UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Pokročilý algoritmus prohledávání stavového prostoru

Javkhlan Naranbaatar

Bakalářská práce
2013

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2012/2013**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Javkhlan Naranbaatar**
Osobní číslo:	**I08121**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Pokročilý algoritmus prohledávání stavového prostoru**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

TEORETICKÁ ČÁST:
Teoretická část bude obsahovat úvod do problematiky prohledávání stavového prostoru (princip, příklady využití) a přehled metod prohledávání stavového prostoru s jejich stručným popisem. Algoritmy, které budou použity v praktické části práce, budou popsány podrobně.
PRAKTICKÁ ČÁST:
Cílem práce je vytvoření informovaného algoritmu prohledávání stavového prostoru a jeho aplikace na zadaný problém (hledání nejefektivnější cesty robota v předem známém prostředí s překážkami). Pro zadaný úkol bude navrženo několik hodnotících funkcí. Jejich vliv na efektivitu algoritmu bude vyhodnocen na základě dat získaných při řešení modelových situací.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

MAŘÍK V., ŠTĚPÁNKOVÁ O., LAŽANSKÝ J., Umělá inteligence (1), 1. vydání - dotisk, Praha : Academia, 2004. s. 264. ISBN 80-200-0496-3.

Vedoucí bakalářské práce: **Ing. Pavel Škrabánek**
Katedra řízení procesů

Datum zadání bakalářské práce: **21. prosince 2012**
Termín odevzdání bakalářské práce: **10. května 2013**

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 29. března 2013

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracovala samostatně. Veškeré literární prameny a informace, které jsem v práci využila, jsou uvedeny v seznamu použité literatury.

Byla jsem seznámena s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 5. 5. 2013 Javkhlan Naranbaatar

Poděkování

Na tomto místě bych ráda poděkovala svému vedoucímu panu Ing. Pavlu Škrabánkovi za pomoc, který mi poskytoval v průběhu zpracování bakalářské práce. Dále musím poděkovat své rodině za podporu během celého studia.

Anotace

Primárním cílem této bakalářské práce je navrhnout a programově realizovat několik různých hodnotících funkcí určených k vyhledávání cest a provést jejich srovnání na zadaných experimentech. Data získané touto cestou zpracovat a následně vyhodnotit vliv heuristické funkce na chování použitého algoritmu.

Klíčová slova

Prohledání stavového prostoru, A star algoritmus, plánování pohybu robota

Title

Advanced state space search algorithm

Annotation

The primary objective of this bachelor thesis is to design and implement several different heuristic functions designed for path search problems and compare them based on the experiments. Data obtained in this way is used to evaluate the impact of heuristic functions on the behavior of the algorithm.

Keywords

State space search, A star algorithm, robot motion planning

Obsah

Seznam zkratek a použitých symbolů

MS	množina stavů
MO	množina operátorů
MCS	množina cílových stavů
P	plán
U	úloha
S	stavový prostor
C	situace
A	akce
n	délka seznamu
$r(i)$	funkce pro vypočítání počtu otáčení i-tého stavu
$f(i)$	hodnotící funkce i-tého stavu
$g(i)$	cena vykonaného pohybu i-tého stavu
$h(i)$	heuristická funkce i-tého stavu pro odhad ceny budoucího pohybu
s_n	n-tý stav
o_n	n-tý operátor
x_c	souřadnice x cílového stavu
x_i	souřadnice x aktuálního stavu
y_c	souřadnice y cílového stavu
y_i	souřadnice y aktuálního stavu

Seznam obrázků

Seznam tabulek

1 Úvod

Vědní obor umělá inteligence prodělává v posledních desetiletích bouřlivý rozvoj. Pod pojem umělá inteligence spadá celá řada různých aplikačních oblastí. Příkladem mohou být expertní systémy či rozpoznávání obrazu a zvuku. Aplikace s umělou inteligencí využívají základní techniky a nástroje, mezi které bezesporu patří i metody prohledávání stavového prostoru.

Technikám prohledávání stavového prostoru se věnuje i tato práce. Základní pojmy a významné algoritmy jsou popsány v teoretické části práce. Praktická část je věnována aplikaci informovaného algoritmu prohledávání stavového prostoru na zadaný úkol. Prostor je zde věnována i posouzení vlivu formulace hodnotící funkce na běh algoritmu při řešení tohoto problému.

2 Cíl práce

Tato práce má hned několik cílů. Prvním je vytvoření obecného algoritmu informovaného prohledávání stavového prostoru.

Dalším cílem je aplikace tohoto algoritmu při hledání časově nejefektivnější cesty robota z bodu A do bodu B v předem známém prostředím s překážkami. Tento cíl vyžaduje mimo jiné i vytvoření nástavby s grafickým rozhraním, která bude umožňovat využití vytvořeného algoritmu prohledávání stavového prostoru na zadaný úkol. Součástí nástavby je i editor pro tvorbu map. Samozřejmým krokem při řešení tohoto cíle je i analýza a návrh řešení zadaného problému.

Obecně, součástí analýzy a návrhu řešení zadaného problému je i formulace hodnotící funkce. V této práci je nutné navrhnout hned několik hodnotících funkcí, protože třetím cílem je posouzení jejich vlivu na efektivitu algoritmu. Efektivita je hodnocena na základě dat získaných při řešení modelových situacích.

3 Teoretická část

3.1 Stavový prostor

Stavový prostor je uspořádaná dvojice, kterou tvoří množina stavů a množina operátorů, jejichž přesná formulace je popsána v následující podkapitole. Stavový prostor lze znázornit jako orientovaný graf, jehož každý uzel reprezentuje stav a každá orientovaná hrana přechod mezi stavy. Přechod mezi stavy je akce, jejímž vykonáním se dostaneme z jednoho stavu do druhého. Řešení úloh pak představuje, hledání cesty mezi počátečním a cílovým uzlem. Cílových stavů může být obecně více. Navíc cílový stav nemusí být popsán explicitně, může být popsán pouze podmínkami, které musí splňovat [1].

Stavový prostor může být buď konečný, anebo v některých případech i nekonečný. Konečný stavový prostor má omezené množství stavů, a to znamená, že od začátku je známo jeho celkový počet stavů.

3.1.1 Formulace úloh

V této části se pokusíme přesněji popsat formulaci problémů, při řešení úlohy ve stavovém prostoru. Informace uvedené v této podkapitole byla čerpána z knihy [1].

Stavový prostor $S = (MS, MO)$ kde:

- MS je konečná množina stavů, $MS = \{s_i\}$
- MO konečná množina operátorů reprezentujících přechod mezi stavy, $MO = \{o_j\}$

Úloha U nad stavovým prostorem S je dvojice $U = (s_0, MCS)$ kde:

- s_0 je počáteční stav, $s_0 \in MCS$
- MCS je množina cílových stavů, $MCS \subseteq MS$

Plánem P pro danou úlohu U (tj. řešením úlohy U) je taková posloupnost operátorů $P = (o_1, o_2, \dots, o_n)$ ke které lze přiřadit posloupnost stavů $(s_1, s_2, \dots, s_n)$, pro kterou platí

$$s_1 = o_1(s_0),$$
$$s_2 = o_2(s_1),$$
$$s_n = o_n(s_{n-1}),\ s_n \in MCS$$

Řešení úlohy je řízeno podle určité řídící strategie, realizované ve formě algoritmu. Jednou z velmi obecných možností popisu úlohy je *produkční systém*.

Produkční systém je tvořen:

- bází dat,
- souborem produkčních pravidel,
- řídicími strategií.

Báze dat popisuje okamžitý stav řešené úlohy. Některá část báze dat může být trvalá (když zachycuje trvale platné skutečnosti), zatímco další část zachycuje aktuálně platná data (tj. data platná v okamžiku, ve kterém se produkční systém právě nachází).

Produkční pravidla mají tvar:

$$\{\text{situace C}\} \rightarrow \{\text{akce A}\},$$

což lze slovně interpretovat takto: „Jestliže nastala situace C, vykonej akci A“.

Produkční systém pracuje v cyklech:

$$\{\text{rozpoznání situace}\} \rightarrow \{\text{vykonání akce}\}$$

Cílem provedení pravidla neboli akce je provést operaci, která povede ke změně obsahu báze dat. Produkční pravidla odpovídají operátorům o_j ve formulaci úlohy U.

Řídicí strategie určuje jak a v jakém pořadí aplikovat pravidla na bázi dat.

Principiálně rozlišujeme následující typy režimů řízení:

- přímý režim řízení (data-driven strategy) – postup od počátečního k některému z cílových stavů,
- zpětný režim řízení (goal-driven strategy) – postup od cílů k počátečnímu stavu.

V prvním případě se hovoří o strategii řízené daty a v druhém případě o strategii řízené cílem. Při prohledávání stavového prostoru se většinou používá režim přímý.

Každá řídící strategie, má-li být úspěšnou, musí splňovat tyto dvě základní vlastnosti:

- vést k prohledávání (zabránit zacyklení),
- být systematická.

3.2 Prohledávání stavového prostoru

Prohledávání stavového prostoru je obecná technika k nalezení řešení úloh, jež lze reprezentovat pomocí stavového prostoru. Sem spadají i některé úlohy z oblasti umělé inteligence. Rozlišujeme dva základní typy prohledávacích metod:

- neinformované metody prohledávání,
- informované metody prohledávání.

Oba tyto typy algoritmů jsou založeny na obdobném principu činnosti (jejich principy budou podrobně popsány v kapitolách 3.3 a 3.4), přičemž pro aktuální stav se provádí tzv. expanze. Expanzí stavu se myslí aplikace všech použitelných operátorů na daný stav, čímž se získají všichni jeho následníci. Dále algoritmy prohledávání stavového prostoru obvykle využívají dva následující seznamy [6]:

- seznam expandovaných stavů,
- seznam neexpandovaných stavů.

V každém kroku prohledávání stavového prostoru řídící mechanismus vybere pravidlo. Jeho aplikací na aktuální stav, vzniká stav nový, čímž se generuje tzv. strom stavů. V případě přímého řízení se nejprve expanduje počáteční stav s_0. V dalším průběhu prohledávání se pak expandují některé z dříve generovaných stavů. Právě výběr vhodného stavu k expanzi představuje ve většině případů hlavní rozdíl mezi jednotlivými algoritmy.

Jestliže nově vygenerovaný stav „s“ náleží do množiny cílových stavů MCS, je prohledávání u většiny algoritmů ukončeno, protože ve stromu existuje orientovaná cesta od s_0 k s [1].

Jestliže má být některý z algoritmů aplikován na úlohu, která je reprezentována rozsáhlým stavovým prostorem, nemusí vést systematické prohledávání k nalezení řešení, a to jak z časových tak i technických důvodů. Je totiž pravděpodobné, že se zbytečně prohledává značná část stavového prostoru, která nevede k cíli. Rozsah prohledávání prostoru můžeme omezit využitím znalostí. Tyto znalosti nazýváme znalostmi heuristickými (heuristikami) a někdy mají empirický charakter. Mohou to být neexaktní poznatky, které jsou často užitečné při řešení, ale nemusí zaručit, že vždy nalezneme řešení. Heuristiky se používají tam, kde není k dispozici exaktní algoritmus.

Způsob hodnocení metod prohledávání stavového prostoru

Existují v podstatě tři základní kritéria, podle kterých lze hodnotit výkonnost zvoleného algoritmu.

- Časová složitost – čas potřebný k vyřešení úlohy danou metodou.
- Prostorová složitost – množství operační paměti potřebné k řešení úlohy.
- Kvalita získaných výsledků – zda je daná metoda úplná (nalezne řešení vždy, když existuje) a optimální (nalezené řešení je nejlepší ze všech).

3.3 Neinformované metody prohledávání

Metody neinformované dělíme z hlediska pořadí, ve kterém jsou uzly expandovány, na slepé prohledávání do šířky a slepé prohledávání do hloubky. Hloubkou uzlu ve stromu řešení rozumíme počet hran na cestě od počátečního uzlu k uzlu aktuálnímu. K popisu všech následujících algoritmu prohledávání, zavedeme dva seznamy, a to seznam neexpandovaných stavů OPEN a seznam již expandovaných stavů CLOSED [1].

3.3.1 Slepé prohledávání do šířky (breadth-first search)

Při tomto algoritmu se nejdříve expanduje uzel s minimální hloubkou a postupně procházíme strom řešení po vrstvách, přičemž se k expanzi vybírají vždy uzly s nejmenší hloubkou. Algoritmus tohoto způsobu prohledávání si pamatuje celou generovanou strukturu stromu řešení. Stavy se kterými, už algoritmus pracoval, se znovu již negenerují. Výhodou slepého prohledávání do šířky je, že vždy nalezneme optimální řešení (koncový stav s nejmenší hloubkou). Tento algoritmus může být realizován jako fronta FIFO [6].

Algoritmus prohledávání do šířky:

1. Zapíšeme počáteční stav do seznamu OPEN, seznam CLOSED je prázdný. Jestli je počáteční stav stavem cílovým – ukončíme prohledávání.
2. Jestli je seznam OPEN prázdný – řešení neexistuje a ukončíme prohledávání.
3. Vymažeme první stav (označíme jej *i*) v seznamu OPEN a zapíšeme tento stav do seznamu CLOSED.
4. Expandujeme stav *i*. Jestliže tento stav nemá následovníky nebo všichni jeho následovníci byli již expandováni (jsou v seznamu CLOSED) – pokračujeme krokem č. 2.
5. Napíšeme všechny následovníky stavu *i*, kteří nejsou v seznamu CLOSED na konec seznamu OPEN.
6. Jestli je některý z následovníků *i* cílovým stavem, bylo nalezeno řešení a ukončíme prohledávání. Jinak pokračujeme krokem č. 2.

3.3.2 Slepé prohledávání do hloubky (depth-first search)

Při tomto prohledávání se přednostně expandují uzly s největší hloubkou. Algoritmus lze realizovat jako zásobník LIFO. Na rozdíl od prohledávání do šířky, při tomto prohledávání si algoritmus v základní verzi pamatuje vždy jen jedinou cestu od počátečního k poslednímu vygenerovanému stavu a můžeme tedy některými uzly procházet vícekrát, neboť se často musíme navracet (tzv. backtracing). Nevýhodou backtrackingu s neomezenou hloubkou prohledávání je možnost nájezdu algoritmu na jakousi nekonečnou cestu, která neobsahuje řešení. Tomu se dá zabránit nastavením maximální přípustné hloubky prohledávání [1] a [6].

Je tedy jasné, že žádná z variant tohoto algoritmu nezaručuje nalezení řešení, natož řešení optimálního.

Výhodou slepého prohledávání do hloubky je snadná implementovatelnost a nižší nárok na paměť. Nevýhodou je, že se musí znovu procházet a rozvíjet stavy, ve kterých už

algoritmus byl, což může mít větší časovou náročnost. Níže je popsána činnost algoritmu s omezením hloubky.

Algoritmus prohledávání do hloubky:

1. Zapíšeme počáteční stav do seznamu OPEN, seznam CLOSED je prázdný. Jestliže je počáteční stav stavem cílovým – ukončíme prohledávání.
2. Jestli je seznam OPEN prázdný – řešení neexistuje a ukončíme prohledávání.
3. Vymažeme první stav (označíme jej *i*) v seznamu OPEN a zapíšeme tento stav do seznamu CLOSED.
4. Jestli hloubka uzlu *i* rovna maximální přípustné hloubce, pokračujeme krokem č. 2.
5. Expandujeme stav *i*. Jestliže tento stav nemá následovníky nebo všichni jeho následovníci byli již expandování (jsou v seznamu CLOSED) – pokračujeme krokem č. 2.
6. Zapíšeme všechny následovníky stavu *i*, kteří nejsou v seznamu CLOSED na začátek seznamu OPEN.
7. Jestli některý z následovníků *i* je cílovým stavem – řešení bylo nalezeno a ukončíme prohledávání. Jinak pokračujeme krokem č. 2.

3.4 Informované metody prohledávání

Informované metody prohledávání využívají kritérií (heuristiky), které umožňují lepší volbu stavu vhodného k expanzi. Snaží se vybrat takový stav, který slibuje co nejrychlejší dosažení některého z cílových stavů. Využívají k tomu tzv. hodnotící funkce.

Podle charakteru úlohy je možné definovat hodnotící funkci $f(i)$, která ke každému stavu s přiřadí číslo, vyjadřující jeho kvalitu z hlediska řešení úlohy. Tyto hodnoty jsou pak používány k výběru uzlu pro expanzi. Je tedy pravděpodobné, že budou expandovány nejperspektivnější uzly a tím se zabrání prohledávání cest, které nevedou k cíli. Čím budou heuristické znalosti o daném problému kvalitnější, tím efektivnější bude prohledávání. Existují různé druhy informovaných metod. V této práci jsou uvedeny jen některé základní [1], [6] a [8].

3.4.1 Gradientní algoritmus

Gradientní algoritmus lze chápat jako heuristické prohledávání do hloubky. Při tomto algoritmu se expanduje ten uzel, který byl doposud vyhodnocen pomocí hodnotící funkce $f(i)$ jako nejlepší, kde $f(i)$ je odhad vzdálenosti k cílovému stavu. Následníky expandovaného uzlu nejprve uspořádáme dle hodnoty funkce $f(i)$ a pak je zařadíme do zásobníku. Následovník, který má nejlepší ohodnocení funkcí $f(i)$, bude vybrán k další expanzi. Rodič i sourozenci tohoto uzlu jsou ihned zapomenuti a v paměti je uložen jen expandovaný uzel. Prohledávání bude zastaveno, pokud dosažený stav má lepší ohodnocení funkcí $f(i)$, než jeho následovníci. Takový postup prohledávání může způsobit řadu problémů: [6]

- uváznutí v lokálním extrému (žádný následník není lepší než expandovaný stav, který ale není cílovým stavem),
- problém plošiny (expandovaný stav i jeho následníci jsou stejně ohodnoceny, není tedy jasné kterým směrem postupovat).

Navíc, vzhledem k tomu, že se algoritmus neuchovává historie prohledávání, není vyloučen ani pohyb po nekonečné dlouhé cestě a tím dojít k zacyklení [1].

3.4.2 Algoritmus uspořádaného prohledávání (best-first search algorithm)

Tento algoritmus vznikl jako rozšíření gradientního algoritmu o paměť a tato paměť je interpretována seznamy OPEN a CLOSED. Prvky těchto seznamu jsou ve formátu:

<jméno uzlu, hodnota *f*, jméno rodičovského uzlu>.

Popis základní verze algoritmu uspořádaného prohledávání:

1. Zapíšeme počáteční stav do seznamu OPEN, seznam CLOSED je prázdný. Jestli je počáteční stav stavem cílovým – ukončíme prohledávání.
2. Jestli je seznam OPEN prázdný – řešení neexistuje a ukončíme prohledávání.
3. Ze seznamu OPEN vybereme stav i s nejmenší hodnotou $f(i)$. V případě většího počtu stavů se stejnou hodnotou funkce $f(i)$ prověříme, zda některý ze stavů není stavem cílovým, v takovém případě jej vybereme. Jinak vybereme libovolně.

4. Vymažeme stav *i* ze seznamu OPEN a zapíšeme tento stav do seznamu CLOSED.
5. Jestli je stav *i* cílovým stavem – řešení je nalezeno a ukončíme prohledávání.
6. Expandujeme stav i, pro každého následovníka j stavu i, vypočteme hodnotu funkce $f(i)$. Jestli stav j není ani v seznamu OPEN nebo CLOSED, zařadíme jej do seznamu OPEN. Jestli stav j je již v seznamu OPEN nebo CLOSED, ale s ohodnocením větším než právě vypočtené $f(i)$, změníme jeho ohodnocení na $f(j)$, změníme rodičovský uzel v zápisu uzlu a zařadíme ho do seznamu OPEN.
7. Pokračujeme krokem č. 2.

