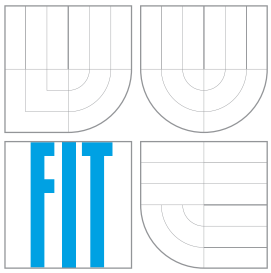
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

ŘEŠENÍ HRY SOKOBAN POMOCÍ GENETICKÝCH ALGORITMŮ

THESIS TITLE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — LEONA NEZVALOVÁ
AUTHOR

VEDOUCÍ PRÁCE — Ing. MILOŠ MINAŘÍK
SUPERVISOR

BRNO 2012

Abstrakt

Tato práce se zabývá tvorbou automatického řešení hry Sokoban s využitím gentických algoritmů. Důraz je kladen především na reprezentaci chromozomu, která pomáhá řešit hlavní problémy hry – velikost stavového prostoru a prezence uváznutí. Na tyto problémy je také zaměřena speciální operace křížení a fitness ohodnocující funkce. Vedlejším cílem je optimalizace nalezeného řešení pomocí optimalizační funkce i samotné evoluce.

Abstract

This work proposes an automatic Sokoban solver based on genetic algorithms. Emphasis is placed on the chromosome representation, that helps to deal with main problems related to automatic solving – size of a state space and presence of deadlocks. These problems are also addressed by specialized crossover operation and fitness function. The secondary objective is to optimize the solution using the optimization function and evolution itself.

Klíčová slova

Genetické algoritmy, Sokoban, evoluce, IDA*, automatické řešení

Keywords

Genetic algorithms, Sokoban, evolution, IDA*, automatic solver

Citace

Leona Nezvalová: Řešení hry Sokoban
pomocí genetických algoritmů, bakalářská práce, Brno, FIT VUT v Brně, 2012

Řešení hry Sokoban pomocí genetických algoritmů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením pana Ing. Miloše Minaříka.

.....
Leona Nezvalová
18. května 2012

Poděkování

Děkuji mému vedoucímu bakalářské práce Ing. Miloši Minaříkovi za odbornou pomoc a cenné rady při zpracování této práce. Dále děkuji své rodině a příteli za podporu a trpělivost.

© Leona Nezvalová, 2012.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

1 Úvod **2**
- 1.1 Motivace . . . 2
- 1.2 Přehled kapitol . . . 2
- 1.3 Současný stav . . . 3

2 Problematika hry Sokoban **4**
- 2.1 Pravidla . . . 4
- 2.2 Uváznutí . . . 5

3 Genetické algoritmy **7**
- 3.1 Inspirace evoluční biologií . . . 7
- 3.2 Reprezentace jedince . . . 8
- 3.3 Selekce . . . 8
- 3.4 Variační operátory . . . 9

4 Návrh **10**
- 4.1 Příprava hrací plochy . . . 10
- 4.2 Reprezentace jedince . . . 11
- 4.3 Fitness ohodnocující funkce . . . 13
- 4.4 Selekce a křížení jedinců . . . 16
- 4.5 Mutace . . . 18
- 4.6 Maximální délka chromozomu . . . 19
- 4.7 Optimalizace . . . 20

5 Implementace **21**
- 5.1 Reprezentace hry . . . 21
- 5.2 Reprezentace jedince . . . 22
- 5.3 Genetický algoritmus . . . 22

6 Experimenty **24**
- 6.1 Velikost populace . . . 25
- 6.2 Typ selekce . . . 27
- 6.3 Počet bodů mutace . . . 27
- 6.4 Problematické typy instancí . . . 28
- 6.5 Shrnutí experimentů . . . 29

7 Závěr **30**

Kapitola 1

Úvod

Sokoban je rozšířená logická počítačová hra. Díky své oblíbenosti a prostému grafickému zpracování vzniká mnoho jejích oficiálních i neoficiálních variant. V této práci se budeme věnovat pouze originální verzi.

Ačkoli pravidla hry jsou jednoduchá, najít správné řešení může být velice obtížné. I rozlohou malé instance mohou obsahovat optimální řešení obsahující desítky tahů. Běžná jsou také řešení čítající tahů stovky. Například nejlepší nalezené řešení první úrovně hry originální sady instancí z roku 1984 obsahuje 230 tahů. Příčinou těchto komplikovaných řešení je rozsáhlý stavový prostor a nutnost vyhnout se určitým stavům hry, ze kterých není možné instanci úspěšně dořešit.

Vzhledem k tomuto charakteru hry se Sokoban stává výzvou i pro oblast inteligentních systémů. V minulosti již byla navržena automatická řešení založená například na principu A* nebo IDA* [9], [10]. Další, méně zkoumanou možností, jak řešit problematiku hry, jsou genetické algoritmy.

1.1 Motivace

Genetické algoritmy jsou často používány pro řešení úloh s velkým stavovým prostorem. Jejich pomocí již byly řešeny hlavolamy Art of Zen [5] a v rámci evolučních algoritmů velice oblíbený Tartatus [1], [4]. Co se hry Sokoban týče, existují studie, které pomocí evolučních algoritmů vytvářejí řešitelné instance hry Sokoban [11], nebo se snaží odhadnout jejich obtížnost [3]. Možnost vytvořit automatické řešení této hry pomocí genetických algoritmů však není dostatečně prozkoumána.

Cílem této práce je rozšířit poznatky týkající se aplikace genetických algoritmů na řešení hry Sokoban. Zaměříme se především na problematiku uváznutí a efektivní zkoumání stavového prostoru hry.

1.2 Přehled kapitol

V kapitole 2 se blíže seznámíme s hrou Sokoban, jejími pravidly a problematikou. V kapitole 3 jsou popsány principy genetických algoritmů a jejich inspirace evoluční biologií. V kapitole 4 následuje návrh vhodného řešení problematiky hry Sokoban pomocí genetických algoritmů. Kapitola 5 popisuje implementaci návrhu. Experimenty s programem jsou shrnuty v kapitole 6, jejich závěry a celkové hodnocení návrhu obsahuje kapitola 7.

1.3 Současný stav

Článek [10] shrnuje výsledky tříleté studie (1996 - 1999), zabývající se tvorbou automatického řešení hry Sokoban. Pravděpodobně se jedná o nejrozsáhlejší výzkum v této oblasti. Autoři zvolili pro prohledávání stavového prostoru algoritmus IDA*. V základním tvaru tento princip nebyl schopný řešit žádnou z instancí v testovací sadě. Tato sada obsahuje celkem 90 úrovní hry, které jsou téměř všechny totožné s původními instancemi zveřejněnými firmou Thinking Rabbit. Postupným přidáváním pomocných algoritmů během výzkumu však konečný program dokázal řešit celkem 57 úrovní. Na webových stránkách kanadské univerzity Alberta nalezneme stručný přehled všech těchto přidaných technik [13]. Na této stránce je také možné stáhnout zdrojové soubory programu, který byl autory nazván *Rolling Stone*.

Studie [3] se zabývá hrou Sokoban z jiného pohledu. Cílem autorů je porovnávat obtížnosti jednotlivých instancí hry. Součástí návrhu je i automatické řešení hry, pracující na principech genetického programování s použitím struktury ISAc List [2]. Obtížnost instance je určena na základě počtu neúspěchů automatického řešení a průměrného času, potřebného k nalezení řešení pro danou instanci hry. Pravidla hry, nad kterými článek pracuje, se však liší od originálních, popsaných v podkapitole 2.1. Vzhledem k zaměření článku je použita testovací sada s instancemi hry, které obsahují jednoduché řešení. Srovnávány jsou například instance obsahující vnitřní zdi s takovými, jejichž plocha obsahuje zdi pouze po obvodu hrací plochy.

Kapitola 2

Problematika hry Sokoban

Sokoban byl navržen roku 1981 Hiroyukim Imabayashim, zaměstnancem japonské softwarové firmy Thinking Rabbit, která tuto hru zveřejnila v roce 1982. Známějším se stalo však pozdější zpracování hry firmou Spectrum Holobyte, které je kompatibilní s počítači IBM.

Bylo dokázáno, že Sokoban je NP-těžký [8] a PSPACE-úplný [6]. Pravidla hry jsou prostá, ale kvůli rozsáhlému stavovému prostoru a prezenci uváznutí může být hledání správného řešení obtížné.

