlov grafu (bez 10 grafov, pri ktorých beh trval najdlhšie) .

Na Obr. 49 je zachytená závislosť dĺžky riešenia pri originálnej dekompozícii od počtu uzlov grafu. Vidieť, že so stúpajúcim počtom uzlov stúpa aj dĺžka riešenia pri originálnej dekompozícii, veľkosť nárastu je však rozdielna pri rôznych skupinách grafov. Pri grafoch, u ktorých sa zväčšovala dĺžka počiatočného cyklu, bol nárast dĺžky riešenia pri originálnej dekompozícii väčší v porovnaní s ostatnými skupinami grafov. Ako už bolo spomenuté, je to spôsobené tým, že dĺžka riešenia pri originálnej dekompozícii veľmi závisí od dĺžky počiatočného cyklu (analýza v kapitole 6.3). Ďalej môžeme vidieť, že so zväčšovaním úch narastá dĺžka riešenia pri originálnej dekompozícii viac ako pri pridávaní úch. Ide o podobný záver ako pri závislosti od času, ktorá je zachytená na Obr. 47, kde pri predlžovaní úch čas riešenia narastal výraznejšie ako pri zväčšovaní počtu úch.



Obr. 49. Závislosť dĺžky riešenia pri originálnej dekompozícii od počtu uzlov grafu.

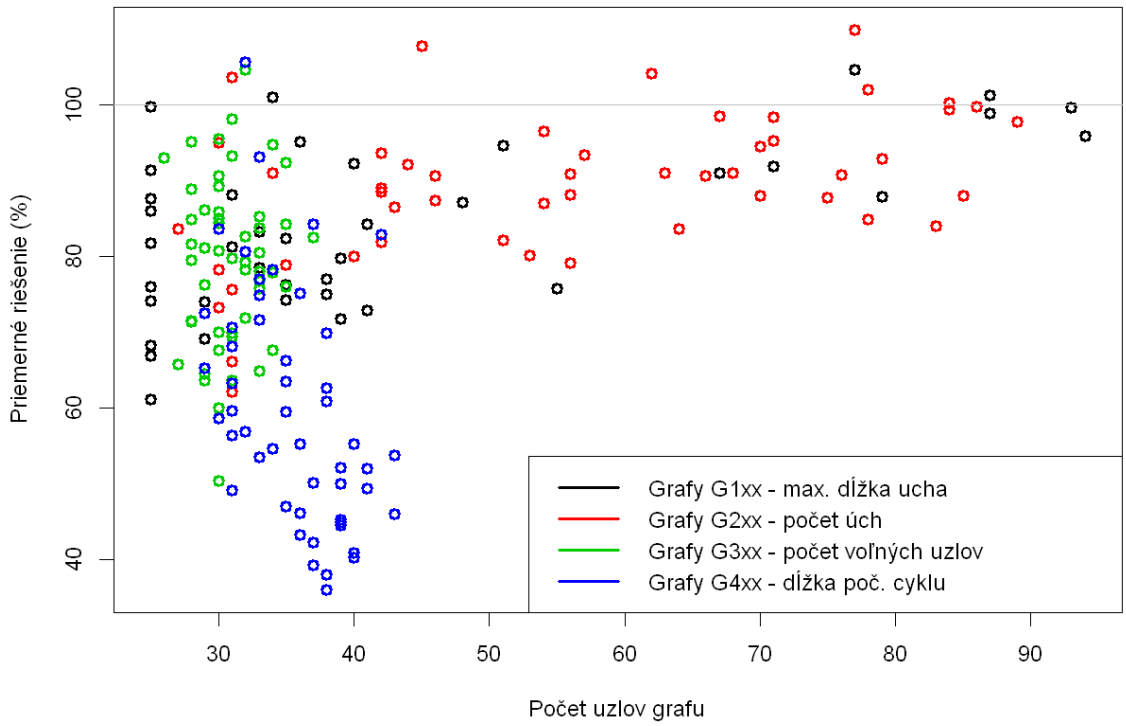
Tieto pozorovania sa dajú vysvetliť množstvom kolízií. Kolízia nastáva, keď má robot cieľovú pozíciu v rovnakom uchu, v akom sa práve nachádza. Algoritmus BIBOX v takýchto situáciách potrebuje na umiestnenie robota na správne miesto viac krokov ako v prípade, že robot sa nachádza v inom uchu. Pri grafe s určitým počtom uzlov (napr. 100), môžu nastať 2 situácie:

1. Dekompozícia grafu má málo dlhých úch (napr. 5 úch, každé má 20 vnútorných uzlov)

2. Dekompozícia grafu má veľa krátkych úch (napr. 20 úch, každé má 5 vnútorných uzlov)

Ak predpokladáme, že pozície robotov a ich cieľové pozície sú náhodne rozmiestnené po grafe, tak pri prvej možnosti je pravdepodobnosť kolízie pre jedného robota $\frac{1}{5}$. V druhom prípade je to $\frac{1}{20}$. Vo všeobecnosti je tak pravdepodobnosť kolízie určená vzťahom $\frac{1}{počet\ úch}$. Pri prvej možnosti tak nastane 4 krát viac kolízií ako pri druhej možnosti. Čím viac kolízií, tým bude dlhšie riešenie a takisto aj jeho nájdenie bude časové náročnejšie.

Závislosť priemerného riešenia nájdeného prehľadávaním (ako percenta z dĺžky riešenia pri originálnej dekompozícii) od počtu uzlov grafu je na Obr. 50. Pri približne 94% skúmaných grafov sa v priemere našla lepšia dekompozícia.



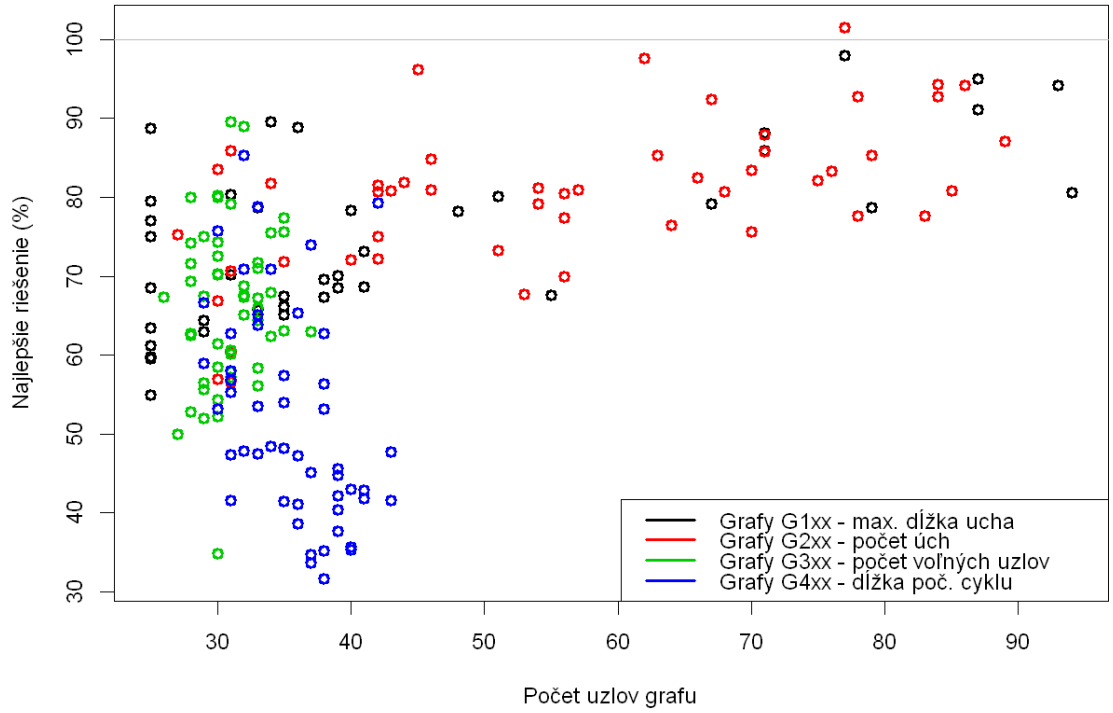
Obr. 50. Závislosť priemerného riešenia od počtu uzlov grafu.

Pri grafoch, kde sa mení dĺžka počiatočného cyklu (modrá farba) vidieť, že tam je najľahšie prekonať originálnu dekompozíciu, ktorá je znevýhodnená dlhým počiatočným cyklom.

Pri grafoch, kde sa mení počet voľných uzlov (zelená farba) je to rôzne, nie je závislosť medzi počtom voľných uzlov a zlepšením, ktoré je možné dosiahnuť.

Pri grafoch, kde sa mení počet úch (červená farba) a maximálna dĺžka ucha (čierna farba) vidieť, že so zväčšujúcim sa počtom uzlov grafu sa priemerné riešenie nájdené prehľadávaním približuje riešeniu pri originálnej dekompozícii.

Pri najlepšom riešení je závislosť znázornená na Obr. 51. Pri 99% grafov sa podarilo nájsť lepšie riešenie ako pri originálnej dekompozícii.



Obr. 51. Závislosť najlepšieho riešenia od počtu uzlov grafu.

