

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

URČENÍ VELIKOSTI STAVOVÉHO PROSTORU

DETERMINING THE SIZE OF THE STATE SPACE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Marian Rybanský
AUTHOR

VEDOUCÍ PRÁCE — Ing. Jan Roupec, Ph.D.
SUPERVISOR

BRNO 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav automatizace a informatiky
Student:	**Marian Rybanský**
Studijní program:	Strojírenství
Studijní obor:	Aplikovaná informatika a řízení
Vedoucí práce:	**Ing. Jan Roupec, Ph.D.**
Akademický rok:	2015/16

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Určení velikosti stavového prostoru

Stručná charakteristika problematiky úkolu:

Znalost velikosti stavového prostoru je v některých úlohách předpokladem k určení optimálních algoritmů, velikosti datových uložišť, odhad času zpracování úlohy, množství výpočetních jednotek atd. Pro určení velikosti stavového prostoru lze vybrat vhodnou metodu na základě prohledání určité části stavového prostoru.

Cíle bakalářské práce:

1. Rešerše existujících metod k určení velikosti stavového prostoru.
2. Otestování vybraných metod na příkladu.
3. Vyhodnocení testů.

Seznam literatury:

Sedláček, J. (1981): Úvod do teorie grafů. Praha.

Tureček, M. (2008): Demonstration of State Space Search Methods. Vysoké učení technické, Fakulta informačních technologií, Brno.

Krč, M. (2006): Vizualizace algoritmů na prohledávání stavového prostoru. Vysoké učení technické, Brno.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2015/16

V Brně, dne

L. S.

doc. Ing. Radomil Matoušek, Ph.D. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

ABSTRAKT

Cílem práce je popsat a ohodnotit vhodné metody na určení velikosti stavového prostoru. Stavový prostor vytvoříme pomocí programu (generátoru), který ze vstupních hodnot a předpisů (modelu) vygeneruje stavový prostor. K objektivnímu posouzení klasifikujeme stavové prostory na základě jejich parametrů. Klasifikovaný stavový prostor zařadíme do určité třídy, v jejímž vymezení provedeme hodnocení metody.

ABSTRACT

Objective of the thesis is to describe and evaluate the appropriate methods to determine the size of the state space. State space will be created by using a program (generator) which generates a state space, from the input values and regulations (model). To objectively assess the state space we classifies according to their parameters. Classified state space will be added to a class, in which the demarcation perform evaluation methods will be done.

KLÍČOVÁ SLOVA

Stavový prostor, parametr, náhodná procházka, teorie grafů.

KEYWORDS

State space, parameter, random walk, graph theory.

RYBANSKÝ, M. Určení velikosti stavového prostoru. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2016. 29 s. Vedoucí Ing. Jan Roupec, Ph.D.

Prohlašuji, že jsem svou bakalářskou práci Určení velikosti stavového prostoru vypracoval samostatně pod vedením Ing. Jana Roupce, Ph.D. a uvedl v ní všechny použité literární a jiné odborné zdroje v souladu s právními předpisy a vnitřními předpisy Vysokého učení technického.

V Brně dne 20. 5. 2016 Marian Rybanský

Poděkování

Touto cestou bych rád poděkoval rodině za její trvalou podporu během celého studia.

Obsah:

1 Úvod 6

2 Stavový prostor vyjádřený grafem 7
2.1 Zobrazení grafu 7
2.2 Graf vyjádřený množinou 8
2.3 Typy grafů 8
2.3.1 Podgraf 9

3 Metody na prohledávání a určení VSP 9
3.1 Procházení grafu výpočetním systémem 9
3.1.1 Určení parametrů pomocí procházení SP 11
3.2 Matice vzdáleností 12
3.3 Zařazení SP do velikostní třídy 13

4 Další metody pro určení VSP 15
4.1 Prohledávání grafu do hloubky 15
4.2 Náhodná procházka 16
4.3 Metody klasifikace 17
4.4 Porovnání parametrů 18
4.5 Průnik vzorků stavového prostoru 18
4.6 Určení BFS parametrů 21

5 Generátor stavových prostorů 23
5.1 Příkazy k ovládání programu Divine 23

6 Závěr 26
7 Seznam použité literatury 27
8 Přílohy 29
9 Seznam použitých zkratek 35

1 ÚVOD

Tato práce se zabývá určením počtu všech možných variant (stavů), které mohou nastat v průběhu vykonávání zadané úlohy. Předmětem této práce tedy bude zhodnocení metod na určení velikosti stavového prostoru (SP). Úloha neboli model bude v našem případě reprezentována orientovaným grafem. K vytvoření a prozkoumání SP softwarem jsem si vybral program Divine, který dokáže pracovat s daným datovým modelem (úlohou).

Různé charakteristiky modelu a zejména množství stavů, kterých proces nabývá, přímo ovlivňují rychlost výpočtu a nároky na technické vybavení počítače. Motivací je tedy využití poznatků o vlastnostech stavového prostoru na optimalizaci modelu, technického vybavení počítače, nebo dalších komponent využívaného systému.

Znalost velikosti stavového prostoru (VSP) je důležitá zvláště u náročnějších úloh, s velkým množstvím dosažitelných výsledků. Dále pokud chceme optimalizovat algoritmus programu, určit velikosti datových úložišť nebo pro odhad času na zpracování úlohy. Odhad času na zpracování úlohy je v mnoha případech zásadní veličinou. Určením velikosti stavového prostoru se také dá stanovit množství výpočetních jednotek v závislosti na ostatních specifikacích výpočetního systému. Zajímal jsem se hlavně o případy (typy) modelů, u kterých je obtížné, nebo nemožné velikost SP určit odhadem, nebo jednoduchým výpočtem. Tyto úlohy mají často spoustu proměnných, a proto je vhodné využít výpočetní techniky a specializovaného softwaru.

Rozhodl jsem se porovnat nejpoužívanější metody a zjistit jejich vhodnost pro vybrané úlohy. Tyto úlohy jsem vybíral s ohledem na znalost samotného modelu.

SP, jeho logické uspořádání a prohledávání lze zjednodušit orientovaným grafem. Struktura těchto grafů se skládá z uzlů, které vyjadřují stavy, a z hran jako logických propojení mezi jednotlivými stavy a možnými přechody, podle zadaných pravidel. Obecně lze SP nahradit jednoduchým orientovaným grafem vyjádřeným jako: G = (V, E, v_0), kde V je množina vrcholů, E množina orientovaných hran (přechodů mezi jednotlivými stavy) a v_0 je počáteční stav (vrchol ze kterého se začíná při provádění úlohy). Toto grafické vyjádření je názorné a zjednodušuje systém řešení dané úlohy a relace mezi jednotlivými stavy.

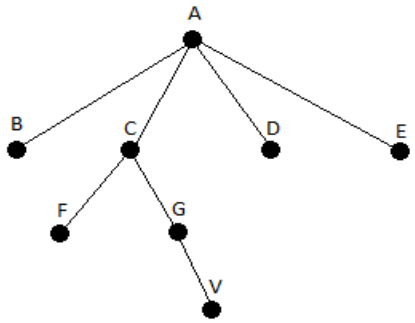
Pomocí teorie zabývající se průchodem grafů můžeme tedy odvodit metody používané na určení velikosti grafu. Prohledání celého stavového prostoru poskytne přesný výsledek, ale toto řešení je s ohledem na velikost a časovou náročnost často nemožné. Určení velikosti lze ve většině případů s odvozenou nejistotou provést z prohledání malé části a určení lokálních parametrů SP. Graficky vyjádřeno, tato část musí být reprezentativní, tedy zastupovat většinovou strukturu zkoumané oblasti, nebo celého grafu. Na závěr je potřeba určené stavové prostory roztřídit a zařadit do velikostních skupin, čemuž se věnuji v kapitole 3.

2 STAVOVÝ PROSTOR VYJÁDŘENÝ GRAFEM

Stavový prostor (SP) je dán uspořádáním množiny stavů příslušících k určitému systému a množinou přechodů mezi těmito stavy. Stavem je pak konfigurace daného systému v konkrétním čase. Jeden stav tedy popisuje například rozložení figur na hracím poli u hry šachy.

2.1 Zobrazení grafu

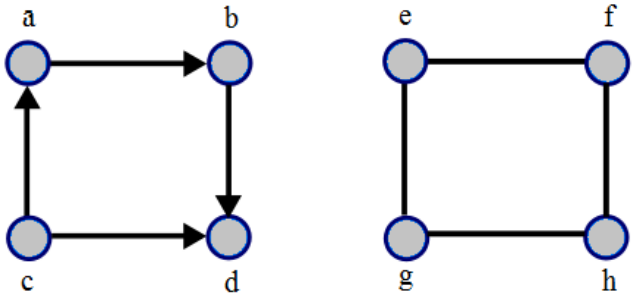
Grafy lze kreslit názorně, hrany grafu (přechody mezi stavy) tvoří úsečky a vrcholy jsou zobrazeny, jako tečky viz obr. 1. Graf se dá také vyjádřit množinou, případně maticí.



Obr. 1 Vyjádření stavového prostoru grafem.

Své důležité místo v informatice si grafy získaly dobře vyváženou kombinací svých vlastností – svou názorností a zároveň jednoduchou zpracovatelností ve výpočetních systémech. Díky těmto vlastnostem se grafy prosadily jako vhodný matematický model pro popis i velice komplikovaných, vztahů mezi objekty jak uvádí Hliněný v [2].

V případě datových modelů použitých pro vytvoření stavových prostorů vyjádřených grafem se ve většině případů jednalo o orientovaný graf. U orientovaných grafů mají hrany danou orientaci. Při popisu reálných situací grafem je obvyklé, že hrana může být jednosměrná. Neformálně řečeno, například při hře piškvorky není podle pravidel možné dostat se po odehrání do předchozího stavu, neboli vzít tah zpátky viz obr. 2.



Obr. 2 Orientovaný a neorientovaný graf.

Nechť *G* je dvojice (*V, E), kde E* je podmnožina kartézského součinu V × V. Prvky *E* nazýváme šipky nebo orientované hrany. Orientovaná hrana *e* má tvar (*x*, *y*). Říkáme, že tato orientovaná hrana vychází z *x* a končí v *y*.

Máme-li zobrazit SP jako jednoduchý orientovaný graf G = (V, E, v_0) se souborem vrcholů V, bude soubor orientovaných hran E μ V × V, s počátečním vrcholem v_0. Hrany reprezentují platné přechody mezi stavy.

