Masarykova univerzita
Fakulta informatiky

Realizace Rabinovy hry na konečných grafech

Bakalářská práce

Filip Bártek

Brno, podzim 2011

Prohlášení

Prohlašuji, že tato bakalářská práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Filip Bártek

Vedoucí práce: prof. RNDr. Ivana Černá, CSc.

Poděkování

Za poznámky k předmětu a obsahu této práce děkuji vedoucí práce prof. RNDr. Ivaně Černé, CSc., autorce použitého MATLABového řešiče Rabinových her RNDr. Janě Tůmové a svému otci Ing. Jiřímu Bártkovi, CSc.

Shrnutí

Předmětem práce je implementace a experimentální srovnání dvojice algoritmů pro řešení Rabinových her. Vyřešení Rabinovy hry zahrnuje určení množiny vyhrávajících stavů hry, tj. ověření splnění Rabinovy podmínky na těchto stavech, a syntézu řídící strategie z těchto stavů.

Klíčová slova

Rabinova hra, řešič, syntéza řídící strategie, Rabin game, solver, control strategy synthesis, C++

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Od výpočetních systémů zpravidla požadujeme, aby jejich chování splňovalo určité požadavky. Jeden ze způsobů specifikace požadavků na chování systému nabízejí tzv. Rabinovy podmínky. Tento model postihuje konečněstavové interaktivní systémy, u kterých hodnotíme nekonečné běhy systému (jde o tzv. liveness podmínky [4, s. 7]). Rabinova podmínka formalizuje požadavek typu „na alespoň jeden nekonečně častý požadavek systém odpoví pouze konečně často“.[1] Zajímá nás, které (resp. jestli všechny) stavy systému splňují danou Rabinovu podmínku ve smyslu, že každý běh z takového stavu splňuje danou podmínku.

Rabinovy podmínky nacházejí uplatnění v řešení podmínek specifikovaných nedeterministickými Büchiho automaty. Nedeterministický Büchiho automat lze pomocí tzv. Safrovy konstrukce [6, 10] převést na (deterministický) Rabinův automat. Vyřešení tohoto automatu vede k vyřešení původního Büchiho automatu.

Rabinova hra je zobecněním Rabinova automatu – stavy systému jsou v ní rozdělené mezi dva hráče, z nichž jeden (Adam, reprezentující (výsledný) program) usiluje o splnění Rabinovy podmínky, zatímco u jeho protivníka (Eva, reprezentující okolní prostředí) počítáme s tím, že usiluje o porušení podmínky. Řešením Rabinovy hry zjišťujeme, které stavy systému splňují danou Rabinovu podmínku za předpokladu, že hráč Adam napomáhá jejímu splnění, a recept pro Adama k takovému napomáhání.[2] Jde tedy o syntézu jednoduchého programu v rámci zadaném řešenou Rabinovou hrou. Recepty pro Adama takto získané jsou bezpaměťové [9, s. 2], tedy Adamovo rozhodnutí závisí vždy pouze na stavu, ve kterém se systém nachází.

1. Duální podmínka Rabinovy podmínky se nazývá Streettova a má (jako přímá realizace tzv. strong fairness [4, s. 12]) širší uplatnění.
2. Rabinův automat odpovídá Rabinově hře, kde všechny vrcholy patří hráči Eva.

1.1 Cíle

1.1.1 Implementace řešičů

Hlavním cílem této práce je implementace algoritmů pro řešení Rabinových her, tj. výpočet výherního regionu a vyhrávající strategie pro hráče Adam, představených v článcích [9] a [5]. Omezím se stejně jako tyto články na Rabinovy hry nad konečnými grafy (viz definici 1) a konečnými podmínkami (viz definici 7). Při tvorbě implementace se zaměřím na časovou efektivitu.

1.1.2 Implementace uživatelského rozhraní pro řešiče

Řešiče zastřeším prakticky použitelným programem pro operační systém Windows.

1.1.3 Implementace rozhraní pro MATLABový řešič

Druhotným cílem je vytvoření rozhraní mezi mým zastřešujícím programem pro řešení Rabinových her a MATLABovou implementací řešiče Rabinových her RNDr. Jany Tůmové.

1.1.4 Srovnání řešičů

Na závěr experimentálně srovnám časovou efektivitu jednotlivých řešičů.

1.2 Struktura práce

V kapitole 2 definuji základní pojmy problematiky řešení Rabinových her.

V kapitole 3 specifikuji použitý programovací jazyk a popisuji datové struktury společné implementacím všech řešičů.

V kapitole 4, resp. 5, popisuji algoritmus pro řešení Rabinových her představený v článku [9], resp. [5], a vysvětluji zajímavá specifika jeho implementace.

V kapitole 6 popisuji řešič Rabinových her implementovaný ve skriptovacím jazyce programu MATLAB RNDr. Janou Tůmovou a

komunikační rozhraní mezi tímto řešičem a mým zastřešujícím programem `rgs.exe`.

V kapitole 7 popisuji použitou metodiku srovnání řešičů a představuji výsledky experimentů.

V kapitole 8 stručně shrnuji výsledek této práce a navrhuji možnosti dalšího vývoje.

2 Definice

Terminologii čerpám zejména z [6, 5, 9].

2.1 Rabinova hra

2.1.1 Hra dvou hráčů

Definice 1. Hra dvou hráčů je uspořádaná trojice $G_2 = (Gr, P, p)$, kde

- $Gr = (V, E)$ je orientovaný graf,
- P je uspořádaná dvojice hráčů a
- $p : V \to P$ je rozdělení vrcholů grafu Gr mezi hráče z P.

V dalším textu budu uvažovat pouze hry nad konečnými grafy, jak jsem již předeslal v kapitole 1. Počet vrcholů grafu rozebírané hry budu značit symbolem n, tedy $n = |V|$.

V dalším textu budu hráče rozebírané hry dvou hráčů nazývat $Adam$ a Eva ve smyslu $P = (Adam, Eva)$.

Tabulka 2.1: Označení hráčů hry dvou hráčů

Hráč	Výherní podmínka	[5]	[9]	C++	MATLAB
$Adam$	Rabinova	Adam	Rabin	`true`	[protagonist]
Eva	Streettova	Eve	Streett	`false`	adv[ersary]

Pro snadnější pochopení významu hry dvou hráčů si představme žeton, který v každém okamžiku běhu hry leží na některém z jejích vrcholů. V průběhu hry se potom tento žeton přesouvá mezi vrcholy podle určitých pravidel. Jeden přesun žetonu odpovídá jednomu tahu. Přesuny (resp. tahy) jsou diskrétní.

2.1.2 Rabinova výherní podmínka

Definice 2. Běh nad grafem $Gr = (V, E)$ je posloupnost vrcholů $\rho \in V^* \cup V^\omega$, $\rho = (v_0, v_1, \dots)$, pro kterou platí:

- $\forall i \in \mathbb{N}_0.(v_i, v_{i+1}) \in E$ a
- $\rho \in V^* \Rightarrow v_{|\rho|-1}$ nemá následníka v Gr.

Tedy běh (jako posloupnost vrcholů) musí procházet graf v souladu s přechodovou funkcí danou relací E a smí skončit pouze ve vrcholu, který nemá žádného následníka.

Jinými slovy se žeton v každém tahu z libovolného vrcholu v smí přesunout pouze na následníka vrcholu v a pokud takový následník existuje, žeton se v tom tahu přesunout musí. Posloupnost vrcholů navštívených takovýmto postupem se nazývá během.

Zejména pokud má každý vrchol v Gr alespoň jednoho následníka, žeton se v každém běhu nad Gr přesune nekonečněkrát.[1]

Definice 3. $runs(Gr)$ je množina všech běhů nad grafem Gr.

Definice 4. Vítězná podmínka pro hru dvou hráčů $G_2 = (Gr, P, p)$ je funkce $win : runs(Gr) \to P$, která každému běhu ρ nad Gr přiřazuje hráče z P vyhrávajícího (hru G_2 pro) běh ρ.

Definice 5. Nechť Σ je abeceda. **Množinu nekonečně častých znaků** $inf : \Sigma^* \cup \Sigma^\omega \to 2^\Sigma$ slova $(a_0, a_1, \dots)$ definujeme následovně:

$$inf((a_0, a_1, \dots)) = \{a \in \Sigma | \exists^\omega i \in \mathbb{N}_0 : a_i = a\} \tag{2.1}$$

Množina nekonečně častých znaků běhu je tedy množina právě těch vrcholů, kterými žeton v tomto běhu projde nekonečněkrát.

Zejména platí $\rho \in V^* \Rightarrow inf(\rho) = \emptyset$, tedy množina nekonečně častých znaků konečného běhu je prázdná.

Protože $runs(Gr = (V, E)) \subseteq V^* \cup V^\omega$, lze funkci inf zúžit na $runs(Gr)$. V dalším textu budu funkci inf bez přejmenovávání zužovat na $runs(Gr)$ pro různé grafy Gr zřejmé z kontextu.