Kromě výše uvedené verze algoritmu existuje i varianta *paprskovité prohledávání s aperturou n*, která je jeho modifikací. Rozdíl spočívá v tom, že se užívá seznamu OPEN konečné délky *n*, přičemž v každém kroku se do seznamu OPEN zapíší jen ty nově expandované uzly, které mají lepší ohodnocení než uzly dosud zachycené v seznamu OPEN [1].

Další možností je použití algoritmu A, která pracuje s hodnotící funkcí $f()$ ve tvaru:

$$f(i) = g(i) + h(i) \tag{3.1}$$

Kde

- $g(i)$ - optimální cena cesty do stavu i z počátečního stavu s_0,
- $h(i)$ - optimální cena cesty ze stavu i do cílového stavu.

Pokud položíme $h(i) = 0$, je prohledávání řízeno pouze funkcí $g(i)$ – získáme tzv. algoritmus se stejnoměrnou cenou.

Jestli $h(i) = 0$ a $g(i) = 0$ pro všechna i, degeneruje algoritmus na náhodné prohledávání. Pokud volíme cenu každého kroku rovnou jedné a $h(i) = 0$, odpovídá $g(i)$ hloubce i-tého uzlu a algoritmus degeneruje na slepé prohledávání do šířky.

Ve většině praktických úloh však funkce $g(i)$ a $h(i)$ neznáme, a proto je nahrazujeme jejich odhady, v tomto případě hovoříme o algoritmu A*, který je blíže popsán v následující podkapitole.

3.4.3 Algoritmus A^*

Jak již bylo výše zmíněno, algoritmus A^* (A star) je modifikací algoritmu A, také je velmi podobný algoritmu prohledávání do šířky, avšak s rozdílem, že místo obyčejné fronty využívá frontu prioritní, ve které jsou stavy seřazeny podle hodnoty hodnotící funkce $f(i)$. Hodnotící funkce $f(i)$ má tvar:

$$f(i) = g^*(i) + h^*(i) \tag{3.2}$$

Kde

- $g^*(i)$ - odhad funkce $g(i)$
- $h^*(i)$ - odhad funkce $h(i)$

Funkci $g(i)$ odhadneme minimální dosud zjištěnou cenou $g^*(i)$ přechodu z počátečního stavu do stavu i a funkci $h(i)$ nahrazujeme funkcí $h^*(i)$, která kvantitativně vyjadřuje náš

odhad cesty z uzlu i do cíle. Tento odhad vyjadřuje naši heuristickou znalost o tom, jaké jsou šance nalézt optimální řešení z uzlu i. Je zřejmé, že tato funkce je pro efektivitu prohledávání velmi podstatná. Pro přípustné algoritmy platí podmínka $0 \leq h^*(i) \leq h(i)$ přitom, ale skutečnou $h(i)$ neznáme [1] a [8].

Algoritmus A*:

1. Zapíšeme počáteční stav do seznamu OPEN, seznam CLOSED je prázdný. Jestli je počáteční stav stavem cílovým – konec prohledávání.
2. Jestli je OPEN prázdný – řešení neexistuje – konec prohledávání.
3. Vybereme stavy s nejmenší hodnotou $f(i)$ ze seznamu OPEN.
4. Pokud některý z nich je cílovým stavem, zjisti cestu od počátečního stavu do cílového stavu a ukončíme prohledávání.
5. Jinak vybereme první stav i ze seznamu stavů z předchozího bodu, které mají nejmenší hodnotu $f(i)$. Vymažeme stav i ze seznamu OPEN, zařadíme ho do seznamu CLOSED.
6. Expandujeme vybraný stav i a pro všechny jeho následníky j vyhodnocujeme hodnotu $f(j)$. Do seznamu OPEN zapíšeme ty, kteří nejsou v seznamu OPEN ani v CLOSED. Jestliže některý z následníků j je už v seznamu OPEN (stav k), ale s větší hodnotou hodnotící funkce ($f(k) > f(j)$), tak odebereme existující stav k a přidáme následníka j do seznamu OPEN. Jinak pokračujeme krokem 3.

3.4.4 Algoritmus větví a mezi (branch-and-bound)

Tento algoritmus je rozšířeným algoritmu uspořádaného prohledávání. Prohledává strom stavového prostoru tak dlouho, dokud není nalezeno optimální řešení. Jestliže je nalezen nějaký cílový stav, algoritmus si zapamatuje jeho „cenu“ a pokračuje dále v prohledávání stavového prostoru. Ze seznamu OPEN automaticky vyřazuje všechny stavy s vyšší cenou, než je cena zapamatovaného řešení. Jestli je nalezeno řešení s nižší „cenou“, zapamatuje se toto řešení a pokračuje se až do situace, kdy je zásobník OPEN prázdný. Takový algoritmus nalezne vždy optimální cestu k cíli, přitom však vede k expanzi uzlů i na cestách, které nevedou k cíli, ale mají nižší „cenu“ [1] a [7].

Algoritmus větví a mezí:

1. Zapíšeme počáteční stav do seznamu OPEN, seznam CLOSED je prázdný. Jestli je počáteční stav stavem cílovým – konec prohledávání. Nastavíme hodnotu aktuální hodnotu f_{nej} na plus nekonečno.
2. Jestli je OPEN prázdný – řešení neexistuje – konec prohledávání.
3. Ze seznamu OPEN vybereme stav i s nejmenší hodnotou $f(i)$. V případě většího počtu stavů se stejnou hodnotou funkce $f(i)$ prověříme, zda některý ze stavů není stavem cílovým, v takovém případě jej vybereme. Jinak vybereme libovolně.
4. Vymažeme stav i ze seznamu OPEN a zapíšeme tento stav do seznamu CLOSED.
5. Jestli je stav i cílovým a hodnotící funkce $f(i)$ je nižší než f_{nej}, zapamatujeme si tento stav a nastavíme $f_{nej} = f(i)$. Jestliže hodnota f_{nej} byla změněna, vyřazujeme všechny stavy v seznamu OPEN, které mají vyšší hodnotu $f()$ než nová f_{nej}.

6. Expandujeme stav i a pro všechny následníky j vyhodnocujeme hodnotící funkci $f(j)$ a zapíšeme je do seznamu OPEN. Pokračujeme krokem 3.

Všechny doposud uvedené algoritmy mají charakter základních algoritmů prohledávání stavového prostoru. Při řešení reálných úloh a při tvorbě praktických systémů bývají tyto nejjednodušší algoritmy kombinovány s některými dalšími technikami. V následující podkapitole se seznámíme s některými z těchto technik.

3.5 Pokročilé techniky

3.5.1 Pravidla se složitějšími předpoklady

Jednou z možností využití znalostí při prohledávání stavového prostoru je zahrnout znalosti do předpokladů pravidel v podobě dalších předpokladů. Jelikož jsou pravidla specifičtější, sníží se počet generovaných stavů, tím pádem se prohledává menší část stavového prostoru a případné řešení je nalezeno rychleji [1].

3.5.2 Hledání v hierarchicky uspořádaném stavovém prostoru

Tato technika je založena na myšlence, že lze danou úlohu popsat v různých stavových reprezentacích, které se liší popisem a jsou hierarchicky uspořádány. *Obecnější prostor* je menší, protože každý jeho stav odpovídá více stavům méně obecného prostoru. Proto je hledání v něm jednodušší. Každý krok je rozložen na posloupnost kroků v méně obecném prostoru, až je nalezeno výsledné řešení v nejméně obecném. Přitom ty části obecnějšího prostoru, které nejsou součástí řešení nalezeného v tomto prostoru, nejsou již uvažovány v prostorech méně obecných [1].

3.5.3 Metaznalosti

Velmi efektním způsobem využívání znalostí o úloze je vyjádření znalostí ve formě samostatných pravidel (metapravidel). Těmito se pak řídí používání pravidel, která definují úlohu. Tento způsob dovoluje oddělit pravidla, která definují úlohu, od pravidel, která pomáhají tuto úlohu řešit, tj. od řídící strategie. Metapravidla jsou obvykle popsána ve tvaru *{Preferuj pravidlo1 před pravidlo2}* [1].

3.5.4 Metoda generování a testování

Tato metoda je jinou alternativou heuristického hledání. Generátor nám nabízí jednotlivá řešení (provádí expanzi stavu), tato jednotlivá řešení jsou testovaná a odmítána, jestli nesplňují zadaná omezení. Znalosti o dané úloze můžeme vhodně rozdělit mezi generátorem a testerem, což může zvýšit účinnost prohledávání. Velmi efektivní je metoda hierarchicky uspořádaného generování a testování, která umožňuje odmítnout řešení na základě jenom jeho částečného popisu (jediným testem lze naráz zamítnout větší počet nevyhovujících řešení) [1].

3.5.5 Metoda užití analogie

Další důležitou metodou je užití analogie. Jestliže je odhalena určitá podobnost mezi dvěma stavovými prostory, lze řešení úlohy v jednom prostoru použít jako podklad pro řešení úlohy v druhém prostoru, nebo můžeme využít analogie s řešením obdobného případu [1].

4 Praktická část

Tato část práce se zabývá analýzou, realizací a vyhodnocením zadaného příkladu. Hlavními úkoly praktické části této práce jsou vytvoření informovaného algoritmu a navržení několika různých hodnotících funkcí pro zadaný příklad. Na tomto příkladu je také sledován vliv hodnotících funkcí na efektivitu algoritmu při řešení úlohy.

4.1 Analýza zadání

Úkolem je nalézt časově nejefektivnější cestu robota v předem známém prostředí z počáteční pozice (viz obrázek 1) do pozice cílové (viz obrázek 2).

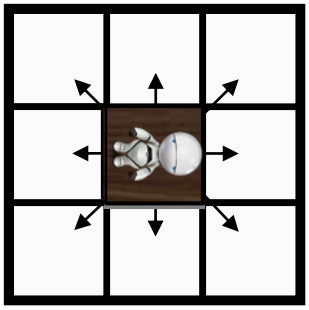
Obrázek 1 – Reprezentace robota

Obrázek 2 – Reprezentace cíle

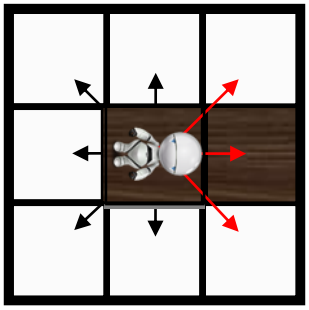
V prostředí, ve kterém se robot pohybuje, se mohou vyskytovat různé nepřekonatelné překážky (nábytek, zdi a podobně). Příklad takovéhoto prostředí znázorněného pomocí mapy je uveden na následujícím obrázku, kde všechna neprázdná políčka, kromě dvou výše uvedených, představují nepřekonatelné překážky.

Obrázek 3 – Příklad reprezentace prostředí

Robot se může pohybovat celkem osmy směry (do sousedních 8 polí), pokud mu nepřekáží v cestě žádné překážky (budoucí pole je prázdná). Robot je směrován tam, kam míří jeho hlava (v ukázkovém příkladu na obrázcích 4 a 5 je robot orientován směrem doprava) a může se pohybovat pouze směrem vpřed (nelze couvat). Pro diagonální posun je třeba, aby i sousední pole byla prázdná (příklad nepovoleného posunu je uveden níže (viz obrázek 5), kde jsou nepovolené posuny označeny červenými šipkami).



Obrázek 4 – Směr robota



Obrázek 5 – Příklad nepovoleného posunu

4.2 Hodnotící funkce *f*

Všechny hodnotící funkce f byly pro tuto úlohu navrženy tak, aby zachycovaly časovou náročnost přesunu robota. Výpočet f pro *i*-tý stav je obecně popsán vzorcem (3.2), kde $g^*(i)$ představuje odhad časové náročnosti přesunu robota z počátečního stavu do stavu aktuálního tj. cenu vykonaného pohybu. Hodnota funkce $h^*(i)$ představuje odhad času potřebného k dosažení stavu cílového (odhad budoucího pohybu).

4.2.1 Cena vykonaného pohybu

Cena pohybu nutného k dosažení pozice i je označena jako $g^*(i)$ a vypočítá se podle vzorce (4.1).

$$g^*(i) = g^*(i-1) + g_p(i) \tag{4.1}$$

Kde:

- $g^*(i-1)$ je cena pohybu robota z počáteční pozice do pozice $i-1$.
- $g_p(i)$ je cena pohybu robota z pozice $i-1$ do pozice i, určuje se podle vzorce (4.2).

$$g_p(i) = p(i) + r(i) \tag{4.2}$$

Proměnná $p(i)$ v tomto vzorci odpovídá ceně samotného posunu a může nabývat hodnotu 1 (přímý posun) nebo $\sqrt{2}$ (diagonální posun). Proměnná $r(i)$ je nejmenší možný počet otočení robota o 45° k dosažení požadovaného směru. Proměnná $r(i)$ je určována vůči natočení robota v pozici $i-1$ a může nabývat celočíselných hodnot z intervalu $< 0{,}4 >$. Výpočet ohodnocení ceny otáčení robota při přesunu z pozice $i-1$ do pozice i bude nejlépe patrný z následujícího příkladu.

Mějme situaci, kde bude robot orientován směrem doprava, viz obrázek 4, pak cena pohybu jednotlivými směry bude určena podle následující tabulky.

Tabulka 1 – Příklad vypočítání ceny pohybů

Směr	Ohodnocení pohybu
Vpravo	$g_p(i) = 1 + 0 = 1$
Vpravo nahoru	$g_p(i) = \sqrt{2} + 1 \approx 2{,}4142$
Vpravo dolů	$g_p(i) = \sqrt{2} + 1 \approx 2{,}4142$
Nahoru	$g_p(i) = 1 + 2 = 3$
Dolů	$g_p(i) = 1 + 2 = 3$
Vlevo nahoru	$g_p(i) = \sqrt{2} + 3 \approx 4{,}4142$
Vlevo dolů	$g_p(i) = \sqrt{2} + 3 \approx 4{,}4142$
Vlevo	$g_p(i) = 1 + 4 = 5$

4.2.2 Odhad ceny budoucího pohybu

Odhad ceny budoucího pohybu se nazývá heuristická funkce a označuje se jako h^*. Právě tento člen posloužil k vytvoření různých variant hodnotící funkce f. Byly navrženy celkem tři heuristické funkce, které jsou popsány následujícími vztahy:

1. Bez ceny odhadu budoucího pohybu (nulový odhad)
$$h^*(i) = 0 \tag{4.3}$$

2. Hodnocení podle Manhattanské vzdálenosti
Toto kritérium předpokládá, že je pohyb možný jen ve 4 směrech [3].
$$h^*(i) = |x_c - x_i| + |y_c - y_i| \tag{4.4}$$

3. Hodnocení podle Euklidovské vzdálenosti
Toto kritérium předpokládá, že je pohyb možný ve všech 8 směrech [3].
$$h^*(i) = \sqrt{(x_c - x_i)^2 + (y_c - y_i)^2} \tag{4.5}$$

kde**:**

- x_c – souřadnice x cílového stavu,
- x_i – souřadnice x aktuálního stavu,
- y_c – souřadnice y cílového stavu,
- y_i – souřadnice y aktuálního stavu.

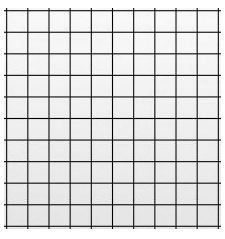
4.3 Algoritmus A^*

Nezbytným úkolem této práce byla realizace informovaného algoritmu prohledávání stavového prostoru. Jako nejvhodnější algoritmus pro zadaný úkol se jevil algoritmus A^*. Ten byl implementován do výsledné aplikace podle postupu, který byl již uveden v podkapitole 3.4.3. Ukázka zdrojového kódu tohoto algoritmu je uvedena v příloze A. 4.

4.4 Popis aplikace

Dalším, logicky vyplývajícím úkolem této práce, byla aplikace zvoleného algoritmu prohledávání stavového prostoru na zadaný problém za účelem získání dat potřebných k vyhodnocení vlivu formulace hodnotící funkce f na efektivitu algoritmu při hledání řešení. Pro tento účel byla vytvořena aplikace, která splňuje všechny požadavky uvedené v zadání práce. Celá aplikace byla napsaná v programovacím jazyce Java a pro vývoj bylo použito vývojové prostředí Netbeans IDE 7.1. Jazyk Java byl zvolen kvůli velkému rozšíření mezi uživateli, kvalitní dokumentaci, objektově orientovanému přístupu a hlavně přenositelnosti.

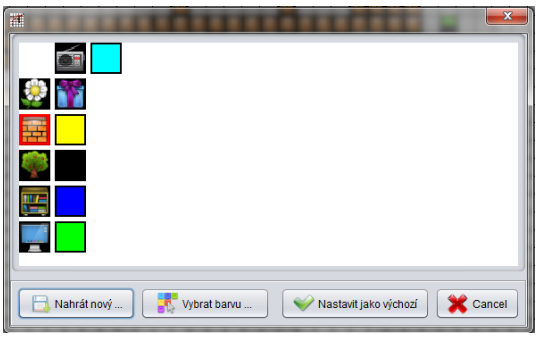
Prostředí, ve kterém se robot pohybuje, je znázorněno čtvercovou mřížkou. Její velikost se může pohybovat v rozmezí 10x10 až 100x100. Jako datová struktura reprezentující tuto mřížku, bylo vybráno dvourozměrné pole a to z důvodu přehlednosti a rychlého přístupu do paměti. Nad polem operuje třída `ProstrediRobota`.



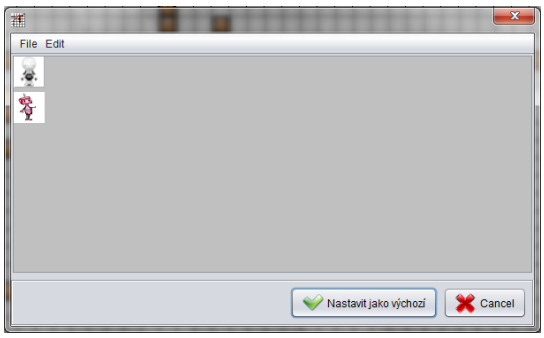
Obrázek 6 – Čtvercová mřížka 10x10

Základní charakteristiky aplikace:

1. Jádro tvoří algoritmus A*.
2. Interakce s uživatelem je realizována pomocí grafického rozhraní.
3. Aplikace umožňuje výběr obrázků reprezentujících překážky a robota (viz obrázek 7 a obrázek 8).

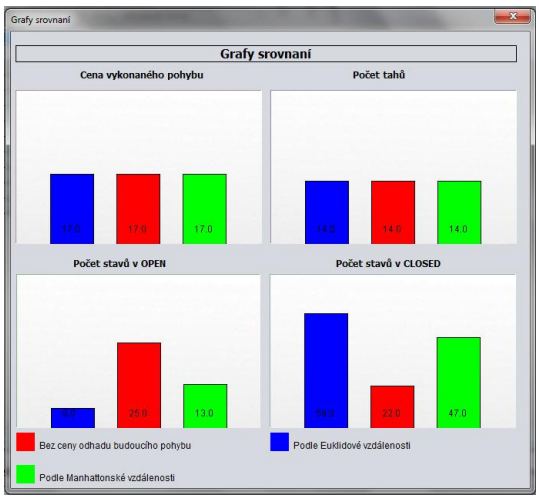


Obrázek 7 – Ukázka editor překážky

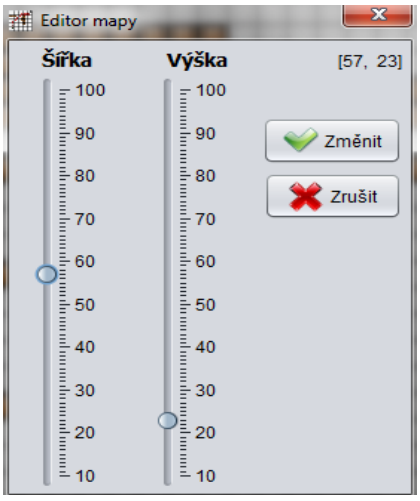


Obrázek 8 – Ukázka editor robotu

4. Kritéria hodnotící efektivitu navržených hodnotících funkcí lze zobrazit pomocí grafů, viz obrázek 9.
5. Součástí aplikace je také editor map, který umožňuje uživateli vytvářet vlastní mapy libovolných rozměrů 10x10 až 100x100 viz obrázek 10.