2.1 Pravidla

Hrací plocha se skládá ze čtvercových polí. Každé z nich nabývá jednoho ze sedmi typů (zeď, podlaha, agent, krabice, cílové místo, cílové místo s krabicí, cílové místo s agentem) 2.1. Agent je na hrací ploše pouze jeden a počet cílových míst je shodný s počtem krabic. Přesun agenta na jedno políčko z jeho čtyřokolí nazýváme jedním *tahem*. Přesuny agenta v jiném než vodorovném nebo svislém směru nejsou dovoleny. Agent je schopen krabicemi pohybovat, ale nejvýše jednou krabicí zároveň v každém tahu. Krabice lze pouze tlačit, nikoli táhnout. Základním cílem hry je přesunout všechny krabice pomocí agenta z počátečních pozic na místa cílová, přičemž krabice i agent se mohou pohybovat pouze po podlaze.

Od vzniku hry Sokoban jsou jeho pravidla často upravována, studie [3] např. uvažuje cílová místa jako jámy, ze kterých se agent nemůže dostat ven. Až po umístění krabice se z takovéto jámy stává podlaha, po které se lze pohybovat, případně přesouvat zbylé krabice. V této práci uvažujeme pouze originální pravidla hry.

Konečné řešení hry lze ohodnotit několika způsoby — dle počtu tahů agenta, posunů krabic, popřípadě obojího. Na obrázku 3 je znázorněna první úroveň hry původní sady instancí, jejíž nejlepší dosažené řešení obsahuje 230 tahů a 97 posunů.

(a) podlaha (b) zeď (c) krabice (d) cílové místo (e) agent (f) cílové místo s krabicí (g) cílové místo s agentem

Obrázek 2.1: Znázornění jednotlivých typů políček.

2.2 Uváznutí

Uváznutí je stav hrací plochy, který vznikne přemístěním krabice na určité místo a ze kterého nelze hru úspěšně dořešit. *Stavem hrací plochy* chápeme pozici agenta a rozmístění krabic na ploše. Stavům způsobujícím uváznutím je třeba se během řešení vyhýbat. Jejich rozpoznání přináší zmenšení stavového prostoru, ve kterém řešení hledáme, ale tato detekce je zvláště u některých typů obtížná. Pro potřeby této práce jsou uváznutí rozdělena na statická a dynamická. Dynamická uváznutí se dále dělí na lokální a globální.

2.2.1 Statické uváznutí

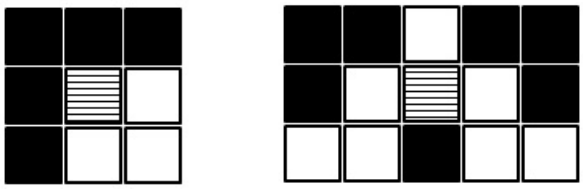
Statickým uváznutím rozumíme umístění krabice do rohu tvořeného zdmi nebo na spojnici mezi dvěma rohy, která je alespoň z jedné strany vždy ohraničená zdí a neobsahuje cílové místo (obrázek 2.2). Uváznutí pak způsobuje právě krabice ležící v takovéto oblasti, protože ji již nelze přesunout na cílové místo. Místa, která způsobují statické uváznutí, lze označit při inicializaci hrací plochy. Agent se vyhýbá tahům, které by způsobili přesun krabice na takto označená místa a tím se výrazně zmenšuje stavový prostor.

2.2.2 Dynamické uváznutí

Pokud se krabice, případně i agent, dostanou do takových pozic, kde vzájemným působením brání v úspěšném řešení hry, mluvíme o *dynamickém uváznutí* (obrázek 2.3). Vznik dynamického uváznutí neovlivňuje jen rozestavení zdí a nastává proto v průběhu hry.

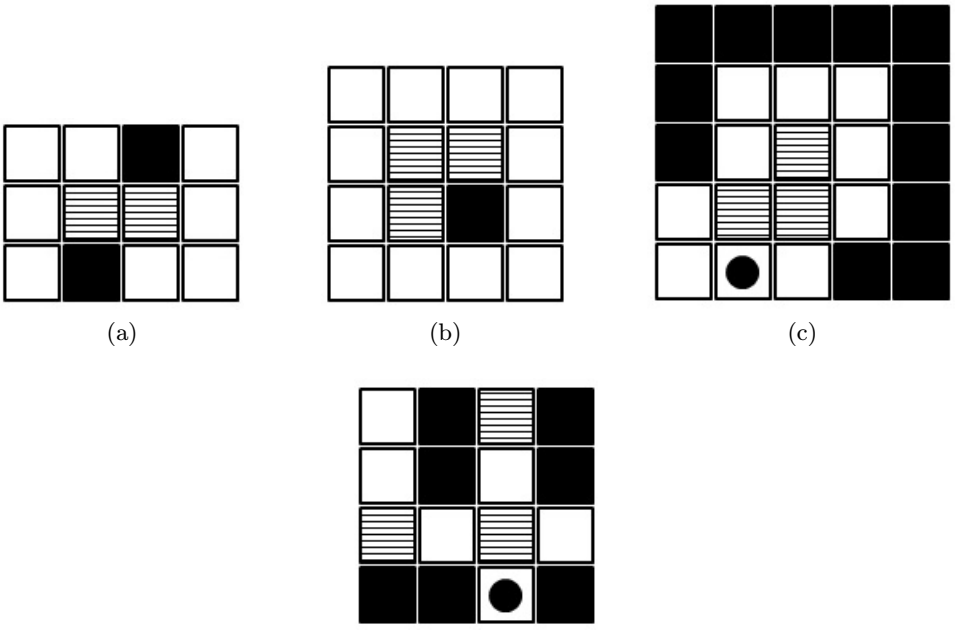
Jestliže na ploše vznikne shluk dvou nebo více krabic sousedících hranou, který je neměnný — všechny zahrnuté krabice jsou nepohyblivé — označujeme takovýto stav jako *dynamické uváznutí lokálního typu. Pohyblivá krabice* je taková, která alespoň z jednoho páru protilehlých stran není obklopena krabicemi nebo zdmi (krabici je odkud tlačit) a dále alespoň jedna z této stran není označena uváznutím (krabici je kam tlačit).

Uváznutí globálního typu je dáno celkovým stavem hrací plochy. Krabice se dostávají do vzájemného postavení (aniž by spolu musely sousedit), ze kterého je z pozice agenta již nelze umístit na cílová místa. Pro detekci globálního uváznutí je třeba analyzovat stav celé hrací plochy. Tento typ uváznutí se proto stává hlavním problémem hry.



(a) (b)

Obrázek 2.2: Příklady statických uváznutí.



Obrázek 2.3: Příklady dynamických uváznutí. Na obrázku (a) a (b) se jedná o uváznutí lokální. Na obrázku (c) vidíme uváznutí globální. Na obrázku (d) jsou krabice seskupeny, ale jedná se také o dynamické uváznutí globální, protože vliv má i pozice agenta.

Kapitola 3

Genetické algoritmy

Genetické algoritmy patří do třídy evolučních algoritmů, inspirovaných evoluční biologií. Populace je zde reprezentována jako konečná množina kandidátních řešení. Nad touto populací jsou prováděny operace selekce, mutace a křížení. Genetický algoritmus můžeme popsat následujícím pseudokódem:

```
vytvoř počáteční populaci
while(není splněna ukončující podmínka){
    ohodnoť jedince v populaci
    vyber rodiče pro křížení
    vytvoř nové jedince operací křížení
    nahraď část populace potomky
    proveď operaci mutace
}
```

Genetické algoritmy jsou vhodné pro řešení problémů, o kterých nemáme příliš mnoho informací. Nezbytné jsou pouze znalosti potřebné k ohodnocení kandidátního řešení. Vzhledem k velikosti prohledávaného prostoru, který je často velice rozsáhlý, nemusí být řešení vůbec nalezeno. Právě pro minimalizaci tohoto rizika je nutné vytvořit nejen efektivní algoritmus pro výpočet fitness ohodnocení a zvolit vhodné přístupy variačních operátorů, ale i vhodně reprezentovat jedince.

3.1 Inspirace evoluční biologií

Zakladatelem evoluční biologie je Robert Charles Darwin [7]. Jeho teorie předpokládá, že organismy procházejí vývojem, který je dán přirozeným výběrem nejsilnějších jedinců, jejich množením a mutací.