Definice 6. Základní Rabinova vítězná podmínka pro hru dvou hráčů $G_2 = ((V, E), (Adam, Eva), p)$ je přesně určená uspořádanou dvojicí $c = (g, r)$, kde

1. Při řešení Rabinových her se běžně uvažují pouze hry, ve kterých má každý vrchol alespoň jednoho následníka [5], mj. protože vrcholy bez následníků jsou řešitelné triviálně (viz 5.2.1). Já jsem se rozhodl ve své implementaci uvažovat i hry s vrcholy bez následníků, abych nekladl žádnou netriviální podmínku na vstupní hru a zejména abych umožnil řešení (všech) náhodných her generovaných jednoduchým způsobem popsaným v sekci 7.1.

- $g \subseteq V$ je množina vrcholů žádoucích pro hráče $Adam$ a
- $r \subseteq V$ je množina vrcholů nežádoucích pro hráče $Adam$:

$$win_c(\rho) = Adam \Leftrightarrow inf(\rho) \cap g \neq \emptyset \wedge inf(\rho) \cap r = \emptyset \tag{2.2}$$

Tedy $Adam$ vyhrává podle $c = (g, r)$ běh ρ, pokud žeton v tomto běhu nekonečněkrát stane na některém z vrcholů z g a zároveň stane na každém z vrcholů z r jen konečněkrát.

Definice 7. (Obecná) **Rabinova vítězná podmínka** pro hru dvou hráčů G_2 je přesně určená množinou C základních Rabinových podmínek pro G_2:

$$win_C(\rho) = Adam \Leftrightarrow \bigvee_{c \in C} win_c(\rho) \tag{2.3}$$

Tedy $Adam$ vyhrává podle $C = \{(g_0, r_0), (g_1, r_1), \dots\}$ běh ρ, pokud žeton v tomto běhu nekonečněkrát stane na některém z vrcholů z g_i pro některé i a zároveň stane na každém z vrcholů z odpovídající r_i jen konečněkrát.[2]

V dalším textu budu uvažovat pouze konečné Rabinovy vítězné podmínky, jak jsem již předeslal v kapitole 1. Počet základních Rabinových podmínek obecné Rabinovy podmínky rozebírané hry budu značit symbolem k, tedy $k = |C|$.

Definice 8. Rabinova hra je uspořádaná dvojice $G = (G_2, C)$, kde

- G_2 je hra dvou hráčů a
- C je Rabinova vítězná podmínka pro G_2.

2. Hráč $Adam$ vyhrává Rabinovu hru právě při splnění splnění Rabinovy vítězné podmínky. Duální vítězná podmínka se nazývá Streettova [9, s. 2] – hráč Eva tedy vyhrává právě při splnění odpovídající Streettovy podmínky. Na základě toho mohou být hráči Rabinovy (nebo Streettovy) hry označeni jako Rabin a Streett [9, s. 2].

2.2 Řešení Rabinovy hry

Definice 9. Nechť $G_2 = ((V, E), (tento, onen), p)$ je hra dvou hráčů. Množiny vrcholů patřících jednotlivým hráčům pojmenujeme V_{tento} a V_{onen} podle následujících pravidel:

$$V_{tento} = \{v \in V | p(v) = tento\} \tag{2.4}$$

$$V_{onen} = \{v \in V | p(v) = onen\} \tag{2.5}$$

Tedy V_{hrac} je množina vrcholů patřících hráči $hrac$.

Definice 10. Strategie hráče $tento$ (resp. $onen$) hry dvou hráčů $G_2 = ((V, E), (tento, onen), p)$ je funkce $s : runs(G_2) \cap \{(v_0, v_1, \ldots, v_m) \in V^*.V_{tento}$ (resp. $V^*.V_{onen}$)$\} \to V$, pro kterou platí: $s((v_0, v_1, \ldots, v_m)) \in \{v \in V | (v_c, v) \in E \wedge v_c = v_m\}$.

Tedy strategie danému hráči přesně určuje, na kterého následníka má poslat žeton v kterémkoliv okamžiku hry, kdy je na tahu (tedy kdy žeton leží na některém z vrcholů tomuto hráči patřících).

Definice 11. Nechť s je strategie hráče $hrac$. **Běh** $\rho = (v_0, v_1, \ldots)$ je **s-konformní** právě tehdy, když $\forall i \in \mathbb{N}_0.p(v_i) = hrac \Rightarrow s((v_0, v_1, \ldots, v_i)) = v_{i+1}$.

Tedy běh je konformní vzhledem ke strategii hráče $hrac$, pokud hráč $hrac$ pokaždé, když je na tahu, pošle žeton na následníka určeného touto strategií (podle dosavadního průběhu hry).

Definice 12. Strategie s hráče $hrac$ je **vyhrávající** z výchozího vrcholu v právě tehdy, když hráč $hrac$ vyhrává každý běh, který začíná ve vrcholu v a je s-konformní.

Tedy pokud se hráč řídí strategií, která je vyhrávající, jistě zvítězí.

Definice 13. Vrchol je **výherní** pro hráče $hrac$ právě tehdy, když existuje strategie vyhrávající pro hráče $hrac$ z tohoto (výchozího) vrcholu.

Každý vrchol Rabinovy hry je výherní pro právě jednoho z hráčů.[9, s. 3] Tedy vrchol výherní pro hráče $Adam$ je proherním pro hráče Eva a vrchol výherní pro hráče Eva je proherním pro hráče $Adam$.

Definice 14. Výherní region $W_{hrac} \subseteq V$ hráče $hrac$ je množina všech výherních vrcholů hráče $hrac$.

Tedy pokud žeton začíná hru na vrcholu z W_{hrac}, $hrac$ může jistě zvítězit.

Věta 1. *Nechť $\bar{W}_{hrac} \subseteq W_{hrac}$, tedy pro každý z vrcholů z $\bar{W}_{hrac}$ existuje vyhrávající strategie hráče $hrac$. Potom existuje strategie hráče $hrac$, která je vyhrávající pro každý z vrcholů z $\bar{W}_{hrac}$.*

Důkaz. Pro $\bar{W}_{hrac} = \emptyset$ platí triviálně (požadovanou vlastnost má každá strategie).

Nechť $\bar{W}_{hrac} \neq \emptyset$. Nechť $v \in \bar{W}_{hrac}$ – obecný. Nechť s_v je strategie vyhrávající pro hráče $hrac$ a výchozí vrchol v. Nechť $v_{fix} \in \bar{W}_{hrac}$ – libovolný určitý. Nechť $s_{\bar{W}_{hrac}}$ je strategie definovaná následovně:

$$s_{\bar{W}_{hrac}}((v_0, v_1, \ldots, v_m)) = \begin{cases} s_{v_0}((v_0, v_1, \ldots, v_m)) & \text{pro } v_0 \in \bar{W}_{hrac} \\ s_{v_{fix}}((v_0, v_1, \ldots, v_m)) & \text{pro } v_0 \notin \bar{W}_{hrac} \end{cases} \tag{2.6}$$

Potom $s_{\bar{W}_{hrac}}$ je strategie hráče $hrac$, která je vyhrávající z každého z vrcholů z $\bar{W}_{hrac}$. □

Zejména platí pro $\bar{W}_{hrac} = W_{hrac}$, tedy existuje strategie, která je vyhrávající z každého výherního vrcholu. Říkejme takové strategii dále prostě vyhrávající strategie (bez upřesnění výchozího vrcholu).

Definice 15. (Úplné) **řešení Rabinovy hry** $G = ((V, E), (Adam, Eva), p), C)$ je uspořádaná dvojice (W_{Adam}, s_{Adam}), kde

- $W_{Adam} \subseteq V$ je $Adam$ův výherní region a
- s_{Adam} je $Adam$ova vyhrávající strategie.

(Úplným) vyřešením Rabinovy hry míníme určení $Adam$ova výherního regionu a $Adam$ovy vyhrávající strategie.

Definice 16. Částečné řešení Rabinovy hry je $Adam$ův výherní region.

Částečným vyřešením Rabinovy hry míníme určení $Adam$ova výherního regionu.

Definice 17. Bezpaměťová[3] **strategie** je strategie s, pro kterou platí: $\forall(v_0, v_1, \ldots, v_m), (\bar{v}_0, \bar{v}_1, \ldots, \bar{v}_l) \in dom(s).v_m = \bar{v}_l \Rightarrow s((v_0, v_1, \ldots, v_m)) = s((\bar{v}_0, \bar{v}_1, \ldots, \bar{v}_l))$

Tedy bezpaměťová strategie rozhoduje o následujícím tahu hráče pouze na základě vrcholu, na kterém leží žeton (a nezávisle na vrcholech navštívených v předchozích tazích).

Sjednocením konečných běhů, které končí ve stejném vrcholu, získáme stručnější a praktičtější reprezentaci bezpaměťové strategie pro (libovolného) hráče $hrac$: $s : V_{hrac} \to V$.

Podle [9, s. 2] existuje pro hráče $Adam$a každé Rabinovy hry bezpaměťová výherní strategie. Protože Rabinovu hru řeším právě pro hráče $Adam$a a protože články [9] a [5] řeší pro $Adam$a v Rabinových hrách právě (jen) bezpaměťové strategie, budu ve zbytku práce uvažovat jen bezpaměťové strategie (reprezentované podle předchozího odstavce).