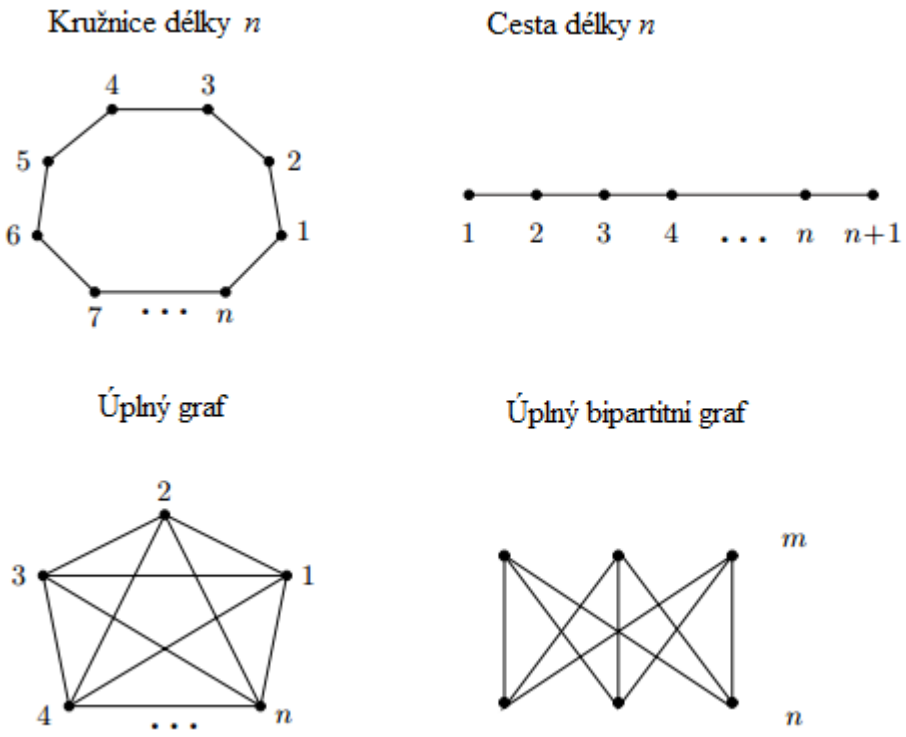
2.2 Graf vyjádřený množinou

Hrany a vrcholy grafu lze zapsat množinou, např.: E = {{A, B}, {A, C}, {A, D}, {A, E}, {C, F}, {C, G},{G, V}}, V = {A, B, C, D, E, F, G, V} kde množina E vyjadřuje možná spojení mezi jednotlivými uzly a množina V je výčet všech vrcholů grafu.

2.3 Typy grafů

Jak uvádí Hliněný v [2] běžné typy grafů jsou:

- Kružnice délky n má n ≥ 3 vrcholů spojených do jednoho cyklu n hranami:
- Cesta délky n má n + 1 vrcholů spojených za sebou n hranami:
- Úplný graf na n ≥ 1 vrcholech má n vrcholů, všechny navzájem pospojované.
- Úplný bipartitní graf na m ≥ 1 a n ≥ 1 vrcholech má m+n vrcholů ve dvou skupinách (partitách), přičemž hranami jsou pospojované všechny m-n dvojice z různých skupin:

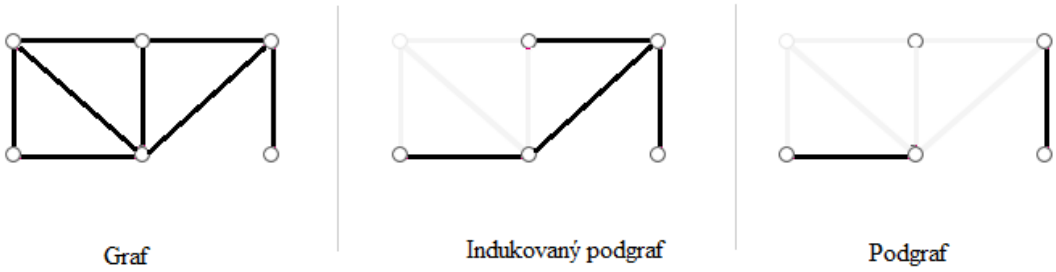


Obr. 3 Běžné typy grafů.

2.3.1 Podgraf

Graf H je podgrafem grafu G, jestliže V(H) $\subseteq$ V(G) a E(H) $\subseteq$ E(G). Graf H je potom indukovaným podgrafem grafu G, pokud V(H) $\subseteq$ V(G) a za předpokladu, že platí vztah 2.1 (Graf, indukovaný podgraf, podgraf viz obr. 4).

$$E(H) = E(G) \cap \binom{V(H)}{2} \tag{2.1}$$



Obr. 4 Graf, indukovaný podgraf, podgraf.

3 METODY NA PROHLEDÁVÁNÍ A URČENÍ VSP

Uvažujme prohledávání do šířky (BFS z anglického breadth first search) z počátečního vrcholu v_0. Úroveň BFS s indexem k je množina stavů se vzdáleností od v_0 rovnající se k. Výška BFS je největší hodnota / index úrovně.

Za předpokladu, že vyjádříme graf pomocí matice sousednosti, je zapotřebí brát v úvahu, že první číslo určuje odkud a druhé kam hrana směřuje. Pravidlo u neorientovaných grafů, kdy hodnota na pozici (3,2) se rovná hodnotě na pozici (2,3) neplatí. Matice sousednosti orientovaného grafu tedy nemusí být symetrická.

Na grafech můžeme zkoumat různé parametry, například kolik hran vychází z určitého vrcholu. Tento parametr se nazývá stupeň vrcholu. V grafu na obr. 1 jsou tyto stupně vrcholů: A = 4, B = 1, C = 3, D = 1, E = 1, F = 1, G = 2, V = 1. Součet stupňů jednotlivých uzlů je 14 s polovičním počtem hran 7 a počtem vrcholů 8. Podle Hliněného v [2] pomocí součtu stupňů můžeme určit počet hran (přechodů) v grafu, mezi jednotlivými uzly (stavy).

3.1 Procházení grafu výpočetním systémem

Velké grafy je nutné prohledávat postupně (prozkoumat jeho dílčí - lokální části) v případě, že prohledání celého grafu je nevýhodné (časová, výpočetní, nebo jiná náročnost), nebo nemožné. Tohoto postupu se využívá při odhadování velikosti SP. Zjednodušeně řečeno, pokud bychom například chtěli odhadnout počet bytových jednotek v obytném domě, odhad bychom provedli na základě znalosti dílčí části. Prozkoumáním jednoho patra dostaneme počet jednotek

na patro. Na základě dalších znalostí (počet pater, velikost pater) odhadneme celkový počet. Toto by nám poskytlo odhad s určitou přesností. Přesné určení počtu stavů se tedy s ohledem na nehomogenní vyjádření jednotlivých SP jeví jako nemožné.

Z toho vyplívá, že bez toho aniž by byl graf prozkoumán celý, je přesný výsledek nedosažitelný. Musíme tedy vyjít ze znalosti jednotlivých částí, které prozkoumáme, například jak uvádí Hliněný v [2] při procházení souvislé komponenty grafu - algoritmus projde a zpracuje každou hranu a vrchol souvislého grafu G. Průchod vyjádříme algoritmem (alg. 1).

- Stavy vrcholu

  iniciační – stav na začátku,
  nalezený – poté, co jsme jej přes některou hranu nalezli,
  zpracovaný – poté, co jsme už probrali všechny hrany z něj vycházející.

- Stavy hrany

  iniciační – počáteční hrana,
  zpracovaná – poté, co už byla probrána od jednoho ze svých vrcholů.

- Úschovna je pomocná datová struktura (množina),
  udržuje nalezené a ještě nezpracované vrcholy.

Alg. 1:

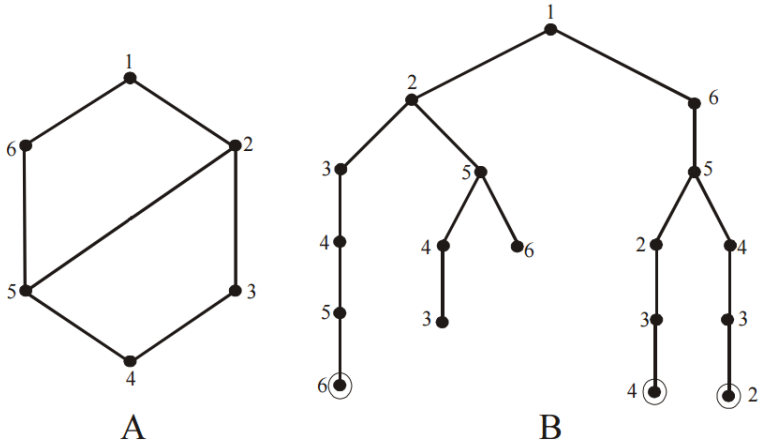
```
vstup < graf G;
Stav (všechny vrcholy a hrany G) = iniciační;
paměť P = {libovolný vrchol v0 grafu G};
Stav (v0) = nalezený;
// zpracování vybrané komponenty G
while (P = 0) {
        vybrat v ∈ U; U = U \ {v};
        ZPRACUJ (v);
        foreach (e hrana vycházející z v) {

                if (stav (e)==iniciační) ZPRACUJ (e);

                w = opačný vrchol hrany e = vw;
                if (stav (w)==iniciační) {
                        Stav (w) = nalezený;
                        U = U ∪{w};
                                        }
                Stav (e) = zpracovaná;
            }
        Stav (v) = zpracovaný;
// případný přechod na další komponentu G
}
G zpracovaný;
```

SP vygenerovaný z datových modelů (viz [5]) pro účely měření bývá řádově větší (přibližně až 20 milionů stavů). O výpočtu velikosti uvažovat nelze, proto pro určení velikosti použijeme odhad, přestože odchylka u zjištěných parametrů může nabývat vysokých hodnot. Zjistil jsem, že nepřesnost je přímo úměrná tvaru (nehomogenitě) SP. Tvarem je myšleno logické rozložení jednotlivých stavů v prostoru. Zjištění parametrů z lokální části SP je u nesymetrických tvarů nevypovídající a odhad může být značně ovlivněn. Pro názornější představu o rozložení SP je vhodné část grafu vygenerovat a následně určit nejpoužitelnější metodu.

Názorný příklad nehomogenity vyjádření SP je na obr. 5 (A - hamiltonova kružnice, B – stromová struktura grafu). U grafu na obr. 5B cesty končící vrcholem v kroužku jsou uzavřené. V případě, že bychom se celou dobu prohledávání vzorku pohybovali v pravé větvi grafu na obr. 5B, zkreslilo by to parametry SP.



Obr. 5 Vyjádření stavového prostoru grafem [8].