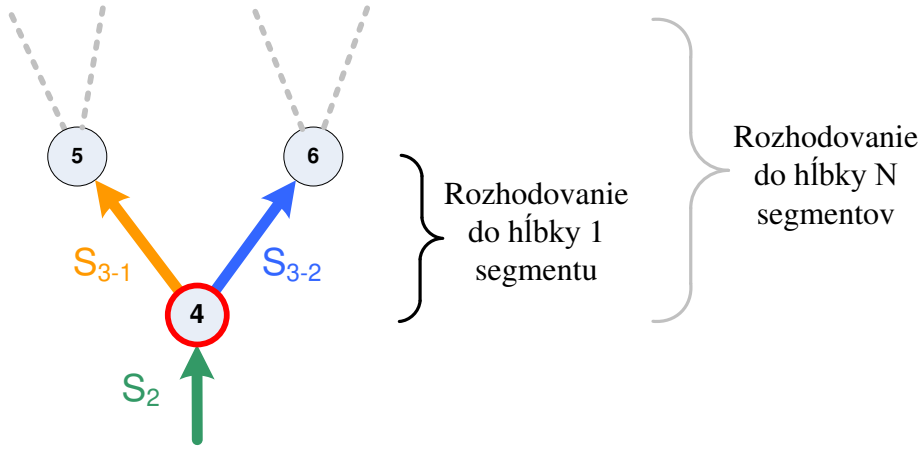
Rozsah dosiahnutých výsledkov v porovnaní s originálnou dekompozíciou je zachytený v Tab. 17. Údaje sú uvedené pre každú skupinu grafov samostatne. Pri najlepšom riešení sa uvažuje najlepšie nájdené riešenie zo všetkých behov prehľadávania. Priemerné riešenie predstavuje priemer z výsledkov, ktoré dosiahli jednotlivé behy prehľadávania.

Tab. 17. Rozsah výsledkov pri jednotlivých skupinách grafov.

Skupina grafov	Najlepšie riešenie (% z dĺžky riešenia pri originálnej dekompozícii)	Priemerné riešenie (% z dĺžky riešenia pri originálnej dekompozícii)
G1xx – max. dĺžka ucha	55-98%	61-104%
G2xx – počet úch	57-102%	62-110%
G3xx – počet voľných uzlov	35-90%	50-105%
G4xx – dĺžka počiatočného cyklu	32-85%	36-106%

12. Možné smery rozvoja problematiky do budúcna

Hľadanie vhodnej dekompozície môže byť vylepšené efektívnejšími heuristikami, priestor na vylepšenie vidíme hlavne pri heuristike minimálneho počtu kolízií. Tak ako bola táto heuristika opísaná v kapitole 8.2, sa rozhoduje iba podľa priľahlých segmentov. Môžeme teda povedať, že ide o prehľadávanie v strome do hĺbky 1. Lepšie výsledky by heuristika mohla dosiahnuť, ak by sa prehľadávalo do väčšej hĺbky. Znázornené je to na Obr. 52. Ak sme v uzle č. 4, tak sa heuristika teraz rozhoduje iba podľa počtu kolízií, ktoré prinesú najbližšie segmenty S_{3-1} a S_{3-2}. Vylepšená heuristika by sa rozhodovala podľa toho, koľko kolízií by prinieslo nasledujúcich N segmentov.



Obr. 52. Návrh vylepšenia heuristiky minimálneho počtu kolízií.

Ďalšou možnosťou vylepšenia je vyskúšanie zavedenia selekčnej stratégie pri MCPS na väčších grafoch. V pokusoch v kapitole 11.3 boli totiž použité pomerne malé grafy, kde zavedenie selekčnej stratégie neprinieslo signifikantné zlepšenie. Ako už bolo napísané vo vyhodnotení uvedených pokusov, pri väčších grafoch by selekčná stratégia mohla prispieť k lepším riešeniam.

Spôsob prehľadávania opisovaný v tejto práci je možné použiť aj pre algoritmus BIBOX- θ (spomenutý v kapitole 4.2), ktorý dokáže riešiť inštancie problému plánovania cesty pre skupinu robotov, pri ktorých je v grafe iba jeden voľný uzol. Heuristika krátkeho počiatočného cyklu by sa však musela modifikovať, aby nehľadala čo najkratší počiatočný cyklus, ale aby sa zamerala na nájdenie čo najmenšieho θ-grafu, čo je spojenie prvého ucha a počiatočného cyklu. Pri algoritme BIBOX-θ sa totiž θ-graf rieši špeciálne (podobne ako počiatočný cyklus pri algoritme BIBOX).

V publikovaných výsledkoch, opísaných v kapitole 4.4, boli okrem náhodných 2-súvislých grafov použité na experimenty aj grafy typu mriežky. Ďalším pokračovaním tejto práce by mohlo byť otestovanie hľadania vhodnej dekompozície práve na takýchto grafoch. Špecifikom originálnej dekompozície pri týchto grafoch je krátky počiatočný cyklus (4 uzly) a veľa krátkych úch (1 alebo 2 vnútorné uzly). Bolo by zaujímavé zistiť, ako sa dá vhodnou dekompozíciou pri takýchto grafoch skrátiť výsledné riešenie.

13. Zhodnotenie

Cieľom práce bolo zoptimalizovať riešenie problému plánovania cesty pre skupinu robotov. Práca sa zameriava na algoritmus BIBOX, navrhnutý P. Surynekom, ktorý je určený pre špecifické inštancie uvedeného problému, kde grafom je 2-súvislý graf a v grafe sa nachádza málo voľných uzlov. Algoritmus BIBOX na svoje fungovanie vyžaduje, aby bola pre graf určená dekompozícia na počiatočný cyklus a uchá. Pri publikovaných pokusoch sa žiadny algoritmus na dekompozíciu nepoužil, namiesto toho bola dekompozícia určená konštrukciou 2-súvislého grafu. Optimalizácia dekompozície grafu je aj samotným autorom algoritmu BIBOX považovaná za otvorenú otázku na ďalšie práce.

Táto práca sa teda venuje optimalizácii dekompozície grafu tak, aby algoritmus BIBOX dosiahol s danou dekompozíciou čo najkratšie riešenie. Na účely dekompozície bol implementovaný algoritmus od Brandesa [19], ktorý určuje dekompozíciu postupne pri prehľadávaní grafu do hĺbky. Keďže možných dekompozícií je veľké množstvo, na prehľadávanie stavového priestoru bol zvolený algoritmus Monte-Carlo prehľadávania stromu.

Na urýchlenie prehľadávania boli na základe analýzy výsledkov náhodného prehľadávania, či analýzy samotného algoritmu BIBOX, navrhnuté dve heuristiky. Prvá heuristika sa snaží prehľadávanie smerovať tak, aby počiatočný cyklus výslednej dekompozície bol čo najmenší. Druhá heuristika sa snaží minimalizovať prípady, ktoré pri algoritme BIBOX predlžujú počet krokov výsledného riešenia. Pri vyhodnocovaní najvhodnejších nastavení prehľadávania vychádza ako najlepšia kombinácia oboch heuristík, ktorá v porovnaní s náhodným prehľadávaním dosiahla pri rôznych grafoch o 27-44% kratšie riešenia.

Nasledujúce pokusy sa venovali vyhodnocovaniu výsledkov, ktoré prehľadávanie dosiahne s určeným nastavením na testovacej sade, zloženej zo 190 rôznych grafoch. Prehľadávanie dokázalo v priemere nájsť lepšiu dekompozíciu pri 94% grafov, pričom zlepšenie oproti originálnej dekompozícii bolo na úrovni 38-64% v závislosti od typu grafu.

Nevýhodou opisovaného prístupu je časová náročnosť, keďže sa jedná o prehľadávanie, pri ktorom sa algoritmus BIBOX spúšťa viackrát. Jeho použitie preto

vidíme hlavne v situáciách, v ktorých sa rieši inštancia problému plánovania cesty pre skupinu robotov v reálnom svete a s presunom robotov sú tak spojené isté náklady. V takejto situácii môže byť efektívnejšie venovať viac času nájdeniu čo najlepšieho riešenia.

Algoritmus hľadania dekompozície, opísaný v tejto práci, je prvým takým algoritmom, ktorý sa špecificky venuje hľadaniu vhodnej dekompozície pre algoritmus BIBOX. Ako ukazujú výsledky, takýto spôsob optimalizácie výsledného riešenia je veľmi sľubný. Pre budúcu prácu sme preto navrhli v predchádzajúcej kapitole ďalšie vylepšenia, s ktorými by sa dali dosiahnuť ešte lepšie výsledky.

14. Literatúra

[1] SURYNEK, P.: A Novel Approach to Path Planning for Multiple Robots in Bi-connected Graphs. In *Proceedings of the 2009 IEEE International Conference on Robotics and Automation (ICRA 2009)*. Piscataway, USA: IEEE Press, 2009, pp. 3613-3619.