Není potřeba, aby strategie určovala následníky proherních vrcholů, protože pokud běh následující z proherního vrcholu bude konformní vzhledem k výherní strategii hráče Eva z tohoto vrcholu (která jistě existuje, protože je ten vrchol proherní), bude tento běh proherní pro $Adam$a, tedy žádný výběr následníka proherního vrcholu nezajistí, že běh bude výherní. Je tedy možné reprezentaci bezpaměťové výherní strategie dále zúžit na $V_{hrac} \cap W_{hrac}$.

3. Bezpaměťová strategie bývá někdy zvána poziční [5, s. 3].

3 Implementace

3.1 Programovací jazyk

K implementaci jsem použil programovací jazyk C++, protože od něj očekávám velkou rychlost výpočtu ve výsledném programu[1], která může být vzhledem k velké teoretické časové náročnosti použitých algoritmů (viz 4.1.4 a 5.1.3) potřebná.

3.2 Realizace množin

Množiny, přesněji podmnožiny určitých konečných množin, jsou reprezentovány tzv. bitsety, tj. vektory (poli) pravdivostních příznaků (bitů) s přidruženými operátory základních množinových operací.

Každý bitset má určitou délku, tedy počet pravdivostních příznaků, které obsahuje. Bitset délky l nativně reprezentuje podmnožinu množiny čísel $\{0, 1, \ldots, l-1\}$. V pořadí m-tý pravdivostní příznak bitsetu (pro $0 \leq m < l$) určuje přítomnost čísla m v množině reprezentované tímto bitsetem.

Pro reprezentaci podmnožiny konečné množiny A stačí prvky A očíslovat indexy $\{0, 1, \ldots, |A|-1\}$. V mé implementaci takto čísluji vrcholy grafu hry (bitsety reprezentující podmnožiny V mají délky n) a základní Rabinovy podmínky hry (bitsety reprezentující podmnožiny C mají délky k).

Pro implementaci bitsetů jsem použil třídu `boost::dynamic_bitset` z knihovny Dynamic Bitset [11] z balíku Boost [2]. Vybral jsem ji na základě studie [8], která `boost::dynamic_bitset` vyhodnotila jako časově nejefektivnější variantu z několika běžně používaných C++ implementací bitsetu (včetně standardního `std::vector<bool>`).

Vzhledem k jednoduchosti operací na bitsetech použitých v mých implementacích řešičů považuji za nepravděpodobné, že jsou třídou `boost::dynamic_bitset` realizovány s polynomiální časo-

1. Programovací jazyk C++ jsem si vybral na základě doporučení vedoucí práce. Pro ne nezbytně zcela relevantní srovnání „výkonu“ programovacích jazyků příznivé pro C++ viz např. [3].

vou složitostí. Vzhledem k velké asymptotické složitosti implementovaných algoritmů, viz 4.1.4 a 5.1.3, jsem se rozhodl od časových náročností operací na bitsetech abstrahovat a považovat je pro účely analýzy časové složitosti řešičů za konstantní.

3.3 Rabinova hra

Rabinova hra G je reprezentována objektem třídy `RabinGame`, který obsahuje jako atributy:

- hru dvou hráčů (G_2) a
- Rabinovu vítěznou podmínku (C).

3.3.1 Hra (dvou hráčů)

Hra dvou hráčů je reprezentována vektorem (délky n) vrcholů v atributu `RabinGame::game_` typu `std::vector<Vertex>`.

Vrchol (orientovaného grafu hry dvou hráčů)

Vrchol v je reprezentován objektem třídy `Vertex`, který obsahuje jako atributy:

- množinu následníků daného vrcholu ($\{w \in V | (v, w) \in E\}$) a
- (pravdivostní) příznak hráče ($p(v)$).

Množina následníků Množina následníků vrcholu je (jako podmnožina V) reprezentována bitsetem délky n v atributu `Vertex::successors_` typu `RabinGame::BitsetType`. Implicitně je prázdná.

Příznak hráče Příznak hráče je reprezentován pravdivostní hodnotou v atributu `Vertex::player_` typu `bool`. Nabývá hodnoty pravda (`true`) ve vrcholech hráče $Adam$ a hodnoty nepravda (`false`) ve vrcholech hráče Eva. Implicitně je nepravdivý.

3.3.2 Rabinova vítězná podmínka

Rabinova vítězná podmínka je reprezentována vektorem (délky k) základních Rabinových (vítězných) podmínek v atributu `RabinGame::condition_` typu `std::vector<RabinStreettPair>`.

Základní Rabinova podmínka

Základní Rabinova podmínka c je reprezentována objektem třídy `RabinStreettPair`, který obsahuje jako atributy:

- množinu žádoucích vrcholů (g) a
- množinu nežádoucích vrcholů (r).

Množina žádoucích vrcholů Množina žádoucích vrcholů je reprezentována bitsetem délky n v atributu `RabinStreettPair::g_` typu `RabinGame::BitsetType`. Implicitně je prázdná.

Množina nežádoucích vrcholů Množina nežádoucích vrcholů je reprezentována bitsetem délky n v atributu `RabinStreettPair::r_` typu `RabinGame::BitsetType`. Implicitně je prázdná.

3.4 Řešení Rabinovy hry

3.4.1 Výherní region

Výherní region je reprezentován bitsetem délky n. Implicitně je prázdný.

3.4.2 Výherní strategie

Výherní strategie je reprezentována vektorem (délky n) čísel z $\{0, 1, \ldots, n\}$. Číslo j na pozici i značí pokyn k přechodu z vrcholu i do vrcholu j. Ve správně utvořené strategii jsou čísla na pozicích odpovídajících vrcholům z $V_{Adam} \cap W_{Adam}$ různá od n a čísla na ostatních pozicích rovna n. Implicitně jsou všechna čísla rovna n (což odpovídá implicitnímu (prázdnému) výhernímu regionu).

4 Pitermanův řešič

Pitermanův řešič jsem implementoval podle článku [9].

4.1 Pitermanův algoritmus

4.1.1 Pomocná definice

Definice 18. Nechť $W \subseteq V$. Množina **řídících předchůdců**[1] množiny W je množina všech vrcholů v takových, že:

- $P(v) = Adam \Rightarrow \exists w \in W.(v, w) \in E$
- $P(v) = Eva \Rightarrow \forall w \in V.(v, w) \in E \Rightarrow w \in W$

Tedy řídící předchůdci množiny W jsou právě ty vrcholy, ze kterých se žeton v jednom kroku dostane do množiny W, pokud tomu Adam chce.

4.1.2 Popis algoritmu

Pitermanův algoritmus je rekurzivní prostřednictvím funkce `Rabin`. Vstupními argumenty funkce `Rabin` jsou množina základních Rabinových podmínek `Set`, množina nevyloučených vrcholů `seqnr` (v prvním volání plná) a množina výherních vrcholů `right` (v prvním volání prázdná).

`Rabin` vždy počítá pouze na množině nevyloučených vrcholů `seqnr`.

V každém volání si `Rabin` vybere základní Rabinovu podmínku $(g, r) \in$ `Set` a spočítá rekurzivním voláním `Rabin(Set`$-(g, r)$`, seqnr`$-r$`, right)` předvýherní vrcholy, tj. vrcholy, které vyhrávají podle ostatních podmínek v `Set` v podhře ochuzené o r. Poté do množiny výherních vrcholů `right` přidá řídící předchůdce takto spočtených předvýherních vrcholů a opakuje volání.

Jakmile toto vnořené volání nevrátí žádné nové vrcholy, algoritmus vrátí množinu výherních vrcholů `right` do původního stavu

1. V [9] vystupuje množina řídících předchůdců pod jmény „control predecessor“ a `cpred`.

a přidá do ní ty vrcholy z g, které jsou zároveň řídícími předchůdci spočtených předvýherních vrcholů, a opakuje výpočet množiny předvýherních vrcholů. Tak činí opakovaně, dokud se vypočtená množina předvýherních vrcholů neustálí – takto ustálenou množinu přidá k výsledku (návratové hodnotě) a opakuje celý výpočet pro další základní Rabinovu podmínku.

Zastřešující funkce `main_Rabin` opakuje volání `Rabin` nad plnou množinou základních Rabinových podmínek (`Set` $= C$), plnou množinou nevyloučených vrcholů (`seqnr` $= V$) a množinou výherních vrcholů (v prvním volání prázdnou) obohacenou o řídící předchůdce množiny předvýherních vrcholů spočtených předchozím voláním `Rabin`, dokud `Rabin` vrací různé výsledky. Jakmile se výsledek `Rabin` ustálí, je prohlášen za úplné řešení a vrácen funkcí `main_Rabin`.

Pro přesný popis Pitermanova algoritmu pro řešení Rabinových her a důkaz jeho korektnosti nahlédněte do [9].

4.1.3 Řešení strategie

Vyhrávající strategie pro hráče Adama je indukována voláními operace `cpred` (pro výpočet množiny řídících předchůdců) na množinách předvýherních vrcholů vracených voláními funkce `Rabin`.

4.1.4 Časová složitost

Tento algoritmus řeší Rabinovy hry v čase $O(mn^{2k}k!)$, kde $m = |E|$ [9, s. 16].