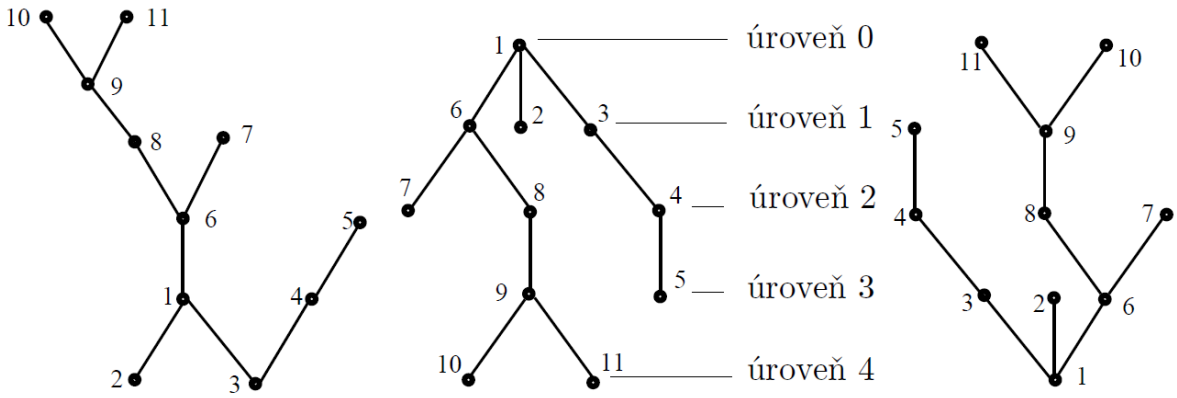
3.1.1 Určení parametrů pomocí procházení SP

Grafy kořenových stromů obvykle začínají od vrcholů nejnižší úrovně a větví se směrem dolů na vrcholy vyšší úrovně. Používaný je i opačný způsob. Zvolme si strom s kořenem v_0. Jak uvádí Lepš v [8] (v_0, $\{v_0, v_n\}$, $v_1,\ldots,v_n - 1$, $\{v_n-1, v_n\}$, v_n) je (jediná) $v_0 - v_n$ cesta. Pak v_0, $v_1,\ldots,v_n - 1$ označme jako předky vrcholu v_n, vrchol v_n-1 se nazývá otec vrcholu v_n, vrchol v_n se pak nazývá syn vrcholu v_n-1. Když je x předkem y, potom je y následníkem neboli potomkem vrcholu x. Rozložení a úrovně grafu na obr. 6.

Pro odhad velikosti SP je zjištění úrovně grafu důležitý parametr. Je ale zapotřebí zvolit vhodnou metodu. Pro ohodnocení se nabízí lokální průzkum SP pomocí gradientního algoritmu (Hill-climbing). Tento algoritmus zahájí expanzi z náhodného uzlu a vygeneruje svoji nejbližší úroveň všemi směry. Následně provede ohodnocení nejbližších uzlů a pokračuje expanzí z nejlépe hodnoceného. Toho se dá využít za předpokladu, že by se prozkoumaná oblast byla uložena do paměti (gradientní algoritmus se využívá pouze k nalezení cíle bez nutnosti ukládat do paměti).

U určitých modelů by se na základě těchto hodnot dalo určit, v které části (úrovni) SP k expanzi došlo. Například pokud bychom dokázali odhadnout parametry modelu v určitém stádiu vykonávání jeho algoritmu. Pro zjednodušení například se znalostí expanze v náhodných

místech SP hry piškvorky můžeme odhadnout dosaženou úroveň (jedná se o jednoduché větvení). Na základě násobného spuštění (více průchodů zpřesňuje odhad) získáme základní představu o velikosti SP ze znalosti modelu a velikosti dosažených úrovní. Z toho se dají odvodit počáteční hodnoty datového modelu (u piškvorek třeba velikost hrací plochy – počet políček na začátku).



Obr. 6 Úrovně grafu [8].

3.2 Matice vzdáleností (vzdálenost dvojice vrcholů)

Jak zmiňuje autor v [16] v každém grafu lze přirozeným způsobem definovat vzdálenost libovolné dvojice vrcholů. Vzdálenosti v grafu lze zjistit z mocnin jeho matice sousednosti. Matice sousednosti a její mocniny umožňují zjistit počet sledů dané délky mezi dvěma vrcholy. Vzhledem k tomu, že vzdálenost libovolných dvou vrcholů je buď menší než n, nebo nekonečná (graf na n vrcholech neobsahuje žádnou cestu délky ≥ n), stačí pro zjištění matice D(G) spočítat n − 1 mocnin matice sousednosti S (G). Výpočet každé mocniny spočívá ve vynásobení dvou matic o rozměrech n × n.

Definice 3.1. Vzdálenost d(x, y) vrcholů x, y orientovaného grafu G je délka nejkratší cesty z x do y. Pokud taková cesta neexistuje, položíme d(x, y) = ∞.

Definice 3.2. Nechť G je souvislý neorientovaný graf. Pak funkce d(x, y) je metrikou na množině V (G), tj. má následující vlastnosti:

$$d(x, y) \geq 0, \text{ přičemž } d(x, y) = 0, \text{ právě když } x = y, \tag{3.1}$$

$$d(x, y) = d(y, x), \tag{3.2}$$

$$d(x, y) + d(y, z) \geq d(x, z) \text{ („trojúhelníková nerovnost“).} \tag{3.3}$$

Definice 3.2. Distanční matice orientovaného grafu G s vrcholy $v_1, \ldots, v_n$ je matice o rozměrech n × n.

Distanční matice neorientovaného grafu je definována jako distanční matice jeho symetrické orientace.

U větších grafů je však mnohem jednodušší použít následující obecný postup. Spočítáme první čtyři mocniny matice sousednosti (včetně nulté). Jednotlivé prvky jsou zvýrazněny v nejnižší mocnině, kde jsou nenulové (obr. 7).

$$(S(G))^0 = \begin{bmatrix} \underline{1} & 0 & 0 & 0 \\ 0 & \underline{1} & 0 & 0 \\ 0 & 0 & \underline{1} & 0 \\ 0 & 0 & 0 & \underline{1} \end{bmatrix}, \qquad (S(G))^1 = \begin{bmatrix} 1 & \underline{1} & 0 & 0 \\ 0 & 0 & 0 & \underline{1} \\ 0 & \underline{1} & 0 & \underline{1} \\ 0 & 0 & \underline{1} & 0 \end{bmatrix},$$

$$(S(G))^2 = \begin{bmatrix} 1 & 1 & 0 & \underline{1} \\ 0 & 0 & \underline{1} & 0 \\ 0 & 0 & 1 & 1 \\ 0 & \underline{1} & 0 & 1 \end{bmatrix}, \qquad (S(G))^3 = \begin{bmatrix} 1 & 1 & \underline{1} & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

Obr. 7 Zvýrazněné prvky v nejnižší mocnině [16].

Pro každý prvek nyní zapíšeme, ve které mocnině je zvýrazněn a dostaneme distanční matici D(G) viz obr. 8.

$$D(G) = \begin{bmatrix} 0 & 1 & 3 & 2 \\ \infty & 0 & 2 & 1 \\ \infty & 1 & 0 & 1 \\ \infty & 2 & 1 & 0 \end{bmatrix}.$$

Obr. 8 Výsledná distanční matice [16].

Podle Palůcha v [7] můžeme na vypočet matice vzdáleností vrcholů v hranově orientovaném grafu použít Floydův algoritmus. Ten je aplikovatelný i na digrafy s všeobecným (kladným i záporným) ohodnocením hran. Při použití Floydova algoritmu je potřeba dát si pozor na počítačovou reprezentaci ∞. Je vhodné zvolit za ∞ tak velké číslo, aby jeho dvojnásobek nepřesáhnul největší zobrazitelné číslo. Potom je podle O'Neilla a Riana v [13] možné uvažovat o vynechání kontroly na nekonečno.

3.3 *Zařazení SP do velikostní třídy*

Autoři doporučují zařadit SP do velikostních tříd. Tento přístup zjednodušuje vyhodnocení a porovnání techniky na odhad VSP. Třídy jsou definovány na základě intervalů. Čas na vygenerování SP u různých modelů se velmi mění. Relativní odhad časových potřeb je v praxi užitečnější realizovat za běhu. Ze zadání problému určíme počet velikostních tříd a jejich intervaly. Autoři v [1] se rozhodli rozdělit VSP na tři intervaly. Nechť je R poměr celkového počtu stavů k počtu odebraných vzorků viz tab. 1.

Tab 1. [1]

Třída	***Interval***	***Popis***
1	$1 \leq R < 1000$	Modely v této třídě mohou být ověřeny. V případě, že tento interval nebude dostatečný, doladí se parametry modelu, např. vhodná velikost „hash“ tabulky.
2	$1001 \leq R < 10^5$	Chceme-li zkontrolovat model v této třídě, je nezbytné použít redukce, nebo distribuované výpočty.
3	$(10^5) + 1 \leq R < 10^7$	Vzorek v této třídě je natolik velký, že je nezbytné použít abstrakci.

4 DALŠÍ METODY PRO URČENÍ VSP

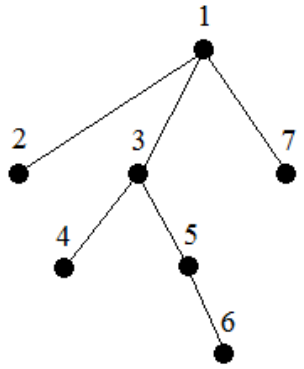
Další metody pro určení VSP jsou také založeny na odhadu z parametrů SP. Tyto parametry lze získat průchodem grafem. Jak bylo uvedeno v kapitole 3, při vhodné volbě a použití (zvolení počtu průchodů, kombinace s jinými metodami) způsobu průzkumu části SP odhadneme jeho parametry. Tyto data jsou závislá na zvolené metodě.

4.1 Prohledávání grafu do hloubky

Při průchodu do hloubky (DFS z anglického Depth First Search) se z počátečního uzlu zvolí jedna cesta (hrana) a přejde se do dalšího vrcholu. Zde dochází k expanzi do prvního následujícího vrcholu, kterým se ještě neprošlo. Když algoritmus dojde na konec větve, ze které není možnost expandovat do vyšší úrovně, vrací se zpět o jednu úroveň a snaží se postoupit na následující nenavštívený vrchol, jak uvádí autoři v [6]. Tento algoritmus je znázorněn na obr. 4. Správnou orientaci v průchodu grafem a zamezení zacyklení zajistí označení hrany proměnnou, aby byla možnost se v prohledávání vrátit na vyšší úroveň. Hodnota se po průchodu do vyšší úrovně přidá na vrchol zásobníku, při průchodu zpět na nižší úroveň se hodnota odebere. Vyjádřeno procedurou pruchod(g): navstiveno[g] = aktualne, cas = cas + 1; in [g] = cas; pro každé w ∈ sousedi[g]; pokud navštíveno[g] = ne, potom projdi (w); navstiveno[g] = ano; cas = cas + 1; out[g] = cas.
Procedura DFS ():

```
∀v∈V: navstiveno[g] = ne
cas = 0
foreach g ∈ G  do
        if navstiveno[w]= ne then
                projdi (w)
```

Nechť je kořenem výchozí vrchol g (na obr. 9 je výchozí vrchol 1). Hrany jdoucí do navštíveno[g] = ne tvoří strom (DFS strom). Jak uvádí Černý v [3] tyto hrany označíme jako stromové, ostatní jako zpětné. Množina stromu (DFS stromů) tvoří les (DFS les). Pokud při průchodu dojdeme z kořenu do vrcholu v označíme v jako následníka kořenu g.



Obr. 9 Průchod grafem do hloubky

Průchod metodou BFS začíná na první úrovni, z které se pokračuje na úroveň druhou. Třetí úrovní (vzdálenost 2 od začátku prohledávání) se nepokračuje až do doby, dokud se neprohledá úroveň druhá. Tento postup se někdy označuje jako algoritmus vlny, neboť v *k*-té vlně (přechodu na vyšší úroveň) se dostaneme do výchozího vrcholu, do vrcholu, který je vzdálen o *k*. Při simulaci vlny po přechodu do další úrovně, uložíme všechny možné cesty do fronty. Neformálně řečeno BFS na rozdíl od DFS umožňuje nalezení nejkratší cesty, která vede k řešení.