Obrázek 9 – Ukázka zobrazení grafů



Obrázek 10 – Ukázka editor mapy

6. Aplikace umožňuje uložení a načtení prostředí.

Popis GUI:

1. Tlačítky „Přidat“ a „Odebrat“ se zapínají funkcionality sloužící k přidání a odebraní překážek.
2. Tlačítkem „Generuj“ se vygeneruje nové prostředí s náhodnými překážkami, počátečním a cílovým stavem robota.
3. Tlačítkem „Restart“ se restartuje již vyřešený úkol.
4. Tlačítkem „Začátek“ a „Cíl“ se zapínají funkcionality sloužící k volení startovací a cílovou pozici robota.

Diagramy tříd základních tříd včetně popisu jsou uvedeny v Příloha A – Programátorská dokumentace, a kompletní popis uživatelského rozhraní je popsán v Příloha B - Uživatelská dokumentace.

4.5 Realizace experimentů

Experimenty nutné pro získání potřebných dat, byly provedeny na počítači s následující konfigurací.:

- Procesor: Intel® Core™ i3-2310M Processor (3M Cache, 2.10 GHz)
- Operační paměť: 4,0GB RAM

Celkem bylo provedeno 100 pokusů na 10 různých mapách s rozdílnou obtížností a s rozdílnou počáteční a koncovou pozicí robota. Použité mapy jsou uvedeny v dalším textu, obrázek 15 a další. Pro každou mapu bylo zvoleno 10 různých kombinací počáteční a cílové pozice robota.

4.6 Vyhodnocení experimentálních dat

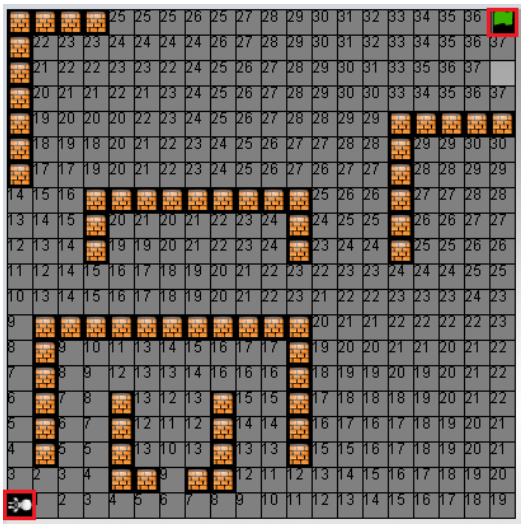
Jednotlivé experimenty byly hodnoceny podle následujících kritérií:

- Časová složitost - počet expandovaných stavů (počet stavů v seznamu CLOSED).
- Prostorová složitost – maximální počet stavů, které byly současně uloženy v paměti (součet stavů v seznamech OPEN a CLOSED).
- Cena nalezené cesty – skutečný čas potřebný k přesunu robota z počátečního stavu do stavu cílového.

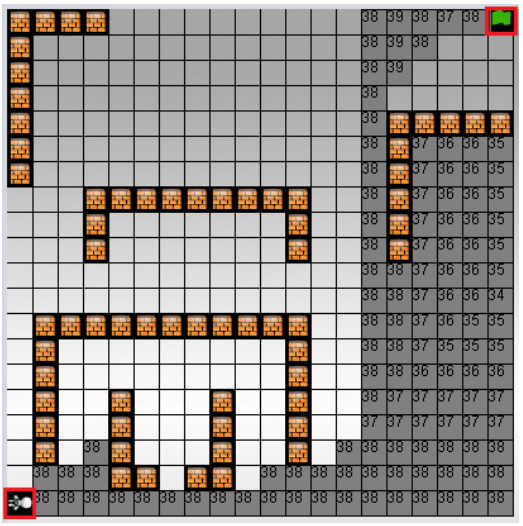
Z experimentálních dat byly pro každé prostředí vypočítány průměrné hodnoty výše uvedených kritérií. Veškerá data a výsledky jsou uvedeny v tabulkách 3 až 12. Souhrnné průměrné hodnoty vypočítané z dat ze všech sta experimentů jsou uvedeny v tabulce 13.

Ukázkový příklad

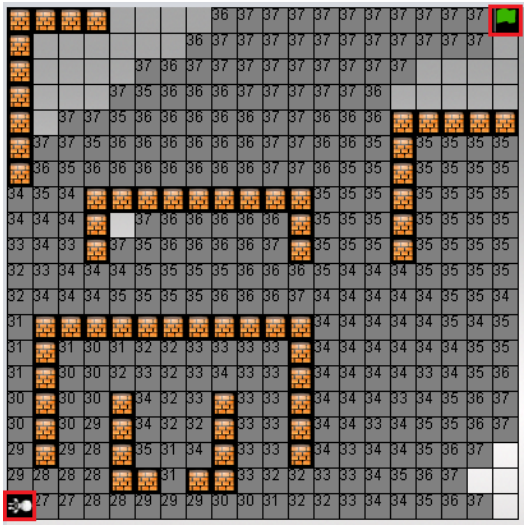
Mějme situaci, která je znázorněna na obrázcích 11, 12 a 13. Šedou barvou jsou označeny stavy, které byly navštíveny a ohodnoceny v průběhu řešení. Na obrázku 14 můžeme vidět nalezené cesty.



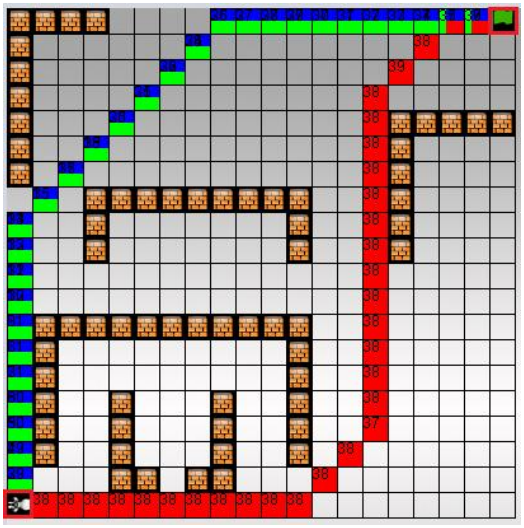
Obrázek 11 – Expandované stavy podle kritéria s nulovým odhadem



Obrázek 13 – Expandované stavy podle Manhattanské vzdálenosti



Obrázek 12 – Expandované stavy podle Euklidovské vzdálenosti



Obrázek 14 – Nalezené cesty jednotlivých hodnotících funkcí

Legenda barev:

- – expandované stavy
- – nalezená cesta podle Manhattanské vzdálenosti
- – nalezená cesta podle Euklidovské vzdálenosti
- – nalezená cesta podle kritérií s nulovou cenou odhadu budoucího pohybu

Tabulka 2 – Srovnání hodnotících funkcí ukázkového příkladu

Název kritéria	Cena cesty	Prostorová složitost [ks]	Časová složitost [ks]
Nulový odhad	37.3	334	334
Manhattanská vzdálenost	38.5	153	120
Euklidovská vzdálenost	37.3	324	302

Z výše uvedených obrázků a tabulky patrné, že výsledky Euklidovské vzdálenosti a kritéria s nulovým odhadem jsou z hlediska ceny cesty stejné, ale Manhattanská vzdálenost

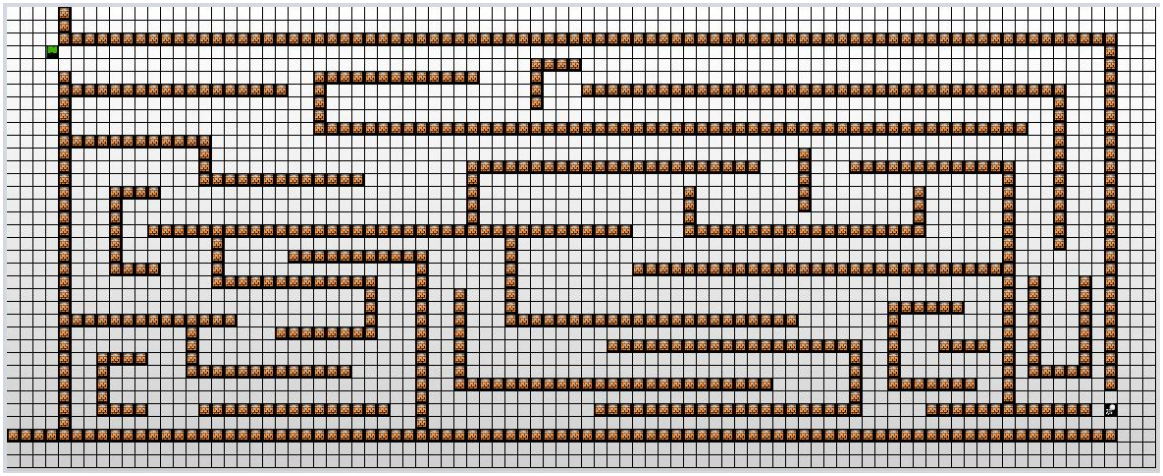
vykazuje vyšší hodnoty než ostatní dvě kritéria. Z hlediska časové a prostorové složitosti, nejhůře dopadlo kritérium s nulovým odhadem.

4.6.1 Výsledky testování

Tato podkapitola obsahuje mapy jednotlivých experimentálních prostředí i data získaná při experimentech. Součástí této podkapitoly je i statistické vyhodnocení získaných dat pro jednotlivá prostředí i celkové zhodnocení navržených kritérií.

Kompletní data i grafy jsou uvedeny i v přiloženém souboru „experiment.xlsx“.

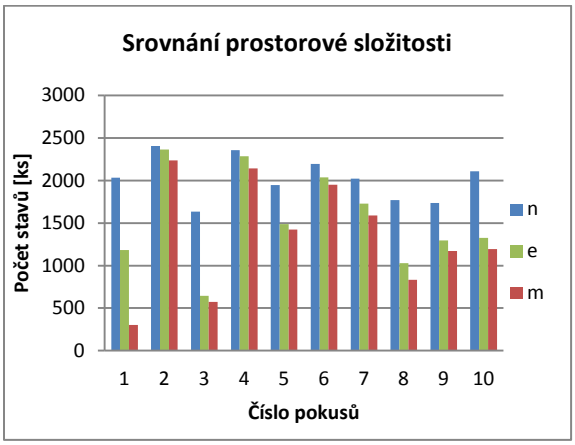
Prostředí č. 1



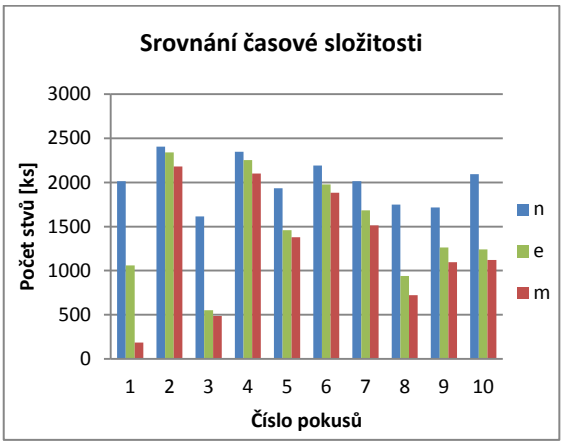
Obrázek 15 – Ukázka experimentální prostředí 90x36 č. 1

Tabulka 3 – Naměřené hodnoty experimentu na prostředí č. 1

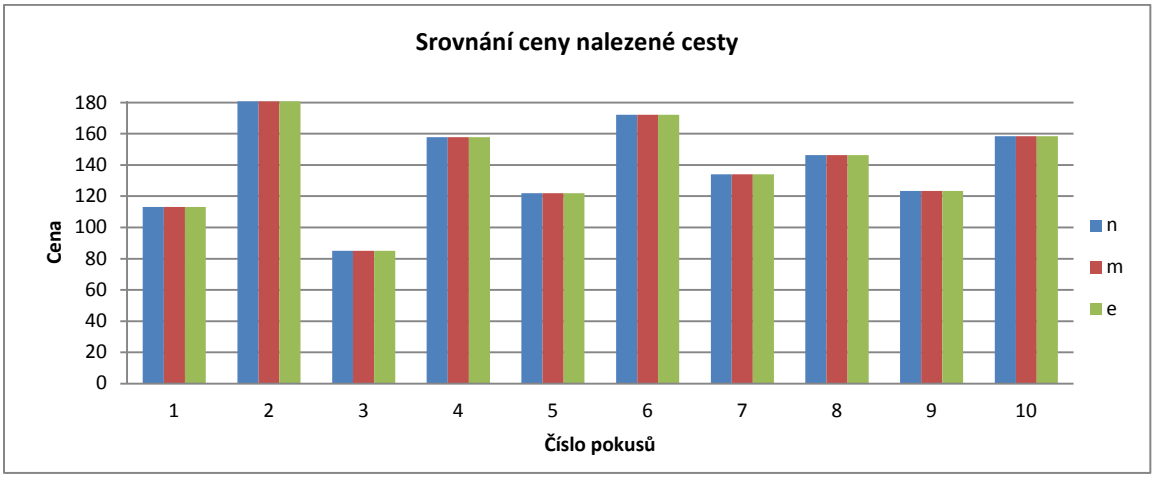
Název kritéria	1	2	3	4	5	6	7	8	9	10	Průměr
Nulový odhad											
cena	113,1	180,7	85,1	157,9	121,9	172,2	134,0	146,3	123,4	158,5	139,3
prostor	2031	2405	1634	2357	1945	2193	2022	1770	1734	2106	2019,7
čas	2013	2406	1613	2347	1932	2190	2014	1747	1715	2093	2007
Manhattanská vzdálenost											
cena	113,1	180,7	85,1	157,9	121,9	172,2	134,0	146,3	123,4	158,5	139,3
prostor	302	2237	575	2141	1424	1950	1587	831	1171	1192	1341
čas	185	2180	487	2102	1380	1881	1513	723	1095	1120	1266,6
Euklidovská vzdálenost											
cena	113,1	180,7	85,1	157,9	121,9	172,2	134,0	146,3	123,4	158,5	139,3
prostor	1183	2362	643	2283	1488	2037	1726	1030	1295	1325	1537,2
čas	1060	2339	551	2253	1457	1978	1683	939	1263	1242	1476,5



Obrázek 16 – Srovnání prostorové složitosti experimentu na prostředí č. 1

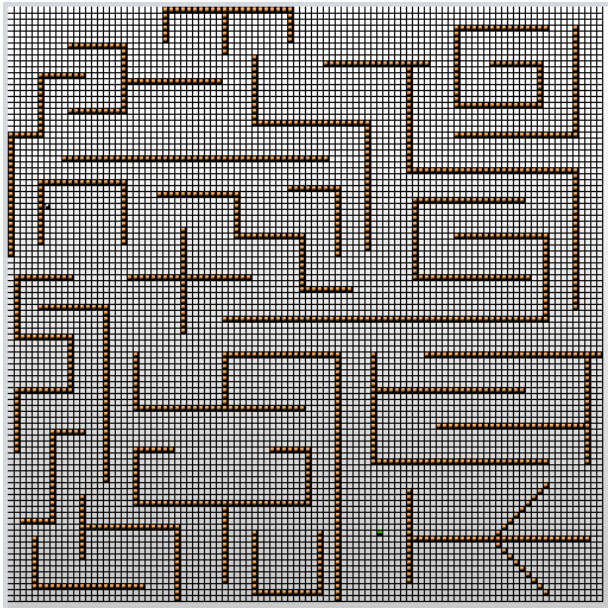


Obrázek 17 – Srovnání časové složitosti experimentu na prostředí č. 1



Obrázek 18 – Srovnání ceny pohybu nalezené cesty na prostředí č. 1

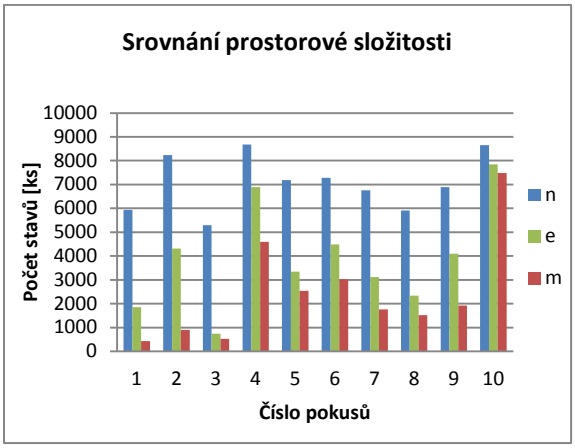
Prostředí č. 2



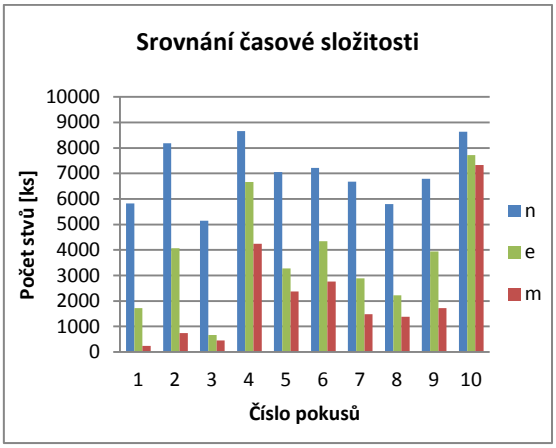
Obrázek 19 – Ukázka experimentální prostředí 100x100 č. 2

Tabulka 4 – Naměřené hodnoty experimentu na prostředí č. 2

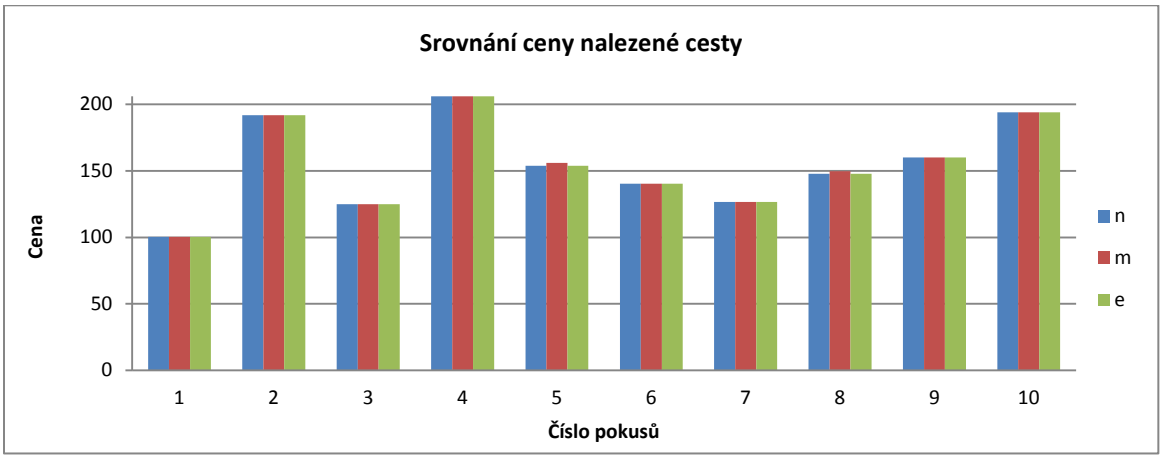
Název kritéria	1	2	3	4	5	6	7	8	9	10	Průměr
Nulový odhad											
cena	100,4	191,8	124,9	206,0	153,9	140,5	126,6	147,7	160,2	194,0	154,6
prostor	5943	8228	5298	8673	7191	7275	6756	5913	6896	8652	7082,5
čas	5832	8182	5153	8659	7059	7225	6682	5795	6790	8642	7001,9
Manhattanská vzdálenost											
cena	100,4	191,8	124,9	206,0	156,0	140,5	126,6	149,8	160,2	194,0	155,0
prostor	426	906	525	4596	2542	3027	1759	1519	1925	7483	2470,8
čas	236	739	448	4243	2375	2767	1477	1378	1717	7334	2271,4
Euklidovská vzdálenost											
cena	100,4	191,8	124,9	206,0	153,9	140,5	126,6	147,7	160,2	194,0	154,6
prostor	1853	4319	738	6895	3349	4482	3116	2335	4103	7845	3903,5
čas	1723	4065	664	6672	3274	4348	2887	2218	3939	7724	3751,4