[2] SURYNEK, P.: *An Application of Pebble Motion on Graphs to Abstract Multi-robot Path Planning*. Technical Report, ITI Series, 2010-504. MFF, Karlova Univerzita v Prahe, 2010. Dostupné na: http://ktiml.mff.cuni.cz/~surynek/download/download.php?file=publications/files/SurynekPavel_Multirobots-Practice_ITI-TechReport-2010.pdf

[3] DIESTEL, R.: *Graph Theory*. New York: Springer-Verlag, 2000.

[4] WEST, D. B.: *Introduction to Graph Theory*. Prentice Hall, 2001.

[5] RATNER, D., WARMUTH, M. K.: The (n^2-1)-Puzzle and Related Relocation Problems. In *Journal of Symbolic Computation*. Duluth, USA: Academic Press, Inc., 1990, vol. 10, issue 2, pp. 111-138.

[6] KORNHAUSER, D., MILLER, G. L., SPIRAKIS, P. G.: Coordinating Pebble Motion on Graphs, the Diameter of Permutation Groups, and Applications. In *Proceedings of the 25th Annual Symposium on Foundations of Computer Science (FOCS 1984)*. Washington, USA: IEEE Computer Society, 1984, pp. 241-250.

[7] WILSON, R. M.: Graph Puzzles, Homotopy, and the Alternating Group. In *Journal of Combinatorial Theory*, *Series B*. Elsevier, 1974, vol. 16, pp. 86-96.

[8] SURYNEK, P.: An Optimization Variant of Multi-Robot Path Planning is Intractable. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI 2010)*. Atlanta, USA: AAAI Press, 2010, pp. 1261-1263.

[9] PAPADIMITRIOU, C. H., RAGHAVAN, P., SUDAN, M., TAMAKI, H.: Motion Planning on a Graph. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science (FOCS 1994)*. Washington, USA: IEEE Computer Society, 1994, pp. 511-520.

[10] AULETTA, V., PERSIANO, P.: Optimal pebble motion on a tree. In *Journal of Information and Computation*. Duluth, USA: Academic Press, Inc., 2001, vol. 165, issue 1.

[11] AULETTA, V., MONTI, A., PARENTE, M., PERSIANO, P.: A Linear Time Algorithm for the Feasibility of Pebble Motion on Trees. In *SWAT '96 Proceedings of the 5th Scandinavian Workshop on Algorithm Theory*. London: Springer-Verlag, 1996.

[12] RYAN, M. R. K.: Exploiting Subgraph Structure in Multi-Robot Path Planning. In *Journal of Artificial Intelligence Research (JAIR)*. USA: AAAI Press, 2008, vol. 31, issue 1, pp. 497-542.

[13] WU, Z., GRUMBACH, S.: Feasibility of Motion Planning on Directed Graphs. In *TAMC '09 Proceedings of the 6th Annual Conference on Theory and Applications of Models of Computation*. Berlin: Springer-Verlag, 2009, pp. 430-439.

[14] GORALY, G., HASSIN, R.: Multi-Color Pebble Motion on Graphs. In *Algorithmica*. New York, USA: Springer New York, 2009, vol. 58, num. 3, pp. 610-636.

[15] SURYNEK, P.: An Application of Pebble Motion on Graphs to Abstract Multi-robot Path Planning. In *Proceedings of the 21st International Conference on Tools with Artificial Intelligence (ICTAI 2009)*. Newark, USA: IEEE Press, 2009, pp. 151-158.

[16] SURYNEK, P.: Making Solutions of Multi-robot Path Planning Problems Shorter Using Weak Transpositions and Critical Path Parallelism. In *Proceedings of the 2009 International Symposium on Combinatorial Search (SoCS 2009)*. Lake Arrowhead, USA: University of Southern California, 2009.

[17] KOUPÝ, P.: *GraphRec – a visualization tool for entity movement on graph*. Stránka projektu: http://www.koupy.net/graphrec.php, 2010.

[18] SURYNEK, P., KOUPÝ, P.: Improving Solutions of Problems of Motion on Graphs by Redundancy Elimination. In *Proceedings of the ECAI 2010 Workshop on Spatio-Temporal Dynamics*. SFB/TR 8, Universität Bremen, 2010, pp. 37-42.

[19] BRANDES, U.: Eager st-Ordering. In *Proceedings of the 10th Annual European Symposium on Algorithms (ESA '02)*. London, UK: Springer-Verlag, 2002, pp. 247-256.

[20] TARJAN, R. E.: Two streamlined depth-first search algorithms. In *Fundamenta Informaticae*. Amsterdam: North Holland, 1986, vol. 9, issue 1, pp. 85-94.

[21] NÁVRAT, P. et al.: *Umelá inteligencia*. Bratislava: STU, 2007.

[22] CHASLOT, G. et al.: Monte-Carlo Tree Search : A New Framework for Game AI. In *Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference*. Stanford, USA: The AAAI Press, 2008, pp. 216-217.

[23] SCHADD, M. P. et al.: Single-Player Monte-Carlo Tree Search. In *Proceedings of the 6th international conference on Computers and Games (CG '08)*. Heidelberg: Springer-Verlag Berlin, 2008, pp. 1-12.

[24] COULOM, R.: Efficient selectivity and backup operators in Monte-Carlo tree search. In *Proceedings of the 5th International Conference on Computer and Games*. Heidelberg: Springer-Verlag, 2007.

[25] ROSS, S. M.: Introductory Statistics, Third Edition. Burlington, MA: Academic Press, 2010.

Príloha A Technická dokumentácia

V tejto časti je detailnejšie opísaná implementácia algoritmu pre dekompozíciu grafu a pre prehľadávanie priestoru dekompozícií. Ďalej je tu uvedená používateľská príručka pre spúšťanie uvedených algoritmov. Tiež sú tu opísané pomocné nástroje použité na vyhodnocovanie, či vizualizáciu výsledkov.

A.1 Opis implementácie

Implementácia alternatívnej dekompozície vychádza zo zdrojového kódu autora algoritmu BIBOX v jazyku C++. Tento zdrojový kód je dostupný na <http://ktiml.mff.cuni.cz/~surynek/research/ecaiw2010/> a je zverejnený pod licenciou GPL 2. Zdrojový kód, ktorý sa nachádza na priloženom CD v adresári </source/multirobot/> je teda tiež zverejnený pod licenciou GPL 2. Text licencie je uvedený v súbore </source/multirobot/LICENCE>.

V pôvodnom kóde bolo veľmi málo komentárov, takže v rámci prechádzania kódu, boli do neho pridávané komentáre a rôzne pomocné funkcie a pomocné výpisy. Okrem toho boli vykonané zmeny, ktoré sú ďalej podrobnejšie opísané.

Boli vymazané súbory pre pravdepodobne staršie experimenty, ktoré neboli potrebné pre implementáciu algoritmov z tejto práce. Išlo o súbory: *bibox_main.cpp, tbox_main.cpp, slider_main.cpp, mit_main.cpp, multirobot.cpp, multirobot.h, obox_main.cpp* (z tohto súboru boli presunuté funkcie *print_IDS_list* a *print_IDS_set* do *obox.cpp*).

Najdôležitejšími úpravami v súbore *bibox.h* sú atribúty pridané kvôli algoritmu pre alternatívnu dekompozíciu:

- Pre hrany bolo treba pridať zoznam závislých hrán (trieda *EDGE2*).
- Ku grafu (trieda *GRAPH*) boli pridané tieto atribúty:
    - Množina hrán stromu a spätných hrán (pre prehľadávanie grafu do hĺbky).
    - Zoznam závislých hrán pre každú hranu grafu.
    - Zoznam cyklov alternatívnej dekompozície, ktorý sa postupne napĺňa.
    - Množina spracovaných uzlov.

    - Číslo prvého uzlu, ktorým sa začala dekompozícia (pri originálnej dekompozícii sa vždy začínalo uzlom č. 0).
    - Smerník na zoznam segmentov pre práve vytvárané ucho.
    - Počet robotov, ktorí sa nachádzali v uchu, v ktorom mali cieľovú pozíciu.
    - Počet robotov, ktorí sa nachádzali mimo ucha, v ktorom mali cieľovú pozíciu.
- K uzlu (trieda *VERTEX*) boli pridané tieto atribúty:
    - Príznak, či už bol uzol navštívený (*m_visited*).
    - Algoritmus dekompozície pri prehľadávaní do hĺbky vytvára DFS strom, pridaný atribút *m_child* vtedy určuje aktuálne spracovávaného potomka daného uzlu, atribút *m_parent* určuje rodiča daného uzlu.