Principem přirozeného výběru je přežití dostatečně kvalitních jedinců do produktivního věku. Tito jedinci své vlastnosti přenáší na potomky. Potomků je více než je potřeba pro udržení stabilní populace, přežívá tedy pouze část s vlastnostmi vhodnějšími pro dané životní podmínky a cyklus se opakuje.

Mutace je změna genetické informace jedince. Její vliv na vznik nových živočišných druhů (makroevoluce) nebyl dokázán, ale v rámci vědecké oblasti je chápána jako jedna z příčin vzniku nových variací daného druhu (mikroevoluce) [12].

Výsledkem mutace a přirozeného výběru je vznik druhových variací, které jsou adaptovány na prostředí a podmínky, ve kterých žijí.

3.2 Reprezentace jedince

Informace jedince – *chromozom* – je nejčastěji reprezentována jako uspořádaná struktura prvků stejného datového typu. Tyto prvky se nazývají geny a popisují jednu vlastnost jedince. Nejčastěji je chromozom zakódován jako znakový, číselný nebo bitový řetězec.

Prvky množiny přípustných hodnot, kterých gen může nabývat, se nazývají *alely*. Význam takto zakódovaného řetězce se nazývá *fenotyp*.

Reprezentace chromozomu zásadně ovlivňuje úspěšnost algoritmu, jeho rychlost, čitelnost i paměťové nároky a pro jeho volbu je třeba stanovit priority těchto atributů. Komplexní datové struktury pravděpodobně povedou k jednoduchému zjištění fenotypu chromozomu, ale časové nároky programu se mohou stát neúnosnými.

3.3 Selekce

Selekce je operátor, který vybírá jedince vhodné ke křížení, a podporuje tak konvergenci k úspěšnému řešení.

Míra, která určuje upřednostňování *elity* (jedinců s vysokým fitness ohodnocením) se nazývá selekční tlak. Díky vysokému selekčnímu tlaku může algoritmus najít řešení v krátkém čase, zvyšuje se ale riziko konvergence k lokálnímu optimu. Naopak s nízkým selekčním tlakem rychlost konvergence k úspěšnému řešení klesá.

Existuje několik přístupů k selekci, které společně s nastavením jejich parametrů představují různou míru kompromisu mezi zmíněnými problémy:

- Turnaj – jedinci jsou náhodně rozčleněni do skupin velikosti n, kde jsou seřazeni podle fitness ohodnocení. Vítězové těchto skupin jsou vybráni jako jedinci vhodní ke křížení. Velikostí skupiny je určován selekční tlak — čím početnější skupina, tím kvalitnější bývá průměrně vítěz.

- Ruleta – pravděpodobnost, že jedinec i bude vybrán ke křížení, je dána vzorcem:

$$p_i = \frac{f_i}{\sum_{j=1}^{n} f_j}, \tag{3.1}$$

  kde f_i je fitness hodnota jedince i a n je počet jedinců v populaci.

- Podle pořadí – jedinci jsou seřazeni podle jejich fitness ohodnocení a následně jim je dle jejich postavení přiřazena hodnost z intervalu $1 - N$, kde N je počet jedinců v populaci a náleží nejkvalitnějšímu jedinci z populace. Pravděpodobnost, že jedinec i bude vybrán ke křížení je dána vzorcem:

$$p_i = \frac{r_i}{\sum_{j=1}^{n} r_j}, \tag{3.2}$$

  kde r_i je hodnost jedince i a n je počet jedinců v populaci. Tato metoda se v důsledku liší od rulety zanedbáváním velikosti rozdílu ve fitness ohodnocení mezi jedinci.

3.4 Variační operátory

Zatímco počáteční populace je generována náhodně, popřípadě je výstupem jiného programu, její další vývoj a obměna se řídí variačními operátory – křížením a mutací.

Křížení kombinuje vlastnosti jedinců, a vytváří tak nové jedince. Mutace pozměňuje jedince stávající a přináší nové informace do populace.

3.4.1 Křížení

Operátor křížení dává vzniknout novým jedincům (potomkům), kteří dědí vlastnosti svých rodičů. Vzhledem k tomu, že selekce rodičů upřednostňuje kvalitnější jedince před slabšími, je pravděpodobné, že potomci budou mít vyšší ohodnocení než nejslabší jedinci v populaci. V ideálním případě je dosaženo kvalitnějšího jedince než jsou samotní rodiče.

Principem křížení je kopírování částí chromozomu rodičů a vytvoření nových (nejčastěji dvou) chromozomů. Způsobů, jak tyto části určit, je několik:

- Jednobodové křížení – rodiče jsou rozděleni na dvě části v náhodně určené pozici.
- n-bodové křížení – podobný postup jako u jednobodového křížení s tím rozdílem, že počet pozic, kde je rodič rozdělen, určuje parametr n.
- Uniformní křížení – pro každou pozici genu rodičů je generováno náhodné číslo p, dle kterého je rozhodnuto který potomek od kterého rodiče daný gen zdědí. Tento přístup přináší různorodost do populace, ale není vhodný pro problémy, kde kopírováním rodičů na úrovni genů rozbíjí logické celky chromozomu.
- Adaptivní křížení – každý jedinec má předlohu určující geny, které budou kopírovány do prvního potomka. Tato operace je vždy řízena jedním z rodičů. Nově vzniklí jedinci získají vlastní šablonu křížením šablon rodičů.

3.4.2 Mutace

Mutace je proces, který změní hodnotu jednoho nebo více genů v chromozomu. Přispívá tak k různorodosti populace a snižuje riziko uváznutí v lokálním optimu. Rozlišujeme několik základních principů pro výběr genů k mutaci:

- Mutace n-bodová – počet genů určených k mutaci je dán parametrem n.
- Mutace pravděpodobnostní – každý gen chromozomu je mutován s pravděpodobností p.

Princip mutace je volen na základě reprezentace chromozomu. U reprezentace chromozomu číselnými hodnotami je často používána mutace Gaussova, která mění hodnotu genu o hodnotu s Gaussovým rozložením. Pokud gen nabývá bitových hodnot, hodnota genu je při mutaci invertována. V případech, kdy není možné určit míru s jakou měníme význam genu, je nová hodnota vybrána náhodně z množiny přípustných hodnot.

Kapitola 4

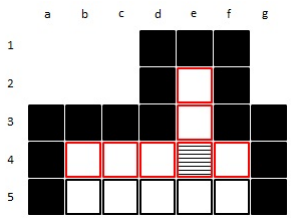
Návrh

V předchozí kapitole jsme zmínili jako hlavní problémy hry rozsáhlý stavový prostor a přítomnost dynamických globálních uváznutí. V této kapitole najdeme návrh řešení hry Sokoban a těchto problémů. Na zmenšení stavového prostoru se zaměřuje především reprezentace kandidátního řešení a označení políček na ploše, která způsobují uváznutí. Problematická detekce dynamického globálního uváznutí je částečně řešena ve fitness ohodnocující funkci.

4.1 Příprava hrací plochy

Než je spuštěna samotná evoluce, jsou na hrací ploše označena místa způsobující uváznutí. Jedná se o statická uváznutí a malou podmnožinu globálních dynamických uváznutí.

Jak bylo zmíněno v kapitole 2.2, rozpoznání míst způsobujících statické uváznutí je prosté, protože nemusíme uvažovat pozici agenta ani ostatních krabic. U uváznutí dynamického globálního tyto informace již musíme uvažovat. V některých případech je však zřejmé, že agent se do pozice, ze které by krabici bylo možné posouvat, nemůže nikdy dostat. Jedná se o z jedné strany uzavřené tunely o šířce jedné buňky. Příklad vidíme na obrázku 4.1. Takováto místa jsou také označena uváznutím. U původní sady instancí hry se s tunely způsobujícími uváznutí příliš často nesetkáváme. Jejich odhalení je pro hráče i inteligentní systémy triviální. Mohou však vznikat jednodušší instance, které by se bez této detekce staly pro program obtížně řešitelnými.



Obrázek 4.1: Dynamické globální uváznutí, které však můžeme předvídat již před začátkem hry. Agent může krabici na *e4* odsunout z červeně vyznačených míst pouze z pozice *e2* nebo *e3*. Z obrázku je však zřejmé, že takového umístění nemůže kvůli krabici nikdy dosáhnout.