Obrázek 20 – Srovnání prostorové složitosti experimentu na prostředí č. 2

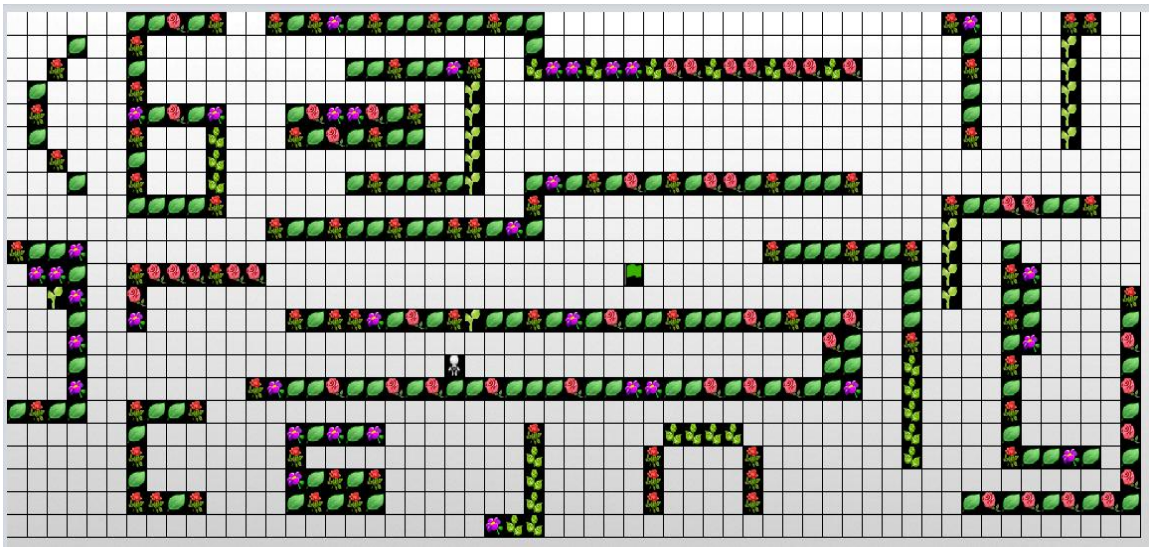


Obrázek 21 – Srovnání časové složitosti experimentu na prostředí č. 2



Obrázek 22 – Srovnání ceny pohybu nalezené cesty na prostředí č. 2

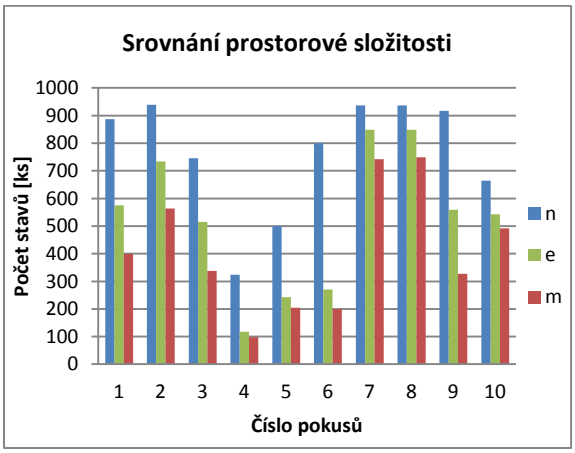
Prostředí č. 3



Obrázek 23 – Ukázka experimentální prostředí 57x23 č. 3

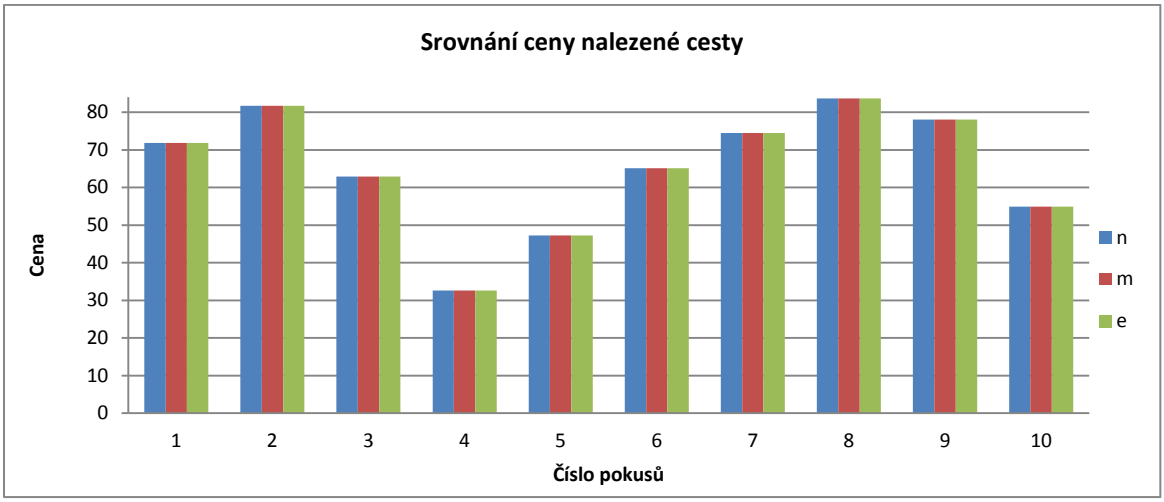
Tabulka 5 – Naměřené hodnoty experimentu na prostředí č. 3

Název kritéria	1	2	3	4	5	6	7	8	9	10	Průměr
Nulový odhad											
cena	71,9	81,7	62,9	32,7	47,2	65,1	74,5	83,7	78,0	54,9	65,3
prostor	887	939	746	324	500	800	937	937	917	664	765,1
čas	875	935	712	304	490	788	926	926	912	645	751,3
Manhattanská vzdálenost											
cena	71,9	81,7	62,9	32,7	47,2	65,1	74,5	83,7	78,0	54,9	65,3
prostor	399	564	338	98	204	200	742	749	327	492	411,3
čas	307	507	274	85	188	159	647	661	247	443	351,8
Euklidovská vzdálenost											
cena	71,9	81,7	62,9	32,7	47,2	65,1	74,5	83,7	78,0	54,9	65,3
prostor	575	734	515	117	243	270	849	849	559	543	525,4
čas	495	698	481	104	225	225	808	808	501	508	485,3



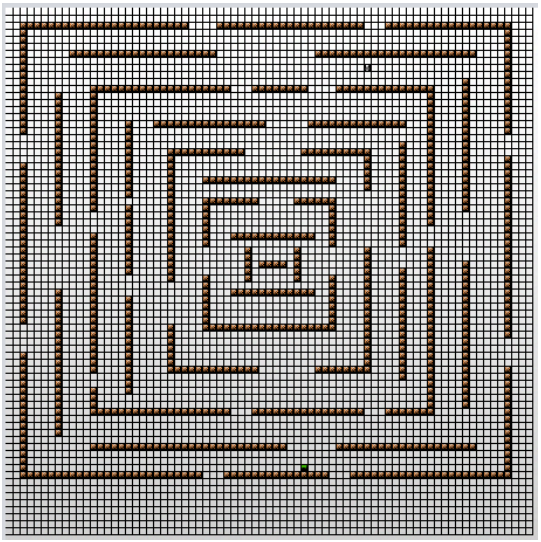
Obrázek 24 – Srovnání prostorové složitosti experimentu na prostředí č. 3

Obrázek 25 – Srovnání časové složitosti experimentu na prostředí č. 3



Obrázek 26 – Srovnání ceny pohybu nalezené cesty na prostředí č. 3

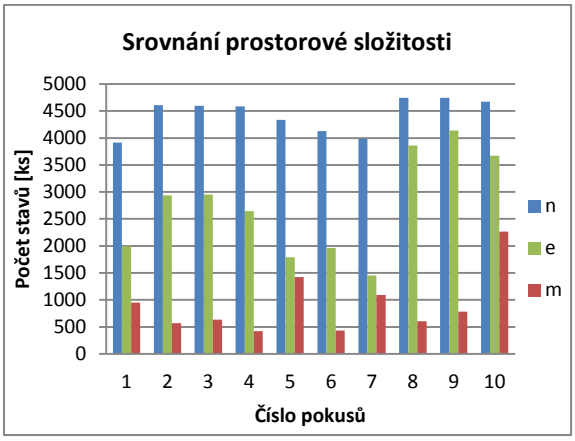
Prostředí č. 4



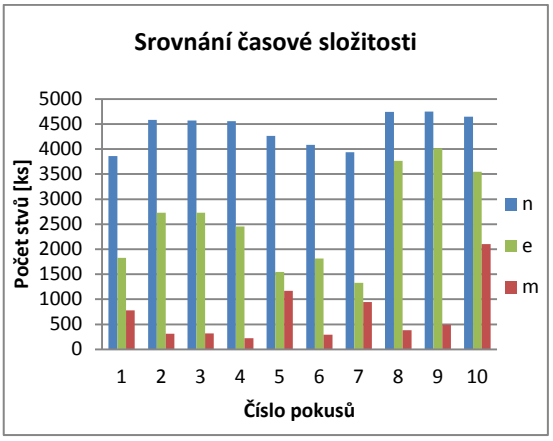
Obrázek 27 – Ukázka experimentální prostředí 75x75 č. 4

Tabulka 6 – Naměřené hodnoty experimentu na prostředí č. 4

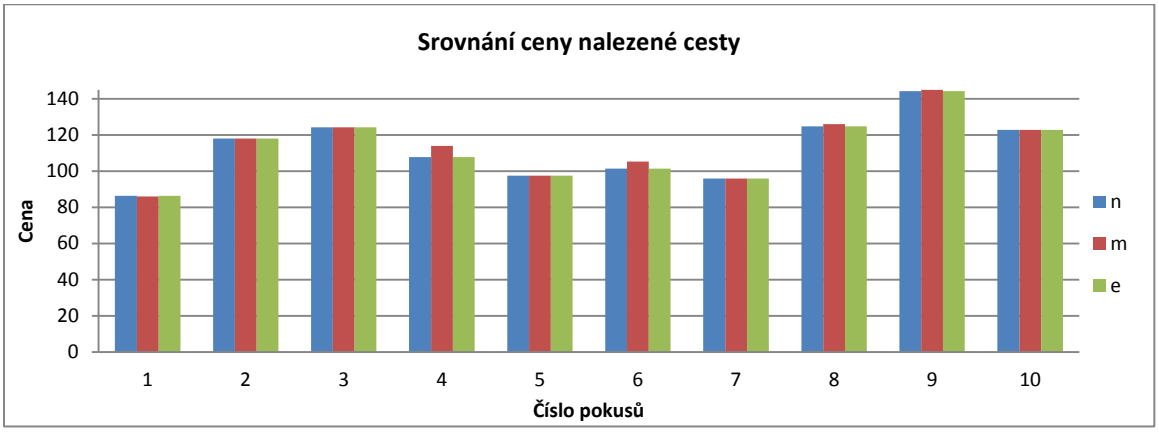
Název	1	2	3	4	5	6	7	8	9	10	Průměr
Nulový odhad											
cena	86,5	118,1	124,4	107,8	97,6	101,4	96,0	124,8	144,3	122,8	112,4
prostor	3914	4608	4597	4586	4336	4127	3985	4746	4746	4674	4431,9
čas	3858	4586	4568	4561	4262	4082	3935	4743	4747	4648	4399
Manhattanská vzdálenost											
cena	86,1	118,1	124,4	114,1	97,6	105,4	96,0	126,0	145,0	122,8	113,5
prostor	947	571	633	419	1425	433	1092	604	785	2266	917,5
čas	780	309	316	225	1167	294	946	380	493	2101	701,1
Euklidovská vzdálenost											
cena	86,5	118,1	124,4	107,8	97,6	101,4	96,0	124,8	144,3	122,8	112,4
prostor	1992	2933	2953	2647	1790	1962	1450	3863	4139	3672	2740,1
čas	1826	2732	2731	2457	1543	1817	1329	3763	4013	3546	2575,7



Obrázek 28 – Srovnání prostorové složitosti experimentu na prostředí č. 4

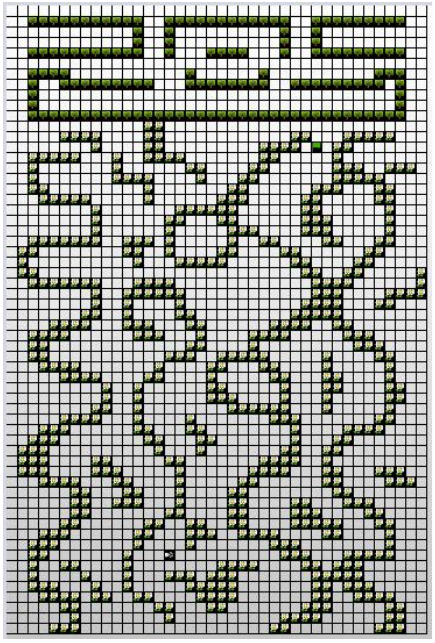


Obrázek 29 – Srovnání časové složitosti experimentu na prostředí č. 4



Obrázek 30 – Srovnání ceny pohybu nalezené cesty na prostředí č. 4

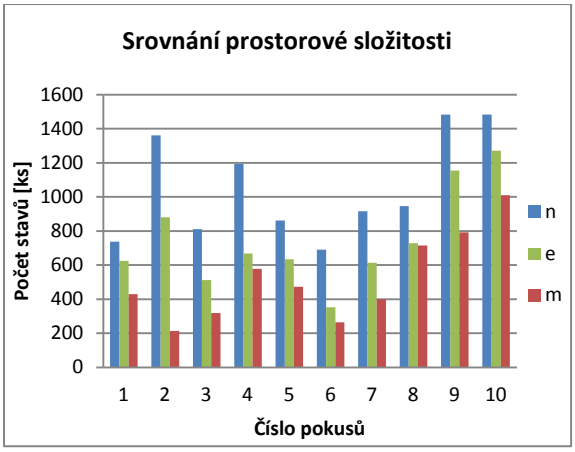
Prostředí č. 5



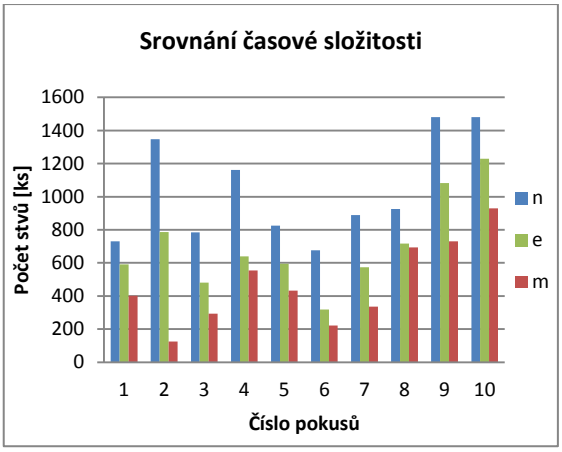
Obrázek 31 – Ukázka experimentální prostředí 40x60 č. 5

Tabulka 7 – Naměřené hodnoty experimentu na prostředí č. 5

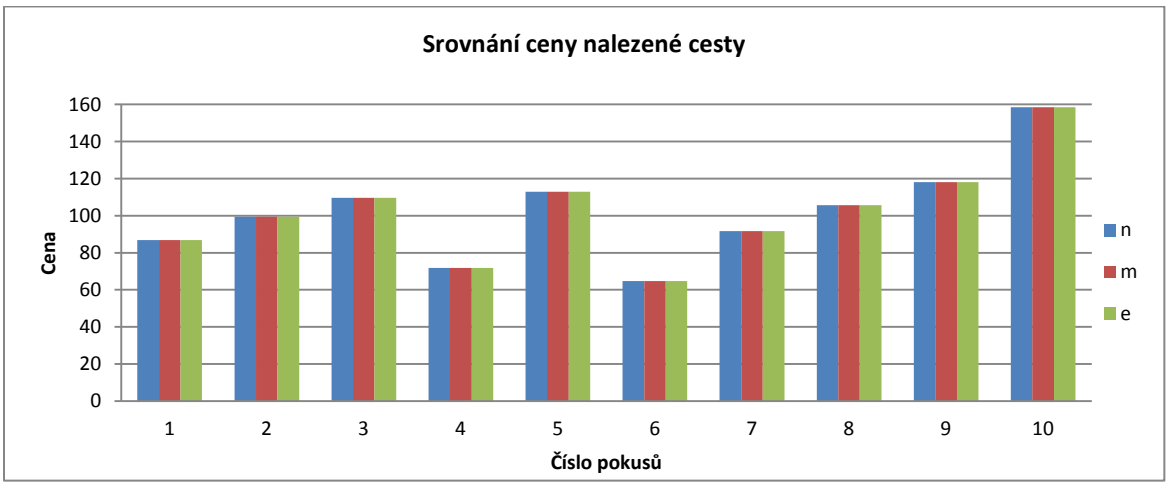
Název kritéria	1	2	3	4	5	6	7	8	9	10	Průměr
Nulový odhad											
cena	86,8	99,4	109,6	71,8	113,0	64,7	91,6	105,6	118,0	158,4	101,9
prostor	737	1361	811	1195	861	691	916	947	1484	1484	1048,7
čas	730	1347	785	1161	825	676	889	925	1480	1480	1029,8
Manhattanská vzdálenost											
cena	86,8	99,4	109,6	71,8	113,0	64,7	91,6	105,6	118,0	158,4	101,9
prostor	430	213	318	579	473	265	400	715	793	1010	519,6
čas	401	124	294	554	432	221	336	693	731	929	471,5
Euklidovská vzdálenost											
cena	86,8	99,4	109,6	71,8	113,0	64,7	91,6	105,6	118,0	158,4	101,9
prostor	625	881	513	669	634	353	613	729	1155	1271	744,3
čas	590	786	481	640	594	319	573	717	1083	1229	701,2