V súbore *bibox*.cpp, ktorý obsahuje kód samotného algoritmu BIBOX, boli vykonané nasledovné zmeny, ktoré súviseli s tým, že autor kódu nepredpokladal inú dekompozíciu (napr. počiatočný cyklus začínal vždy v uzle č. 0, čo pri alternatívnej dekompozícii neplatí):

- Pridaný príznak *ALLOW_EMPTY_EAR_TRAVERSAL*. Tento príznak určuje, či je možné pri hľadaní cesty za účelom presunutia robota, použiť uchá bez vnútorných uzlov. Dôvod je bližšie vysvetlený v kapitole 6.1. Tento príznak je normálne zapnutý, avšak pri riešení počiatočného cyklu (metóda *solve_critical_last_free*) je nutné ho vypnúť. Je to kvôli tomu, že implementácia algoritmu BIBOX s možnosťou existencie prázdneho ucha nepočítala a presun robota cez prázdne ucho tak pokazí počítanie rotácií počiatočného cyklu (tento proces je bližšie opísaný v kapitole 4.1.2), a algoritmus potom nefunguje správne.
- V metóde *find_shortest_path*, ktorá hľadá najkratšiu cestu v grafe, sa pri prechode do ďalšieho uzlu kontroluje, či daný uzol nie je zo zoznamu *fvertices* a nepatrí cyklu zo zoznamu *fcycles* (do tohto zoznamu sa dávajú čísla už spracovaných cyklov, teda cyklov, ktoré sa nemajú používať na presun robotov). Táto podmienka povoľovala hľadanie cesty aj cez prázdne ucho, ktoré nemá vnútorné uzly. Ako však bolo spomenuté v predchádzajúcom bode, takéto správanie je potrebné v niektorých okamihoch zakázať. Bola preto pridaná podmienka, ktorá v prípade, že je

nastavený príznak *ALLOW_EMPTY_EAR_TRAVERSAL* skontroluje či hrana z aktuálneho uzla do nasledujúceho uzla nepatrí niektorému z cyklov v zozname *fcycles*. Takýmto spôsobom sa zakáže prechod cez prázdne ucho, ktoré má len jednu hranu (každá hrana má uložené číslo cyklu, ku ktorému patrí).

- V metóde *exchange_last_cycle* a *solve_critical_last_free* sa na dočasné ukladanie robotov z počiatočného cyklu používal prvý vnútorný uzol prvého ucha. To však pri alternatívnej dekompozícii môže byť prázdne, musí sa preto nájsť prvé ucho, ktoré prázdne nie je.
- V metóde *solve_critical_last_free* je pridané zakázanie prechádzania prázdnych úch (zapnutie a vypnutie príznaku *ALLOW_EMPTY_EAR_TRAVERSAL*).

Vlastný kód pre alternatívnu dekompozíciu začína vo funkcii *mainDecomposition* (súbor *main.cpp*), ktorej štruktúra je jednoduchá:

```
GRAPH g;
Context* c = new Context(&g, &config);
generateBiconnected(config, &g);

if (!config.originalDecomposition)
{
      randomizeDfs(&g, config.decompositionSeed);
      decompose(c, &selectInOrder);
}

Result result;
runMultirobot(stdout, config, result, g);
```

Najprv sa prečíta konfigurácia experimentu z parametrov príkazového riadku do premennej *config*, čo nie je vo výpise znázornené. Následne sa vygeneruje 2-súvislý graf funkciou *generateBiconnected* (pripájanie úch k počiatočnému cyklu). Ak je požiadavka na alternatívnu dekompozíciu, náhodne sa premieša v metóde *randomizeDfs* poradie zoznamu susedných uzlov, ktorý sa nachádza v každom uzle grafu. Prehľadávanie do hĺbky a teda aj dekompozícia tak budú prebiehať náhodne. Následne sa vykoná samotná dekompozícia funkciou *decompose* (*selectInOrder* je funkcia ktorá vyberá zo zoznamu susedných uzlov nasledujúci uzol na spracovanie podľa poradia,

resp. na základe heuristiky). Následne sa funkciou *runMultirobot* spustí samotný algoritmus BIBOX a jeho výsledok sa uloží do premennej *result*. Výsledky, ktoré sú pred paralelizáciou dlhšie ako 10000 krokov sa neparalelizujú, pretože proces paralelizácie je potom príliš časovo náročný.

V súbore *ear_decomposition.cpp* je implementovaná dekompozícia grafu algoritmom z kapitoly 5. Hlavná funkcia je *decompose*, ktorá vyberie prvý uzol dekompozície a spustí prehľadávanie do hĺbky (funkcia *dfs*), ktoré vytvorí novú dekompozíciu. Do pôvodnej dekompozície sa však zatiaľ nezasiahlo, nová dekompozícia prepíše pôvodnú až vo funkcii *replaceCycles*.

V súbore *config.cpp* sú funkcie pre načítavanie parametrov experimentu z príkazového riadku.

A.1.1 Monte-Carlo prehľadávanie stromu

Kód pre Monte-Carlo prehľadávanie stromu (MCPS) začína v metóde *mainMcts* v súbore *main.cpp*. Kostra metódy v pseudokóde vyzerá približne nasledovne (na začiatku sa vyskúša originálna dekompozícia pre porovnanie):

```
Spusti algoritmus BIBOX s originálnou dekompozíciou
Opakuj pre počet behov MCPS:
     Vytvor strom prehľadávania pre MCPS
     Opakuj pre počet iterácií v behu:
          Spusti iteráciu
          Zaznamenaj výsledky
```

Spustenie jednej iterácie je tiež pomerne jednoduché, skladá sa z týchto krokov:

```
Znáhodni prehľadávanie do hĺbky
Dekomponuj graf
Spusti algoritmus BIBOX
Aktualizuj štatistiku v uzloch stromu
```

Znáhodnenie prehľadávania do hĺbky slúži kvôli simulačnej stratégii, ktorá postupuje do istej miery náhodne. Zapojenie MCPS sa deje pri dekompozícii grafu, čo je v zdrojovom kóde zapísané nasledovne:

```
decompose(&h, &selectByMcts);
```

Tento riadok hovorí o tom, že výber nasledujúceho uzlu pri prehľadávaní grafu do hĺbky bude riadiť algoritmom MCPS. Samotný MCPS algoritmus je implementovaný v súboroch *mcts.h* a *mcts.cpp*. Základnou triedou je *MCTSNode*, ktorá predstavuje jeden uzol stromu prehľadávania, uchováva štatistické údaje a odkaz na rodiča a potomkov. Trieda *MCTS* predstavuje samotný strom prehľadávania. Obsahuje tiež zoznam všetkých uzlov stromu. Hlavné kroky algoritmu (podľa kapitoly 9) sú realizované nasledovnými metódami. Selekcia aj simulácia sa vykonáva v metóde:

```
int MCTS::select(Context* context, int v)
```

V tejto metóde je potrebné rozhodnúť sa, ktorý zo susedných uzlov uzla *v*, bude ďalším uzlom pri prehľadávaní do hĺbky. Parameter *context* reprezentuje graf a konfiguráciu prehľadávania.

Na výber medzi možnosťami vo fáze selekcie aj simulácie je použitá jednoduchá metóda opísaná v kapitole 10.1. Implementovaná je v súbore *selection.cpp* v triede *Selector*, ktorá na vstup požaduje zoznam možností spolu s ich kvalitou (*SelectionData*) a hodnotu koeficientu K (*divider*). Metóda *select()* triedy *Selector* potom vráti číslo vybranej možnosti.

Fáza rozvitia uzlov je pokrytá metódou:

```
MCTSNode* MCTS::expandNode(MCTSNode* node, int vertex,
                                          int nextVertex)
```

Táto metóda pridá do stromu prehľadávania nový uzol, ktorý reprezentuje skutočnosť, že pri prehľadávaní do hĺbky sa v uzle grafu *vertex* ako nasledujúci uzol vybral uzol *nextVertex*. Aktualizácia štatistiky je implementovaná metódou:

```
void MCTS::updateStatistics(MCTSNode* node,
                                int solutionLength)
```

Táto metóda v uzle stromu prehľadávania *node* aktualizuje štatistiku výsledkov a následne sa rekurzívne zavolá na rodičovskom uzle. *SolutionLength* je dĺžka riešenia, teda výsledok behu algoritmu BIBOX.

Heuristika krátkeho počiatočného cyklu je implementovaná v súbore *short_cycle_heuristics.cpp* triedou *ShortCycleHeuristics*. Samotný výber je implementovaný v metóde:

```
int selectNextVertex(int currentVertex,
                     IDS_vector* allowedNext,
                     Context *context)
```

Parameter *currentVertex* je aktuálny uzol, *allowedNext* je zoznam nasledujúcich uzlov, z ktorých sa vyberá. Parameter *context* reprezentuje graf a konfiguráciu prehľadávania. Táto metóda vypočíta vzdialenosť od každého uzla zo zoznamu *allowedNext* k počiatočnému uzlu dekompozície. Na základe tejto vzdialenosti sa vykoná rozhodnutie, ktorým uzlom pokračovať. Číslo zvoleného uzlu je potom návratovou hodnotou.

Heuristika minimálneho počtu kolízií je implementovaná v súbore *segments.cpp*. V tomto súbore sa nachádza trieda *Segment*, ktorá predstavuje jeden segment v grafe. Metódou *calculateCollisions()* sa dá vypočítať počet kolízií, ktoré daný segment prinesie do práve vytváraného ucha. Ďalšou triedou je trieda *Segments*, ktorá predstavuje všetky segmenty práve vytváraného ucha. Hlavnou metódou je metóda:

```
int select(int v, IDS_vector allowedNext, Context *context)
```

Parameter *v* je aktuálny uzol, *allowedNext* je zoznam nasledujúcich uzlov, z ktorých sa vyberá. Parameter *context* reprezentuje graf a konfiguráciu prehľadávania. Táto metóda vytvorí dočasný segment pre každý uzol zo zoznamu *allowedNext*. Pre každý takýto segment sa vypočíta počet kolízií, ktoré daný segment prinesie do práve vytváraného ucha. Na základe vypočítanej hodnoty, ktorá zahŕňa počet kolízií a dĺžku segmentov sa vykoná rozhodnutie, ktorým uzlom pokračovať. Číslo zvoleného uzlu je potom návratovou hodnotou.