4.2 Reprezentace jedince

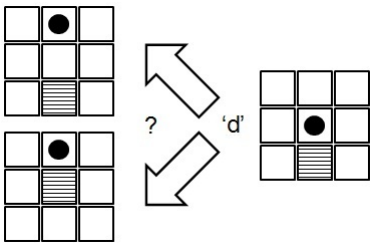
Jedinec obsahuje jeden chromozom, který reprezentuje kandidátní řešení, tzn. pohyb agenta po ploše. Existuje několik způsobů, jak řešení zakódovat. Nejjednodušším principem je reprezentovat řešení jako sled instrukcí definujících kterým směrem se agent pohybuje. Přístup, kdy je taková hodnota genu volena náhodně bez ohledu na stav hrací plochy, je vzhledem k rozsáhlému stavovému prostoru nevhodný. V této práci proto budeme řešení generovat pomocí agenta, který má k dispozici informace o hrací ploše.

Gen v tomto řetězci dává informaci o jednom tahu agenta a nabývá jedné z osmi znakových hodnot, které jsou běžně používané pro reprezentaci řešení hry Sokoban:

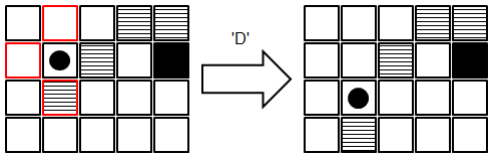
- u – pohyb směrem nahoru
- r – pohyb směrem doprava
- d – pohyb směrem dolů
- l – pohyb směrem doleva
- U – pohyb směrem nahoru s posuvem krabice
- R – pohyb směrem doprava s posuvem krabice
- D – pohyb směrem dolů s posuvem krabice
- L – pohyb směrem doleva s posuvem krabice

Velikostní rozlišení znaků pro tah s krabicí a bez krabice je potřebné pro zpětné vyhodnocování, kde se krabice místo tlačení táhne. V tomto případě je třeba určit, zda agent při pohybu posouvá i krabici (obrázek 4.2). Grafická zpracování hry Sokoban zpětné řešení často používají, protože se tak lze kdykoliv vrátit na některý z předchozích tahů. Použití popsaného formátu umožňuje řešení zobrazit i v takovýchto programech.

Hodnoty genu jsou voleny s ohledem na stav hrací plochy. Agent nejdříve označí ta políčka ve svém čtyřokolí, která jsou pro tah vhodná. Teprve z takto označených políček agent náhodně vybere jedno, určující směr tahu (obrázek 4.3).



Obrázek 4.2: Zpětné vyhodnocení pro tah směrem dolů bez rozlišení znaku pro posuv krabicí. Je jisté, že agent přišel ze shora, ale nevíme, zda posouval krabici nebo ne.

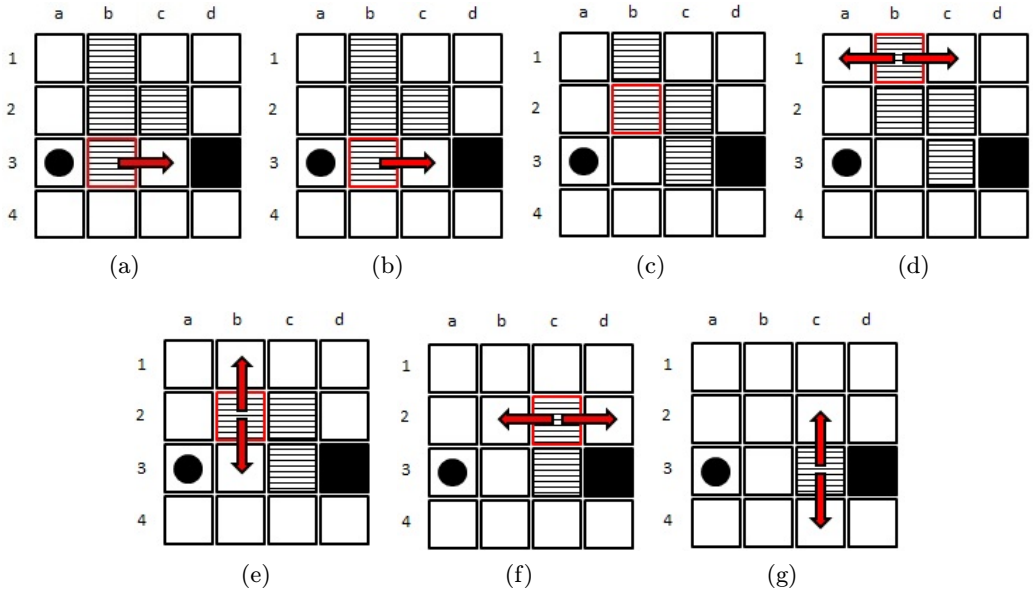


Obrázek 4.3: Volba tahu. Červeně jsou označena políčka, která agent označil jako vhodná pro tah. Vlevo vidíme stav hrací plochy, kdy bylo náhodným způsobem zvoleno spodní políčko – tah směrem dolů.

4.2.1 Vhodnost tahu

Podmínky, na jejichž základě je políčko označeno jako vhodné pro tah, jsou dány jeho typem. Pokud políčko obsahuje zeď, není možné se na něj přesunout a stává se pro tah nevhodným. Políčko, které obsahuje krabici, musí splňovat následující podmínky:

- Krabici lze posunout – *koncové místo krabice* (políčko, které leží ve vzdálenosti 2 od agenta ve stejném směru jako zkoumaná krabice) nesmí obsahovat zeď či jinou krabici. Posuv krabicí by v takovém případě odporoval pravidlům hry.
- Posuv krabicí nezpůsobí uváznutí, které je možné rozpoznat – koncové místo krabice nesmí být označeno jako políčko způsobující uváznutí.
- Posuv krabicí nezpůsobí dynamické lokální uváznutí – pro detekci dynamického lokálního uváznutí je třeba zjistit, zda by krabici na koncovém místě šlo dále posouvat. Pokud ne, kontroluje se pohyblivost krabic, které v tomto pohybu brání. Algoritmus je rekurzivní a pracuje dokud není ověřeno, že krabice bránící v pohybu krabici v koncovém místě lze přemístit tak, aby tento pohyb byl umožněn, nebo dokud nejsou zkontrolovány všechny možnosti jak takovýto pohyb umožnit. Jediná výjimka nastává v případě, kdy nepohyblivý blok je tvořen pouze krabicemi na cílových místech. V takovém případě se o uváznutí nejedná. Grafické znázornění tohoto algoritmu můžeme vidět na obrázku 4.4.



Obrázek 4.4: Grafické znázornění algoritmu pro kontrolu dynamického lokálního uváznutí. Agent posouvá krabici $b3$ doprava (a). Po posunutí jí v pohyblivosti brání krabice $c2$ (b), která je také nepohyblivá kvůli krabici $b2$ (c). Podobně krabici $b2$ brání v pohybu krabice $b1$, která již je pohyblivá (d). Zpětně se tak stává pohyblivou i krabice $b2$ (e), $c2$ (f) a konečně i $c3$ (g). Je ověřeno že tento tah nezpůsobí dynamické lokální uváznutí.

Pokud políčko obsahuje volnou plochu, tedy podlahu nebo cílové místo, agent vyžaduje splnění jediné následující podmínky:

- Pokud agent v předchozím tahu neposunul krabici, políčko nesmí ležet ve směru, ze kterého agent přišel.

Tato podmínka brání zbytečným tahům agenta tam a zpět bez posuvu krabice. Tímto způsobem se prohledávání stavového prostoru hry stává efektivnějším. Situace, kdy se agent přesune na políčko, na kterém se nacházel v předchozím tahu, nastává jedině v případě, kdy žádné jiné políčko není označeno jako vhodné.

4.2.2 Vývoj jedince během evoluce

Výše popsaným způsobem je vytvořena počáteční populace. Každý jedinec obsahuje zakódovaný pohyb agenta po ploše, kde geny obsahují vždy proveditelné instrukce.