4.1.5 Store optimalizace

V [9, kap. 6] Piterman představuje optimalizaci výše uvedeného algoritmu. Ta spočívá v ukládání průběžně vypočtených množin předvýherních vrcholů. Uložené množiny lze za určitých podmínek použít jako výchozí v dalších výpočtech množin předvýherních vrcholů.

Pitermanův algoritmus se store optimalizací řeší Rabinovy hry v čase $O(n^{(k+1)}k!)$ a prostoru $O(n^{(k+1)}k!)$.

4.2 Implementace

Implementace je realizovaná třídou `PitermanRabinGameSolver` definovanou v souboru `main/PitermanRabinGameSolver.cpp`.

4.2.1 Store optimalizace

Pro realizaci tzv. store proměnných, tj. proměnných, které slouží k ukládání množin předvýherních vrcholů s vlastnostmi popsanými v [9, kap. 6], jsem použil knihovnu `tree.hh` [7].

5 Hornův řešič

Hornův řešič jsem implementoval podle článku [5].

5.1 Hornův algoritmus

5.1.1 Pomocné definice

Definice 19. Nechť $G = (((V, E), P, p), C)$ je Rabinova hra a $\bar{V} \subseteq V$. **Podhra** hry G indukovaná množinou $\bar{V}$ je hra $G_{\bar{V}} = (((\bar{V}, \bar{E}), P, \bar{p}), \bar{C})$, na které je přirozeným způsobem definovaná zúžená relace přechodu $\bar{E}$, zúžené rozdělení vrcholů mezi hráče $\bar{p}$ a zúžená Rabinova podmínka $\bar{C}$.

Dále budu podhru indukovanou množinou $\bar{V}$ stručně označovat „podhra $\bar{V}$“.

Definice 20. Nechť $\bar{V} \subseteq V$, $W \subseteq \bar{V}$ a $hrac \in P$. $hrac$ův **atraktor** množiny W v podhře $\bar{V}$ (značíme $Attr^{\bar{V}}_{hrac}(W)$) je množina všech vrcholů, ze kterých dokáže $hrac$ v podhře $\bar{V}$ dostat žeton do vrcholu ve W v konečně mnoha tazích, aniž by mu v tom mohl jeho oponent zabránit.

5.1.2 Popis algoritmu

Algoritmus je realizován funkcí $SolveSubgame$, která počítá výherní region podhry. Jejími vstupními argumenty jsou podhra $\bar{V} \subseteq V$ a množina základních Rabinových podmínek $\bar{C} \subseteq C$. Vrací Adamův výherní region podhry $\bar{V}$ s Rabinovou podmínkou $\bar{C}$.

$SolveSubgame(\bar{V}, \bar{C})$ si nejdříve vybere základní Rabinovu podmínku $(g, r) \in \bar{C}$. Poté odebráním vrcholů $Attr^{\bar{V}}_{Eva}(r)$ z $\bar{V}$ získá podhru G_i.[1] Z G_i dále odebere $Attr^{G_i}_{Adam}(g)$, čímž získá podhru H_i. Podhru H_i vyřeší zavoláním $SolveSubgame$ s Rabinovou podmínkou $\bar{C}$ ochuzenou o (g, r) (tedy voláním $SolveSubgame(H_i, \bar{C} - (g, r))$).

Komplement výsledku tohoto volání, tj. Evin výherní region v podhře H_i s podmínkou $\bar{C} - (g, r)$, a jeho Evin atraktor v podhře G_i je

1. V [5] index i značí vybranou základní Rabinovu podmínku (g, r).

odebrán z G_i. Algoritmus poté opakuje výpočet pro tuto novou G_i. Takto činí opakovaně, dokud se G_i neustálí.

Ustálená G_i a její Adamův $\bar{V}$-atraktor jsou výherní pro Adama ve $\bar{V}$ podle podmínky $\bar{C}$ (a jsou tedy přidány k výsledku). Při tomto výpočtu atraktoru G_i se ukládá do globální tabulky Adamova vítězná strategie – odpovídá právě způsobu dosažení G_i z vrcholu v atraktoru. Algoritmus odebere $Attr^{\bar{V}}_{Adam}(G_i)$ z $\bar{V}$ a nechá novým voláním $SolveSubgame$ vyřešit takto vzniklou podhru – tentokrát však s plnou podmínkou C (tedy zavolá $SolveSubgame(\bar{V} - G_i, C)$). Výsledek tohoto volání přidá ke svému výsledku.

Tím končí zpracování podmínky (g, r) ve volání $SolveSubgame(\bar{V}, \bar{C})$. Funkce si zvolí další podmínku z $\bar{C}$ a výpočet pro ni opakuje.

Výsledný výherní region nasbíraný ve výpočtech pro jednotlivé základní podmínky je vrácen jako návratová hodnota $SolveSubgame(\bar{V}, \bar{C})$.

Zavoláním funkce $SolveSubgame(V, C)$ získáme řešení plné hry.

Pro přesný popis Hornova algoritmu pro řešení Rabinových her a důkaz jeho korektnosti nahlédněte do [5].

5.1.3 Časová složitost

Tento algoritmus řeší Rabinovy hry v čase $O(n^{2k}k!)$ [5, s. 11].

5.2 Implementace

Implementace je realizovaná třídou `HornRabinGameSolver` definovanou v souboru `main/HornRabinGameSolver.cpp`.

5.2.1 Zobecnění na všechny Rabinovy hry

Hornův algoritmus funguje zaručeně správně pouze na hrách, ve kterých má každý vrchol alespoň jednoho následníka. Aby má implementace dokázala řešit všechny Rabinovy hry, odebírá před výpočtem ze hry množinu všech vrcholů, které nemají žádného následníka, a její Evin atraktor.

Věta 2. *Vrchol, který nemá žádného následníka, je proherní pro Adama.*

Důkaz. Každý běh, který vychází z takového vrcholu, je konečný. Tedy má prázdnou množinu nekonečně častých znaků. Tedy nesplňuje žádnou základní Rabinovu podmínku. Tedy prohrává. □

Věta 3. *Evin atraktor množiny proherních vrcholů je proherní pro Adama.*

Důkaz. Takovýto Evin atraktor z definice (viz definici 20) odpovídá Evině výherní strategii z dané množiny. □

Věta 4. *Výherní region v podhře $\bar{V}$ vzniklé odebráním Evina atraktoru množiny proherních vrcholů v plné hře V ze hry V je výherní i v plné hře V.*

Důkaz. Nechť $G = (((V, E), P, p), C)$ je řešená Rabinova hra. Nechť $V_E \subseteq V$ je množina proherních vrcholů hry G. Nechť $V_0 = Attr^G_{Eva}(V_l)$. Nechť s_0 je odpovídající atraktorová strategie, tj. strategie, podle které Eva dokáže ve hře G dostat žeton z vrcholu ve V_0 do vrcholu ve V_E. Nechť $\bar{G}$ je podhra G indukovaná množinou vrcholů $\bar{V} = V - V_0$.

- Nechť $v \in \bar{V}$ je vrchol výherní v G. Nechť s_v je vyhrávající strategie z vrcholu v ve hře G.

  Nechť s_v-konformní běh ve hře G zavede žeton z vrcholu v do vrcholu ve V_0. Potom Eva dokáže v následujícím běhu hry G použitím strategie s_0 dostat žeton do proherního vrcholu. Tedy s_v není vyhrávající v G – spor.

  Každý s_v-konformní běh ve hře G se tedy vyhýbá vrcholům ve V_0. Tedy s_v je vyhrávající strategie z vrcholu v i v $\bar{G}$.

- Nechť $\bar{v} \in \bar{V}$ je vrchol výherní v $\bar{G}$. Nechť $\bar{s}_{\bar{v}}$ je vyhrávající strategie z vrcholu $\bar{v}$ ve hře $\bar{G}$.

  Nechť $\bar{s}_{\bar{v}}$-konformní běh ρ ve hře $\bar{G}$ zavede žeton z vrcholu $\bar{v}$ do vrcholu $\bar{v_0}$, který má následníka ve V_0.

  Nechť $P(\bar{v}_0) = Adam$. Potom $\bar{s}_{\bar{v}}(\bar{v}_0) \in \bar{V}$, protože $\bar{s}_{\bar{v}}$ je Adamova strategie v $\bar{G}$.

  Nechť $P(\bar{v}_0) = Eva$. Potom Eva dokáže dostat žeton z vrcholu $\bar{v}_0$ do vrcholu ve V_0, tedy $\bar{v}_0 \in Attr^G_{Eva}(V_E)$, tedy $\bar{v}_0 \notin \bar{V}$ – spor.

  Každý $\bar{s}_{\bar{v}}$-konformní běh ve hře G se tedy vyhýbá vrcholům ve V_0. Tedy $\bar{s}_{\bar{v}}$ je vyhrávající strategie z vrcholu $\bar{v}$ i v G.