A.2 Používateľská príručka

Obsahom tejto kapitoly je používateľská príručka pre spúšťanie implementovaných algoritmov pomocou konzolovej aplikácie. Cieľom bol predovšetkým návrh a testovanie algoritmov, preto na odporúčanie vedúceho práce nebolo vytvorené grafické používateľské rozhranie, a ušetrený čas bol venovaný práve návrhu a testovaniu algoritmov.

A.2.1 Kompilácia

Na priloženom CD sa v adresári *</source/multirobot>* nachádzajú zdrojové kódy, ktoré treba skompilovať g++ kompilátorom nasledujúcimi príkazmi:

```
g++ -c *.cpp *.h
g++ -o Multirobot.exe *.o
```

Pre operačný systém Windows je spustiteľný súbor *Multirobot.exe* uložený v adresári *</source/multirobot/bin>*.

A.2.2 Spustenie algoritmu BIBOX (s alternatívnou dekompozíciou)

Prvým režimom, v ktorom sa dá program *Multirobot.exe* využiť, je spustenie algoritmu BIBOX na určitom grafe, s voliteľným zapojením alternatívnej dekompozície. Pre tento účel sú k dispozícii nasledovné parametre príkazového riadku, ktoré je potrebné uviesť pri spustení programu:

--seed : inicializácia generátora náhodných čísel, jednoznačne určuje vygenerovaný graf a rozloženie robotov (ich počiatočné a cieľové pozície).

--decompositionSeed : inicializácia generátora náhodných čísel, jednoznačne určuje alternatívnu dekompozíciu.

--initialCycleLength : dĺžka počiatočného cyklu vygenerovaného grafu.

--earCount : počet úch vygenerovaného grafu.

--maxLengthOfEar : maximálna dĺžka ucha vygenerovaného grafu.

--runId : číselná identifikácia behu, nemá ďalší význam.

--*freeVertices* : počet voľných uzlov v grafe (t.j. uzlov, v ktorých sa nenachádzajú roboty).

--*decomposition* : treba uviesť reťazec ORIGINAL, ak sa má použiť originálna dekompozícia (vtedy parameter *decompositionSeed* nemá význam), alebo reťazec NEW, ak sa má použiť alternatívna dekompozícia.

--*help* : vypíše informácie ku všetkým parametrom.

Ak sa niektoré parametre nezadajú, použijú sa predvolené hodnoty. Medzi parametrami programu je aj nastavovanie inej metódy riešenia (implicitne sa používa BIBOX), či metódy skrátenia riešenia. Nastavenie týchto parametrov však nebolo otestované, preto ho tu neuvádzame.

A.2.3 Výstup algoritmu BIBOX

Výstupom programu je najprv výpis parametrov experimentu vo forme argumentov pre príkazový riadok. Je tak možné jednoducho zopakovať celý experiment. Nasleduje výstup algoritmu BIBOX, ktorý je možné načítať napr. v programe GraphRec [17], v ktorom sa dá spustiť animácia celého riešenia. V dokumentácii k programu GraphRec (dostupnej na <http://koupy.net/download/GraphRec-guide.html>) je možné nájsť detaily formátu. Ďalej sú spomenuté len najdôležitejšie údaje z celého výstupu.

Na začiatku sú tri sekcie (začínajú riadkami V=, E=, C=), ktoré obsahujú zoznam uzlov grafu, hrán a cyklov. Príklad najdôležitejších položiek z výstupu:

```
Solution size:921
Filtered solution size:865
Parallel solution length:143
Parallelism:6.049
```

„*Solution size*“ je počet výmen susedných entít (roboty aj voľné miesta sa chápu na tomto mieste rovnako). „*Filtered solution size*“ je počet pohybov robotov (vyfiltrujú sa teda tie výmeny, ktoré sa uskutočnili medzi voľnými miestami). „*Parallel solution length*“ je počet krokov výsledného riešenia, ktorý sa získal tak, že sa nájdené riešenie paralelizovalo spôsobom uvedeným v kapitole 4.3.1. Táto veličina nás zaujíma najviac. „*Parallelism*“ je potom podiel posledných dvoch spomínaných hodnôt.

A.2.4 Spustenie Monte-Carlo prehľadávania stromu

Druhým režimom, v ktorom sa dá program *Multirobot.exe* využiť, je spustenie Monte-Carlo prehľadávania stromu (MCPS). Pre tento účel sú k dispozícii nasledovné parametre príkazového riadku, ktoré rozširujú zoznam parametrov uvedených v kapitole A.2.2.

--*mcts* : tento parameter treba uviesť, keď sa má spustiť MCPS.

--*mctsRuns* : počet behov MCPS.

--*mctsIterations* : počet iterácií v jednom behu.

--*useMctsSelection* : ak je uvedený tento parameter, použije sa selekčná stratégia.

--*mctsSelectionLimit*: počet návštev uzla, od ktorého sa začne uplatňovať selekčná stratégia.

--*mctsSelectionDivider*: hodnota koeficientu K pre selekčnú stratégiu.

--*selectByBestSolution*: tento parameter treba uviesť, ak sa má selekčná stratégia riadiť podľa najlepšieho riešenia v konkrétnom podstrome. V opačnom prípade sa bude riadiť podľa priemerného riešenia.

--*useShortCycleHeuristics*: bude použitá heuristika krátkeho počiatočného cyklu.

--*shortCycleHeuristicsDivider*: hodnota koeficientu K pre heuristiku krátkeho počiatočného cyklu.

--*useMinimalCollisionsHeuristics*: bude použitá heuristika minimálneho počtu kolízií.

--*minimalCollisionsHeuristicsDivider*: hodnota koeficientu K pre heuristiku minimálneho počtu kolízií.

Parameter *decompositionSeed* sa pri týchto experimentoch používa na jednoznačné určenie priebehu MCPS.

A.2.5 Výstup Monte-Carlo prehľadávania stromu

Výstupom programu je najprv výpis parametrov, s ktorými bolo prehľadávanie spustené. Nasleduje tabuľka hodnôt, kde každý riadok predstavuje jednu iteráciu. Stĺpce a ich význam sú nasledovné:

- RUN: číslo behu.
- ITER: číslo iterácie.
- PSL: dĺžka riešenia po paralelizácii (=1, ak riešenie paralelizované nebolo).
- BSL: najlepšie dosiahnuté riešenie v danej iterácii.
- OSL: dĺžka riešenia pri originálnej dekompozícii (rovnaká vo všetkých iteráciách).
- RIC: počet robotov, ktorí sa nachádzali v uchu, v ktorom majú cieľovú pozíciu.
- ROC: počet robotov, ktorí sa nenachádzali v uchu, v ktorom majú cieľovú pozíciu.
- FCL: dĺžka počiatočného cyklu pri dekompozícii grafu v danej iterácii.
- TIME: čas v milisekundách od začiatku pokusu po skončenie konkrétnej iterácie.

A.3 Pomocné nástroje

V tejto kapitole sú opísané pomocné nástroje implementované na vizualizáciu, či vyhodnocovanie výsledkov.

A.3.1 Vizualizácia dekompozície

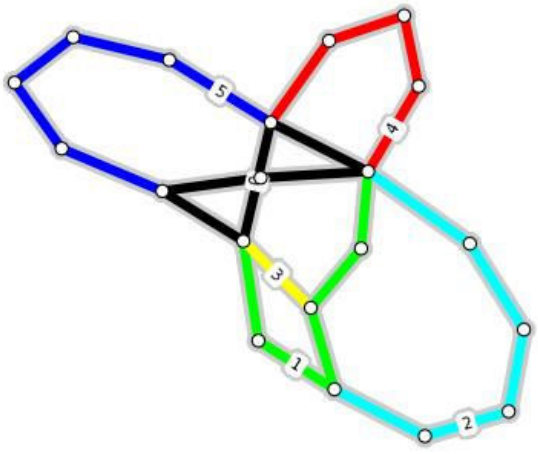
Na vizualizáciu dekompozície grafu bol implementovaný skript v jazyku Python. Tento skript bol skúšaný s verziou Python 2.6.1, a kvôli grafickým výstupom vyžaduje nasledovné knižnice:

- networkX, dostupná na <http://networkx.lanl.gov/>
- matplotlib, dostupná na <http://matplotlib.sourceforge.net/>
- NumPy, dostupná na <http://numpy.scipy.org/>

Samotný skript sa spúšťa nasledovne:

```
python graph.py <vstupný súbor>
```

Vstupný súbor je názov súboru, ktorý vznikol ako výstup programu z kapitoly A.2.2 (resp. štandardný výstup algoritmu BIBOX, ktorý je možné načítať aj programom GraphRec [17]). Výstup algoritmu je znázornený na Obr. 53. Vykreslí sa graf, pričom počiatočný cyklus aj uchá sa farebne zvýraznia a označia číslom. Počiatočný cyklus je vždy čierny a je označený číslom 0. Farby sú priradené napevno, teda uchá s rovnakým číslom majú rovnaké farby, čo je užitočné v prípade, že nás zaujíma porovnanie dvoch rôznych dekompozícií.