Vzhledem k principům hry Sokoban lze logickou stavbu chromozomu lehce narušit. Pokud změníme jedinou hodnotu genu, stav hrací plochy je změněn pro všechny geny následující, které však byly agentem generovány s ohledem na stav původní. Z takové části chromozomu se následně stává běžný řetězec, nerespektující stav hrací plochy. Tento problém nastává při operaci mutace. Chromozom lze od změněné pozice genu znovu vyhodnotit tak, aby řešení bylo opět validní a prohledávání stavového prostoru efektivní (viz následující podkapitola). Z mutace se však stává operace, která přináší zásadní změny do fenotypu jedince. Podobný problém nastává i při operaci křížení, která však byla navržena tak, aby dopady na logickou stavbu chromozomu byly minimalizovány.

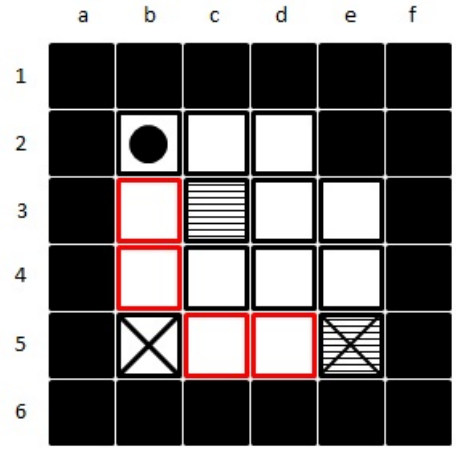
4.2.3 Fenotyp jedince

Fenotyp jedince chápeme jako výsledný průchod plochou, kterého docílíme postupným vyhodnocováním každého genu. Pokud je detekován nevalidní tah, agent je nucen vyhodnotit aktuální stav hrací plochy a zvolit nový vhodný směr tahu. *Nevalidním tahem* chápeme tah, který není na aktuálním stavu hrací plochy uskutečnitelný, tzn. přesun agenta na políčko obsahující zeď nebo posuv v daném směru nepohyblivou krabicí.

4.3 Fitness ohodnocující funkce

Fitness funkce má za úkol určit kvalitu jedince a zásadním způsobem tak ovlivňuje jeho roli v populaci. Míru kvality určuje množství fitness bodů, které jedinec obdrží na základě informací zjištěných z jeho fenotypu.

Nejzásadnější je informace o tom, zda je jedinec označen jako *stagnující*. Stagnující jedinec je takový, jehož řešení potenciálně obsahuje dynamické globální uváznutí. Algoritmus této detekce je popsán dále v samostatné podkapitole. Pokud je jedinec označen jako stagnující, dostává 0 fitness bodů a bodové ohodnocení se dále neprovádí. U ostatních jedinců je dalším faktorem počet krabic na cílových místech a počet krabic na *místech blízkých cílovým*. Takováto místa leží na vodorovné nebo svislé přímce procházející místem cílovým, které není obsazeno krabicí. Přímka je ohraničena zdí nebo políčkem označeným uváznutím (obrázek 4.5). Fitness ohodnocení upravuje i penalizace, která je udělována za délku chromozomu potřebnou pro dosažení určitých parametrů řešení. Protože optimální délka chromozomu není dopředu známa, velikost penalizace je určována relativně dle ostatních jedinců v populaci. Tímto se stává nezbytným zjistit fenotyp všech jedinců v populaci dříve, než začne samotné bodové hodnocení.



Obrázek 4.5: Jednoduchá instance hry Sokoban. Červeně jsou označena místa blízká cílovým. Cílové místo $e5$ žádná blízká místa nedefinuje, protože je již obsazeno. Políčko $b2$ není třeba označovat, protože způsobuje statické uváznutí a agent na něj krabici nikdy neumístí.

4.3.1 Bodové ohodnocení

Pro stanovení fitness ohodnocení jedinců jsou potřebné následující informace o jejich řešení:

- počet krabic umístěných v cílových místech,
- počet krabic umístěných na místech blízkých cílovým,
- délka chromozomu do poslední pozice genu, která mění počet krabic umístěných na cílových místech,
- délka chromozomu do poslední pozice genu, která mění počet krabic umístěných na místech blízkých cílovým.

Počet krabic na cílových místech nejvýznamněji ovlivňuje ohodnocení jedince. Každá takováto krabice přináší 2 fitness body do celkového ohodnocení. Následně je jedinci udělena penalizace, která zohlední jeho pozici ve skupině jedinců se stejným počtem krabic na cílových místech. Skupina je seřazena dle délky chromozomu potřebné k dosažení daného počtu umístěných krabic od nejkratšího po nejdelší. Tento typ penalizace je dán vztahem:

$$P_g = 2/N * (k - 1), \tag{4.1}$$

kde N je počet jedinců se stejným počtem krabic na cílových místech a k je pozice jedince v seřazené skupině. Z rovnice je zřejmé, že první jedinec ve skupině není penalizován. Celkový počet bodů jedince, dosažený v oblasti počtu krabic umístěných na cílových místech, je dán vztahem:

$$F' = B_g * 2 - P_g, \tag{4.2}$$

kde B_g je počet krabic umístěných na cílových místech.

Dalším faktorem určujícím kvalitu jedince je počet krabic umístěných na místech blízkých cílovým. Krabici umístěnou na takovémto místě zpravidla nebývá problém do místa cílového dotlačit a hodnotíme ji proto jedním fitness bodem. Princip udělování penalizace je stejný jako u penalizace v rámci krabic na místech cílových, ale operuje se zde nad skupinou

tvořenou jedinci se stejným počtem krabic umístěných na místech blízkých cílovým. Délka chromozomu, která má vliv na umístění ve skupině, je dána počtem genů, které ovlivňují počet takto umístěných krabic. Velikost penalizace tedy můžeme vyjádřit rovnicí:

$$P_c = 1/N * (k - 1). \tag{4.3}$$

Počet bodů, dosažený v oblasti parametru počtu krabic umístěných na místech blízkých cílovým, je dán tímto vztahem:

$$F'' = \begin{cases} (B_c - I_c) - P_c & \text{pro } B_c > I_c \\ 0 & \text{pro } B_c <= I_c \end{cases}, \tag{4.4}$$

kde B_c je počet krabic umístěných na cílových místech a I_c je počet krabic umístěných na cílových místech v počátečním stavu hrací plochy. Do ohodnocení se tedy promítne jen počet krabic, o který řešení převyšuje počáteční konfiguraci hrací plochy. Toto opatření zabraňuje uváznutí v lokálním optimu, které nastává na samotném začátku evoluce. U instancí, které v počátečním stavu obsahují krabice na místech blízkých cílovým, hrozí riziko, že se budou jedinci pouze vyhýbají tahům, které by krabice z takovýchto míst odstranili. Důsledkem této skutečnosti je i zkracování délky chromozomů v populaci. Elita je následně zaplněna jedinci, jejichž řešení jsou krátká a nemění stav hrací plochy. Stejný problém nastává i při řešení nad instancemi, které ve své počáteční konfiguraci obsahují krabice na místech cílových. Nezapočítáním takovýchto krabic do fitness ohodnocení bychom však ztratili hlavní vodítko, kterým určujeme kvalitu jedince. Tímto řešením by tedy vznikl pouze další problém. Zkracování chromozomu je omezeno stanovením minimální délky chromozomu.

Celkové fitness ohodnocení jedince pak spočítáme jako:

$$F = F' + F''. \tag{4.5}$$

Způsob, kterým je udělována penalizace za délku řešení, zvýhodňuje jedince s kratší délkou. Tímto dochází k optimalizaci řešení, ne však na úkor rychlosti konvergence k řešení. Větší váhu má vždy počet krabic umístěných na cílových nebo jim blízkých místech.

4.3.2 Detekce stagnujících jedinců

Spolehlivě detekovat dynamické uváznutí globálního typu je v rámci genetických algoritmů problematické. Pokud budeme sledovat pohyblivost krabic během agentovy cesty herní plochou, můžeme rozpoznat alespoň takové jedince, kteří potenciálně obsahují dynamické globální uváznutí v konečném stádiu. *Konečným stádiem* rozumíme takový stav globálního uváznutí, kde není možné posunout žádnou krabici na ploše.