Tedy vrchol ve $\bar{V}$ je výherní v G právě tehdy, když je výherní v $\bar{G}$. □

6 MATLABový řešič

6.1 MATLABová implementace Hornova algoritmu

RNDr. Jana Tůmová implementovala ve skriptovacím jazyce programu MATLAB algoritmus pro řešení Rabinových her představený v [5].[1] Tuto MATLABovou implementaci řešiče jsem použil jako vzor pro vytvoření rozhraní mezi mým C++ programem pro řešení Rabinových her, resp. jeho abstraktní třídou `RabinGame`, a arbitrárním řešičem Rabinových her pracujicím v MATLABu, který formátuje vstup a výstup stejně a klade na vstup nejvýše tak přísná omezení, jako zmíněná implementace Jany Tůmové.

6.1.1 Omezení vstupu

Tůmové MATLABová implementace dokáže řešit právě takové Rabinovy hry, které splňují následující podmínky:

1. Každý následník Adamova vrcholu je Evin vrchol.
2. Každý následník Evina vrcholu je Adamův vrchol.
3. Každý Evin vrchol má nejvýše jednoho předchůdce.
4. Rabinova vítězná podmínka zahrnuje v množinách žádoucích i nežádoucích vrcholů pouze Adamovy vrcholy.
5. Počet Eviných vrcholů je n_{Sigma}-násobkem počtu Adamových vrcholů, kde n_{Sigma} je nejméně počet následníků Adamova vrcholu s nejvyšším počtem následníků.
6. Každý vrchol má nejméně jednoho následníka.

S výjimkou šestého jsou tato omezení nucená formátem vstupních dat MATLABové implementace.

---

1. Jde o tentýž algoritmus, který jsem implementoval jako Hornův řešič – viz kapitolu 5.

6.2 Převod řešené hry do MATLABového formátu

6.2.1 Formát vstupních dat MATLABového řešiče

Vstupní hra MATLABové implementace je zadaná jako uspořádaná sedmice $(Q\_list, Sigma, Q\_adv, Q\_adv\_list, Cond, delta, delta\_adv)$, kde:

- Q_list je vektor $(1, 2, \ldots, n)$ indexů Adamových vrcholů
- $Sigma$ je vektor $(1, 2, \ldots, n_{Sigma})$ indexů akcí, tj. možných přechodů z každého Adamova vrcholu
- Q_adv matice rozměrů $|Q\_list \times Sigma| \times 2$ nad indexy, která v lexikograficky uspořádaných řádcích obsahuje všechny dvojice z $Q\_list \times Sigma$
- Q_adv_list je vektor $(1, 2, \ldots, |Q\_adv|)$ indexů Eviných vrcholů, kde $Q\_adv\_list_i$ odpovídá i-tému řádku Q_adv
- $Cond$ je Rabinova vítězná podmínka zadaná maticí rozměrů $2 \times k$, kde každý sloupec obsahuje jednu základní Rabinovu podmínku:
  - první položka je vzestupně uspořádaný vektor indexů žádoucích Adamových vrcholů
  - druhá položka je vzestupně uspořádaný vektor indexů nežádoucích Adamových vrcholů
- $delta$ je matice rozměrů $|Q\_list| \times |Sigma|$ nad pravdivostními hodnotami, která určuje přechodovou funkci z Adamových vrcholů následujícím způsobem:

  $delta_{va} = 1 \Leftrightarrow$ akce a z Adamova vrcholu v (tedy přechod do Evina vrcholu $(v, a) \in Q\_adv$) je umožněna
- $delta\_adv$ je matice rozměrů $|Q\_adv\_list| \times |Q\_list|$ nad pravdivostními hodnotami, která určuje přechodovou funkci z Eviných vrcholů následujícím způsobem:

  $delta\_adv_{vw} = 1 \Leftrightarrow$ přechod z Evina vrcholu $v \in Q\_adv\_list$ do Adamova vrcholu $w \in Q\_list$ je umožněn

6.2.2 Převod

Mějme Rabinovu hru $G = ((Gr = (V = \{v_0, v_1, \ldots, v_{n-1}\}, E), P = (Adam, Eva), p), C = \{(g_0, r_0), (g_1, r_1), \ldots, (g_{k-1}, r_{k-1})\})$. Nechť $V_E \subseteq V$ je množina všech vrcholů, které nemají žádného následníka. Převod na instanci MATLABového řešiče lze provést následovně:

- $Q\_list \cong V \cup \{v_n\}$, kde $v_n \notin V$
- $Sigma \cong V \cup \{a_n\}$, kde $a_n \notin V$
- Q_adv vyplývá z Q_list a $Sigma$, viz 6.2.1
- Q_adv_list vyplývá z Q_adv, viz 6.2.1
- $Cond$ je matice rozměrů $2 \times k$, kde i-tý sloupec (indexovaný od nuly) obsahuje:
  - první složka $\cong g_i$ (jako vzestupně uspořádaný vektor indexů odpovídajících vrcholů z Q_list)
  - druhá složka $\cong r_i$ (jako vzestupně uspořádaný vektor indexů odpovídajících vrcholů z Q_list)
- $delta \cong$ matice rozměrů $Q\_list \times Sigma$, kde:

$$\begin{gathered} delta_{va} = 1 \Leftrightarrow \\ ((v = v_n \vee v \in V_E \vee P(v) = Eva) \wedge a = a_n) \vee \\ (P(v) = Adam \wedge (v, a) \in E) \end{gathered}$$

- $delta\_adv \cong$ matice rozměrů $(Q\_list \times Sigma) \times Q\_list$, kde:

$$\begin{gathered} delta\_adv_{(v,a)w} = 1 \Leftrightarrow \\ ((a \neq a_n \vee v = v_n \vee P(v) = Adam \vee v \in V_E) \wedge a = w) \vee \\ ((v \neq v_n \wedge P(v) = Eva \wedge a = a_n) \wedge (v, w) \in E) \end{gathered}$$

Tedy každý vrchol v_i původní hry a přidaný vrchol v_n zastoupím skupinou jednoho Adamova vrcholu Q_list_i (nazývejme Q_list_i obrazem v_i) a $n+1$ Eviných vrcholů (odkazovaných $n+1$ akcemi) nové hry. Nová Rabinova podmínka je zadaná množinami (resp. vektory) obrazů vrcholů původní Rabinovy podmínky.

delta

Prvních n akcí odpovídá možným následníkům Adamových vrcholů v původní hře. Maticí $delta$ je akce $Sigma_j$ z vrcholu Q_list_i umožněna pouze v případě, že v_i v původní hře je Adamův a má následníka v_j.

Akce a_n je speciální akce umožněná právě v obrazech Eviných vrcholů a v obrazech vrcholů, které nemají v původní hře žádného následníka (vysvětlení viz níže).

delta_adv

Evinou odpovědí na akci, kódovanou v $delta\_adv$, je přechod do Adamova vrcholu. Evina odpověď na akci závisí jak na akci samotné, tak na vrcholu, ze kterého byla akce zavolána (nazývejme jej zdrojovým). Proto je počet Eviných vrcholů v nové hře roven součinu počtů Adamových vrcholů a akcí.

- Pokud je zdrojový vrchol obrazem Adamova vrcholu, je akce žádostí o přechod do vrcholu a Eva tuto žádost poslechne, protože celá skupina zdrojového vrcholu a jeho akcí (tj. Eviných vrcholů nové hry) zastupuje (Adamův) vrchol původní hry. Tedy Eva v odpovědi na akci $Sigma_j$ přejde do vrcholu Q_list_j.

  Toto pravidlo je shodou okolností v pořádku i pro akci $Sigma_n$, která směřuje do vrcholu Q_list_n, který se neobjevuje v původní hře – vysvětlení viz níže.

- Pokud je zdrojový vrchol Q_list_i obrazem Evina vrcholu, podle definice $delta$ je jediná umožněná akce $Sigma_n$. Eva si může v odpovědi na tuto akci vybrat libovolný z Adamových vrcholů, které jsou obrazy následníků vrcholu v_i v původní hře.

Q_list_n **a** $Sigma_n$

Pokud v v původní hře neměl žádného následníka, je z jeho obrazu v nové hře umožněna pouze akce $Sigma_n$. Z Eviných odpovědí na tuto akci je umožněn právě přechod do Q_list_n. Z vrcholu Q_list_n je taktéž umožněna pouze akce $Sigma_n$ a na ni pouze odpověď Q_list_n. Tedy jakmile žeton vstoupí do skupiny s vrcholem

Q_list_n, už ji nikdy neopustí a navždy bude cyklit mezi vrcholy Q_list_n a $(Q\_list_n, Sigma_n)$. Poněvadž Q_list_n nepatří do množiny žádoucích vrcholů žádné ze základních Rabinových podmínek v $Cond$, běh, který přes Q_list_n projde, je zákonitě prohrávající pro Adama. Protože jsou obrazy vrcholů, které v původní hře nemají žádného následníka, nucené přejít do Q_list_n, jsou v nové hře proherní, což je v souladu s proherností vrcholů bez následníků (viz větu 2). Důležité je, že po této úpravě má každý vrchol v nové hře nejméně jednoho následníka, čímž je naplněn vstupní požadavek MATLABové implementace č. 6.

Tento převod jsem implementoval ve třídě `MatlabRabinGameSolver` – pro podrobnosti implementace nahlédněte do souboru `main/MatlabRabinGameSolver.cpp`.