4.2 Náhodná procházka

Další technikou, pomocí které by se dala určit VSP, nebo alespoň extrakce některých parametrů SP je použití náhodné procházky (RW – z anglického random walk). RW můžeme popsat tak, že v jednotlivých krocích se náhodně z jednoho stavu dostáváme do jiného s určitou pravděpodobností. Je důležité, jaká je pravděpodobnost přechodu mezi jednotlivými stavy, od čehož se ovíjí pravděpodobnostní tabulky výsledku. Často je pravděpodobnost jestli se dostaneme do jednoho ze dvou vrcholů stejná, tedy pravděpodobnost pohybu v obou směrech je 50%. Tato náhodná procházka se nazývá symetrická. Výhodou algoritmu RW vidím v malých nárocích na paměť a jednoduchost.

Definice 4.1. Jednoduchá náhodná procházka je prostorově homogenní

$$P(S_n = j \mid S_0 = a) = P(S_n = j + b \mid S_0 = a + b). \tag{4.1}$$

Definice 4.2. Jednoduchá náhodná procházka je časově homogenní

$$P(S_n = j \mid S_0 = a) = P(S_m + n = j \mid S_m = a). \tag{4.2}$$

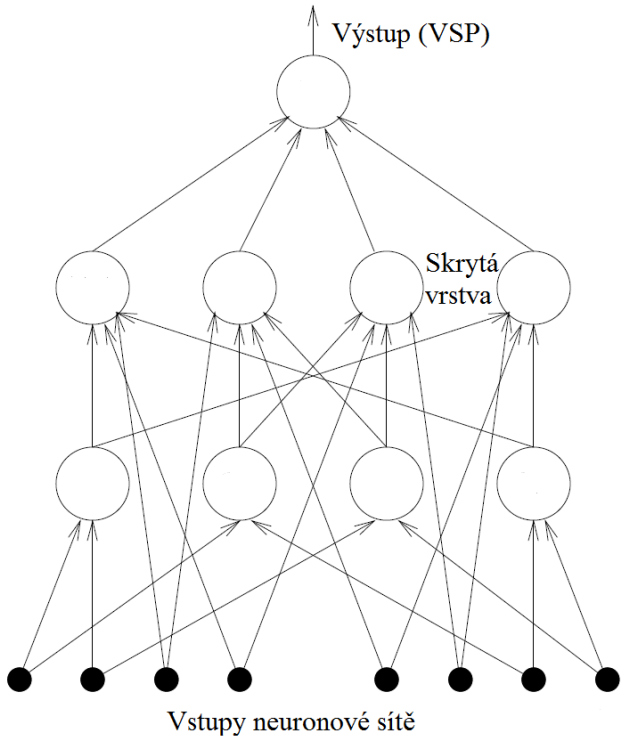
Kde posloupnost $\{Sn\} \infty n=0$ vyjadřuje jednoduchou náhodnou procházku

Cílem této techniky je projít malou, ale reprezentativní část SP. Podle autora v [9] by bylo řešením hash-RW nástroj pracující na podobném principu jako BFS. Metoda využívá „hashovaní“ k rozhodnutí, které stavy by měly být uloženy a prozkoumány. Na rozdíl od RW bez zápisu do paměti, je použito úložiště pro rozpoznání navštívených stavů. Ke zvýšení pravděpodobnosti, že procházkou navštívíme stavy nacházející se na různých cestách a v různých vzdálenostech od počátečního stavu v_0, vyhledávání přejde do *C* - stavů paralelním způsobem kde *C* je konstantní číslo. Podmnožina stavů je určena zvláštní rozhodovací funkcí. Funkce vypočítá „hash“ přidělenému stavu, a pokud je menší, než určený limit, rozhodne se o uložení stavu do úložiště a prohledávácí fronty. Limit je aktualizován po výpočtu všech následovníků poslední úrovně, aby se počet stavů v další úrovni blížil ke konstantě *C*. Cílem těchto rozhodovacích funkcí je snížit počet překročených zpětných hran.

S pomocí nástroje Divine [4] a benchmarkové sady Beem [5] je možné porovnávat jednotlivé metody s vygenerovanými SP. Webový portál benchmarkové sady Beem obsahuje všechny použité modely a také popis vlastností datových modelů. Experimenty v [1] ukázaly, že jednoduché techniky nejsou použitelné pro přesný výpočet VSP.

4.3 Metody klasifikace

Neuronová síť je metoda, pomocí které se systém může učit klasifikovat SP. Tento proces lze automatizovat pomocí vhodného algoritmu. Metoda je lehce uskutečnitelná na vícevrstvé síti, ve které jsou neurony rozděleny do několika vrstev. Neurony jsou spojené mezi dvěma sousedními vrstvami, ale nejsou spojeny mezi sebou ve vrstvě. Vícevrstvá síť obsahuje 1-2 řady skrytých neuronů a jeden výstupní neuron. Používání sítí s menším počtem vrstev je vhodnější, jelikož se sítě rychleji učí. Aktivační funkcí může být sigmoida nebo ostrá nelinearita. Je vhodné zvolit jeden výstupní neuron pro vrácení určitého čísla (odhad VSP) viz obr 10.



Obr. 10 *Neuronová síť s osmy vstupy (8 parametrů) a jedním výstupem (VSP)* [9].

Parametry lze zjistit také prostřednictvím genetického programování (GP). GP je oblastí výpočetních technik (evolučních), využívajících interpretaci problému používanou v genetických algoritmech. Uvažujeme-li o požadovaných strukturách jako o optimalizačním problému, potom je reprezentace pojata jako generovaná rovnice funkce na základě symbolické regrese. Tato metoda má širší použití než neuronové sítě. Stromy, skládající se z listů (parametry SP), zobrazíme generace jedinců. Můžeme použít periodické goniometrické funkce, s jejichž pomocí dosáhneme lepších výsledků. Vstupní data je vhodné roztřídit do *s* skupin a podle toho zvolit *s* neuronů, kde každému vstupu přísluší jeden neuron. Zvolené vstupy bude neuron spojovat s jednotlivými skupinami [9], [12].

4.4 Porovnání parametrů

Některé parametry SP jsou použitelnější než jiné. V této kapitole bude probrána relevance jednotlivých parametrů při hodnocení VSP. Pokud rozdělíme zkoumané SP do dvou skupin, kde první obsahuje všechny naměřené hodnoty a druhá statisticky roztříděné SP, získáme tím podklad pro hodnocení jednotlivých parametrů, jak zmiňují autoři v [9]. Poté se testuje, jestli je hodnota parametru ve vhodném intervalu viz tab. 1. Níže uvedené parametry s popisem a zkratkou viz tabulka 2.

Tab 2. [9]

Parametr	***Zkratka***
Prohledané stavy	ST
Existence inflexních bodů (průběh funkce BFS úrovní)	SI
Existence lokálního maxima	SLoM
Poměr velikosti poslední prozkoumané BFS úrovně k průměrné velikosti již prozkoumaných úrovní.	SLA
Poměr velikosti poslední prozkoumané BFS úrovně k největší BFS úrovni	SLM
Prohledané dopředné přechody	FT
Existence inflexního bodu v průběhu funkce přechodů (dopředných) BFS úrovně.	FI
Existence lokálního maxima přechodů (dopřených) BFS úrovně.	FLoM
Poměr dopředných přechodů s poslední prozkoumanou BFS úrovní k průměrnému počtu dopředných přechodů prozkoumaných BFS úrovní.	FLA
Poměr dopředných přechodů s poslední prozkoumanou BFS úrovní k největšímu počtu dopředných přechodů prozkoumané BFS úrovně.	FLM

Procentuální úspěšnost určení VSP při použití parametrů viz tab. 3.

Tab 3. [9]

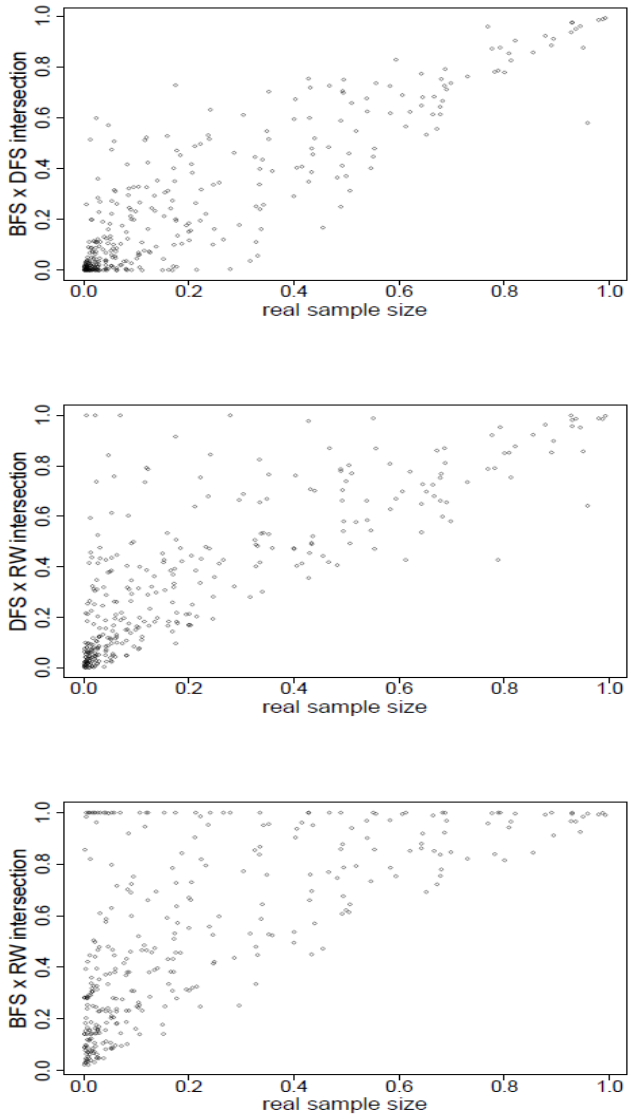
	ST	SI	SLoM	SLA	SLM	FT	FI	FLoM	FLA	FLM
min	0,44	0,41	0,66	0,58	0,56	0,41	0,41	0,63	0,5	0,47
průměr	0,49	0,5	0,73	0,61	0,59	0,47	0,49	0,69	0,53	0,51
max	0,51	0,53	0,78	0,63	0,62	0,5	0,52	0,75	0,57	0,56

4.5 Průnik vzorků stavového prostoru

Jak uvádí autor v [9] na začátku se zvolí limit L počtu stavů a ze SP se náhodně vyberou dva vzorky, každý s počtem L stavů. Nechť je O počet stavů vyskytujících se v obou vzorcích. Pak očekáváme, že poměr stavů vyskytujících se v obou vzorcích ku velikosti vzorků (O/L)