Obrázek 32 – Srovnání prostorové složitosti experimentu na prostředí č. 5

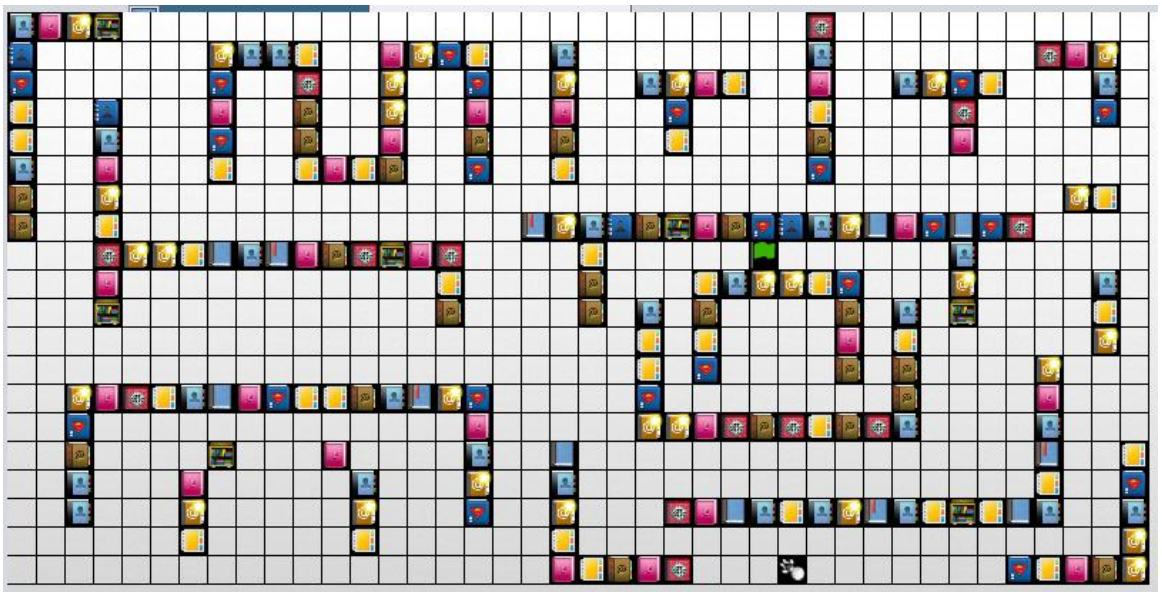


Obrázek 33 – Srovnání časové složitosti experimentu na prostředí č. 5



Obrázek 34 – Srovnání ceny pohybu nalezené cesty na prostředí č. 5

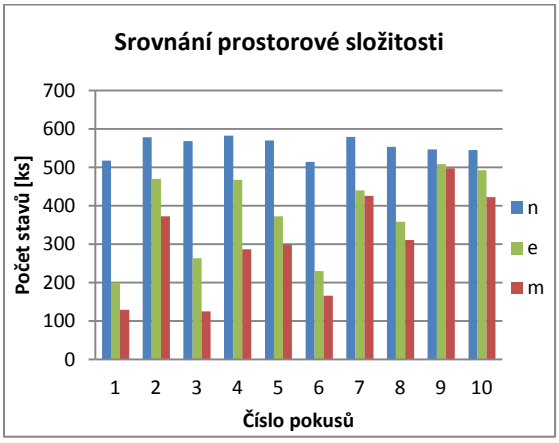
Prostředí č. 6



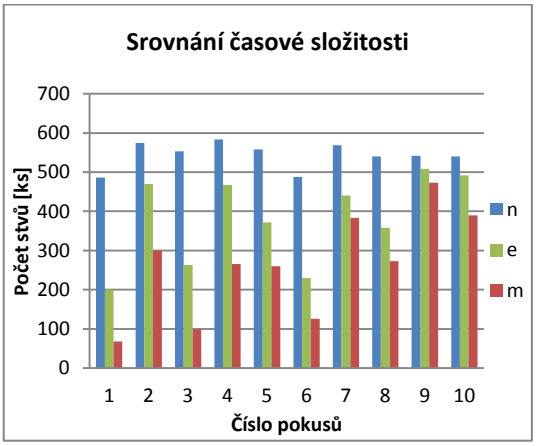
Obrázek 35 – Ukázka experimentální prostředí 40x20 č. 6

Tabulka 8 – Naměřené hodnoty experimentu na prostředí č. 6

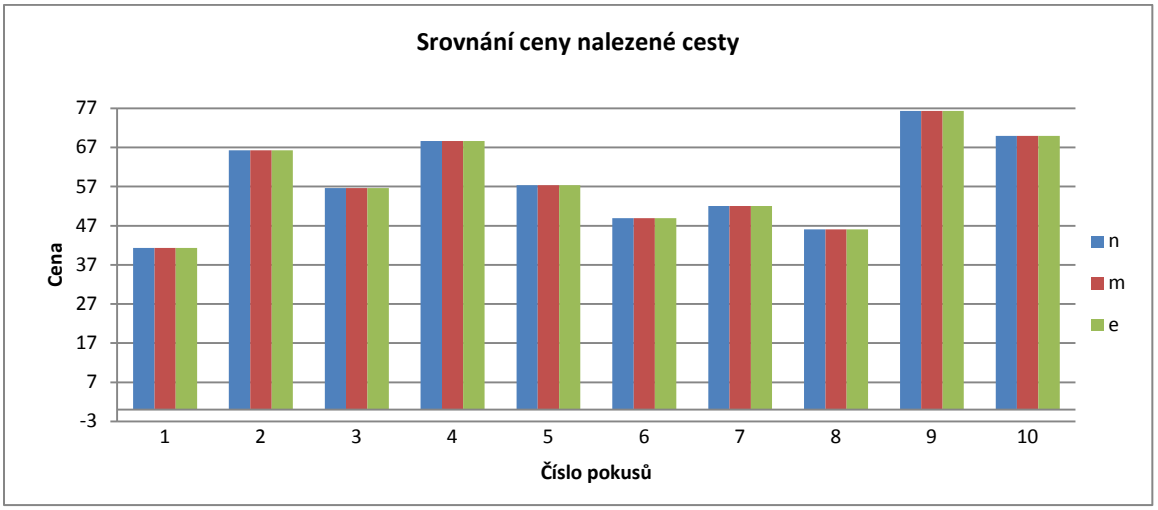
Název kritéria	1	2	3	4	5	6	7	8	9	10	Průměr
Nulový odhad											
cena	41,3	66,2	56,7	68,7	57,3	48,9	52,1	46,1	76,3	70,0	58,4
prostor	517	578	568	582	570	514	579	553	546	545	555,2
čas	486	574	553	583	558	488	569	540	542	540	543,3
Manhattanská vzdálenost											
cena	41,3	66,2	56,7	68,7	57,3	48,9	52,1	46,1	76,3	70,0	58,4
prostor	129	372	125	287	299	166	426	311	497	422	303,4
čas	68	300	99	266	260	126	383	273	473	390	263,8
Euklidovská vzdálenost											
cena	41,3	66,2	56,7	68,7	57,3	48,9	52,1	46,1	76,3	70,0	58,4
prostor	200	470	263	467	372	230	440	358	508	492	380
čas	162	441	214	446	338	193	394	323	493	465	346,9



Obrázek 36 – Srovnání prostorové složitosti experimentu na prostředí č. 6

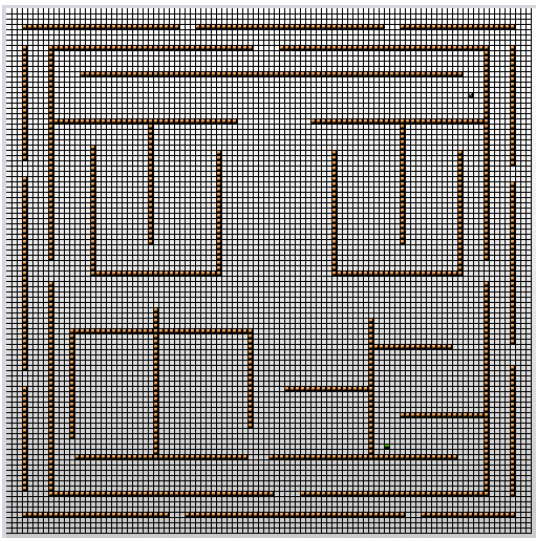


Obrázek 37 – Srovnání časové složitosti experimentu na prostředí č. 6



Obrázek 38 – Srovnání ceny pohybu nalezené cesty na prostředí č. 6

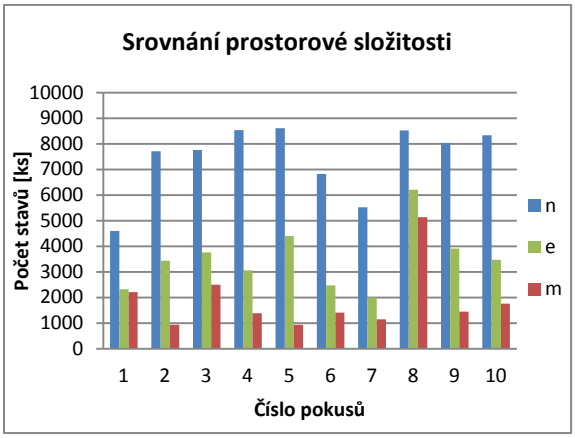
Prostředí č. 7



Obrázek 39 – Ukázka experimentální prostředí 100x100 č. 7

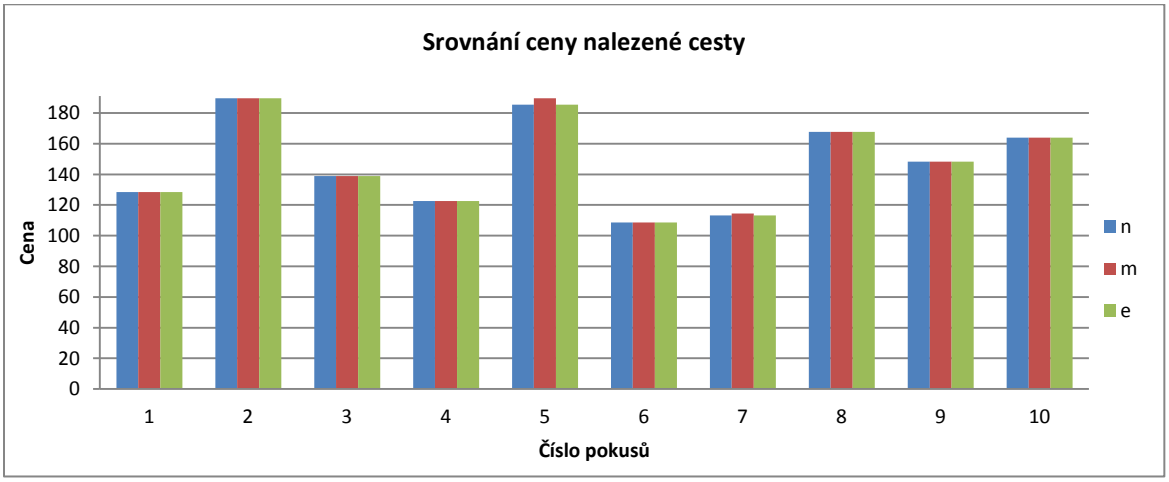
Tabulka 9 – Naměřené hodnoty experimentu na prostředí č. 7

Název kritéria	1	2	3	4	5	6	7	8	9	10	Průměr
Nulový odhad											
cena	128,5	189,7	138,8	122,6	185,5	108,6	113,3	167,7	148,2	163,9	146,7
prostor	4598	7709	7765	8536	8607	6829	5519	8530	8035	8331	7445,9
čas	4539	7637	7672	8492	8565	6731	5352	8494	7970	8258	7371
Manhattanská vzdálenost											
cena	128,5	189,7	138,8	122,6	189,7	108,6	114,5	167,7	148,2	163,9	147,2
prostor	2206	932	2492	1380	932	1406	1146	5138	1451	1763	1884,6
čas	2109	597	2267	1118	597	1163	963	4975	1084	1268	1614,1
Euklidovská vzdálenost											
cena	128,5	189,7	138,8	122,6	185,5	108,6	113,3	167,7	148,2	163,9	146,7
prostor	2320	3437	3754	3061	4402	2469	1993	6215	3914	3467	3503,2
čas	2208	3343	3574	2895	4205	2304	1870	6068	3733	3282	3348,2



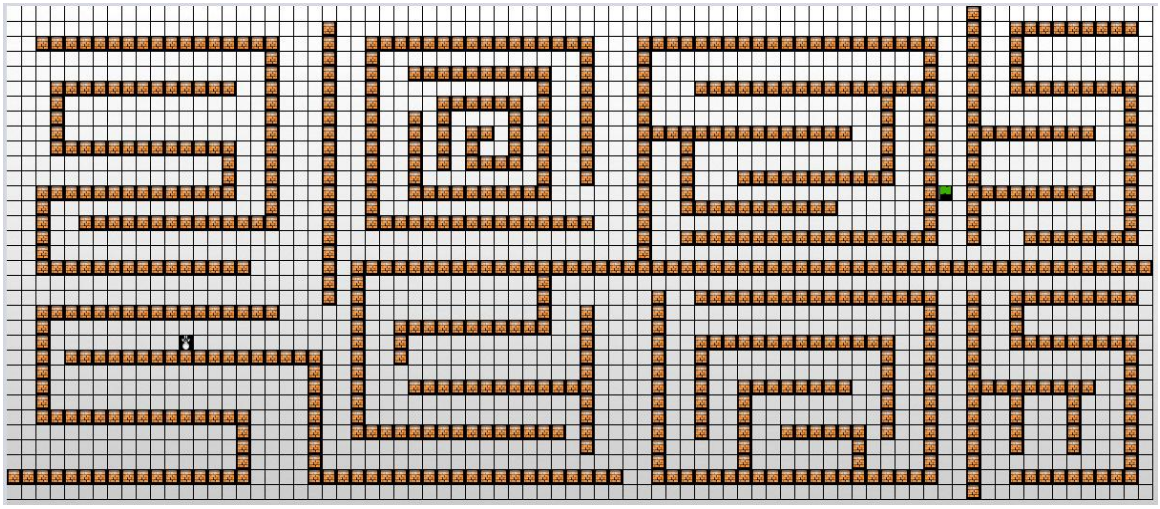
Obrázek 40 – Srovnání prostorové složitosti experimentu na prostředí č. 7

Obrázek 41 – Srovnání časové složitosti experimentu na prostředí č. 7



Obrázek 42 – Srovnání ceny pohybu nalezené cesty na prostředí č. 7

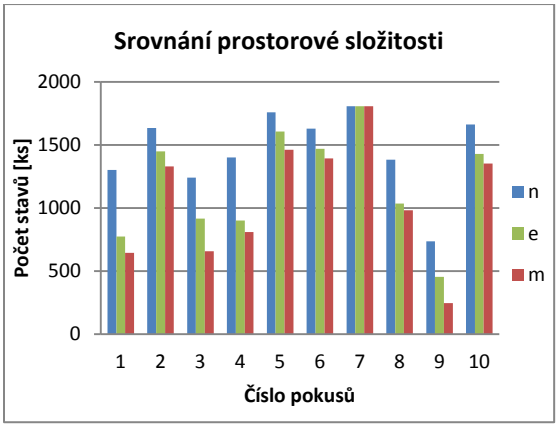
Prostředí č. 8



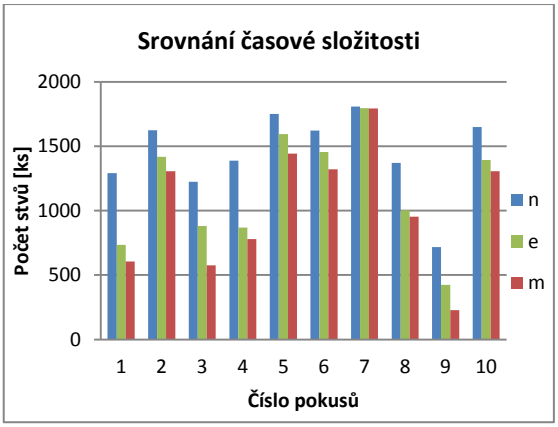
Obrázek 43 – Ukázka experimentální prostředí 80x33 č. 8

Tabulka 10 – Naměřené hodnoty experimentu na prostředí č. 8

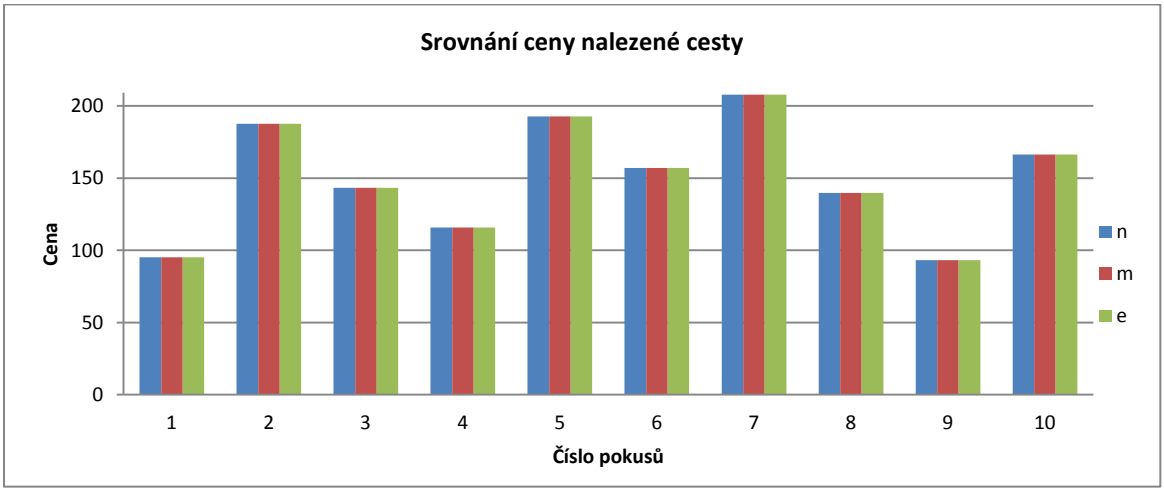
Název kritéria	1	2	3	4	5	6	7	8	9	10	Průměr
Nulový odhad											
cena	95,2	187,5	143,1	115,7	192,7	157,0	207,7	139,8	93,1	166,2	149,8
prostor	1302	1633	1240	1400	1759	1628	1807	1383	736	1662	1455
čas	1291	1625	1224	1387	1750	1622	1807	1370	717	1649	1444,2
Manhattanská vzdálenost											
cena	95,2	187,5	143,1	115,7	192,7	157,0	207,7	139,8	93,1	166,2	149,8
prostor	644	1329	658	810	1462	1393	1807	982	245	1351	1068,1
čas	605	1307	576	778	1442	1322	1792	953	227	1306	1030,8
Euklidovská vzdálenost											
cena	95,2	187,5	143,1	115,7	192,7	157,0	207,7	139,8	93,1	166,2	149,8
prostor	774	1449	915	900	1607	1469	1807	1036	455	1428	1184
čas	735	1417	880	868	1595	1455	1795	1004	424	1394	1156,7