6.2.3 Nárůst velikosti instance

Tento převod zřejmě způsobuje kvadratický nárůst velikosti instance vzhledem k n. Fakt, že nelze převod (bez částečného vyřešení hry) realizovat prostorově asymptoticky efektivněji, dokládá hra, kde Gr je úplný graf a všechny vrcholy patří Adamovi.

Tento převod nebere v potaz vnitřní strukturu hry a i hry, které splňují vstupní podmínky MATLABové implementace, kvadraticky zvětší. Tím výrazně vzroste i asymptotická časová náročnost výpočtu. Vzhledem k tomu, že význam MATLABového řešiče je v rámci této bakalářské práce spíše okrajový, jsem se spokojil s takto neefektivním převodem. Pro přiblížení rychlosti běhu MATLABové implementace nad hrami pro ni určenými použiji v odpovídajících tabulkách a grafech pro MATLABový řešič odmocninnou škálu.

6.3 Výstupní data MATLABové implementace

6.3.1 Formát

MATLABová implementace vrací uspořádanou desetici $(Q\_list, Sigma, Q\_adv, Q\_adv\_list, Cond, delta, delta\_adv, W, W\_adv, pi)$, kde:

- počátečních sedm složek kóduje řešenou hru – viz 6.2.1

- W je vektor indexů z Q_list právě těch Adamových vrcholů, ze kterých vyhrává Adam
- W_adv je vektor indexů z Q_list právě těch Adamových vrcholů, ze kterých vyhrává Eva
- pi je vektor indexů akcí ze $Sigma$, kde i-tou složkou pi je index akce (přechodu), kterou má Adam provést ve vrcholu s indexem i (platí $|pi| = n + 1$)

W, W_adv a Q_list můžeme chápat jako množiny. Potom musí ve správném řešení platit $W \cup W\_adv = Q\_list$ a $W \cap W\_adv = \emptyset$.

Také musí platit, že i-tá složka pi, kde $i \in W$, obsahovat platný index vrcholu. Ostatní složky ovšem mohou obsahovat i neplatné indexy (mimo rozsah vektoru $Sigma$).

6.3.2 Převod

Vrcholy z W převedeme na vrcholy z V postupem opačným k postupu uvedenému v 6.2.2. U všech vrcholů kromě Q_list_n je tento převod možný. Vrchol Q_list_n se ve výherním regionu správného řešení nikdy neobjeví (argumentaci viz výše).

Analogickým mapováním provedeme převod z pi na vektor délky n vrcholů z V. Není potřeba brát v potaz hodnotu pi_n, protože Q_list_n nikdy není výherní. V pi se ve složce odpovídající výhernímu vrcholu nikdy neobjeví akce $Sigma_n$, protože vede do proherního vrcholu (vysvětlení viz výše).

7 Srovnání praktické časové náročnosti řešičů

7.1 Metodika

7.1.1 Testovací hry

Pro porovnání řešičů jsem vytvořil program `bench.exe`. Ten při běžném použití postupně generuje náhodné Rabinovy hry různých velikostí, nechává je vyřešit vybranými řešiči a zapisuje časy běhů řešičů na jednotlivých hrách do tabulky. Generování her je parametrizováno přes n, k a `seed` generátoru náhodných čísel `rand` použitého jako zdroj náhodnosti.

Pro dané n, k a `seed` je hra generována po jednotlivých bitech ve vnitřních bitsetech hry. Protože řešič píši pro obecné Rabinovy hry, zvolil jsem pro účely porovnání řešičů pro všechny bity uniformní rozdělení pravděpodobnosti. Tedy mezi dvěma vrcholy vygenerované náhodné hry je hrana s pravděpodobností $\frac{1}{2}$ a do množiny g, resp. r základní Rabinovy podmínky patří daný vrchol s pravděpodobností $\frac{1}{2}$.

Velikost testovaných her pokryla rozmezí od triviálních ($n = 0$ nebo $k = 0$) po největší, které bylo lze na použitém počítači vyřešit do řádově jednotek minut.

7.1.2 Varianty řešičů

U každého řešiče jsem testoval zvlášť vyřešení pouhého výherního regionu („částečné řešení“) a vyřešení výherního regionu s odpovídající (Adamovou) vyhrávající strategií („plné řešení“).

U Pitermanova řešiče jsem testoval zvlášť variantu bez store optimalizace a variantu se store optimalizací (viz sekci 4.1.5).

7.1.3 Testovací prostředí

Testování jsem provedl na počítači specifikovaném v tabulce 7.1.3.

Čas řešení jsem měřil v tzv. clock ticks, tj. nejmenší časové jednotce měřitelné pomocí standardních knihoven jazyka C++ (v tomto případě funkcí `clock()`). Na použitém počítači odpovídá 1000 clock ticks času jedné sekundy.

Tabulka 7.1: Specifikace počítače použitého k testování

Procesor	Intel Core2 Duo E7300 2,66 GHz 1000 clock ticks za sekundu
Paměť (RAM)	4,00 GB
Operační systém	Windows 7 Professional 64-bit

Program jsem zkompiloval překladačem G++ z balíku GCC [1].

7.2 Výsledky

7.2.1 MATLABový řešič

Při testování výkonu jsem zjistil, že MATLABová implementace vrací pro některé hry chybné výsledky. Jako prakticky nepoužitelnou jsem ji proto ze srovnání vynechal. V digitální příloze jsem zahrnul soubor `program/matlab/b2-2-1.mat` s nejmenší v MATLABové implementaci chybně řešenou hrou, se kterou jsem se setkal.[1]

7.2.2 Pitermanův a Hornův řešič

V grafech označuji řešiče z důvodů prostorových omezení zkrácenými jmény. Vysvětlení těchto jmen najdete v tabulce 7.2.

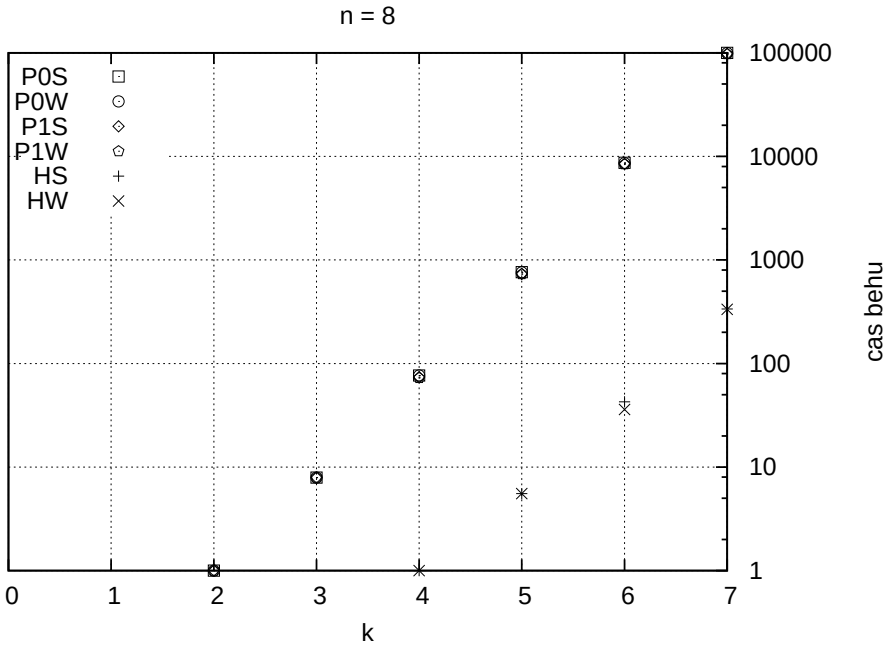
Časy řešení pro $n = 8$

V grafu 7.1 jsou zachyceny časy běhů řešičů nad hrami pro fixní $n = 8$ a k pokrývající interval od 0 do 7. Pro každé k bylo vygenerováno 16 náhodných her. Vynesené časy běhů jsou průměry přes

1. Na správně vyřešených hrách MATLABový řešič vykazoval výrazně horší výsledky, než ostatní řešiče. Pravděpodobně se na tom významnou měrou podepsal použitý převod hry do vstupního formátu MATLABové implementace, který způsobuje kvadratický nárůst velikosti hry vzhledem k n (viz 6.2.3), takže ani v případě, že by MATLABová impelementace byla korektní, nejspíš by nebylo její porovnání zajímavé.

Tabulka 7.2: Označení řešičů v grafech

Popis	Označení
Pitermanův řešič plného řešení bez store optimalizace	PS0
Pitermanův řešič částečného řešení bez store optimalizace	PW0
Pitermanův řešič plného řešení se store optimalizací	PS1
Pitermanův řešič částečného řešení se store optimalizací	PW1
Hornův řešič plného řešení	HS
Hornův řešič částečného řešení	HW



Obrázek 7.1: Průměrné časy běhů řešičů na 16 náhodných Rabinových hrách pro $n = 8$

tyto šestnáctice. Naměřené časy běhů jsou vyneseny v logaritmické škále.

Fixní $n = 8$ jsem vybral experimentálně jako dostatečně velké pro „zajímavé“ chování vzhledem ke k.