Obr. 53. Výstup skriptu na vykreslenie dekompozície.

A.3.2 Vyhodnocovanie výsledkov

Na vyhodnocovanie výsledkov a vytváranie grafických výstupov bolo použité prostredie R, dostupné na <http://www.r-project.org/>. V tejto kapitole sú spomenuté najdôležitejšie funkcie zo súboru *<source/R/functions.R>* na vykreslenie priebehov prehľadávania, či číselné vyhodnotenie výsledkov. Detailnejší opis je v komentári pri každej funkcii. V súbore *<source/R/thesis_data.R>* sú potom uvedené príkazy, ktoré boli použité na vygenerovanie obrázkov a číselných výsledkov pre experimenty v kapitole 11.

Ako prvé je v prostredí R potrebné nastaviť premennú *PATH* na cestu k adresáru s výsledkom experimentov (táto premenná sa nastavuje na začiatku súboru *<source/R/functions.R>*). Ostatné funkcie totiž hľadajú súbory v adresári určenom touto premennou.

Funkcia *read.all.runs(g, h)* načíta údaje o všetkých behoch pre experiment s grafom č. *g* a heuristikou č. *h*, následne ich spojí do jednej tabuľky a vyráta priemernú a minimálnu dĺžku riešenia v každej iterácii. Taktiež vypočíta priemerný čas potrebný na dokončenie každej iterácie.

Funkcia *p(g, hList, ...)* vykreslí priebehy prehľadávania pre daný graf *g* a všetky heuristiky v zozname *hList*. Pre použitie čísla iterácie ako mierky na osi X, treba pridať parameter *use.iterations = TRUE*, pre zobrazenie absolútnych hodnôt na osi Y (namiesto percent), treba pridať parameter *use.percents = FALSE*.

Funkcia *e(g, hList)* vyhodnotí výsledky pre graf *g* a heuristiky *hList*. Na výstupe budú všetky potrebné údaje pre tabuľky v kapitole 11.3, teda celkový čas, priemerné riešenie, najlepšie riešenie, ako aj príslušné p-hodnoty.

Príklad: *e(1, c(1:6))* - vyhodnotí výsledky pre graf G1 a heuristiky H1 až H6.

Funkciou *process.all()* je možné načítať výsledky všetkých experimentov z 2. časti (testovacia sada 190 rôznych grafov).

Číslovanie grafov a heuristík bolo zavedené na zjednodušenie práce s výsledkami experimentov. Jednotlivé súbory s výsledkami tak majú názov vo formáte: „G01_H02_R03.txt“, kde prvé číslo označuje graf, druhé číslo označuje heuristiku a tretie číslo označuje beh. Čísla grafov zodpovedajú číslam uvedeným v kapitole 11.

Číslovanie heuristík je uvedené v Tab. 18, kde pre každé číslo heuristiky je uvedené nastavenie, ktoré reprezentuje.

Tab. 18. Opis číslovania heuristík.

Čísla	Opis nastavení
H0	Náhodné prehľadávanie
H1-6	Heuristika krátkeho počiatočného cyklu. Menili sa hodnoty koeficientu K z množiny $1{,}2; 1{,}5; 2; 5; 10; 100$.
H7-12	Heuristika minimálneho počtu kolízií. Menili sa hodnoty koeficientu K z množiny $1{,}2; 1{,}5; 2; 5; 10; 100$.
H13-18	Kombinácia heuristík. Menili sa hodnoty koeficientu K z množiny $1{,}2; 1{,}5; 2; 5; 10; 100$.
H20-25	Selekčná stratégia, L=10, rozhodovanie podľa priemerného riešenia. Menili sa hodnoty koeficientu K z množiny $1{,}2; 1{,}5; 2; 5; 10; 100$.
H30-33	Selekčná stratégia, K=2, rozhodovanie podľa priemerného riešenia. Menil sa limit pre aktiváciu selekčnej stratégie L z množiny $\{5; 10; 20; 50\}$.
H40-45	Selekčná stratégia, L=10, rozhodovanie podľa najlepšieho riešenia. Menili sa hodnoty koeficientu K z množiny $1{,}2; 1{,}5; 2; 5; 10; 100$.
H50-53	Selekčná stratégia, K=2, rozhodovanie podľa najlepšieho riešenia. Menil sa limit pre aktiváciu selekčnej stratégie L z množiny $\{5; 10; 20; 50\}$.

A.3.3 Spúšťanie experimentov

Na spúšťanie viacerých behov pre konkrétny graf a heuristiku slúži skript v jazyku Python (testovaný bol vo verzii Python 2.6.1). Spúšťa sa nasledovne:

```
run.py <číslo grafu> <číslo heuristiky> <počet behov>
```

Čísla grafov je možné nájsť v kapitole 11.3 a 11.4, čísla heuristík sú uvedené v kapitole A.3.2. Pred spustením však treba v skripte nastaviť premennú *EXE* tak, aby ukazovala na súbor *Multirobot.exe* skompilovaný podľa kapitoly A.2.1.

A.4 Obsah priloženého CD

Štruktúra priloženého CD je nasledovná:

- */document* – táto práca vo formátoch DOCX, DOC a PDF.
- */source/multirobot* – zdrojový kód pre program, ktorý umožňuje spúšťať riešenie problému plánovania cesty pre skupinu robotov, vrátane hľadania alternatívnej dekompozície. V podadresári */bin* sa nachádza spustiteľný súbor pre operačný systém Windows.
- */source/python/graph.py* – skript v jazyku Python, ktorý slúži na vizualizáciu dekompozície.
- */source/python/run.py* – skript v jazyku Python, ktorý slúži na spúšťanie experimentov, ktoré sa skladajú z viacerých behov.
- */source/R/functions.R* – funkcie pre prostredie R, určené na vykresľovanie priebehov a spracovanie číselných výsledkov.
- */source/R/thesis_data.R* – príkazy pre prostredie R použité na získanie údajov a obrázkov do tejto práce.
- */results/decomposition* – výsledky pokusov z kapitoly 6.2, pre všetkých 5 grafov. Sú tu tiež uvedené obrázky aj pre tie grafy, ktoré neboli uvedené v práci. V podadresároch */long* sú uvedené tie riešenia, ktoré boli príliš dlhé a teda sa neparalelizovali.
- */results/mcts* – výsledky pokusov z kapitoly 11. V podadresári */charts* sú obrázky s priebehmi prehľadávania pre všetky grafy z kapitoly 11.3 (nie všetky boli totiž uvedené v práci).

Príloha B Výsledky experimentov

V tejto prílohe sú uvedené výsledky experimentov pre tie grafy, ktorých výsledky neboli uvedené v kapitole 11. Opis údajov v tabuľkách sa nachádza v kapitole 11.2.3. Stĺpec p_{bss} použitý iba pri experimentoch so selekčnou stratégiou predstavuje p-hodnotu pre porovnanie výsledku daného nastavenia s výsledkom bez použitia selekčnej stratégie. Zlepšenie je štatisticky významné, ak $p_{bss} < 0{,}05$.

Tab. 19. Graf G2 - Rôzne koeficienty heuristiky krátkeho počiatočného cyklu.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{orig}	p_{nasl}
K = 1,2	26,3	126,28	90,18	1	0,139
K = 1,5	23,81	121,04	90,18	1	0,017
K = 2	20,37	112,32	83,93	1	0
K = 5	12,26	96,55	72,32	0,075	0,276
K = 10	9,26	94,76	69,64	0,004	0,053
K = 100	7,05 (čas porovnávania)	90,65	66,07	0	-

Tab. 20. Graf G3 - Rôzne koeficienty heuristiky krátkeho počiatočného cyklu.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{orig}	p_{nasl}
K = 1,2	7,99	100,66	76,32	0,626	0,005
K = 1,5	6,97	93,25	72,37	0,001	0,326
K = 2	5,79	91,89	68,42	0,001	0,092
K = 5	3,59	88,2	69,74	0	0,365
K = 10	3,04	87,46	68,42	0	0,23
K = 100	2,67 (čas porovnávania)	85,92	69,74	0	-

Tab. 21. Graf G5 - Rôzne koeficienty heuristiky krátkeho počiatočného cyklu.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{orig}	p_{nasl}
K = 1,2	12,66	96,49	65,26	0,173	0,204
K = 1,5	9,43	92,88	71,58	0,002	0,036
K = 2	8,22	86,88	65,26	0	0,001
K = 5	4,1	78,46	64,21	0	0,251
K = 10	3,09	77,44	58,95	0	0,052
K = 100	2,74 (čas porovnávania)	75,05	64,21	0	-