Fitness funkce po zjištění fenotypu jedince pokračuje v generování agentova průchodu plochou nastavováním chromozomu o další geny. Zároveň jsou počítány výskyty krabic, které se nacházely v agentově čtyřokolí a nebylo je možné z pozice agenta posunout. Každá krabice může být započítána vícekrát v závislosti na počtu průchodů agenta kolem ní. Krabice umístěné na cílových místech do výsledku zahrnuty nejsou. Generování genů (tahů) může být ukončeno dvěma způsoby:

- Počet výskytů nepohyblivých krabic překročil stanovenou hranici – jedinec je označen jako stagnující.
- V nastavené části chromozomu došlo k posuvu krabicí – jedinec není označen jako stagnující.

- počet generovaných genů překročil hodnotu stanovenou dle nastavitelného parametru pro nárůst maximální délky chromozomu – jedinec je označen jako stagnující pouze pokud počet výskytů nepohyblivých beden je větší nebo roven koeficientu k.

Hranice definující stagnujícího jedince v závislosti na počtu výskytů nepohyblivých krabic je dána rovnicí:

$$S = B_l * k, \tag{4.6}$$

kde B_l je počet krabic neumístěných na cílových místech a k je koeficient, který byl na základě dále popsaných experimentů zvolen jako hodnota 2 a ovlivňuje i třetí ukončovací podmínku.

V tabulce 4.1 vidíme úspěšnost detekce dynamických globálních uváznutí v konečném stadiu pro různé hodnoty koeficientu. Výsledky jsou získány z experimentů nad instancemi na obrázku 2 a 3. Vyhodnoceny byly první dvě generace s velikostí populace 24. Při hodnotě 1 je přesnost algoritmu v rámci negativní detekce poměrně vysoká, ale téměř polovina jedinců je mylně označena jako stagnující. Ztrácíme tak jedince, kteří mohou být pro populaci přínosem. Pro hodnotu 2 příliš neubývá přesnost při negativní detekci a zároveň vidíme vyšší přesnost při detekci pozitivní. Koeficient o hodnotě tři má vysoké kritérium pro označení jedince jako stagnujícího. Pokud jedinec je označen, jeho řešení s velkou pravděpodobností opravdu uváznutí obsahuje, ale v populaci zůstává mnoho jedinců, v jejichž řešení uváznutí nebylo rozpoznáno.

Chromozom jedince není algoritmem pro detekci globálních uváznutí změněn. Nastavovaná část chromozomu je pouze dočasná a slouží pouze pro účely tohoto algoritmu.

k	P_T	P_F	P
1	53%	90%	72%
2	90%	83%	87%
3	93%	27%	60%

Tabulka 4.1: Přesnost algoritmu rozpoznávajícího dynamické globální uváznutí pro různé hodnoty koeficientu k. Sloupec P_T uvádí přesnost v rámci případů kdy algoritmus jedince označil jako stagnujícího. V sloupci P_F naopak vidíme spolehlivost algoritmu, kdy jedince jako stagnujícího neoznačí. Ve sloupci P je celková úspěšnost algoritmu.

4.4 Selekce a křížení jedinců

Prvním typem selekce, kterým je možné rodiče vybírat, je ruleta. U tohoto typu selekce nastává problém s jedinci, kteří obdrželi fitness ohodnocení 0. Jedinci s takovýmto ohodnocením nemůžou být selekcí ruleta vybráni (viz rovnice 3.1). V případě nedostatku jedinců s větším fitness ohodnocením než 0 proto není vybrán požadovaný počet rodičů, který je pro evoluci závazný (v každé generaci je třeba vybrat stanovený počet rodičů). Tento problém je řešen náhodným výběrem zbylého počtu rodičů.

Druhým volitelným principem selekce je selekce typu turnaj. Z každé skupiny je vybrán takový počet vítězů, aby při dané velikosti turnaje byl celkový počet rodičů roven hodnotě zvolené v konfiguračním souboru. Oba typy selekce jsou srovnány na základě experimentu v podkapitole 6.1.

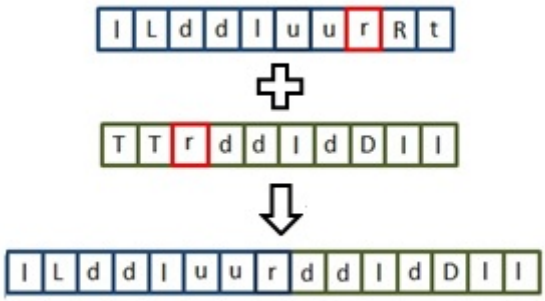
Operace křížení bývá často problematická. Chromozomy, jejichž geny tvoří logické bloky, mohou být křížením rozbity – křížením vzniká sice nový jedinec, ten však nedědí vlastnosti

svých rodičů [1]. Tento problém se týká i reprezentace řešení hry Sokoban, kde je chromozom tvořen s ohledem na stav hrací plochy. U obvyklých způsobů křížení je bod křížení volen náhodným způsobem. Pokud bychom takto vzniklé části různých jedinců spojily, stav hrací plochy se pravděpodobně v bodě křížení bude lišit. Vzniká tak nové řešení, které zanedbává vlastnosti minimálně jednoho z rodičů.

Nabízí se řešení, kde je omezena náhodná volba bodu křížení pouze na geny rodičů, které reagují na stejný stav hrací plochy. Takový bod však nemusí vůbec existovat, neuvažujeme-li počáteční stav hrací plochy. Kompromisem je hledat takový stav hrací plochy, který se shoduje pouze v pozici agenta. Odlišné rozestavení krabic může agentův pohyb pozměnit, nicméně až v době, kdy se takováto krabice objeví v agentově čtyřokolí. Získáváme tak alespoň část vlastností z obou rodičů.

Při samotné operaci křížení je v každém z rodičů náhodným způsobem určen bod v chromozomu. Bod u prvního z rodičů označuje jeho bod pro křížení. U druhého z rodičů se posouváme od bodu oběma směry a hledáme nejbližší pozici genu, která obsahuje stejnou pozici agenta jako u prvního rodiče. Spojením první části z prvního rodiče a druhé části z rodiče druhého vznikne nový jedinec, který může mít odlišnou délku než jeho rodiče (obrázek 4.6). Je podstatné, aby spojené části navazovaly právě v bodech křížení, proto je nezbytné kombinovat části v tomto pořadí. Tímto způsobem jsou vytvořeni dva potomci pro každý pár. V případě, kdy by délka nového jedince převyšovala maximální povolenou délku, jsou odstraňovány koncové geny jedince.

Nově vzniklými jedinci jsou nahrazeni nejslabší jedinci v populaci. Při vytváření nové populace na základě pravděpodobnostních algoritmů (selekce jedinců do nové populace na principu rulety nebo pořadí) evoluce konverguje velice pomalu, u rozsáhlých a složitějších instancí řešení není vůbec nalezeno. U takovýchto hracích ploch je totiž konvergence sama o sobě pomalá a ztráta kvalitního jedince má velký vliv na vývoj populace. Tato skutečnost je pro evoluci důležitější, než i pozitivní přínos pravděpodobnostních selekcí, kterým je možnost vyváznutí z lokálního optima.



Obrázek 4.6: Křížení jedinců pro řešení hry Sokoban. Červeně zvýrazněné chromozomy označují bod křížení. Podmínkou je, aby u obou těchto bodů se agent nacházel na stejné pozici hrací plochy.

4.5 Mutace

Mutace je zvolena n-bodová. Parametr n je volitelným parametrem programu. Na základě experimentů v kapitole 6 můžeme stanovit jako nejvhodnější mutaci dvoubodovou.

Mutace je aplikována na všechny jedince populace. Populaci však budeme dělit do několika skupin. Samotný průběh mutace je upraven pro každou ze skupin tak, aby byla v populaci udržována různorodost, a zároveň aby nebyli ztraceni nejkvalitnější jedinci v populaci.

4.5.1 Mutace stagnujících jedinců

Ačkoli jsou stagnující jedinci hodnoceni 0 fitness body, s nízkou kvalitou ostatních jedinců v populaci se zvyšuje pravděpodobnost, že i stagnující jedinec bude vybrán do populace následující. Dynamické uváznutí, které řešení jedince může obsahovat, je schopna vyřešit vhodně zvolená mutace.

Uváznutí je vždy způsobeno nevhodným posuvem krabicí. Je proto zřejmé, že koncová část chromozomu neobsahující žádné posuvy krabicí není zdrojem uváznutí. Mutace genu v takovéto části žádným způsobem nepřispívá k řešení problému. Náhodný výběr genů k mutaci je proto omezen pouze do pozice genu, který obsahuje poslední posuv krabicí v chromozomu.