Obrázek 44 – Srovnání prostorové složitosti experimentu na prostředí č. 8

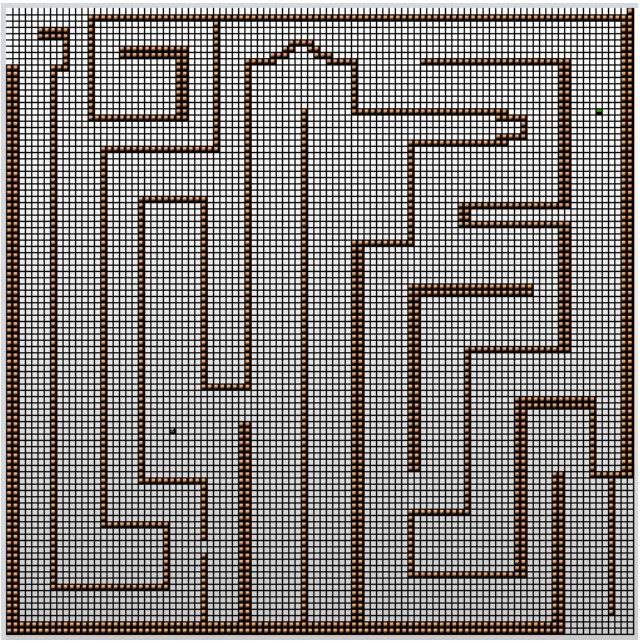


Obrázek 45 – Srovnání časové složitosti experimentu na prostředí č. 8



Obrázek 46 – Srovnání ceny pohybu nalezené cesty na prostředí č. 8

Prostředí č. 9

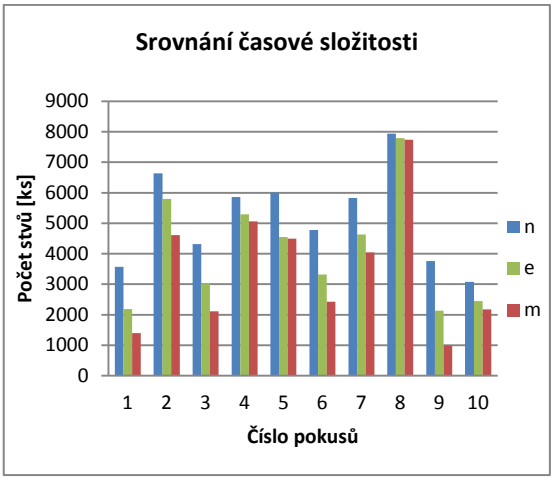


Obrázek 47 – Ukázka experimentální prostředí 100x100 č. 9

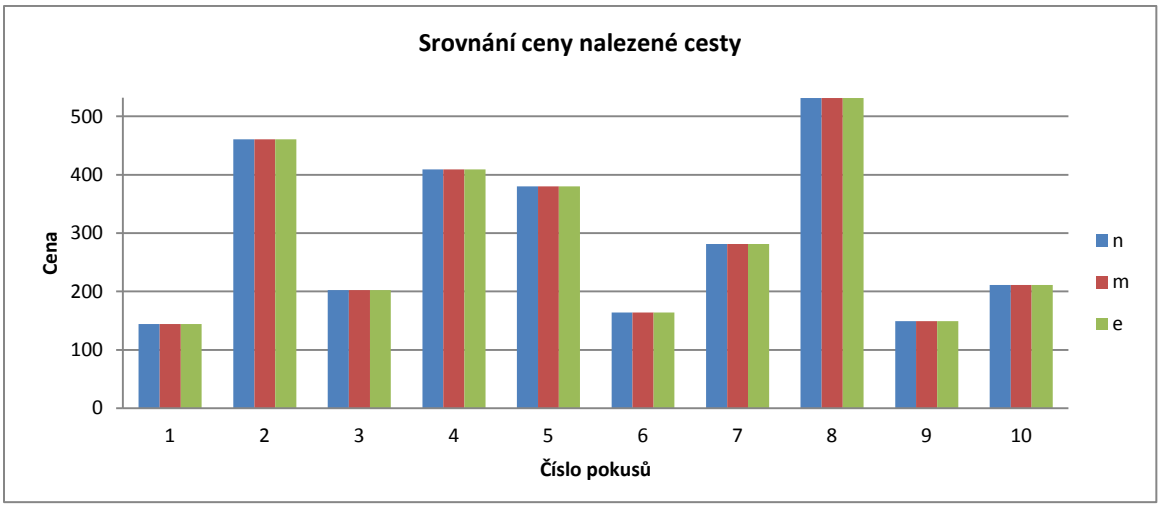
Tabulka 11 – Naměřené hodnoty experimentu na prostředí č. 9

Název kritéria	1	2	3	4	5	6	7	8	9	10	Průměr
Nulový odhad											
cena	144,5	460,9	202,1	409,2	380,2	164,3	281,2	531,2	149,3	211,0	293,4
prostor	3610	6656	4363	5886	6070	4823	5854	7942	3792	3108	5210,4
čas	3572	6639	4313	5856	6011	4774	5829	7935	3757	3077	5176,3
Manhattanská vzdálenost											
cena	144,5	460,9	202,1	409,2	380,2	164,3	281,2	531,2	149,3	211,0	293,4
prostor	1605	4780	2198	5145	4547	2591	4223	7794	1088	2278	3624,9
čas	1397	4615	2115	5061	4491	2429	4042	7743	984	2174	3505,1
Euklidovská vzdálenost											
cena	144,5	460,9	202,1	409,2	380,2	164,3	281,2	531,2	149,3	211,0	293,4
prostor	2281	5851	3090	5341	4600	3415	4760	7834	2227	2487	4188,6
čas	2183	5793	3023	5288	4549	3320	4629	7793	2138	2446	4116,2

Obrázek 48 – Srovnání prostorové složitosti experimentu na prostředí č. 9

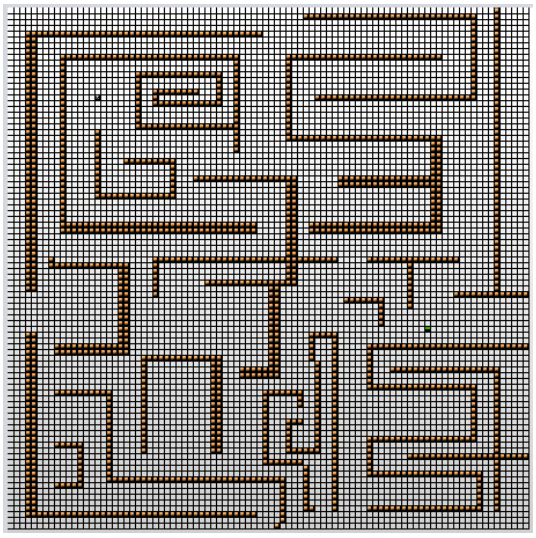


Obrázek 49 – Srovnání časové složitosti experimentu na prostředí č. 9



Obrázek 50 – Srovnání ceny pohybu nalezené cesty na prostředí č. 9

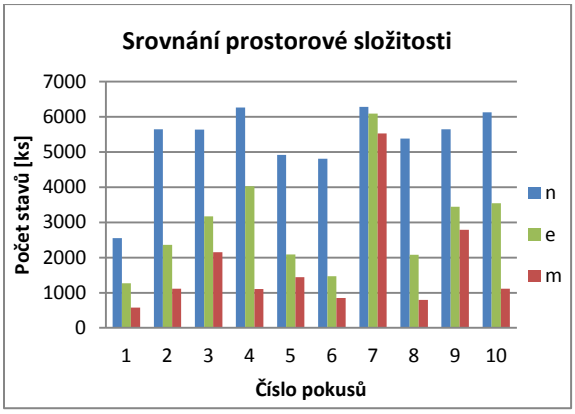
Prostředí č. 10



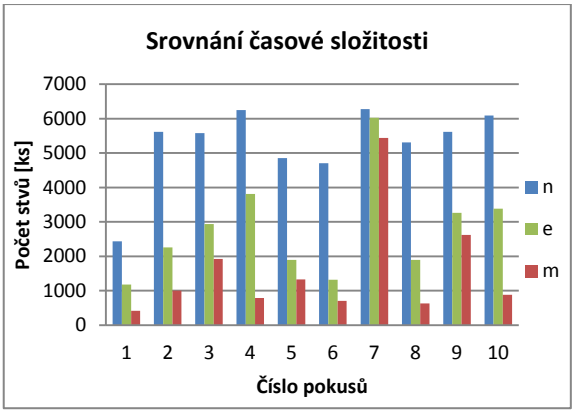
Obrázek 51 – Ukázka experimentální prostředí 90x90 č. 10

Tabulka 12 – Naměřené hodnoty experimentu na prostředí č. 10

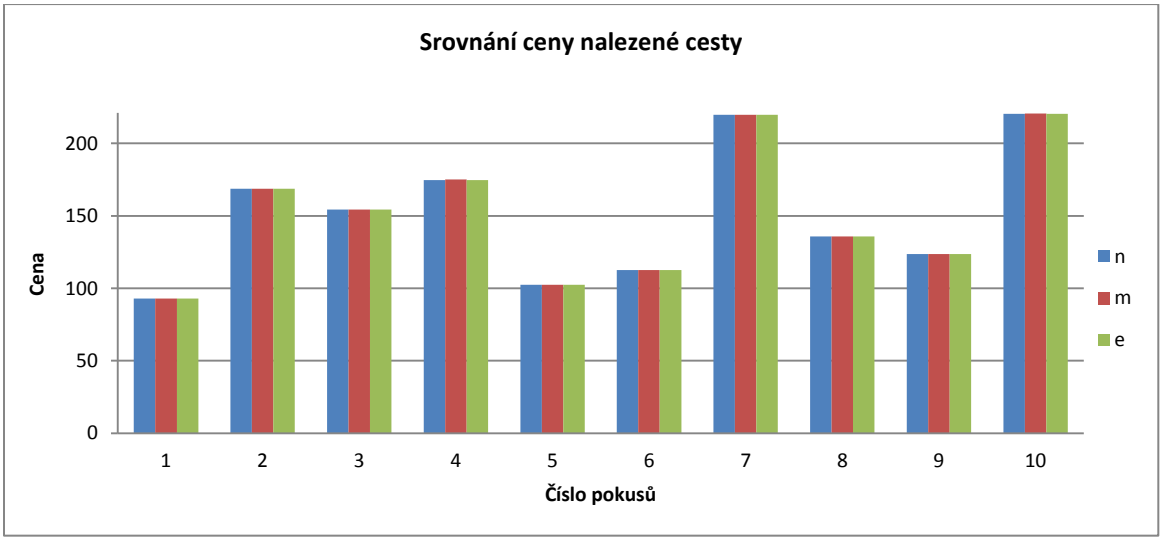
Název kritéria	1	2	3	4	5	6	7	8	9	10	Průměr
Nulový odhad											
cena	93,0	168,6	154,4	174,7	102,4	112,6	219,8	135,8	123,6	220,4	150,5
prostor	2550	5648	5631	6265	4918	4805	6281	5380	5647	6127	5325,2
čas	2437	5616	5577	6253	4856	4708	6279	5313	5613	6089	5274,1
Manhattanská vzdálenost											
cena	93,0	168,6	154,4	175,0	102,4	112,6	219,8	135,8	123,6	220,5	150,6
prostor	581	1114	2151	1103	1443	850	5528	793	2787	1110	1746
čas	417	1004	1925	792	1330	707	5439	627	2618	879	1573,8
Euklidovská vzdálenost											
cena	93,0	168,6	154,4	174,7	102,4	112,6	219,8	135,8	123,6	220,4	150,5
prostor	1272	2364	3167	4028	2085	1467	6086	2078	3442	3541	2953
čas	1182	2258	2938	3810	1895	1321	6028	1892	3267	3389	2798



Obrázek 52 – Srovnání prostorové složitosti experimentu na prostředí č. 10



Obrázek 53 – Srovnání časové složitosti experimentu na prostředí č. 10



Obrázek 54 – Srovnání ceny pohybu nalezené cesty na prostředí č. 10

Souhrnné zhodnocení:

V následující tabulce jsou vypočítány průměrné hodnoty kritérií. Tyto hodnoty byly vypočteny ze všech získaných dat.

Název kritéria	Průměrná cena cesty	Průměrná prostorová složitost [ks]	Průměrná časová složitost [ks]
Nulový odhad	137,2	3534,0	3499,8
Manhattanská vzdálenost	137,4	1428,7	1305,0
Euklidovská vzdálenost	137,2	2165,9	2075,6

Tabulka 13 – Souhrnné průměrné hodnoty kritérií jednotlivých hodnotících funkcí

5 Závěr

Cílem této bakalářské práce bylo posoudit vliv hodnotících funkcí na efektivitu informovaného algoritmu pro prohledávání stavového prostoru. Jako vhodný kandidát pro realizaci tohoto úkolu byl vybrán nejpoužívanější informovaný algoritmus A-star (dále jen A*). Ten byl aplikován na úkol, jehož cílem bylo nalezení časově nejkratší cesty robota z počátečního stavu do stavu cílového v předem známém prostředí s překážkami.

Pro posouzení efektivity byly navrženy celkem tři hodnotící funkce, které se lišily jen heuristickou funkcí h^* tj. odhadem ceny budoucího pohybu. Byla vybrána tři nejčastěji používaná kritéria a to s nulovým odhadem budoucí ceny pohybu, Manhattanská vzdálenost a Euklidovská vzdálenost (bližší informace viz podkapitolu 4.2.2).

V rámci experimentu bylo vytvořeno celkem 100 různých scénářů v 10 různých prostředí s rozdílnou obtížností a velikostí mapy. Pro každé prostředí bylo nastaveno 10 odlišných kombinací počátečních a cílových pozic robota. Každý scénář byl pak vyřešen s užitím všech tří hodnotících funkcí. Výstupy těchto experimentů posloužily jako podklad pro posouzení vlivu hodnotících funkcí na efektivitu realizovaného algoritmu. U jednotlivých hodnotících funkcí byla hodnocena prostorová a časová složitost i cena nalezené cesty. Výsledky jednotlivých scénářů byly zapsány do tabulek 3 až 12. Na základě těchto dat byla vytvořena tabulka 13, která obsahuje průměrné hodnoty spočítané s využitím dat ze všech 100 experimentů.

Vybraný informovaný algoritmus A* má vždy najít optimální řešení, za podmínky že je heuristická funkce dobře zvolena (splňuje podmínku uvedené na straně 22). Čím je hodnota heuristické funkce blíže k nule (dolní mezi), tím je větší časová i prostorová složitost. Důkazem toho je první heuristická funkce s nulovým odhadem, která má nejvyšší časovou a prostorovou složitost, ale vždy nalezne cestu s nejnižší cenou.

Se stoupajícím odhadem hodnoty heuristické funkce časová i prostorová složitost klesá. Odhad budoucí cesty ovšem nesmí přesahovat skutečnou cenu budoucího pohybu (horní mez). Při překročení horní meze již není zaručeno nalezení optimálního řešení. Dokazuje to výsledky Manhattanské heuristické funkce, která vždy odhaduje cenu budoucího pohybu s vyšší nebo stejnou hodnotou, ale zato v provedených experimentech vykazuje menší časové i prostorové nároky než ostatní dvě heuristické funkce. Díky tomu, že je tato účelová funkce nadhodnocená, není zaručeno nalezení optimální cesty. Takto formulovaná hodnotící funkce je tedy pro zadanou úlohu nevhodná. Manhattanská heuristická funkce je vhodná pro úlohy, kde je pohyb možný jen 4 směry.

Poslední heuristickou funkcí je Euklidovská vzdálenost. Tato funkce vždy našla optimální cestu a navíc s mnohem menšími časovými a prostorovými nároky oproti nulovému odhadu, který také vždy našel optimální cestu. Tato heuristická funkce se jeví jako nejvhodnější volba ze všech tří navržených heuristických funkcí.

S využitím vytvořeného algoritmu a navržených hodnotících funkcí se mi podařilo vytvořit aplikaci s přijatelnou časovou režií. Obslužná aplikace je plně funkční a splňuje všechny

požadované funkcionality a navíc poskytuje řadu dalších, jako je editace polohy překážek, zobrazování grafů pro srovnání výsledků jednotlivých hodnotících funkcí, atd.

V případě dalšího vývoje by bylo možné zadaný úkol řešit ještě dalšími algoritmy např. pomocí algoritmu větví a mezí, a porovnávat jejich výsledky pro posouzení vlivu výběru algoritmu. Co se týče obslužné aplikace, bylo by možné přidat generování složitějších překážek.

Při tvorbě bakalářské práce jsem se naučila mnoho nových programátorských dovedností a získala teoretické i praktické znalosti spojené s prohledáváním stavového prostoru.

Literatura

[1] MAŘÍK, Vladimír, ŠTĚPÁNKOVÁ, Olga a LAŽANSKÝ, Jiří. 1993. *Umělá inteligence (1).* Praha: Academia Praha, 1993. ISBN 80-200-0496-3.

[2] ŠEDA, Miloš. *Teorie grafů* [online]. 2003 [cit. 2012-06-26]. Dostupné z: <http://www.uai.fme.vutbr.cz/~mseda/TG03_MS.pdf>.

[3] PATEL, Amit. *Heuristics From Amit's Thoughts on Pathfinding* [online]. Poslední změna: 19.7.2012 [cit. 2012-08-01]. Dostupné z: <http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html>.

[4] CORMEN, Tomas, LEISERSON, Charles, RIVEST, Ronald a STEIN, Clifford. 2009. *Introduction to algorithms.* London: The MIT Press, 2009. ISBN 978-0-262-03384-8, ISBN 978-0-262-53305-8.

[5] VESELÝ, Petr. *Počítačová grafika* [Přednáška]. Pardubice: Univerzita Pardubice. 2010.

[6] BERKA, Petr. *Stavový prostor* [online]. Poslední změna: 18.9.2007 [cit. 2012-08-01]. Dostupné z: <http://sorry.vse.cz/~berka/docs/4iz430/P01-Stavovy%20prostor.pdf>.

[7] *Algoritmy pro prohledávání stavového prostoru s návratem, metoda větví a hranic*í [online]. 2011 [cit. 2011-12-20]. Dostupné z: <http://masuv.blog.cz/1005/algoritmy-pro-prohledavani-stavoveho-prostoru-s-navratem-metoda-vetvi-a-hranic>.

[8] PĚCHOUČEK, Michal, ŽELEZNÝ, Filip. *Prohledávání stavového prostoru* [online]. Poslední změna: 12.10.2004 [cit. 2012-08-10]. Dostupné z: <http://cyber.felk.cvut.cz/gerstner/teaching/kui/sbirka/4_StavPr.doc>.

Příloha A – Programátorská dokumentace

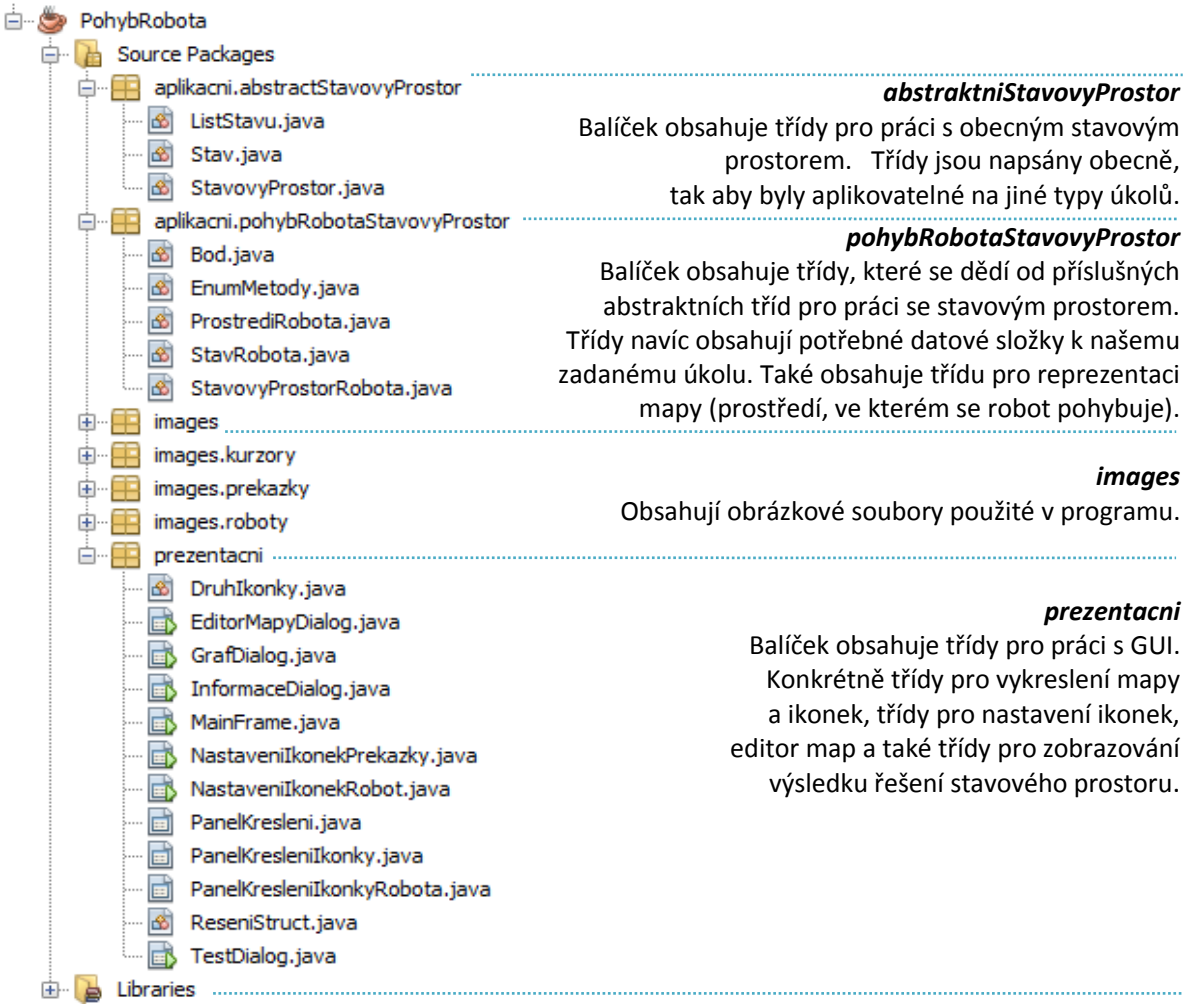
Zde je uveden jen krátký popis balíčků a tříd, které jsou v nich obsažené. Pro podrobný popis všech metod a tříd viz generovány JavaDoc v adresáři `PohybRobota/doc`. Program je rozdělen do dvou balíčků **aplikacni** a **prezentacni**.