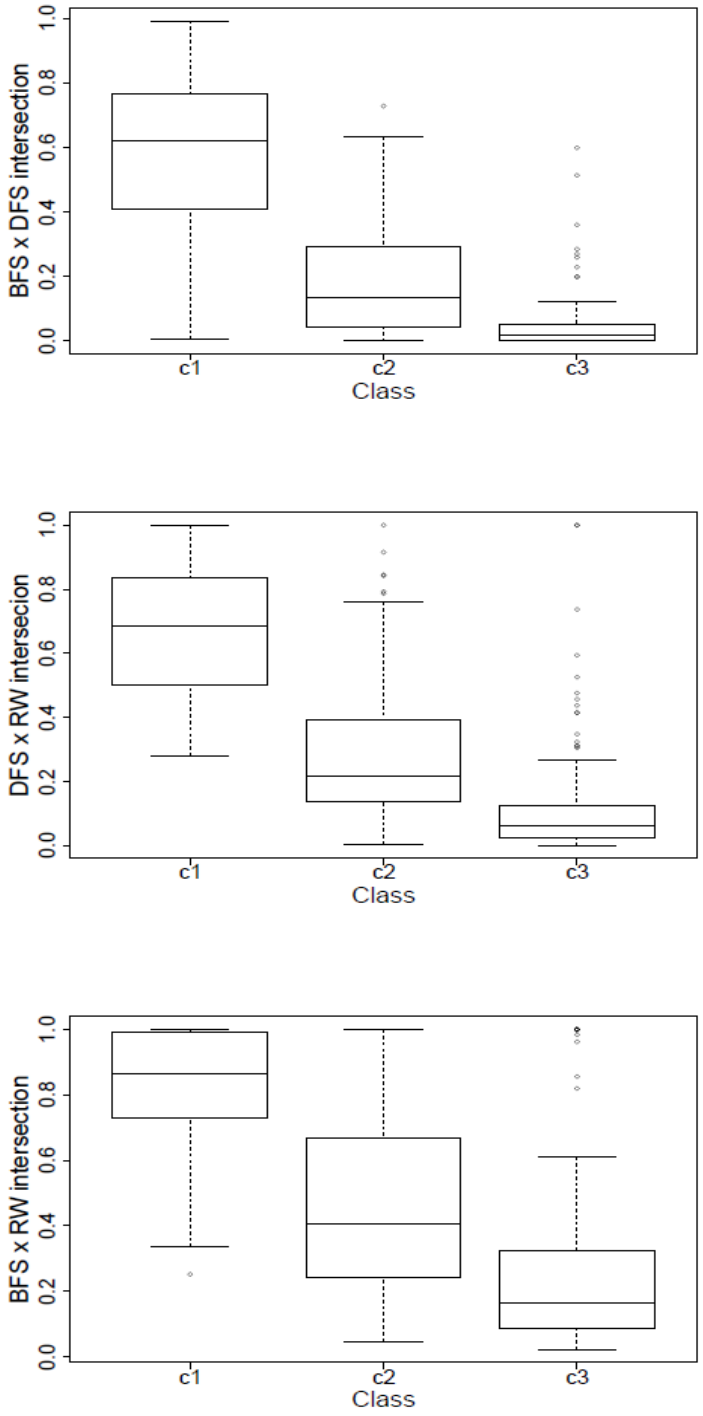
je podobný jako L/N, kde N je celkový počet dosažitelných stavů zkoumaného stavového prostoru. Hodnota L/O by se měla blížit k počtu dosažitelných stavů zvoleného SP. Není snadné zvolit náhodné vzorky, bez možnosti prozkoumání celého stavového prostoru. Lze ale určit dvě metody použitelné pro průchod grafem a pomocí nich spustit prohledávání zkoumaného stavového prostoru s limitem L a takto získané vzorky porovnávat.

Velikost určitého SP by se tedy měla blížit hodnotě $L = N$. Průnik vzorků stavového prostoru může být použit i v případě, že vzorky nejsou zcela nezávislé a náhodné. Z obr. 11 je vidět, že vztah N = L při správném určení velikosti SP odpovídá. Úspěšnost této metody, pro zařazení do velikostní kategorie a použití metod DFS se pohybuje kolem 70% [1]. Na obr. 11 je vykreslen poměr mezi velikostí vzorku L a počtem dosažitelných stavů (VSP) pro různé kombinace metod.



Obr. 11 *Poměr velikosti vzorku a dosažitelných stavů* [1].

Relativní velikosti průsečíků pro tři základní třídy zvoleny v tab. 1 jsou pomocí krabičkového grafu na obr. 12.



Obr. 12 *Relativní velikost průsečíků pro třídu 1,2 a 3* [1].

Na obr. 13 jsou řádky (C1, C2, C3) správné klasifikace, sloupky (E1, E2, E3) jsou odhadované klasifikace. Výsledky jsou zaokrouhlené a v procentech.

BFS x DFS

	E1	E2	E3
C1	31%	6%	0%
C2	8%	19%	12%
C3	3%	9%	12%

DFS x RW

	E1	E2	E3
C1	34%	2%	0%
C2	8%	29%	3%
C3	3%	12%	9%

BFS x RW

	E1	E2	E3
C1	25%	11%	0%
C2	4%	31%	4%
C3	3%	10%	11%

Obr. 13 *Výsledky odhadu na základě odběru vzorků* [1].

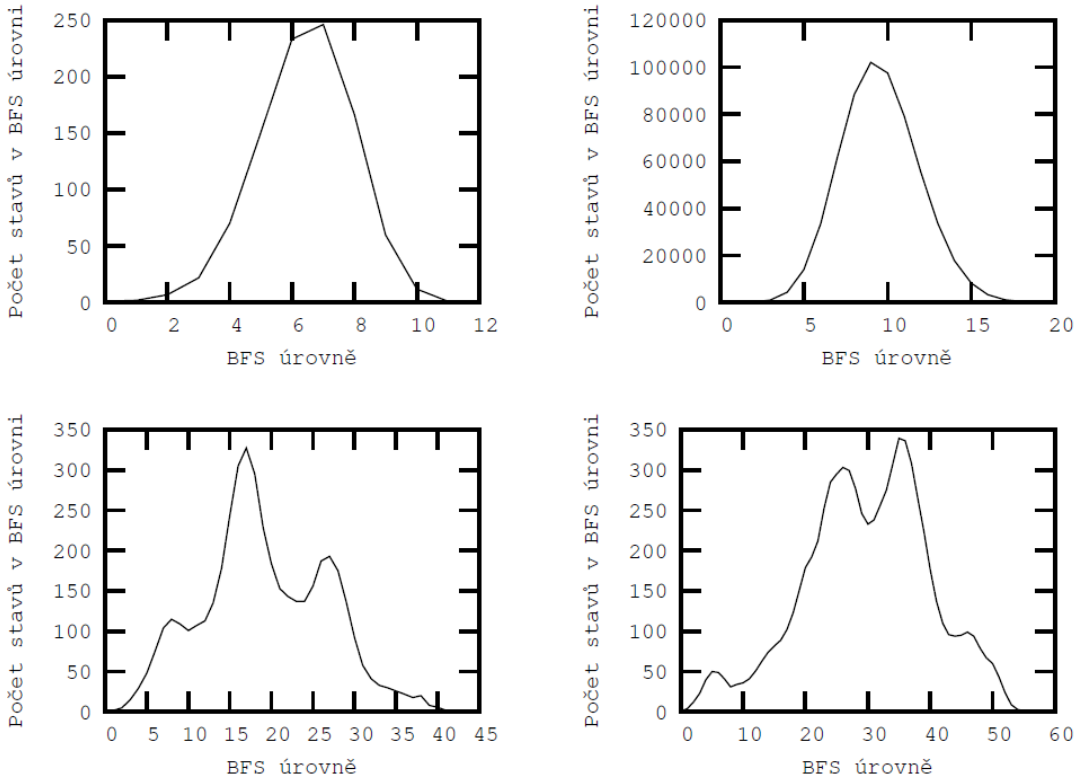
4.6 Určení BFS parametrů

Tato metoda je založená na prohledávání grafu do šířky. Tímto průchodem můžeme určit množinu stavů vzdálených *n* přechodů od počátečního stavu v_0. Nechť je *n* úrovní BFS, potom můžeme uvažovat o každých dvou rozdílných úrovních jako o disjunktní množinách stavů. Jejich sjednocením přes všechna *n* můžeme za určitých předpokladů obdržet celý stavový prostor. Vykreslíme-li počet stavů na úrovni proti indexu úrovně, kterou dostaneme BFS úrovňovým grafem, dostaneme graf s obvykle zvonovitým tvarem. Naším cílem je využít poznatky z tohoto chování při vyhledávání a odhadnout VSP ze znalosti prvních úrovní BFS, kde počet úrovní je určeno podle velikosti vzorku (SP) jak uvádí Katriel a Meyer v [14]. Pozorováním jsem zjistil, že pokud máme poznatky o počtu stavů v jednotlivých úrovní, můžeme u symetrických stavových prostorů predikovat rozvoj do vyšších úrovní. Od první úrovně obvykle počet stavů s vyšší úrovní roste a po dosažení maxima tento počet zase klesá. Na obr. 14 je vyjádřena závislost počtu stavů na úrovni grafu pro čtyři SP vygenerované ze hry sokoban. Výhodu této metody vidím v tom, že lze učinit odhad již ze znalosti několika prvních úrovní SP. Podle Appla v [9] je pomocí integrování, v případě že by se podařilo extrapolovat průběh křivky odpovídající velikosti jednotlivých úrovní, možné určit velikost SP. Použití DFS by pro zjištění parametrů nebylo nejvhodnější. Průchod pomocí DFS je totiž hodně závislý na rozložení následníků ve struktuře a reprezentaci SP jak uvádí Lepš v [8].

Parametry určené z odhadu [1]:

- Poměr velikosti poslední prozkoumané úrovně BFS s průměrnou velikostí prvních *k* - úrovní.
- Poměr velikosti (počet stavů v úrovní) poslední úrovně k maximální velikosti úrovně v první *k*-té úrovni.
- Hodnota, nula znamená, že rozdíl velikostí po sobě jdoucích úrovní se zvyšuje (až do *k*), tedy neexistuje žádný inflexní bod.