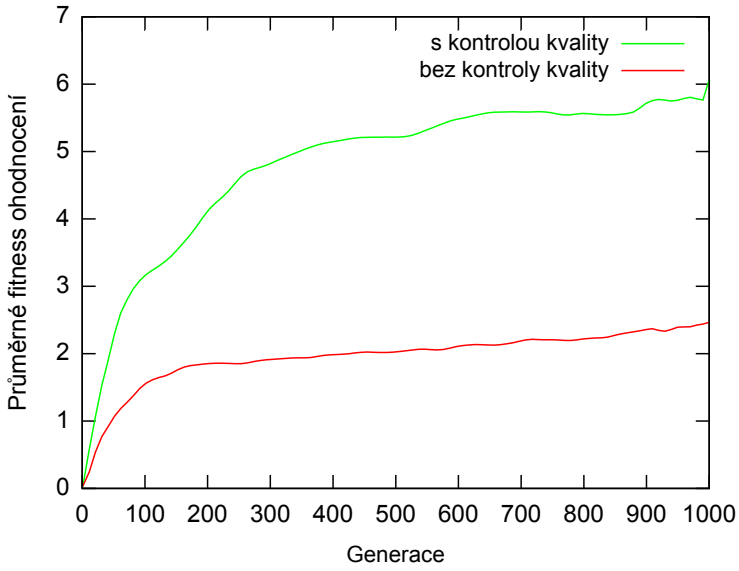
4.5.2 Mutace přežívajících jedinců

Přežívajícími jedinci nazýváme takové jedince, kteří se v populaci nacházeli již před operací křížení. Vzhledem k principům vytváření nové populace se jedná o jedince kvalitní. Mutace je náhodný proces, který by tyto jedince mohl znehodnotit. Proto je vyhodnocen přínos zmutovaného jedince. Pokud dosahuje vyšších kvalit než jedinec původní, zachován je zmutovaný jedinec. V opačném případě jedinec zůstává v populaci ve svém původním tvaru. Tímto principem můžeme přijít o informace, které by vedly během vývoje k dalšímu zkvalitnění jedince. Riziko ztráty stávajících informací je však podstatnější, jak ukazuje srovnání průměrného vývoje evoluce pro mutaci s kontrolováním kvality a mutaci bez této kontroly na obrázku 4.7.

Ohodnocení, na základě kterého se jedinci v rámci mutace porovnávají, není prováděno fitness ohodnocující funkcí, protože ta pracuje nad celou populací. Takto rozsáhlé určování kvality není v tomto případě nutné, postačuje srovnání pouze dvou jedinců. Výběr genů k mutaci probíhá na celé délce chromozomu.

4.5.3 Mutace potomků

Náhodný výběr genů k mutaci u jedinců, kteří vznikli v dané generaci při operaci křížení, probíhá na celé délce chromozomu. Na rozdíl od jedinců přežívajících je tato operace nevratná. Do populace jsou tak přinášeny nové informace a zároveň je udržována její různorodost.



Obrázek 4.7: Srovnání aproximací průměrného fitness ohodnocení jedinců během generací pro dva přístupy k mutaci: s kontrolou výsledné kvality přežívajícího jedince a mutace bez této kontroly. Experiment proběhl nad první úrovní původní sady instancí (obrázek 3). Vstupní data jsou průměrem ze čtyřiceti běhů programu pro každý typ mutace.

4.6 Maximální délka chromozomu

Protože se délka řešení nových i stávajících jedinců v populaci mění vlivem křížení a mutace, je třeba nastavovat maximální počet genů v chromozomu.

První možností, která se nabízí, je stanovení tohoto maxima na začátku programu a v jeho průběhu jej již neměnit. Chromozom, který má však více genů a tím k dispozici více tahů, dostává možnost stát se kvalitnějším než jedinec kratší, který prohledává pouze malou část stavového prostoru. Protože fitness ohodnocující funkce bere v úvahu délku chromozomu pouze v případě, kdy více jedinců dosahuje stejného počtu krabic umístěných na cílových nebo na blízkých místech, elita populace je rychle zaplněna jedinci, jejichž délka se blíží k maximálně stanovené hodnotě. Část efektivity křížení je dána předpokladem, že nový jedinec může dědit velkou část vlastností rodičů. Výsledná délka chromozomu takového jedince je následně větší než délka chromozomu rodičů. Chromozom, který vznikne křížením jedinců dosahujících maximální délky, je tak předem omezen na zdědění pouhé části vlastností rodičů a vývoj se zpomaluje.

Řešením popsaného problému je navyšovat maximální délku chromozomu během evoluce. Chromozomy počáteční populace jsou generovány s délkou danou volitelným parametrem programu. Tato délka je dále násobena počtem krabic v cílových místech, dosažených u nejlepšího jedince v rámci celé historie, tedy:

$$Lenght_{max} = P * G_{max}, \tag{4.7}$$

kde P je volitelný parametr programu a G_{max} je maximální počet krabic umístěných na cílových místech v jednom řešení během všech populací. Tímto je zajištěno, že pokud se v populaci objeví jedinec s novou informací o umístění krabic na cílová místa, uvolní se prostor pro nové jedince, které tuto informaci mohou dědit a dále rozvíjet.

4.7 Optimalizace

Řešení mohou obsahovat zbytečné bloky chromozomů. Tyto bloky jsou tvořeny geny, které ve výsledku nezmění stav hrací plochy – rozmístění krabic i agenta je před blokem stejné jako na jeho konci. Při odstranění těchto bloků je výsledný stav hrací plochy zachován.

Tímto způsobem je optimalizováno každé nalezené řešení. Jedinci, kteří úspěšné řešení neobsahují, zůstávají v původním stavu. Z bloku v chromozomu bezvýznamného může po operaci křížení a mutace vzniknout významná část chromozomu.

Kapitola 5

Implementace

Návrh je realizován v jazyku C/C++, který disponuje jak komfortním objektovým přístupem, tak i vysokou rychlostí. Kvůli minimalizaci časových nároků na běh programu je využívána paměť i pro ukládání hodnot, které jsou z daného stavu v programu zjistitelné. Tento princip je využíván zejména u reprezentace jedince, který si pamatuje celou historii stavů hrací plochy a další potřebné informace zjištěné z jeho fenotypu.

5.1 Reprezentace hry

Reprezentace hry je tvořena třídou *Board*, která obsahuje metody pro inicializaci hrací plochy, informace o hrací ploše a operace nad ní.

Metoda *loadLevel* načítá instanci hry ze souboru, který definuje plochu v obecně rozšířeném textovém formátu:

# – zeď	* – krabice na cílovém místě
@ – agent	. – cílové místo
+ – agent na cílovém místě	mezera – podlaha
$ – krabice	

V metodě *loadLevel* je plocha pomocí slovníku převedena do formátu programu a uložena v dvourozměrném vektoru, do kterého se přistupuje souřadnicemi políčka. Za účelem uchování souřadnic společně s typem políčka slouží struktura *index*. Struktura přetěžuje operátory pro rovnost a nerovnost pro komfortnější porovnávání.

Dvourozměrný vektor, který označuje políčka způsobující uváznutí je nastavován metodou *markDeadlocks*. Algoritmus nejdříve vyhledává dynamická uváznutí, způsobená z jedné strany uzavřenými tunely o šířce jedné buňky (viz kapitola 4.1). Následně jsou označena políčka způsobující uváznutí statická.

Pro orientaci na ploše slouží následující funkce:

- *get4Neighborhood* – vrátí vektor *indexů* políček ve čtyřokolí políčka do zvolené vzdálenosti
- *get8Neighborhood* – vrátí vektor *indexů* políček v osmiokolí políčka do zvolené vzdálenosti

- *getSide* – vrátí *index* políčka ve zvoleném směru

Pohyb po ploše realizuje metoda *move*, která změní pozici agenta a v případě potřeby posunu krabicí volá metodu *pushBox*. Během každého tahu je v objektu třídy *Board* aktualizován stav hrací plochy, uložený jako struktura *state*. Tato struktura obsahuje informace o rozmístění krabic a agenta, jeho předchozím tahu a také přetížené operátory pro porovnávání. Pro výkonnost programu v rámci času potřebného pro výpočet je zde také ukládán počet krabic na cílových místech a na místech blízkých cílovým. Tyto informace jsou pak k dispozici bez toho, abychom opakovaně museli procházet všechna políčka na hrací ploše.