Třídy v balíčku s názvem `prezentacni` tvoří prezentační vrstvu aplikace a využívají GUI prvky (Image, JButton, JMenu, JLabel, JScrollPanel, atd.) připojených grafických JAVA knihoven (javax.swing, javax.imageio, java.awt, atd.). Jednotlivé třídy pak obsahují instance tříd z balíčku `aplikacni`, kromě toho obsahují obrovské množství metod pro zobrazování GUI. V této části aplikace přijme zdrojová data (prostředí, pozice robota, heuristická funkce) od uživatele a předává je aplikační vrstvě k vyřešení.

Třídy v balíčku s názvem `aplikacni` představují aplikační vrstvu programu. Obsahuje v sobě další dva balíčky. Význam těch balíčků a jejich tříd je uveden v dalších kapitolách A. 2 a A. 3.

A. 1 Struktura programu

Zde je znázorněn, co obsahují jednotlivé složky programu a ke každé části je krátký popis k čemu jsou používaná.



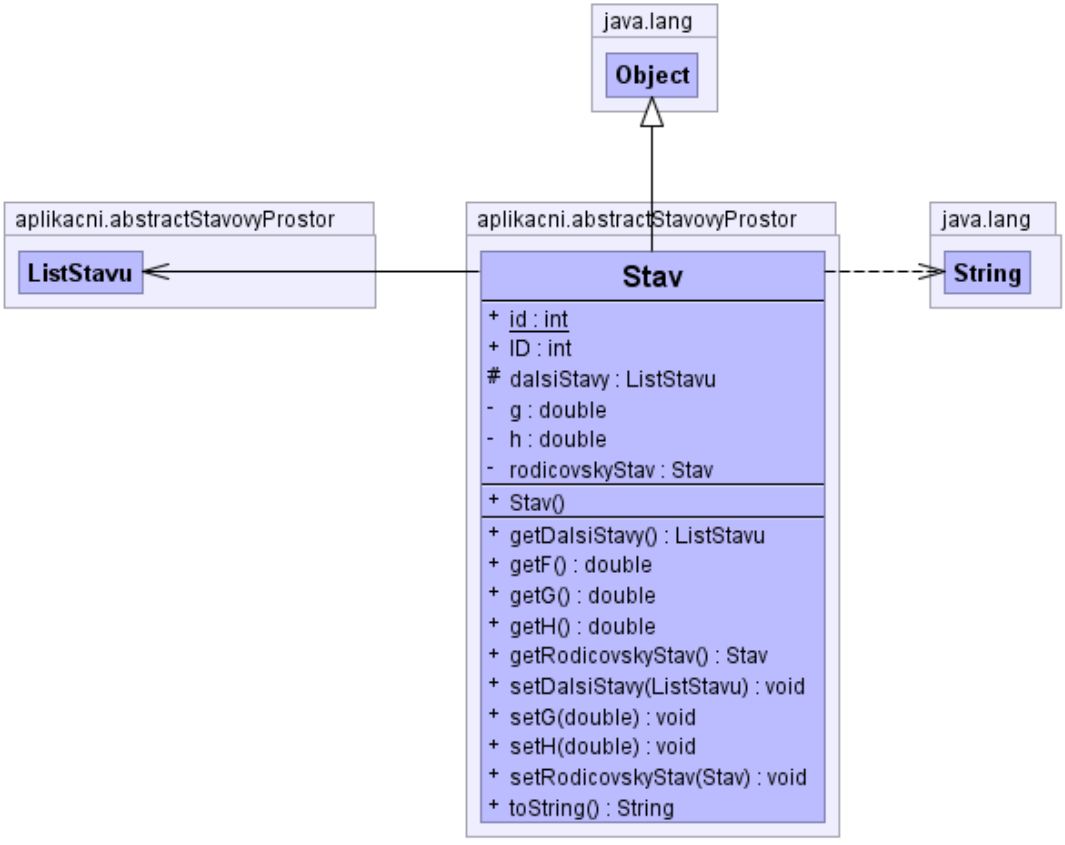
Obrázek 55 – Složková struktura aplikace

A. 2 Popis tříd – abstractStavovyProstor

V této kapitole se detailněji podíváme na třídy, a jejich metody a atributy v balíčku `abstractStavovyProstor`. Zde jsou tři vzájemné spolupracující třídy – `Stav`, `ListStavu` a `StavovýProstor`. Tyto třídy slouží k práci s obecným stavovým prostorem a z těchto se pak dědí třídy v balíčku `pohybRobotaStavovyProstor`.

Obrázek 56 – Balíček abstractStavovyProstor

Třída Stav



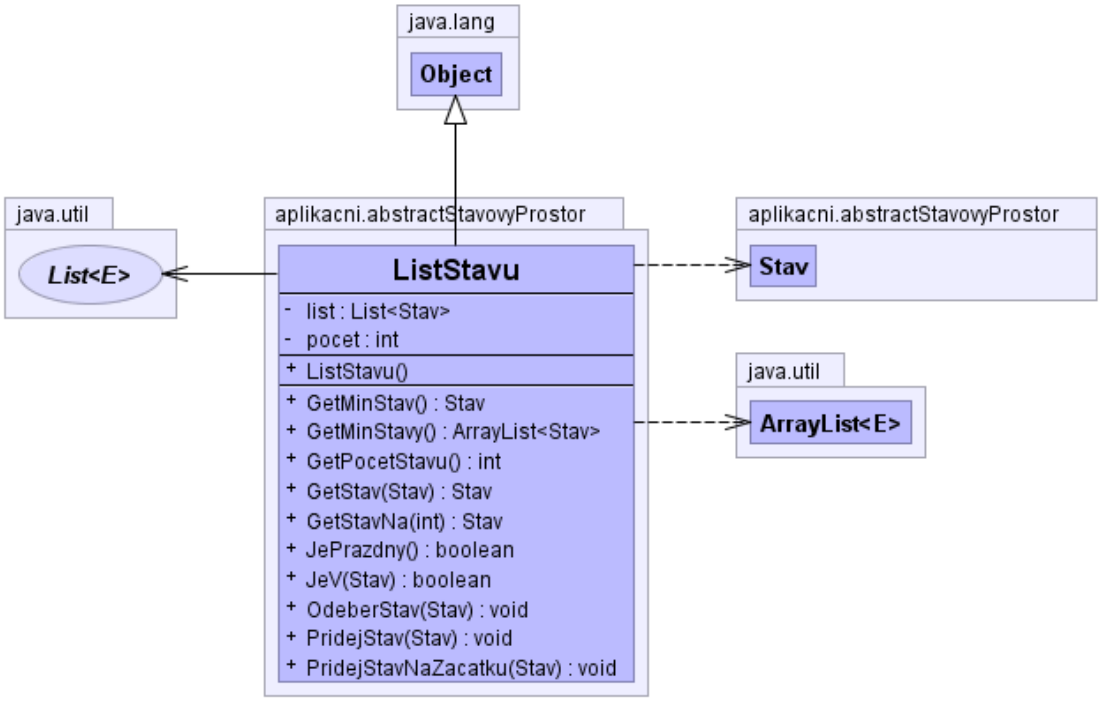
Obrázek 57 – Třída Stav

Tato třída obsahuje datové složky `h`, `g`, `rodicovskyStav` a `dalsiStavy`. Dále bezparametrický konstruktor, pomocí kterého se inicializuje datová složka `dalsiStavy`, abychom potom ve všech dědících třídách nemuseli vytvořit ručně.

Třída ListStavu

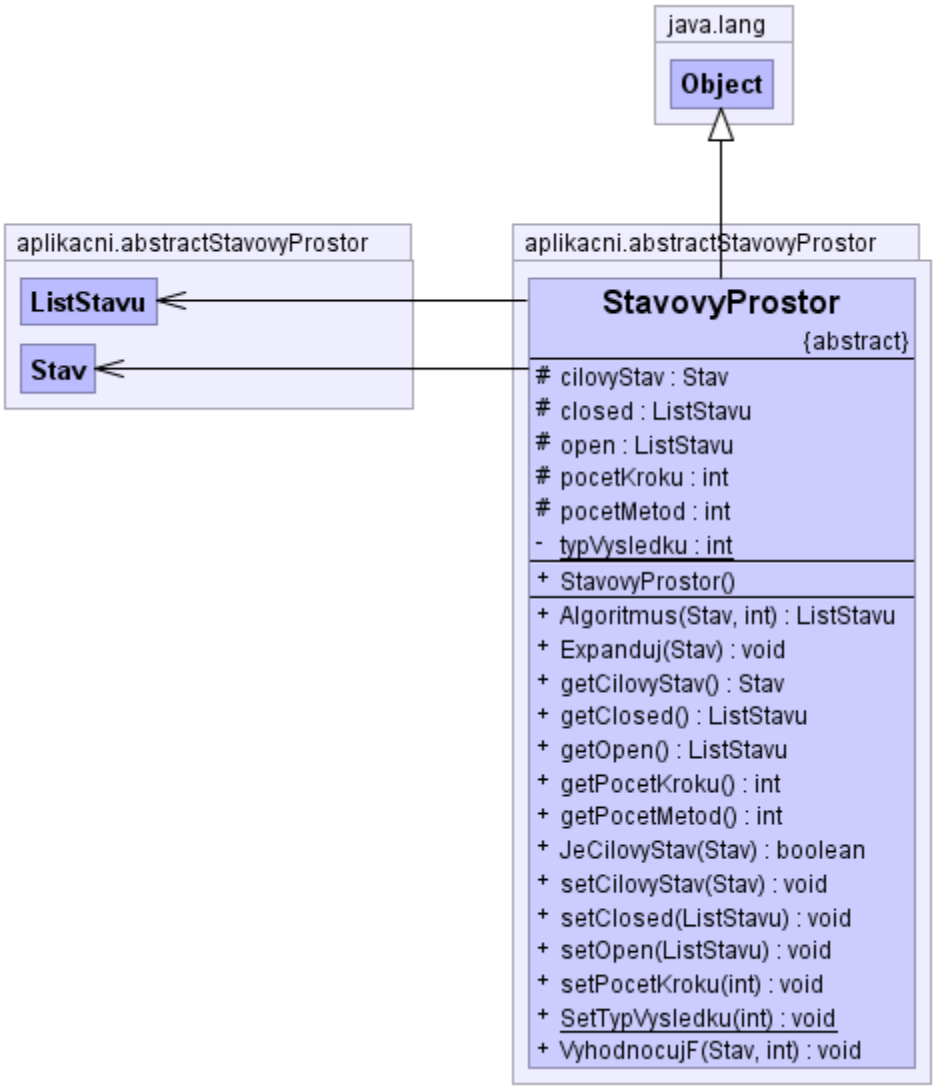
Třída `ListStavu` obstarává seznam stavů a má atributy `počet` a `list`. Atribut `list` používá generickou třídu instancí třídy „`Stav`“ (`List<Stav>`) od Javy. Každá metoda pracuje s tímto seznamem. Kromě konstruktoru, který inicializuje `list`, obsahuje metody:

- `GetMinStav`**()** – vrací stavy s minimálním „*f*“ hodnotou,
- `GetPocetStavu`**()** – vrací počet stavů v listu,
- `GetStavNa(int)` – vrací stav v dané pozici,
- `JePrazdny()` – zjišťuje, zda je list prázdný,
- `JeV(AbstraktniStavovyProstor.Stav)` – zjišťuje, jestli daný stav existuje v listu,
- `OdeberStav(AbstraktniStavovyProstor.Stav)` – odebere daný stav z listu,
- `PridejStav(AbstraktniStavovyProstor.Stav)` – přidá daný stav do listu.



Obrázek 58 – Třída ListStavu

Třída StavovyProstor



Obrázek 59 – Třída StavovyProstor

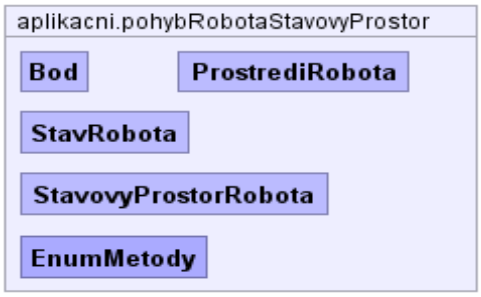
Nyní se podíváme na třídu ***StavovyProstor*** a na obrázku Obrázek 59 můžeme vidět metody a atributy této třídy. Metody „`Expanduj`“, „`JeCilovyStav`“ a „`VyhodnocujF`“ jsou abstraktní a nejsou implementovány v této třídě. Nejdůležitější metodou této třídy je metoda „`Algoritmus`“, která provádí přesun stavů z OPEN do CLOSED a je implementací algoritmu A^*. Dále třída obsahuje get a set metody jednotlivých atributů.

Atributy:

- `cilovyStav` – cílový stav stavového prostoru
- `pocatecniStav` – počáteční stav robota
- `open` – seznam neexpandovaných stavů
- `closed` – seznam už expandovaných stavů
- pocetMetod – počet heuristických funkcí
- pocetKroku – počet posunu, které bylo třeba dosáhnout cílové pozice

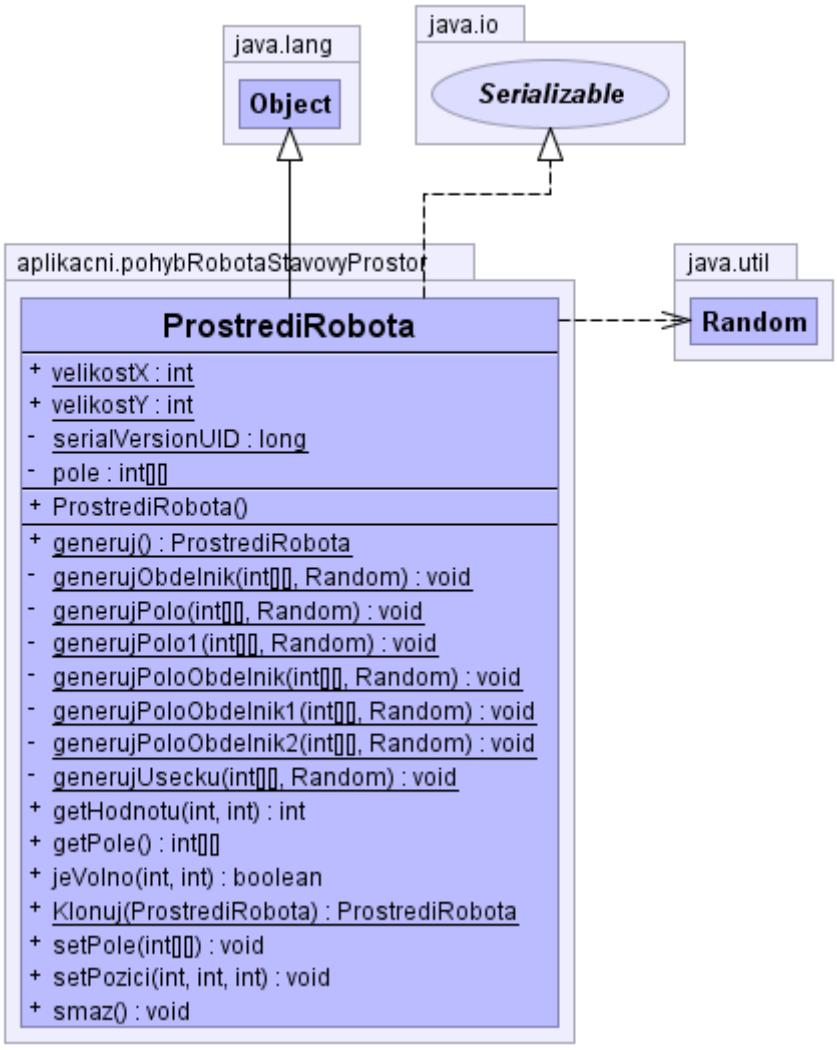
A. 3 Popis tříd – pohybRobotaStavovyProstor

Třídy v balíčku `pohybRobotaStavovyProstor` slouží k řešení našeho zadaného úkolu. Obsahují třídy `PohybRobota`, `StavRobota` a `StavovyProstorRobota`, které jsou odvozené od tříd v balíčku `abstraktniStavovyProstor`.



Obrázek 60 – Balíček pohybRobotaStavovyProstor

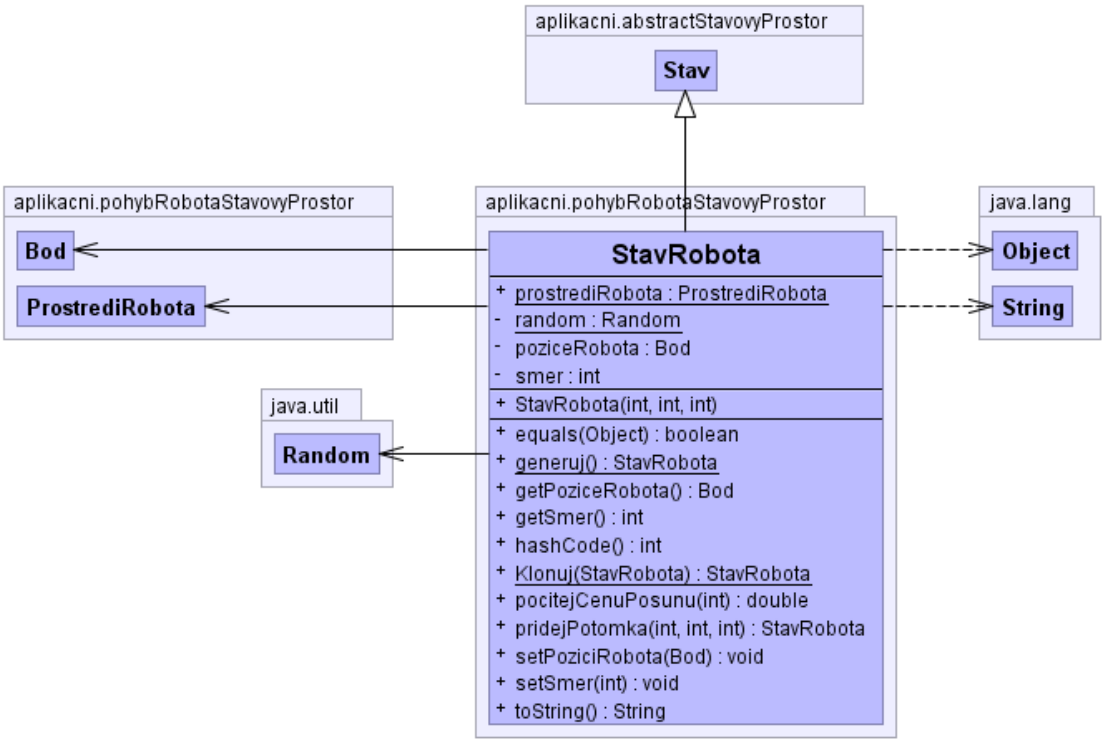
Třída ProstrediRobota



Obrázek 61 – Třída ProstrediRobota reprezentující mapu prostředí

Prostředí, ve kterém se robot pohybuje je znázorněno čtvercovou mřížkou. Jako datová struktura reprezentující tuto mřížku, bylo vybráno dvourozměrné pole a to z důvodu přehlednosti a rychlého přístupu do paměti.