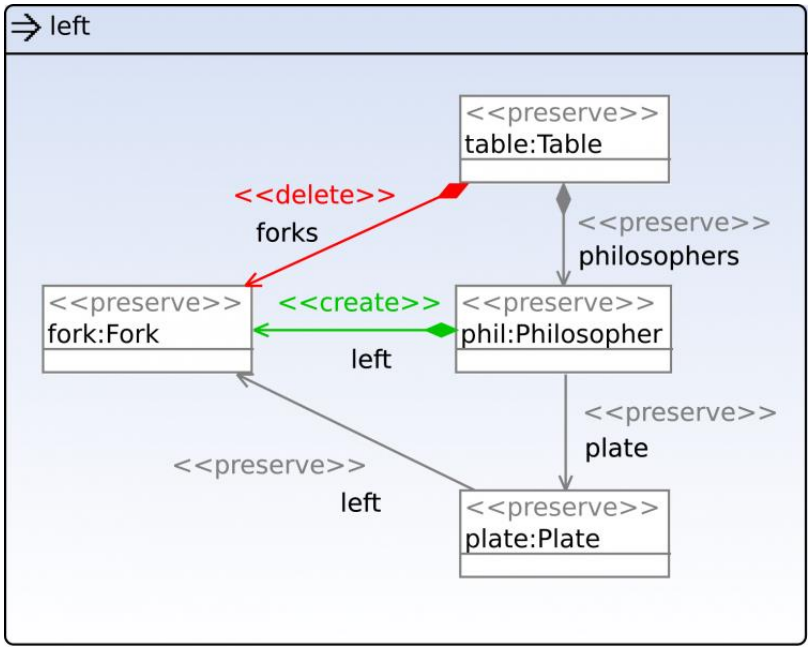
- Poměr *k*, k odhadované výšce BFS. Jako odhad výšky BFS používáme střední výšky ze sady modelů.



Obr. 14 *Výsledky odhadu na základě odběru vzorků* [1].

5 GENERÁTOR STAVOVÝCH PROSTORŮ

Generování SP je vytvoření určité datové struktury pomocí předpisů, udávaných použitým modelem. Můžeme využít již vytvořené generátory, nebo si naprogramovat vlastní. Pro generování SP týkajících se této práce bylo použito generátoru DIVINE. Tento generátor umožnuje testovat a generovat SP ze sady datových modelů BEEM, které obsahují dostatek variability. Využít se dá také vývojového prostředí eclipse a přídavného modulu henshin, nebo MatLabu. Výhoda tvorby v tomto prostředí eclipse je jednoduché (grafické) nastavení relací pro daný model. Pro ověření správnosti transformací, poskytuje Henshin nástroje na tvorbu a analýzu stavových prostorů k modelu transformace. Eclipse a modul Henshin obsahuje nástroje pro výslednou analýzu s grafickým výstupem. Na obr. 15 (převzatý z nápovědy) je ukázka vytváření relací mezi jednotlivými objekty v úloze „problém obědvajících filozofů“ [10].



Obr. 15 *Grafické prostředí modulu Henshin převzato z manuálu* [10].

5.1 Příkazy k ovládání programu Divine

V této části je ukázka příkazů k instalaci a ovládání programu Divine převzatých z manuálu dostupného online [4].

1. Rozbalíme soubor do nově vytvořeného adresáře abc pomocí příkaz $ tar.

$ tar abc divine-3.0.tar.gz

2. Pomocí skriptu zjistíme, jaké možnosti máme k dispozici, případně které balíčky nám chybí.

$ ./configure, pro podrobnější kontrolu pak: $ (cd _build ; ccmake .)

3. Instalace a sestavení. Provedeme příkazem:

$ make

Po sestavení ověříme funkčnost instalace.

$ cd _build/tools

Informace a nápověda:

$ ./divine help

Oveření:
$ ./divine verify ../../examples/dve/shuffle.dve
$ ./divine verify -p LTL ../../examples/dve/peterson-liveness.dve
$ ./divine verify -p LTL ../../examples/dve/peterson-naive.dve

4. Propojení s použitým modelem a nastavení.

divine info <soubor modelu>
divine combine [-f <soubor>] <model file>
divine compile [--cesmi|--llvm] <model file>

Vykreslení:
divine draw [...] <soubor modelu>
divine metrics [...] < soubor modelu >
divine verify [...] < soubor modelu >
divine simulate [...] < soubor modelu >

5. Vykreslení dat z modelu

divine info {model}
$ divine info examples/dve/peterson-liveness.dve

6. Konfigurace jádra kontrola modelu

divine {metrics|verify}
[--statistics [--curses]]
[-w N|--workers=N]
[--max-memory=N]

[--max-time=N]
[--disk-fifo=N]
[--seed=N]
[-i N|--initial-table=N]
[--hash-compaction]
[--compression]

7. Metrika SP vizualizace a prozkoumání

divine metrics [--reduce=R] [--no-reduce] [--fair]
[--report[=<report format>] | -r]
[--property=N] [--fair]
[engine options]
<soubor modelu>

divine draw [--distance=N] [--trace=N,N,N...]
[-l|--labels|--trace-labels]
[--bfs-layout]
[--reduce=R] [--no-reduce] [-f|--fair]
[--render=<cmd>|-r <cmd>]
[--compression]

8. Kontrola modelu

Tento příkaz automaticky vybere nejvhodnější algoritmus na ověření modelu

divine verify [Přepis algoritmu] [Výber vlastnosti] …

Přepis algoritmu:
--reachability, --owcty, --map, --nested-dfs

Výběr vlastnosti, která se na modulu bude kontrolovat:
--property=name | -p name

6 ZÁVĚR

Cílem této práce bylo popsat a ohodnotit vhodné metody na určení velikosti stavového prostoru. V kapitole 3 a 4 jsou tyto metody popsány. Hodnocení jednotlivých metod je však relativní. Každý datový model, nebo úloha vyžaduje samostatný přístup. Je to podmíněno nehomogenitou jednotlivých SP. Jako vhodný postup u komplexních SP se jeví použití gradientní prohledávání (s ukládáním do paměti), s jehož pomocí získáme parametry, podle kterých určíme další postup (volba vhodné metody). Nevýhodu gradientního prohledávání vidím u značně symetrických SP, kde by opakovaný průchod SP mohl vracet stejné hodnoty.

Před samotným hodnocením bylo zapotřebí zajistit data vycházející z určité úlohy (modelu). Vhodné datové modely pro porovnání je možné získat ze sady BEEM tyto modely jsou však značně rozsáhlé a relativně jednoduché postupy na určení VSP je přímo nevyžadují.

I velký SP je možné jednoduše vygenerovat pomocí modelovacího jazyku DiViNe, který má spoustu možností pro analýzu SP.

K objektivnímu posouzení jednotlivých stavových je vhodné zavést třídění podle velikosti do určitých intervalů.

Využití neuronových sítí jsem shledal jako vhodnou metodu, při opakovaných úlohách, na kterých se systém může učit při rozhodování. Tato metoda je popsána ve čtvrté kapitole. Na výstupu byl zvolen jeden neuron pro získání čísla, které reprezentovalo VSP.

Metody klasifikaci SP podle jejich parametrů jsou popsány v kapitole 4. Jednotlivé parametry bylo potřeba vyhodnotit podle jejich relevance k jednotlivým úlohám.

Zvážil jsem taktéž metodu klasifikace pomocí genetického programování. Tato metoda má širší použití než neuronové sítě a zdála se jako nejvhodnější pro klasifikaci stavového prostoru.

S ohledem na zmíněné metody pro určení VSP mohu konstatovat, že jednotlivé metody jsou velice závislé na zadaných parametrech a typech SP. Proto je pro každou úlohu potřeba určit vhodnou metodu zvlášť.

7 SEZNAM POUŽITÉ LITERATURY

[1] PELÁNEK, R.; ŠIMEČEK, P. Estimating State Space Parameters. Technická zpráva. Brno: Fakulta informatiky, Masarykova univerzita. Leden 2008. [FIMU-RS-2008- 01]. [online k 2. 4. 2015].
Dostupné také na http://www.fi.muni.cz/reports/files/2008/FIMU-RS-2008-01.pdf

[2] HLINĚNÝ, Petr. Základy Teorie Grafů. Verze 1.20. Brno: Fakulta informatiky, Masarykova univerzita v Brně, 2010, 135 s. [online k 6. 4. 2015].
http://is.muni.cz/do/rect/el/estud/fi/js10/grafy/Grafy-text10.pdf

[3] ČERNÝ, J. Průchod-grafu. Algoritmy. [online k 4. 3. 2015].
Dostupné z: http://algoritmy.eu/zga/pruchod-grafu/

[4] BARNAT, J.; BRIM, L.; ČERNÁ, I.; MORAVEC, P.; ROCKAI, P.; ŠIMEČEK, P. DiVinE – A Tool for Distributed Verification. In CAV'06, volume 4144 of LNCS, pages 278–281. Springer, 2006. [online k 7. 4. 2015] Nástroj dostupný na: http://anna.fi.muni.cz/divine.

[5] PELÁNEK, R. Web Portal for Benchmarking. Explicit Model Checkers. Technical Report FIMU-RS-2006-03, Masaryk University Brno, 2006. [online k 15. 3. 2015]. Modely dostupné na adrese http://anna.fi.muni.cz/models.

[6] EVANS, R., MINIEKA. E.: Optimization Algorithms for Networks and Graphs, Marcel Dekker, Inc. (1992), second edition, ISBN 0-8247-8602-5.

[7] PALÚCH, S. Algoritmická teória grafov. Žilina.: Žilinská univerzita, 2008. Teória grafov a grafové algoritmy. http://www2.fiit.stuba.sk/. [online k 4. 3. 2015].
Dostupné z:
http://www2.fiit.stuba.sk/~kvasnicka/Mathematics/05.tyzden/GrafyI.pdf

[8] Genetické programování. http://klobouk.fsv.cvut.cz/. [online k 4. 3. 2015].
Dostupné z:
http://klobouk.fsv.cvut.cz/~leps/teaching/mmo/prednasky/prednaska12_GP.pdf

[9] APPL, Jiří. Odhad velikosti stavového prostoru. Brno: Masarykova univerzita, 2009.

[10] Henshin State Space Tools. www.eclipse.org/. [online]. 13.6.2013 [cit. 2015-04-20].
Dostupné z: http://wiki.eclipse.org/Henshin_State_Space_Tools

[11] BARTUŇKOVÁ, Michaela. Náhodná procházka a její aplikace. Brno: Masarykova univerzita, 2007

[12] RIEDMILLER, M.; BRAUN, H. RPROP - A Fast Adaptive Learning Algorithm. Technická zpráva. Proc. of ISCIS VII, Universitat. 1992. [online k 8. 4. 2015].
Dostupné také na:
<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.52.4576&rep=rep1&type=pdf>

[13] O'NEILL, M.; RYAN, C. Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language. Genetic Programming. Springer. První vydání květen 2003. ISBN 978-1-4020-7444-8.