Tab. 22. Graf G6 - Rôzne koeficienty heuristiky krátkeho počiatočného cyklu.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{orig}	p_{nasl}
K = 1,2	27,73 (čas porovnávania)	117,42	98,17	1	0,011
K = 1,5	31,13	112,03	97,25	1	0,072
K = 2	31,32	108,39	90,37	1	0,046
K = 5	29,98	103,81	72,02	0,967	0,372
K = 10	28,99	102,95	86,24	0,956	0,03
K = 100	28,48	98,59	82,57	0,182	-

Tab. 23. Graf G2 - Rôzne koeficienty heuristiky minimálneho počtu kolízií.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{orig}	p_{nasl}
K = 1,2	27,21	117,32	75	1	0,054
K = 1,5	26,59	111,43	94,64	1	0,246
K = 2	26,4	109,02	75	0,999	0,02
K = 100	22,99 (čas porovnávania)	101,49	70,54	0,738	0,489
K = 10	23,25	101,4	70,54	0,723	0,311
K = 5	25,4	99,79	80,36	0,463	-

Tab. 24. Graf G3 - Rôzne koeficienty heuristiky minimálneho počtu kolízií.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{orig}	p_{nasl}
K = 1,2	8,81	88,03	68,42	0	0,422
K = 1,5	8,98	87,46	68,42	0	0,189
K = 2	9,2	85,09	68,42	0	0,005
K = 10	9,49	78,42	68,42	0	0,303
K = 5	9,47	77,06	68,42	0	0,039
K = 100	8,78 (čas porovnávania)	72,76	68,42	0	-

Tab. 25. Graf G4 - Rôzne koeficienty heuristiky minimálneho počtu kolízií.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{orig}	p_{nasl}
K = 1,2	23,12	108,2	85,29	1	0,35
K = 1,5	23,38	107,02	77,06	0,997	0,027
K = 2	22,35	101,31	88,82	0,789	0,403
K = 5	22,23	100,76	83,53	0,69	0,033
K = 10	21,79	96,39	76,47	0,025	0,14
K = 100	21,17 (čas porovnávania)	94	78,24	0	-

Tab. 26. Graf G5 - Rôzne koeficienty heuristiky minimálneho počtu kolízií.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{orig}	p_{nasl}
K = 1,5	15,07	82,18	57,89	0	0,062
K = 1,2	15,5	79,37	66,32	0	0,133
K = 2	16	77,19	58,95	0	0,083
K = 5	14,96 (čas porovnávania)	74,63	63,16	0	0,262
K = 10	15,26	73,68	61,05	0	0,108
K = 100	15,06	71,79	57,89	0	-

Tab. 27. Graf G2 - Rôzne koeficienty pre kombináciu heuristík.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{orig}	p_{nasl}
K = 1,2	21,67	136,88	96,43	1	0,009
K = 1,5	19,84	125	88,39	1	0
K = 2	17,91	110,48	87,5	1	0
K = 5	9,14	90,51	75	0	0,024
K = 10	6,7	86,16	72,32	0	0,375
K = 100	4,21 (čas porovnávania)	85,68	74,11	0	-

Tab. 28. Graf G3 - Rôzne koeficienty pre kombináciu heuristík.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{orig}	p_{nasl}
K = 1,2	8,82	99,74	78,95	0,456	0,001
K = 1,5	7,13	90,22	68,42	0	0,36
K = 2	5,94	89,34	68,42	0	0
K = 5	3,4	77,15	68,42	0	0,1
K = 10	2,92	74,12	68,42	0	0,077
K = 100	2,11 (čas porovnávania)	71,45	68,42	0	-

Tab. 29. Graf G4 - Rôzne koeficienty pre kombináciu heuristík.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{orig}	p_{nasl}
K = 1,2	23,82	117,98	97,06	1	0,006
K = 1,5	22,9	110	82,35	1	0,025
K = 2	20,99	104,47	81,18	0,994	0
K = 5	13,48	91,12	78,24	0	0
K = 10	9,97	86,73	76,47	0	0,024
K = 100	7,55 (čas porovnávania)	84,2	72,35	0	-

Tab. 30. Graf G5 - Rôzne koeficienty pre kombináciu heuristík.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{orig}	p_{nasl}
K = 1,5	10,8	96,7	71,58	0,087	0,212
K = 1,2	13,42	93,96	67,37	0,01	0,001
K = 2	7,58	84,67	67,37	0	0
K = 5	3,3	71,12	62,11	0	0
K = 10	2,31	67,16	57,89	0	0,013
K = 100	1,72 (čas porovnávania)	64,81	60	0	-

Tab. 31. Graf G6 - Rôzne koeficienty pre kombináciu heuristík.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{orig}	p_{nasl}
K = 1,2	23,04	117,8	95,87	1	0,071
K = 1,5	24,57	112,86	86,7	1	0,044
K = 2	25,52	107,22	90,83	0,999	0
K = 5	22,93	97,75	83,49	0,025	0,008
K = 10	19,6	94,22	83,49	0	0,107
K = 100	15,69 (čas porovnávania)	92,97	88,53	0	-

Tab. 32. Graf G2 - Porovnanie heuristík.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{orig}	p_{nasl}
Bez heuristiky	27	154,32	108,04	1	0,003
Heuristika min. počtu kolízií	22,99	130	83,93	1	0
Heuristika krátkeho poč. cyklu	9,14	90,51	75	0	0,008
Kombinácia heuristík	4,21 (čas porovnávania)	85,68	74,11	0	-

Tab. 33. Graf G3 - Porovnanie heuristík.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{orig}	p_{nasl}
Bez heuristiky	9,22	102,63	72,37	0,848	0
Heuristika min. počtu kolízií	8,78	89,34	68,42	0	0
Heuristika krátkeho poč. cyklu	3,4	77,15	68,42	0	0,005
Kombinácia heuristík	2,11 (čas porovnávania)	71,45	68,42	0	-

Tab. 34. Graf G4 - Porovnanie heuristík.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{orig}	p_{nasl}
Bez heuristiky	27,75	126,98	104,12	1	0
Heuristika min. počtu kolízií	21,17	101,69	84,71	0,87	0
Heuristika krátkeho poč. cyklu	13,48	91,12	78,24	0	0
Kombinácia heuristík	7,55 (čas porovnávania)	84,2	72,35	0	-

Tab. 35. Graf G5 - Porovnanie heuristík.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{orig}	p_{nasl}
Bez heuristiky	15,6	107,93	71,58	0,986	0
Heuristika min. počtu kolízií	15,06	90,46	57,89	0,003	0
Heuristika krátkeho poč. cyklu	3,3	71,12	62,11	0	0
Kombinácia heuristík	1,72 (čas porovnávania)	64,81	60	0	-

Tab. 36. Graf G1 - Rôzny koeficient selekčnej stratégie, rozhodovanie podľa priemerného riešenia.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{bss}	p_{nasl}
K = 2	5,39	77,62	65,73	0,796	0,411
K = 100	5,74	77,34	65,73	0,692	0,455
K = 5	5,63	77,18	65,73	0,642	0,485
K = 1,2	4,54	77,13	65,73	0,668	0,415
K = 1,5	4,31 (čas porovnávania)	76,97	69,23	0,614	0,386
Bez sel. stratégie	4,92	76,69	65,73	0,500	0,371
K = 10	5,67	76,25	65,73	0,371	-

Tab. 37. Graf G2 - Rôzny koeficient selekčnej stratégie, rozhodovanie podľa priemerného riešenia.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{bss}	p_{nasl}
Bez sel. stratégie	4,21	85,68	74,11	0,500	0,375
K = 2	4,23	85,27	70,54	0,375	0,478
K = 10	5,05	85,18	74,11	0,365	0,37
K = 100	4,83	84,58	69,64	0,240	0,442
K = 1,5	4,13	84,32	70,54	0,183	0,441
K = 1,2	4 (čas porovnávania)	84,08	74,11	0,108	0,472
K = 5	5,05	83,96	69,64	0,141	-

Tab. 38. Graf G4 - Rôzny koeficient selekčnej stratégie, rozhodovanie podľa priemerného riešenia.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{bss}	p_{nasl}
K = 100	8,01	84,29	71,18	0,533	0,467
Bez sel. stratégie	7,55	84,2	72,35	0,500	0,485
K = 5	7,78	84,16	72,35	0,485	0,335
K = 10	7,96	83,71	74,12	0,333	0,271
K = 1,5	7,57	82,96	71,18	0,153	0,42
K = 2	7,56	82,73	70,59	0,089	0,335
K = 1,2	7,42 (čas porovnávania)	82,24	71,76	0,051	-

Tab. 39. Graf G5 - Rôzny koeficient selekčnej stratégie, rozhodovanie podľa priemerného riešenia.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{bss}	p_{nasl}
Bez sel. stratégie	1,72	64,81	60	0,500	0,396
K = 100	1,84	64,56	57,89	0,396	0,304
K = 1,5	1,72	64,04	57,89	0,209	0,462
K = 5	1,91	63,93	54,74	0,193	0,335
K = 10	1,78	63,47	57,89	0,076	0,44
K = 1,2	1,65 (čas porovnávania)	63,33	57,89	0,045	0,294
K = 2	1,76	62,77	54,74	0,026	-