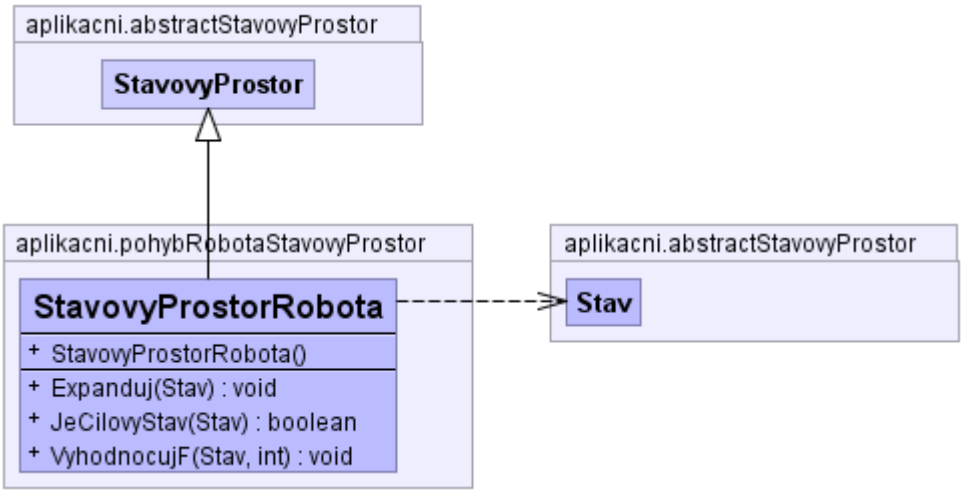
Třída StavRobota



Obrázek 62 – Třída StavRobota

Tato třída je potomkem třídy „Stav“, přičemž dědí všechny jeho atributy. Kromě odvozených, má i další datové složky, a to `poziceRobota` a `směr`. Ty určují polohu a směr robota. Parametrický konstruktor inicializuje nový robot s danými parametry. Metoda „`Equals`“ slouží k porovnání stavu s daným stavem.

Třída StavovyProstorRobota



Obrázek 63 – Třída StavovyProstorRobota

Poslední třídou v balíčku `pohybRobotaStavovyProstor` je třída `StavovyProstorRobot`. Je potomkem třídy `StavovyProstor` a kromě zděděných metod a atributů obsahuje implementaci abstraktních metod přizpůsobené k zadání našeho úkolu.

- `Expanduj(abstraktniStavovyProstor.Stav)` – generuje všechny potomky expandovaného stavu.
- `JeCilovyStav(abstraktniStavovyProstor.Stav)` – testuje, jestli daný stav je stavem cílovým.
- `VyhodnocujF(abstraktniStavovyProstor.Stav)` – pro daný stav vypočítá hodnotu hodnotící funkce *f* podle vzorce uvedené v kapitole 4.2.

A. 4 Ukázka kódu

*Implementace algoritmu A**

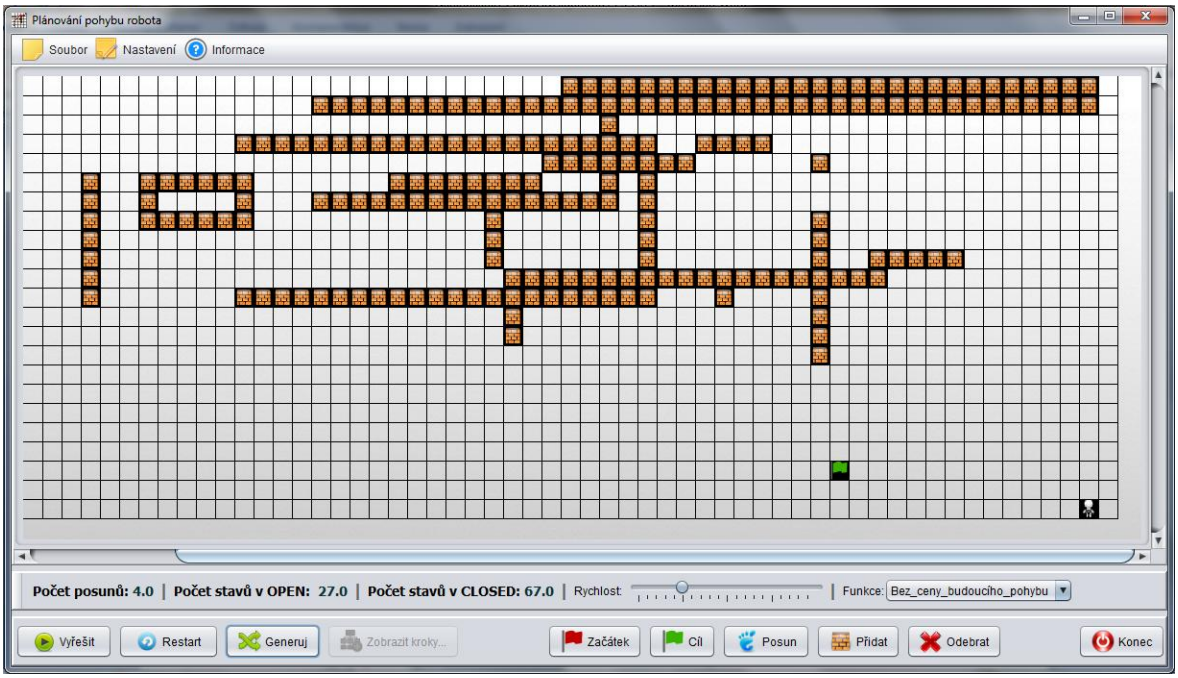
```
public ListStavu Algoritmus(Stav pocatecnyStav, int metoda) {
        …

        do
        {
            // vyber stav s nejmensi hodnotou f ze seznamu OPEN
               minStavy = open.GetMinStavy();

               for (int i = 0; i < minStavy.size(); i++) {
                   if(JeCilovyStav(minStavy.get(i))) {
                       aktualniStav = minStavy.get(i);
                       nalezeno = true;
                       break;
                    }
               }
               if(!nalezeno) {
                   aktualniStav = minStavy.get(0);
               }
```

```
            // zarad stav do seznamu CLOSED
            closed.PridejStav(aktualniStav);
            open.OdeberStav(aktualniStav);
            // jestli je stav cilovym, ukonci prohledavani
             if (JeCilovyStav(aktualniStav)) {
             while (aktualniStav != null) {
                 reseni.PridejStavNaZacatku(aktualniStav);
            //rekonstrujeme cestu od aktualniho az do zacatecniho
                 aktualniStav = aktualniStav.getRodicovskyStav();
             }
             pocetKroku = reseni.GetPocetStavu() - 1;
                 switch (typVysledku) {
                    case 1: return closed;
                    case 2: return open;
                    default: return reseni;
                 }
             }
             // jestli nebylo nalezeno reseni, expanduj stav
             Expanduj(aktualniStav);
             // pro kazdeho naslednika pocitej hodnotu f
             for (int j = 0;
                   j < (aktualniStav.getDalsiStavy().GetPocetStavu());
                   j++)
             {
                 Stav dalsiStav =
                           aktualniStav.getDalsiStavy().GetStavNa(j);
                 VyhodnocujF(dalsiStav, metoda);

                 if(!open.JeV(dalsiStav) && !closed.JeV(dalsiStav)) {
                      open.PridejStav(dalsiStav);
                 }
                 else {
                 Stav existujiciVOpen = open.GetStav(dalsiStav);
                 if(existujiciVOpen!=null&&existujiciVOpen.getG()>
                     dalsiStav.getG()) {
                     open.OdeberStav(existujiciVOpen);
                     minStavy.remove(existujiciVOpen);
                     open.PridejStav(dalsiStav);
                 }

                 Stav existujiciVClosed = closed.GetStav(dalsiStav);
                 if(existujiciVClosed!=null&&existujiciVClosed.getG()>
                     dalsiStav.getG()) {
                     closed.OdeberStav(existujiciVClosed);
                     open.PridejStav(dalsiStav);
                 }
             }
         }
     } while (!open.JePrazdny());
     return null;
 }
```

Příloha B - Uživatelská dokumentace

Tady budu jmenovat popis obsluhy programu z pohledu čtenáře bakalářské práce jako uživatele příslušné úrovně. Program je možný spustit na libovolné platformě, například na Microsoft Windows, většině distribucí Linuxu a dalších, jedinou podmínkou je instalace Javy. Na přiloženém CD je k dispozici spustitelný soubor `PohybRobota.exe`.

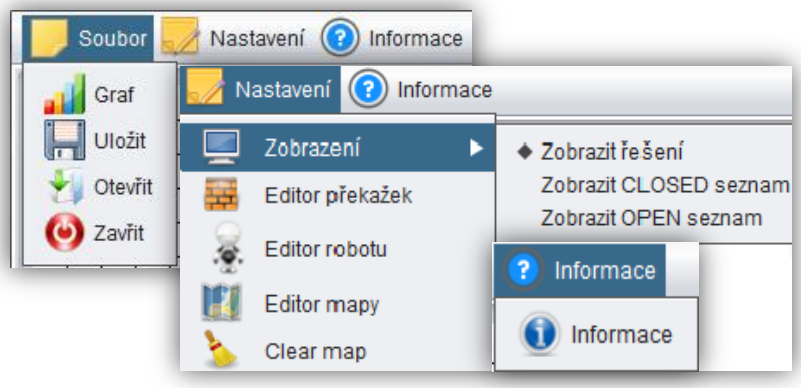
Po spuštění aplikace je k dispozici grafické rozhraní s tlačítky pro ovládání jednotlivých funkcí a je zobrazena čtvercová mřížka s předem generovanými překážkami, počáteční a cílovou pozicí robota.



Obrázek 64 – Náhled uživatelského rozhraní programu

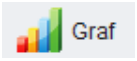
Uživatelské rozhraní se skládá z 3 částí:

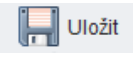
1. Část Menu, která obsahuje položky:

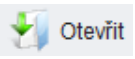


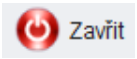
Obrázek 65 – Část menu

a. Menu „Soubor“ obsahuje následující položky.

- 

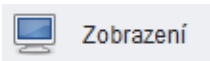
– grafické zobrazení jednotlivých výsledků (obrázek 9).
- 

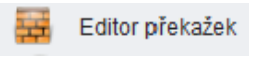
– otevře se dialogové okno pro výběr souboru, kam se uloží aktuální prostředí.
- 

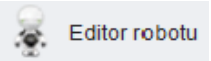
– otevře se dialogové okno pro výběr souboru, ze kterého se načte prostředí a zobrazí ho.
- 

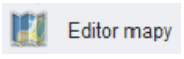
– zavře celý program.

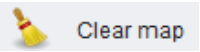
b. Menu „Nastavení“ obsahuje následující položky:

- 

– pomocí tohle tlačítka lze vybrat seznam stavů, které se zobrazí po vyřešení úlohy. Stavy se ukazují animované v takovém pořadí, jak byly uloženy do seznamu. Ve výběru jsou tří možnosti: 1. Nalezení optimální řešení, 2. stavy v seznamu CLOSED – všechny expandované stavy. 3. stavy v seznamu OPEN – zatím neexpandované stavy.
- 

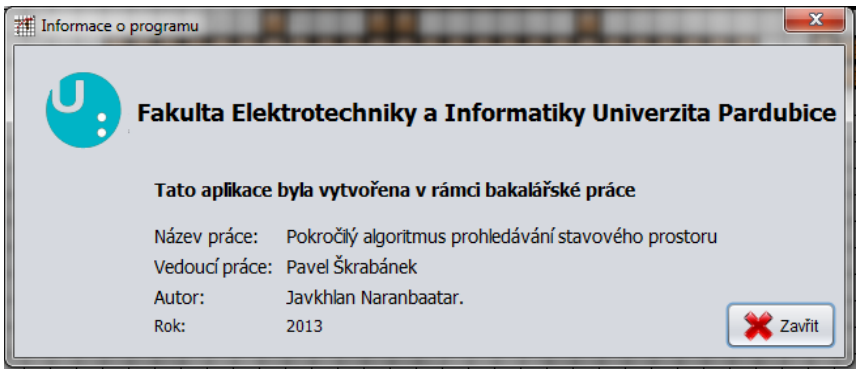
– otevře se dialogové okno (obrázek 7), kde můžeme vybrat obrázek pro budoucí vložené překážky. Obrázky současných překážek se nemění.
- 

– otevře se dialogové okno (obrázek 8), kde můžeme vybrat obrázek robota.
- 

– otevře se dialogové okno (obrázek 10), kde můžeme vybrat velikost mapy v rozmezí 10x10 až 100x100. Po zavření toho dialogu se vygeneruje nové prostředí podle vybrané velikosti a následně se zobrazí na hlavní části.
- 

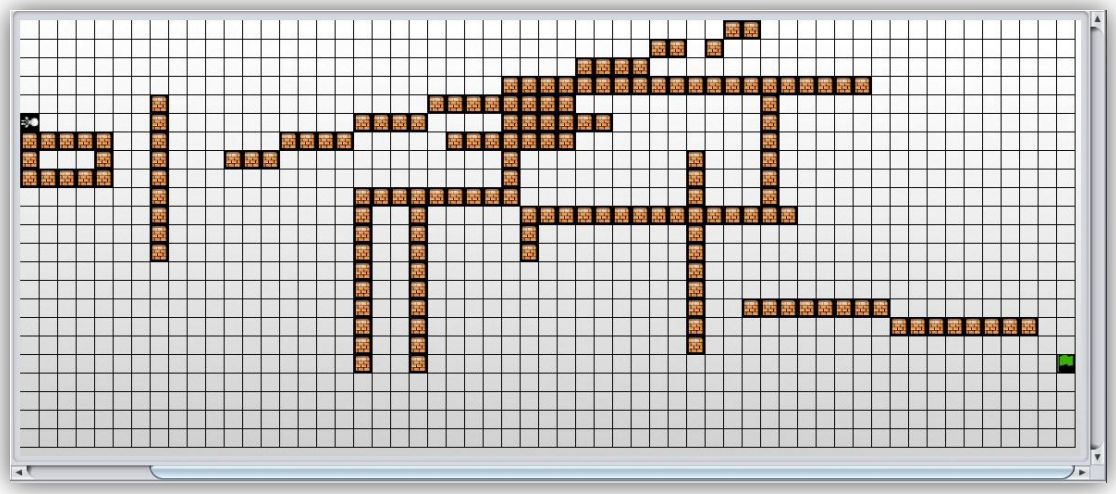
– odstraní všechny překážky.

c. Menu „Informace“ obsahuje údaje o programu.



Obrázek 66 – Informace

2. Část pro zobrazení prostředí, jehož velikost se mění podle aktuální velikosti

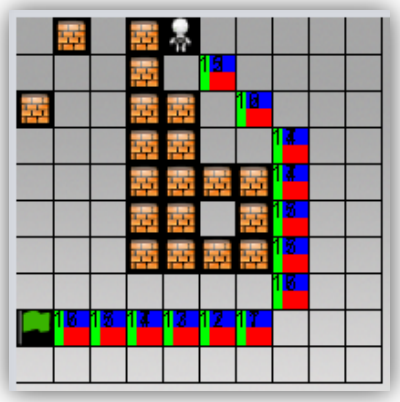


Obrázek 67 – Část pro zobrazení prostředí

3. Část pro ovládání programu

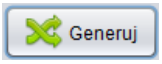
Obrázek 68 – Část ovládání

– Tlačítkem se spustí naprogramovaný algoritmus a vyřeší daný úkol podle zvolené heuristické funkce. Jestliže úkol má řešení, tak se zobrazí jednotlivé posuny robota na mapě předem zvolené rychlosti. Jakmile je úkol vyřešen, není možné editovat prostředí či pozici robota, k tomu je nutné úkol restartovat viz níže.

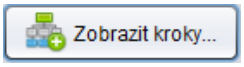


Obrázek 69 – Zobrazení výsledku na mapě

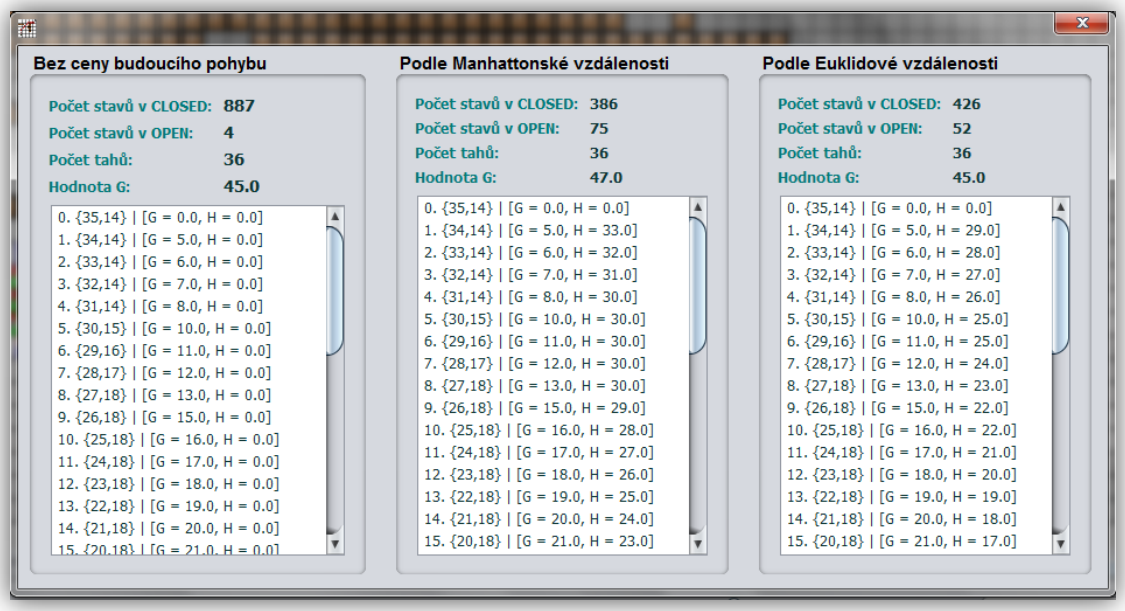
– Tlačítkem se restartuje již vyřešený úkol.



– Tlačítkem se vygeneruje nové prostředí s náhodnými překážkami předem zvolenou velikosti prostředí. A také se vygeneruje začáteční a cílová pozice robota.

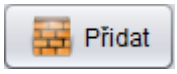


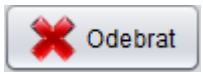
– Jestliže úkol má řešení, bude toto tlačítko aktivní, jinak neaktivní. Po stisknutí tohoto tlačítka se zobrazí dialogové okno, ve kterém můžeme vidět informace, kolik stavů bylo expandováno (počet stavů v seznamu CLOSED), kolik stavů ještě zůstalo v seznamu OPEN a celková cena pohybu robota do cílové pozice atd.

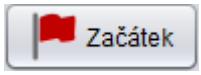


Obrázek 70 – Zobrazení výsledků hodnotících funkcí

Dále jsou k dispozici následující tlačítka:

- 

– stisknutím se aktivuje se funkčnost pro přidání překážky pomocí myši.
- 

– stisknutím se aktivuje se funkčnost pro odebrání překážky pomocí myši.
- 

– stisknutím se aktivuje se funkčnost pro zvolení začáteční pozice robota pomocí myši.
- 

– stisknutím se aktivuje se funkčnost pro zvolení cílové pozice robota pomocí myši.
- 

– stisknutím se aktivuje se funkčnost pro posun robota pomocí myši. Uživatel takhle bude moci porovnávat svoji definovanou cestu s optimální cestou.
- 

– ukončuje celý program.