[14] KATRIEL, I.; MEYER, U. Elementary Graph Algorithms in External Memory. Lecture Notes in Computer Science, ročník 2625. kapitola 4. Berlin, Germany: Springer. 2003. ISBN 3-540-00883-7. s. 62–84. [online k 5. 5. 2015] Dostupné také na <http://www.springerlink.com/content/x4wpcpjwy6r6qadh/fulltext.pdf>

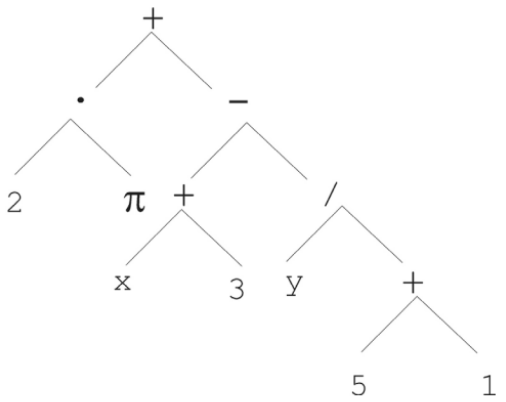
[15] JIROVSKÝ, L.; Teorie-grafu [online]. Praha: Matematicko-fyzikální fakulta Univerzity Karlovy, 2010 [cit. 2016-22-4].
Dostupné z: http://teorie-grafu.cz/zakladni-pojmy/reprezentace-v-pocitaci.php

[16] RYJÁČEK, Z.; Vzdálenost v grafech [online]. Plzeň: Západočeská univerzita v Plzni, [cit. 2016-25-4].
Dostupné z: http://www.cam.zcu.cz/~ryjacek/students/DMA/skripta/11.pdf

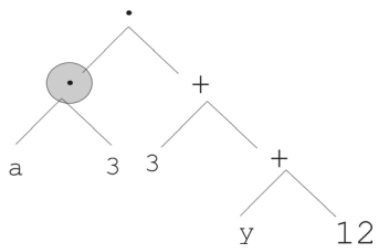
8 PŘÍLOHY

Příloha č. 1 *Genetické programování převod funkce na graf* [3].

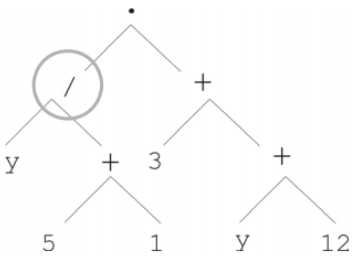
$$2 \cdot \pi + \left((x+3) - \frac{y}{5+1} \right)$$



Příloha č. 2 *Genetické programování rodič a potomek* [3].

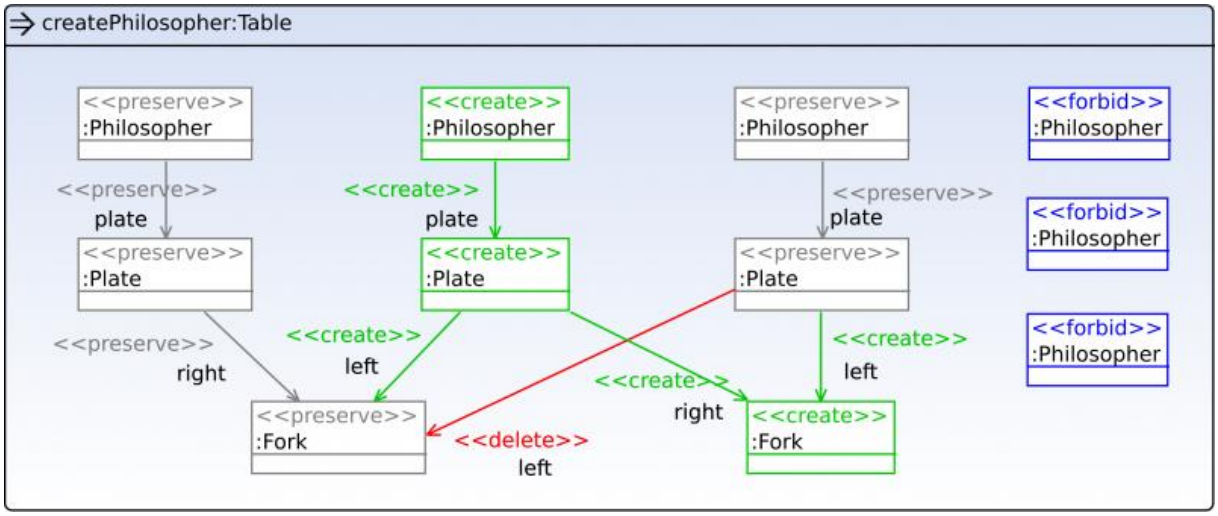


Rodič

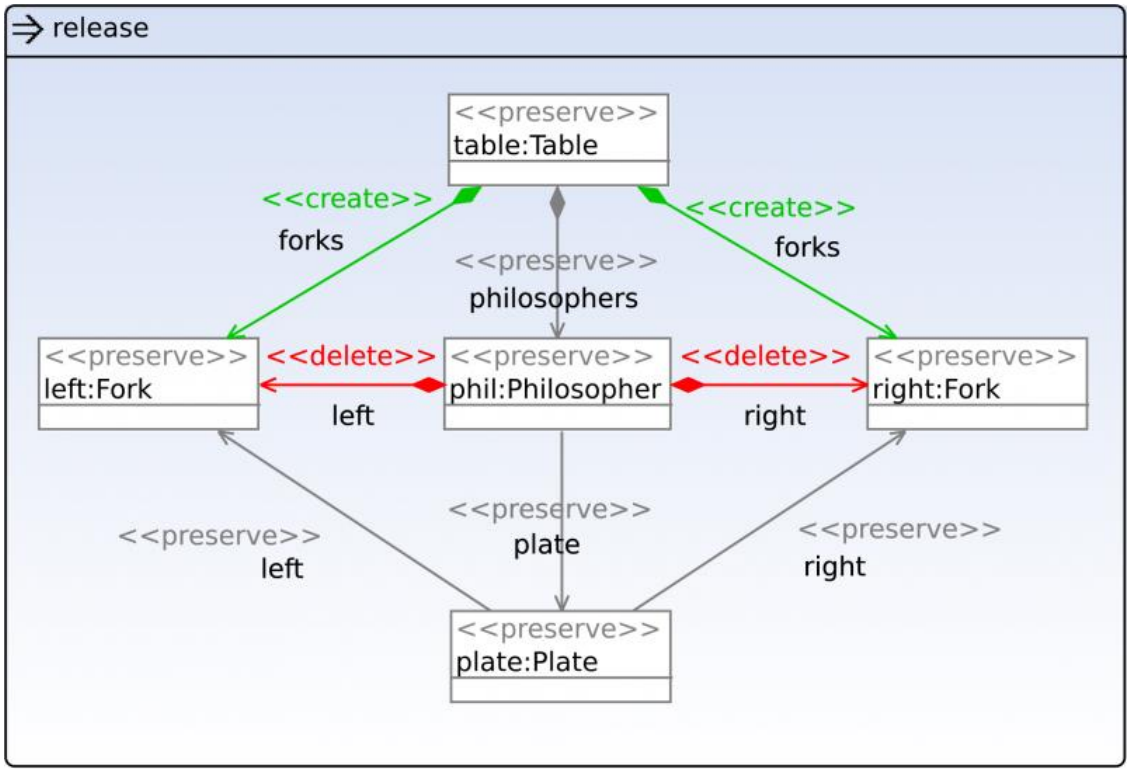


Potomek

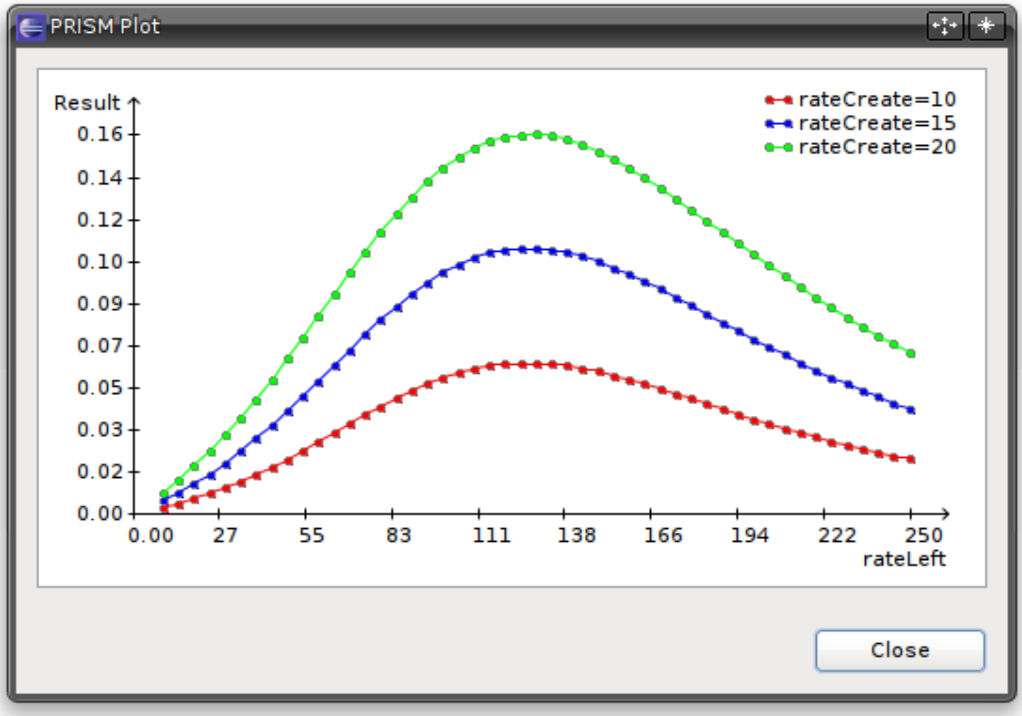
Příloha č. 3 *Ukázka vytváření relací mezi jednotlivými objekty pomocí modulu Henshin* [10].



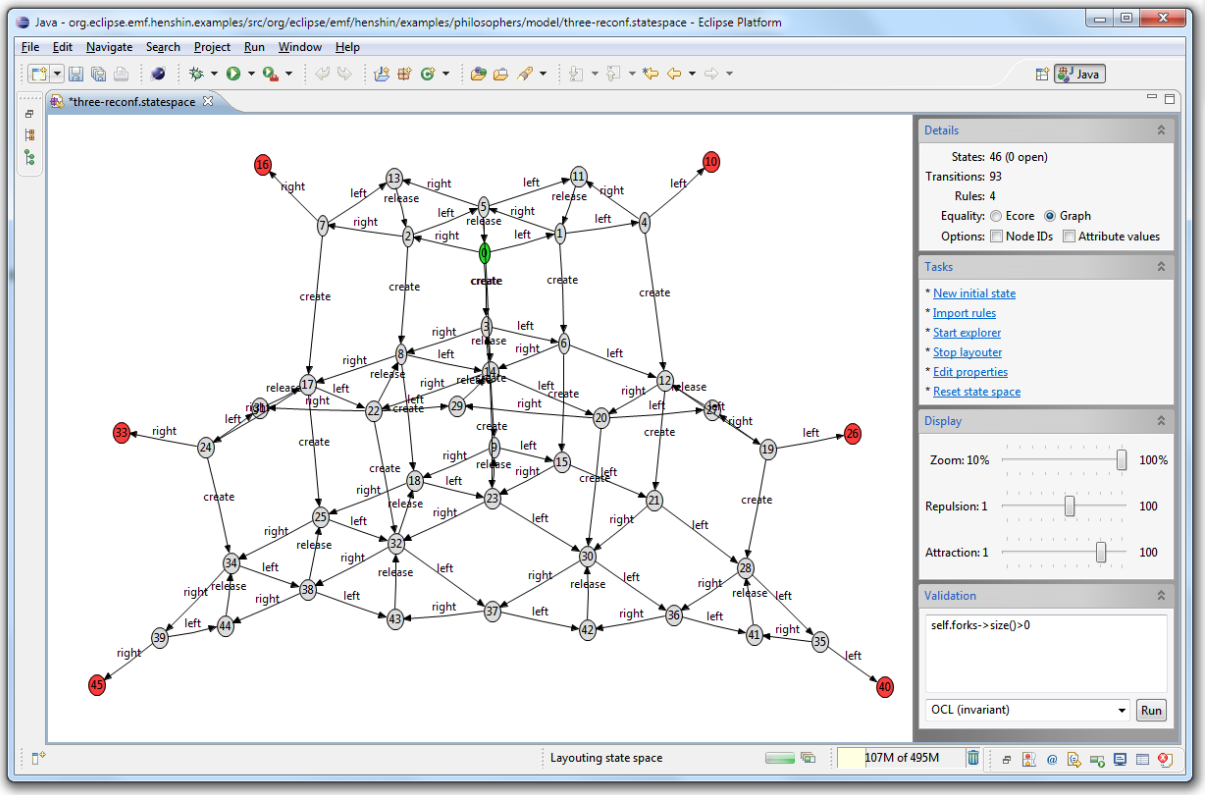
Příloha č. 4 *Ukázka vytváření relací mezi jednotlivými objekty pomocí modulu Henshin* [10].