V grafu jde vidět, že na použitých hodnotách n a k se jednotlivé varianty Pitermanova řešiče a jednotlivé varianty Hornova řešiče výkonem téměř neliší.[2]

Dále lze pozorovat, že Hornovy řešiče mají proti Pitermanovým výrazně lepší časy běhů.

Absenci hodnot v levé části grafu přisuzuji nízké rozlišovací schopnosti použitého systému pro měření času – pro $k \leq 1$ byl na všech řešičích a pro $k \leq 3$ na Hornových řešičích naměřen nulový čas.

Časy řešení pro $k = 4$

V grafu 7.2 jsou zachyceny časy běhů řešičů nad hrami pro fixní $k = 4$ a n pokrývající interval od 1 do 10000 v exponenciálních skocích (o základu 10). Pro každé n bylo vygenerováno 16 náhodných her. Vynesené časy běhů jsou průměry přes tyto šestnáctice. n a naměřené časy běhů jsou vynesené v logaritmických škálách.

Fixní $k = 4$ jsem vybral experimentálně jako dostatečně velké pro „zajímavé“ chování vzhledem k n.

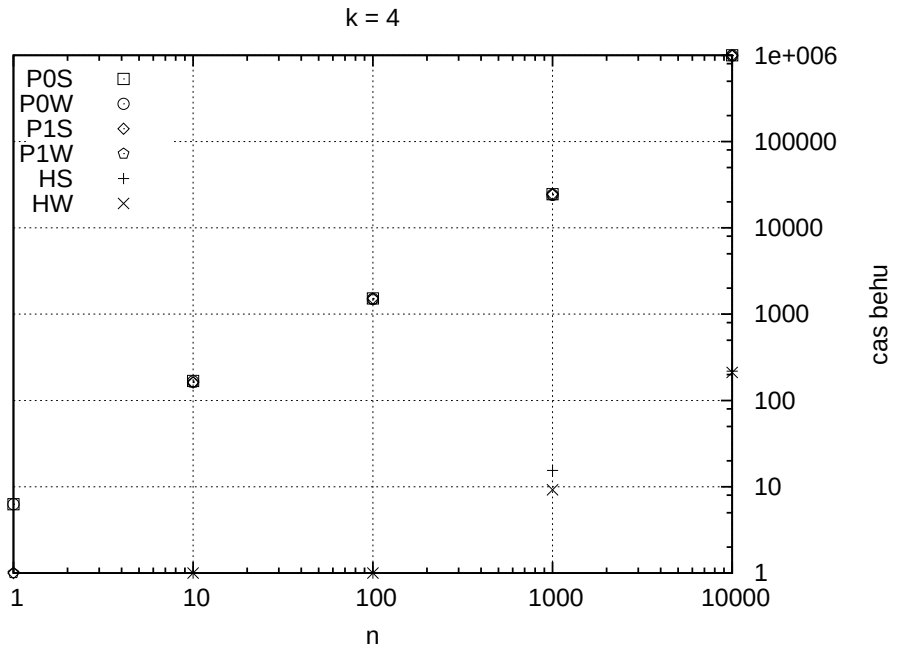
Graf naznačuje podobné relativní vlastnosti řešičů, jako graf 7.1, zejména podobnost jednotlivých variant řešiců a výrazný rozestup výkonu řešičů ve prospěch Hornova.

Dalo se očekávat, že čas řešení bude velmi podobný pro plné a částečné řešení, protože řešení strategie v žádném z použitých algoritmů nezvyšuje asymptotickou složitost. Tento předpoklad analýza potvrdila. Pro další testy jsem se proto rozhodl používat pouze ty varianty řešičů, které řeší plné řešení.

Překvapivým výsledkem je poměrně špatný výkon store-optimalizovaného Pitermanova řešiče vzhledem k Pitermanovu řešiči bez optimalizace. Zejména se nezdá, že by vykazoval asymptoticky menší časovou náročnost. Zůstává otázkou, jestli je tato podobnost zapříčiněna neefektivní implementací nebo výběrem testovacích her.

---

2. Jednotlivé varianty řešičů jsou si výsledky tak podobné, že v grafu téměř splývají. Shluky bodů vynesené výše na ose času běhu patří Pitermanovým řešičům a shluky bodů vynesené níže patří Hornovým řešičům.



Obrázek 7.2: Průměrné časy běhů řešičů na 16 náhodných Rabinových hrách pro $k = 4$

8 Závěr

Úspěšně jsem implementoval algoritmy pro řešení Rabinových her představené v článcích [9] a [5]. Pro tyto implementace jsem vytvořil uživatelské rozhraní ve formě aplikace `rgs.exe` pro operační systém Windows. Implementoval jsem rozhraní mezi touto aplikací a softwarem MATLAB, prostřednictvím kterého lze řešit Rabinovy hry pomocí řešiče implementovaného ve skriptovacím jazyce softwaru MATLAB. Vytvořil jsem aplikaci `bench.exe`, která umožňuje dávkové testování časové efektivity řešičů na náhodných hrách.

V provedených testech se ukázal jako časově výrazně efektivnější Hornův řešič.

Asymptoticky paměťově značně náročná optimalizace Pitermanova algoritmu, která zaručuje výrazné snížení asymptotické časové složitosti, se v praxi na rychlosti řešení testovacích her neprojevila.

Zajímavými možnostmi dalšího vývoje řešiče jsou rozšíření na Streettovy hry a paralelizace výpočtů.

Mohlo by být zajímavé porovnat Pitermanův a Hornův řešič s MATLABovým řešičem speciálně na hrách, které splňují jeho vstupní podmínky, s efektivní metodou převodu (bez nárůstu velikosti instance, viz 6.2.3).

Literatura

[1] GCC, 2010. Verze 4.5.1. [Online; cit. 2012-01-02] Dostupné z WWW: http://gcc.gnu.org/.

[2] Boost, 11 2011. Verze 1.48.0. [Online; cit. 2012-01-02] Dostupné z WWW: http://www.boost.org/.

[3] Christopher W. Cowell-Shah. Nine language performance round-up: Benchmarking math & file i/o, 01 2004. [Online; cit. 2012-01-01] Dostupné z WWW: http://www.osnews.com/story/5602/Nine_Language_Performance_Round-up_Benchmarking_Math_File_I_O/.

[4] Keijo Heljanko. Networks and processes: Safety, liveness, and fairness, 12 2003. [Online; cit. 2012-01-01] Dostupné z WWW: http://www.fmi.uni-stuttgart.de/szs/teaching/ws0304/nets/slides14.pdf.

[5] Florian Horn. Streett games on finite graphs. [Online; cit. 2011-11-24] Dostupné z WWW: http://liafa.jussieu.fr/~horn/publications.html, 2005.

[6] Mojmír Křetínský. Automaty nad nekonečnými slovy, 12 2002. [Online; cit. 2012-01-01] Dostupné z WWW: http://www.fi.muni.cz/usr/kretinsky/automaty.pdf.

[7] Kasper Peeters. tree.hh, 08 2011. Verze 2.81. [Online; cit. 2012-01-02] Dostupné z WWW: http://tree.phi-sci.com/.

[8] Vreda Pieterse, Derrick G. Kourie, Loek Cleophas, and Bruce W. Watson. Performance of C++ bit-vector implementations. In *SAICSIT '10 Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists*, pages 242–250. ACM, 2010. [Online; cit. 2012-01-02] Dostupné z WWW: http://dl.acm.org/citation.cfm?id=1899530.

[9] Nir Piterman and Amir Pnueli. Faster solutions of Rabin and Streett games. In *Proceedings of the 2006 21st Annual*

IEEE Symposium on Logic in Computer Science, pages 275–284. IEEE Computer Society, IEEE Computer Society, 2006. [Online; cit. 2011-11-24] Dostupné z WWW: `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.134.1574`.

[10] Shmuel Safra. On the complexity of ω-automata. In *29th Annual Symposium on Foundations of Computer Science (FOCS 1988)*, pages 319–327, 1988. [Online; cit. 2012-01-01] Dostupné z WWW: `http://www.computer.org/portal/web/csdl/doi/10.1109/SFCS.1988.21948`.

[11] Jeremy Siek, Chuck Allison, and Gennaro Prota. boost::dynamic_bitset, 2008. [Online; cit. 2012-01-02] Dostupné z WWW: `http://www.boost.org/doc/libs/1_48_0/libs/dynamic_bitset/dynamic_bitset.html`.

Rejstřík

A Digitální příloha

A.1 Obsah

V tabulce A.1 najdete výčet adresářů a vybraných souborů digitální přílohy s jejich popisy.