Tab. 40. Graf G6 - Rôzny koeficient selekčnej stratégie, rozhodovanie podľa priemerného riešenia.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{bss}	p_{nasl}
K = 5	17,42	93,87	86,24	0,817	0,388
K = 100	16,54	93,64	90,37	0,789	0,286
K = 10	17,35	93,3	90,37	0,599	0,401
Bez sel. stratégie	15,69	93,17	88,53	0,500	0,481
K = 2	16,71	93,13	86,7	0,481	0,262
K = 1,2	14,96 (čas porovnávania)	92,63	85,32	0,224	0,408
K = 1,5	15,83	92,45	80,73	0,137	-

Tab. 41. Graf G1 - Rôzny koeficient selekčnej stratégie, rozhodovanie podľa najlepšieho riešenia.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{bss}	p_{nasl}
K = 100	7,74	80,98	73,43	0,999	0,26
K = 10	10,06	80,05	67,83	0,993	0,348
K = 5	9,41	79,49	65,73	0,985	0,261
K = 2	6,42	78,6	66,43	0,935	0,067
K = 1,5	5,71	76,78	65,73	0,551	0,449
Bez sel. stratégie	4,92 (čas porovnávania)	76,64	65,73	0,500	0,276
K = 1,2	5,22	75,92	65,73	0,276	-

Tab. 42. Graf G2 - Rôzny koeficient selekčnej stratégie, rozhodovanie podľa najlepšieho riešenia.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{bss}	p_{nasl}
K = 10	6,76	87,59	74,11	0,903	0,457
K = 100	5,69	87,41	74,11	0,886	0,114
Bez sel. stratégie	4,21 (čas porovnávania)	85,68	74,11	0,500	0,284
K = 5	6,75	84,85	69,64	0,284	0,239
K = 1,2	4,4	83,66	70,54	0,078	0,166
K = 2	5,22	81,88	70,54	0,012	0,198
K = 1,5	4,71	80,24	69,64	0	-

Tab. 43. Graf G4 - Rôzny koeficient selekčnej stratégie, rozhodovanie podľa najlepšieho riešenia.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{bss}	p_{nasl}
K = 10	10,35	85,33	76,47	0,825	0,494
K = 100	10,15	85,31	71,18	0,804	0,196
Bez sel. stratégie	7,55	84,2	72,35	0,500	0,404
K = 5	9,73	83,88	71,18	0,404	0,298
K = 1,5	8	83,27	78,82	0,168	0,293
K = 2	8,36	82,67	70,59	0,114	0,417
K = 1,2	7,47 (čas porovnávania)	82,43	77,65	0,036	-

Tab. 44. Graf G5 - Rôzny koeficient selekčnej stratégie, rozhodovanie podľa najlepšieho riešenia.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{bss}	p_{nasl}
K = 100	2,06	66,91	57,89	0,962	0,373
K = 10	2,51	66,46	57,89	0,920	0,08
Bez sel. stratégie	1,72 (čas porovnávania)	64,81	60	0,500	0,357
K = 1,5	1,95	64,46	57,89	0,357	0,318
K = 1,2	1,73	63,96	57,89	0,186	0,486
K = 5	2,94	63,93	57,89	0,163	0,411
K = 2	2,22	63,68	54,74	0,147	-

Tab. 45. Graf G6 - Rôzny koeficient selekčnej stratégie, rozhodovanie podľa najlepšieho riešenia.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{bss}	p_{nasl}
K = 10	18,29	94,43	86,24	0,969	0,391
K = 2	18,15	94,17	90,37	0,932	0,493
K = 5	19,57	94,16	87,61	0,925	0,108
K = 1,5	16,46	93,2	90,37	0,521	0,479
Bez sel. stratégie	15,69	93,17	88,53	0,500	0,416
K = 1,2	14,94 (čas porovnávania)	93,03	90,37	0,416	0,159
K = 100	15,65	92,11	79,36	0,354	-

Tab. 46. Graf G1 - Rôzne hodnoty limitu aktivácie selekčnej stratégie, rozhodovanie podľa priemerného riešenia.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{bss}	p_{nasl}
L = 10	5,32	77,51	65,73	0,779	0,417
L = 5	5,37	77,25	65,73	0,683	0,332
L = 50	4,99	76,71	65,73	0,524	0,476
Bez sel. stratégie	4,92 (čas porovnávania)	76,64	65,73	0,500	0,373
L = 20	5,16	76,22	65,73	0,373	-

Tab. 47. Graf G3 - Rôzne hodnoty limitu aktivácie selekčnej stratégie, rozhodovanie podľa priemerného riešenia.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{bss}	p_{nasl}
L = 5	2,07	73,11	68,42	0,823	0,491
L = 20	2,08	73,07	68,42	0,816	0,482
L = 50	2,06 (čas porovnávania)	72,98	68,42	0,813	0,295
L = 10	2,28	72,02	68,42	0,634	0,366
Bez sel. stratégie	2,11	71,45	68,42	0,500	-

Tab. 48. Graf G4 - Rôzne hodnoty limitu aktivácie selekčnej stratégie, rozhodovanie podľa priemerného riešenia.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{bss}	p_{nasl}
Bez sel. stratégie	7,55	84,2	72,35	0,500	0,367
L = 20	7,41	83,84	77,65	0,367	0,347
L = 5	7,74	83,49	75,29	0,239	0,212
L = 10	7,56	82,73	70,59	0,109	0,423
L = 50	7,17 (čas porovnávania)	82,53	72,35	0,058	-

Tab. 49. Graf G5 - Rôzne hodnoty limitu aktivácie selekčnej stratégie, rozhodovanie podľa priemerného riešenia.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{bss}	p_{nasl}
L = 5	1,76	65,54	57,89	0,772	0,228
Bez sel. stratégie	1,72	64,81	60	0,500	0,348
L = 20	1,69	64,46	61,05	0,348	0,091
L = 50	1,65 (čas porovnávania)	63,26	57,89	0,035	0,315
L = 10	1,77	62,77	54,74	0,026	-

Tab. 50. Graf G6 - Rôzne hodnoty limitu aktivácie selekčnej stratégie, rozhodovanie podľa priemerného riešenia.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{bss}	p_{nasl}
L = 50	14,19 (čas porovnávania)	93,33	90,37	0,609	0,402
L = 5	15,81	93,17	89,91	0,500	0,5
Bez sel. stratégie	15,69	93,17	88,53	0,500	0,481
L = 10	15,86	93,13	86,7	0,481	0,196
L = 20	14,61	92,58	86,24	0,138	-

Tab. 51. Graf G1 rôzne hodnoty limitu aktivácie selekčnej stratégie, rozhodovanie podľa najlepšieho riešenia.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{bss}	p_{nasl}
L = 10	6,3	78,6	66,43	0,935	0,211
L = 5	6,82	77,41	65,73	0,710	0,29
Bez sel. stratégie	4,92 (čas porovnávania)	76,64	65,73	0,500	0,468
L = 20	6,33	76,55	65,73	0,468	0,137
L = 50	5,31	75,24	65,73	0,123	-

Tab. 52. Graf G3 rôzne hodnoty limitu aktivácie selekčnej stratégie, rozhodovanie podľa najlepšieho riešenia.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{bss}	p_{nasl}
L = 5	2,8	76,01	68,42	0,992	0,427
L = 10	2,77	75,61	68,42	0,983	0,442
L = 20	2,75	75,31	68,42	0,981	0,173
L = 50	2,15	73,46	68,42	0,877	0,123
Bez sel. stratégie	2,11 (čas porovnávania)	71,45	68,42	0,500	-

Tab. 53. Graf G4 rôzne hodnoty limitu aktivácie selekčnej stratégie, rozhodovanie podľa najlepšieho riešenia.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{bss}	p_{nasl}
Bez sel. stratégie	7,55 (čas porovnávania)	84,2	72,35	0,500	0,417
L = 5	8,58	83,96	75,29	0,417	0,405
L = 20	8,05	83,65	71,18	0,336	0,4
L = 50	7,55	83,31	72,35	0,220	0,306
L = 10	8,43	82,67	70,59	0,114	-

Tab. 54. Graf G5 rôzne hodnoty limitu aktivácie selekčnej stratégie, rozhodovanie podľa najlepšieho riešenia.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{bss}	p_{nasl}
L = 20	2,16	65,09	61,05	0,617	0,383
Bez sel. stratégie	1,72 (čas porovnávania)	64,81	60	0,500	0,357
L = 5	2,29	64,46	57,89	0,357	0,407
L = 50	1,81	64,21	54,74	0,265	0,323
L = 10	2,22	63,68	54,74	0,147	-

Tab. 55. Graf G6 rôzne hodnoty limitu aktivácie selekčnej stratégie, rozhodovanie podľa najlepšieho riešenia.

	Celkový čas (s)	Priemer (%)	Minimum (%)	p_{bss}	p_{nasl}
L = 10	18,15	94,11	90,37	0,951	0,242
L = 20	17,08	93,64	90,37	0,891	0,442
L = 50	15,49 (čas porovnávania)	93,56	90,37	0,865	0,5
L = 5	18,63	93,56	88,53	0,808	0,192
Bez sel. stratégie	15,69	92,97	88,53	0,500	-