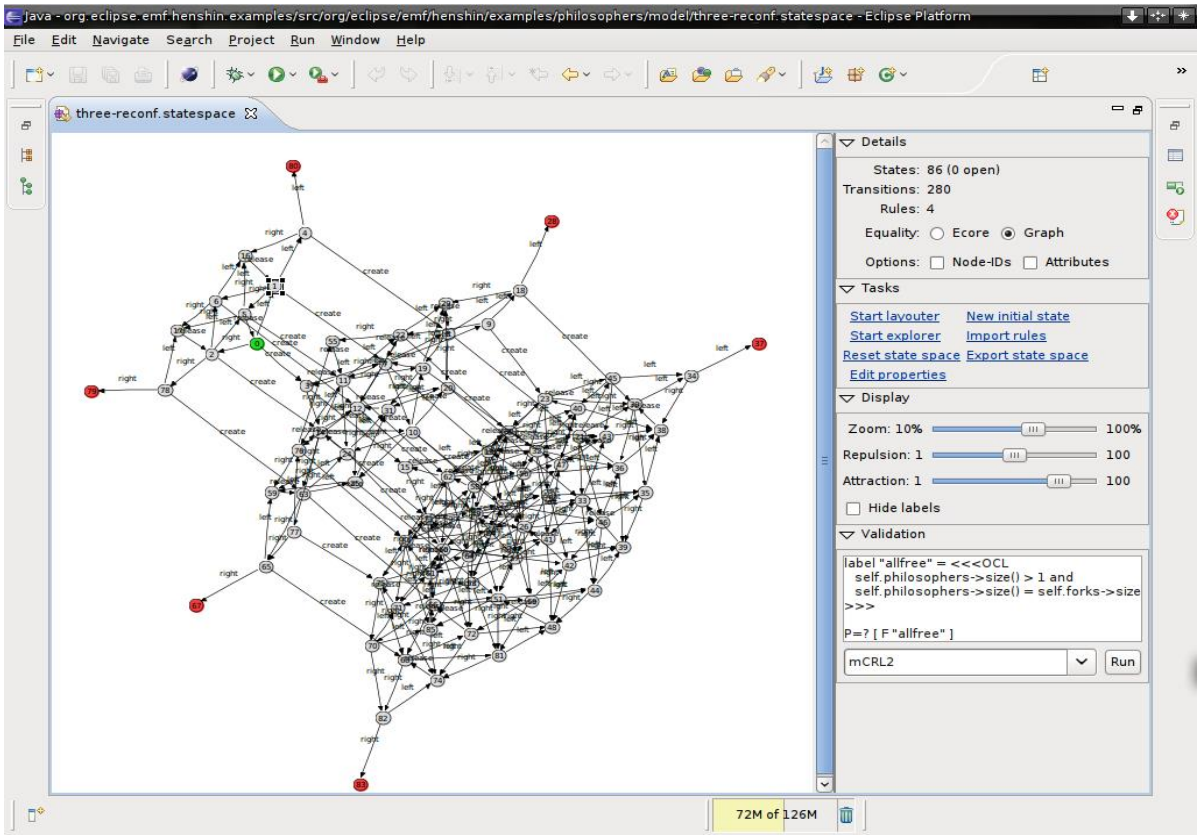
Příloha č. 5 *Vyhodnocení výsledků z úlohy „problém obědvajících filozofů“* [10].



Příloha č. 6 *Vygenerovaný SP v prostřednictvím modulu Henshin* [10].



Příloha č. 7 *Vygenerovaný SP v prostřednictvím modulu Henshin* [10].



Příloha č. 8 *Reprezentace grafu pomocí matice sousednosti převzato z* [15] *(Turbo Pascal 7).*

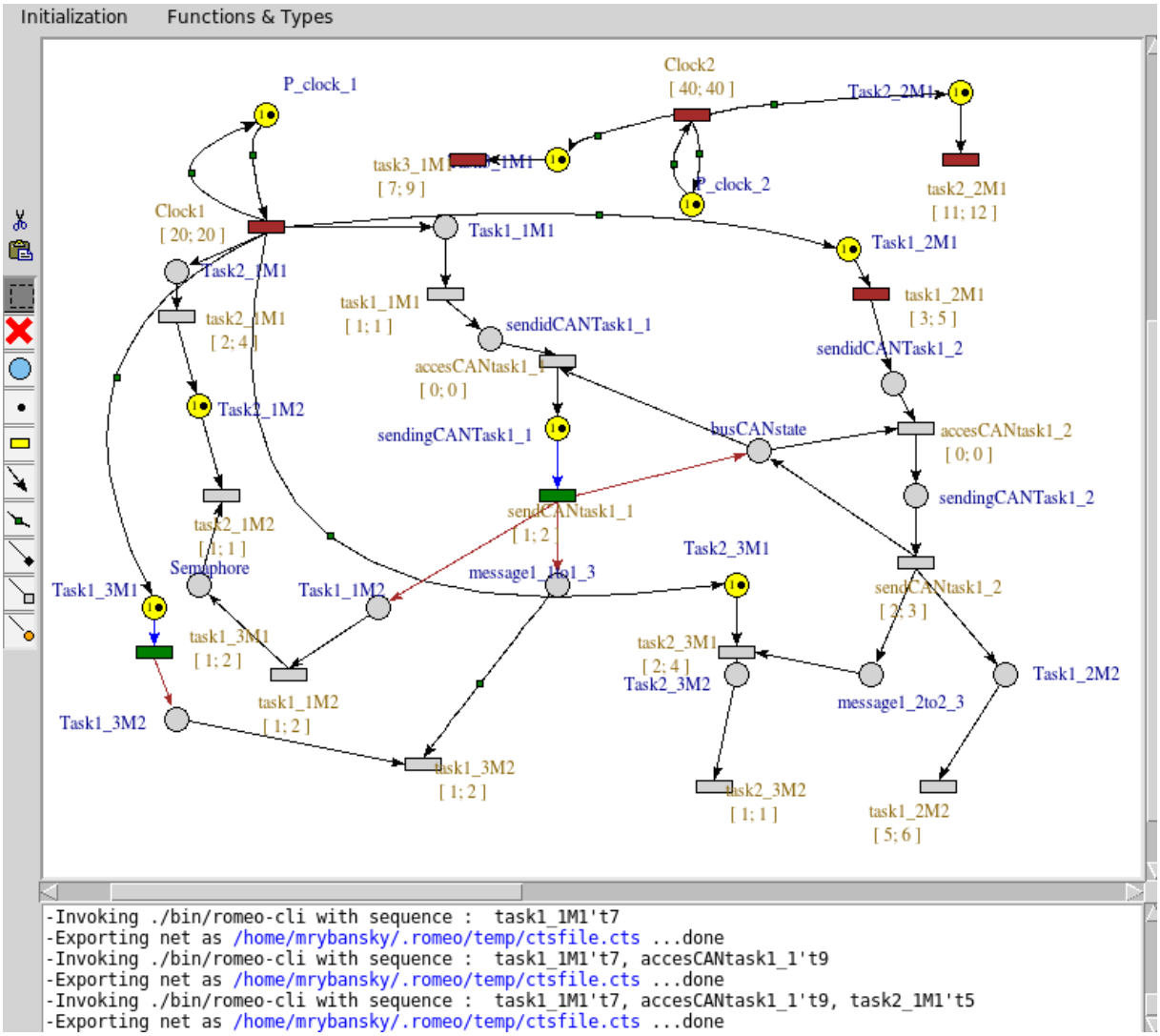
```
program GRAFMATICE1;

const
  velikost_matice = 10;
type
  typ_matice = array[1..velikost_matice,1..velikost_matice] of integer;
var
  matice1: typ_matice;
procedure vycistiMatici(var matice:typ_matice);
var
  i, j: integer;
begin
  for i:=1 to velikost_matice do
    begin
    for j:= 1 to velikost_matice do
      begin
      matice[i,j]:= 0;
      end;
    end;
end;
```

```
procedure zadejHranu(odkud: integer; kam: integer; var matice:typ_matice);
begin
  matice[odkud,kam]:= 1;
  matice[kam,odkud]:= 1;
end;
procedure odeberHranu(odkud: integer; kam: integer; var matice:typ_matice);
begin
  matice[odkud,kam]:= 0;
  matice[kam,odkud]:= 0;
end;
function jeHrana(odkud: integer; kam: integer; var matice:typ_matice):boolean;
begin
  if(matice[odkud,kam] = 1) then jeHrana:=true else
                              jeHrana:=false;
end;
procedure vypisMatici(var matice:typ_matice);
var
   i,j: integer;
begin
   writeln('Matice:');
   for i:=1 to velikost_matice do
    begin
     for j:= 1 to velikost_matice do
      begin
      write(matice[i,j]);
      end;
     writeln;
    end;
    writeln('--->KONEC MATICE---->');
    writeln;
end;

begin
  {samotny program}
  vycistiMatici(Matice1);
  vypisMatici(Matice1);
  {zadani nekolika testovacich hran}
  zadejHranu(1,2,Matice1);
  zadejHranu(2,3,Matice1);
  zadejHranu(2,4,Matice1);
  vypisMatici(Matice1);
  readln;
end.
```

Příloha č. 9 *Stavový prostor vyjádřený časovanou Petriho sítí (TPS) v softwaru Romeo.*



Příloha č. 10 *Model public_subscribe. Model se 2 uživateli.*

Formule	**Počet stavů**
1	1977587
2	3691177
3	2643849
4	4058497
5	4042105
6	4712717
7	4718837
8	3033021
9	3033301
10	3436730
11	1846603

9 SEZNAM POUŽITÝCH ZKRATEK

Seznam použitých zkratek viz tabulka 4.

Tab. 4

Zkratka	Význam
SP	Stavový prostor
NP	Náhodná procházka
DFS	Prohledávání do hloubky
BFS	Prohledávání do šířky
VSP	Velikost stavového prostoru
GP	Genetické programování