Tabulka A.1: Obsah digitální přílohy

Cesta	**Význam**
`program`	Soubory programů
`program/matlab`	Skripty MATLABového řešiče
`program/matlab/*.m`	Skripty MATLABového řešiče
`program/matlab/b2-2-1.mat`	Soubor s nesprávně řešenou hrou
`program/src`	Soubory programů
`program/src/bench`	Soubory programu `bench.exe`
`program/src/bench/*.hpp`	Hlavičkové soubory
`program/src/bench/*.cpp`	Zdrojové soubory
`program/src/bench/bench.exe`	*Program `bench.exe`*
`program/src/main`	Soubory programu `rgs.exe`
`program/src/main/*.hpp`	Hlavičkové soubory
`program/src/main/*.cpp`	Zdrojové soubory
`program/src/main/rgs.exe`	*Program `rgs.exe`*
`program/src/env_local.bat`	Lokální konfigurační skript
`program/src/env_user.bat`	Uživatelský konfigurační skript
`program/bench.bat`	*Zástupce `bench.exe` bez MATLABu*
`program/bench_m.bat`	*Zástupce `bench.exe` s MATLABem*
`program/build.bat`	*Hlavní kompilační skript*
`program/rgs.bat`	*Zástupce `rgs.exe` bez MATLABu*
`program/rgs_m.bat`	*Zástupce `rgs.exe` s MATLABem*
`text`	Soubory textu práce
`text/graph`	Zdrojové soubory grafů
`text/graph/*.dat`	Tabulky s daty z experimentů
`text/graph/*.plt`	Skripty pro generování grafů
`text/tex`	Dílčí zdrojové soubory
`text/tex/*.tex`	Dílčí zdrojové soubory
`text/bachelor.bib`	Zdrojový soubor bibliografie

Tabulka A.1

Cesta	Význam
`text/bachelor.pdf`	*Text práce*
`text/bachelor.tex`	Hlavní zdrojový soubor textu práce
`text/bachelor.sty`	Zdrojový soubor stylu
`text/fi-logo.mf`	Logo Fakulty informatiky
`text/fit12.clo`	Dílčí soubor stylu fithesis2
`text/fithesis2.cls`	Styl fithesis2 pro sazbu
`text/rgs.pdf`	*Příručka programu* `rgs.exe`
`text/rgs.tex`	Zdrojový soubor příručky `rgs.exe`
`readme.txt`	Nápověda k digitální příloze

A.2 Nápověda

Nápovědu k programu `rgs.exe` najdete v kapitole B.

Nápovědu k programu `bench.exe` získáte zavoláním programu `bench.exe` s parametrem `-H` nebo `--help`.

Programy `rgs.exe` a `bench.exe` jsou ve verzi dodané v digitální příloze sestavené bez MATLABového řešiče, aby byly kompatibilní s více systémy. Pokud chcete používat MATLABový řešič, konzultujte sekci B.2.

B Uživatelská příručka programu `rgs.exe`

B.1 Rychlý start

Zadáním `rgs.exe game4-2.txt` do příkazové řádky ve složce s programem `rgs.exe` necháte `rgs.exe` vyřešit hru uloženou v souboru `game4-2.txt` a vypsat řešení na obrazovku.

B.2 Instalace

Program `rgs.exe` sestavíte zavoláním skriptu `build_all.bat` ve složce s programem (implicitně `../program/src/main`). Předem je nutné nastavit proměnné prostředí úpravou skriptu `../program/src/env_local.bat`. Doporučuje se nastavit i uživatelské proměnné kompilace úpravou skriptu `../program/src/env_user.bat`.

Ke kompilaci `rgs.exe` potřebujete mít nainstalované knihovny Boost, CxxTL a tree.hh. Ke kompilaci včetně rozhraní pro MATLAB potřebujete mít nainstalovaný program MATLAB.

B.3 Vstupně-výstupní formát

B.3.1 Vstup

Pokud zavoláte `rgs.exe` s parametrem `filename`, kde `filename` nezačíná pomlčkou (viz tabulku B.4), program se pokusí načíst Rabinovu hru ze souboru `filename` a vyřešit ji.

Příklad obsahu vstupního souboru s Rabinovou hrou (jde o úvodní část ukázkového souboru `game4-2.txt`):

```
4
2
0000 0
0101 1
1010 0
1111 1
0011 1100
0100 0001
```

Na prvním řádku je počet vrcholů (nazývejme jej n, tedy v případě `game4-2.txt` platí $n = 4$). Na druhém řádku je počet základních Rabinových podmínek (nazývejme jej k, tedy v případě `game4-2.txt` platí $k = 2$).

Vrcholy jsou indexované od nuly, tedy nabývají indexů z $\{0, 1, \ldots, n-1\}$. V případě `game4-2.txt` nabývají indexů od 0 do 3.

Na následujících řádcích se objevují podmnožiny množiny vrcholů. Každá taková množina je zadaná posloupností čísel `0` a `1` délky n. `1`, resp. `0` na pozici i v takovéto posloupnosti znamená prezenci, resp. absenci vrcholu s indexem $n - 1 - i$ v dané množině. To znamená, že jsou hodnoty odpovídající vrcholům v poli seřazené v (neobvyklém) pořadí od konce.

Na třetím až n+třetím řádku jsou vrcholy. Na každém řádku s vrcholem je množina následníků tohoto vrcholu, mezera a znak určující hráče vlastnícího daný vrchol – `1` znamená, že vrchol na daném řádku patří Adamovi, a `0` znamená, že patří Evě.

V případě `game4-2.txt` kódují tyto čtyři řádky Evin vrchol 0 bez následníka, Adamův vrchol 1 s následníky 0 a 2, Evin vrchol 2 s následníky 1 a 3 a Adamův vrchol 3 s následníky 0, 1, 2 a 3.

Na následujících k řádcích jsou základní Rabinovy podmínky. Na každém řádku se základní Rabinovou podmínkou je množina žádoucích vrcholů, mezera a množina nežádoucích vrcholů.

V případě `game4-2.txt` kódují tyto dva řádky dvě základní Rabinovy podmínky. První z nich obsahuje v množině žádoucích vrcholů vrcholy 0 a 1 a v množině nežádoucích vrcholů vrcholy 2 a 3. Druhá z nich obsahuje v množině žádoucích vrcholů vrchol 2 a v množině nežádoucích vrcholů vrchol 0.

Na následujících znacích nezáleží, `rgs.exe` je nečte.

B.3.2 Výstup

`rgs.exe` vypíše řešení každé řešené hry na obrazovku. Příklad části výpisu po zavolání `rgs.exe game4-2.txt`:

```
winning set=1110
strategy:
4
2
```

```
4
1
```

Formát výpisu na obrazovku se od formátu výpisu do souboru liší pouze přidaným výrazem `winning set=` a řádkem `strategy:`.

Pokud byl zadán parametr `-o suffix`, `rgs.exe` uloží řešení hry zadané parametrem `filename` (pro libovolné `filename` – může jich být i více v jednom běhu) do souboru `filenamesuffix`.

Příklad obsahu výstupního souboru s řešením Rabinovy hry (jde o celý obsah ukázkového souboru `game4-2.txt`):

```
4
2
0000 0
0101 1
1010 0
1111 1
0011 1100
0100 0001
1110
4
2
4
1
game4-2.txt
0
1
11
```

Počáteční část souboru je prostým výpisem řešené hry ve formátu totožném se vstupním (viz výše). Následuje odřádkování a samotné řešení hry.

Na prvním řádku řešení je množina výherních vrcholů.

V případě `game4-2.txt` jde o vrcholy 1, 2 a 3.

Na následujících n řádcích je Adamova vyhrávající strategie. i-tý řádek strategie obsahuje index některého následníka i-tého vrcholu pro každý vrchol i, který je Adamův a výherní zároveň. Pokud Adam při hraní dané hry pokaždé, když bude na tahu, přejde do vrcholu, který mu takto zadává strategie (pokud mu nějaký zadává),

hru jistě vyhraje. Na řádcích odpovídajících Eviným a proherním vrcholům je číslo n.

V případě `game4-2.txt` je na prvním řádku strategie číslo 4, protože vrchol 0 je prohern\'i (nebo proto, že je Evin), na druhém řádku číslo 2 jako pobídka k přechodu z vrcholu 1 do vrcholu 2, na třetím řádku číslo 4, protože vrchol 2 je Evin, a na čtvrtém řádku číslo 1 jako pobídka k přechodu z vrcholu 3 do vrcholu 1.

Pokud byl zadán parametr `-w0` nebo `-s0` (viz B.4), odpovídající část řešení není vypsána.

B.4 Parametry z příkazové řádky

V tabulce B.4 najdete výčet parametrů, které program `rgs.exe`přijímá prostřednictvím příkazové řádky.

Tabulka B.1: Parametry `rgs.exe` z příkazové řádky

Volba	**Popis**
`-h`	Použít Hornův řešič (implicitní volba)
`-p10`	Použít Pitermanův řešič bez store optimalizace
`-p11`	Použít Pitermanův řešič se store optimalizací
`-m dir`	Použít MATLABový řešič umístěný v `dir` `dir` musí být absolutní cesta bez mezer
`-w1`	Vyřešit výherní region (implicitní volba)
`-w0`	Neřešit výherní region
`-s1`	Vyřešit vyhrávající strategii (implicitní volba)
`-s0`	Neřešit vyhrávající strategii
`-o suffix`	Uložit řešení do souboru `filenamesuffix`
`filename`	Vyřešit hru uloženou v souboru `filename`

Parametry zadané později (na příkazové řádce více vpravo) přepisují parametry zadané dříve. Jedinou výjimkou je volba `filename`, kterou lze zadat vícekrát – program potom vyřeší více her v pořadí, ve kterém byly zadány na příkazové řádce.