Spouštění rekurzivního algoritmu pro kontrolu dynamického lokálního uváznutí najdeme v metodě *dynamicDeadlock*.

Nezbytný je také algoritmus pro zjištění počtu krabic na místech blízkých cílovým. Metoda *getBoxesClose* vyhledá všechna cílová místa, od kterých ve vodorovném i svislém směru označuje políčka blízká cílovým. Zároveň počítá všechny krabice na nich ležící, pokud již políčko nebylo označeno dříve.

5.2 Reprezentace jedince

Každý jedinec v populaci je tvořen objektem třídy *Creature*, která zapouzdřuje následující informace:

- Chromozom – pohyb po ploše zakódovaný jako textový řetězec.
- Vektor struktur *state*, obsahující historii stavů hrací plochy.
- Výsledný počet krabic na cílových místech.
- Výsledný počet krabic na blízkých místech.
- Fitness hodnotu jedince – výchozí hodnota -1 označuje jedince, jehož fenotyp nebyl zjišťován.
- Informaci o tom, zda je jedinec označen jako stagnující.

Metody *lengthGoal* a *lengthClose* slouží pro získání délky chromozomu, který je potřebný k obdržení výsledného počtu krabic na cílových místech, respektive počtu krabic na místech blízkých cílovým. Metoda *lengthPush* slouží pro získání délky chromozomu do pozice genu, který obsahuje poslední posuv krabicí. Tyto metody operují nad zmíněným vektorem, obsahujícím historii hrací plochy během řešení.

5.3 Genetický algoritmus

Běh programu je řízen třídou *GA_Sokoban*, která obsahuje principy genetických algoritmů v následujících metodách:

- *generate* generuje počáteční populaci.
- *evaluate* ohodnocuje jedince v populaci. Součástí této funkce je zjišťování fenotypu jedince voláním funkce *getFenotype*. Pro urychlení běhu programu jsou důležité informace zjištěné na základě fenotypu jedince uloženy v jeho objektu. Není třeba tedy fenotyp opakovaně zjišťovat v každé iteraci algoritmu.

- *selection* vybírá jedince pro křížení zvoleným způsobem.
- *crossover* vytváří jedince pomocí křížení rodičů. Je zde využívána historie stavů hrací plochy, ze které můžeme zjistit pozici agenta pro daný gen.
- *replacement* nahrazuje část populace novými jedinci.
- *mutation* provádí operaci mutace nad jedinci.
- *optimize* – optimalizuje výsledná řešení

Populace je tvořena vektorem objektů *Creature*. Třída obsahuje dva objekty typu *Board*. První z nich zůstává po jeho inicializaci neměnný, slouží pouze k uložení informací o počátečním stavu hry. Nad druhým objektem jsou prováděny tahy jednotlivých řešení. Až daný jedinec skončí s operacemi nad hrací plochou, do objektu je uložena hrací plocha s počátečními informacemi (zjištěné informace o počtu beden na cílových a jim blízkých místech) a stavem hry.

Metoda *run* volá metodu *generate* a spouští iterace algoritmu až do maximálního počtu generací. Pokud je tak v konfiguračním souboru definováno, po skončení genetického algoritmu jsou tisknuty grafy pomocí programu Gnuplot.

Metoda *next* obsahuje jednu iteraci genetického algoritmu a zároveň zapisuje informace o populaci do výstupního souboru:

```
void GA_Sokoban::next(){
    best_population_solution = -1;
    GA_Sokoban::evaluate();
    gn_writer.averageFitness(population_number,averageFitness());
    GA_Sokoban::selection();
    GA_Sokoban::crossOver();
    GA_Sokoban::replacement();
    GA_Sokoban::mutation();
    if (best_population_solution > -1)
       gn_writer.solutionLength(population_number,best_solution);
}
```

V metodě *stagnate* probíhá detekce stagnujících jedinců. Hrací plocha, nad kterou zkoumaný jedinec operoval, je v době volání této funkce v podobě, jakou ji jedinec zanechal. Generuje se tedy potřebný počet tahů, které se uskutečňují nad touto hrací plochou. Tyto tahy mají za účel pouze zjistit zda stav hrací plochy docílený jedincem patří do skupiny potenciálně neměnného (s krabicemi nelze pohybovat). Chromozom jedince zůstává nezměněn.

Kapitola 6

Experimenty

V této kapitole budeme sledovat vývoj evoluce při změně některých parametrů programu. Zaměříme se především na velikost populace, typ selekce a počet bodů mutace. V tabulce 6.1 vidíme základní nastavení parametrů. U jednotlivých experimentů budeme uvádět pouze změny v tomto nastavení.

Experimenty jsou prováděny na následujících instancích hry:

- A – malá instance hry bez zdí uvnitř plochy, riziko globálního dynamického uváznutí je malé (obrázek 1),
- B – rozsáhlá instance hry, řešení je dlouhé, ale bez velkého rizika globálního dynamického uváznutí (obrázek 2),
- C – první úroveň původního setu instancí hry, nejlepší dosažené řešení obsahuje 230 tahů, zejména v horní části hrací plochy je riziko globálního uváznutí vysoké (obrázek 3).

Každý experiment sestává ze čtyřiceti běhů programu nad každou instancí a uváděné statistiky jsou získány průměrováním těchto evolucí.

Paremetr	**Hodnota**
Počet iterací	1000
Velikost populace	24
Počet rodičů	12
Typ selekce	Ruleta
Typ nahrazování	Nejhorší jedinci
Bodů mutace	1
Nárůst chromozomu	300
Minimální délka chromozomu	1

Tabulka 6.1: Výchozí nastavení experimentů.

6.1 Velikost populace

Sledujeme vývoj evoluce pro různé velikosti populací. Poměr počtu rodičů ku velikosti populace zůstává stejný jako u základního nastavení. Pro velikost populace 12 tedy uvažujeme 6 rodičů, pro velikost populace 24 vybíráme 12 rodičů a podobně pro velikost populace 48 je stanoven počet rodičů na 24.

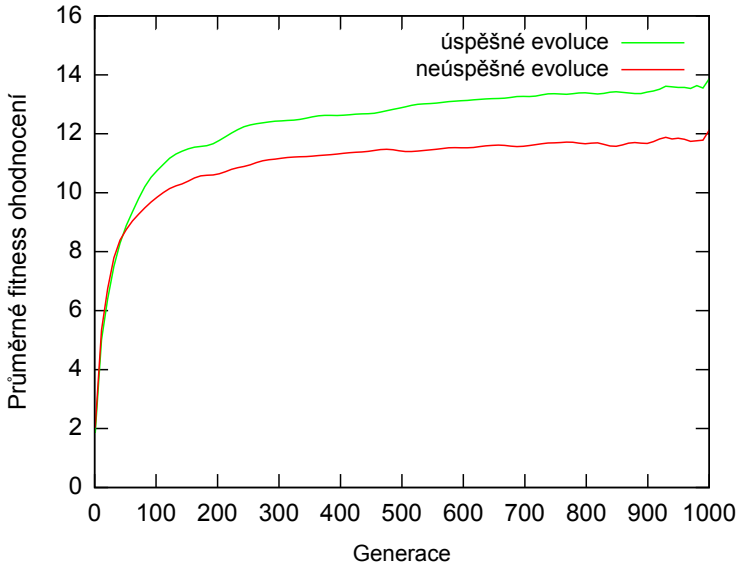
Velikost populace	**A**			**B**			**C**		
	U	**N**	**D**	**U**	**N**	**D**	**U**	**N**	**D**
12	100%	23	68	50%	525	377	55%	558	346
24	100%	18	65	78%	371	377	75%	282	332
48	100%	12	54	95%	215	375	87,5%	192	302

Tabulka 6.2: Statistiky evoluce pro různé velikosti populací. Sloupec *U* definuje procentuální úspěšnost algoritmu (algoritmus našel řešení). Sloupec *N* ukazuje průměrný počet generací potřebný k nalezení řešení (zahrnuta jsou pouze úspěšná řešení). V sloupci *D* vidíme průměr nejkratších dosažených délek řešení.

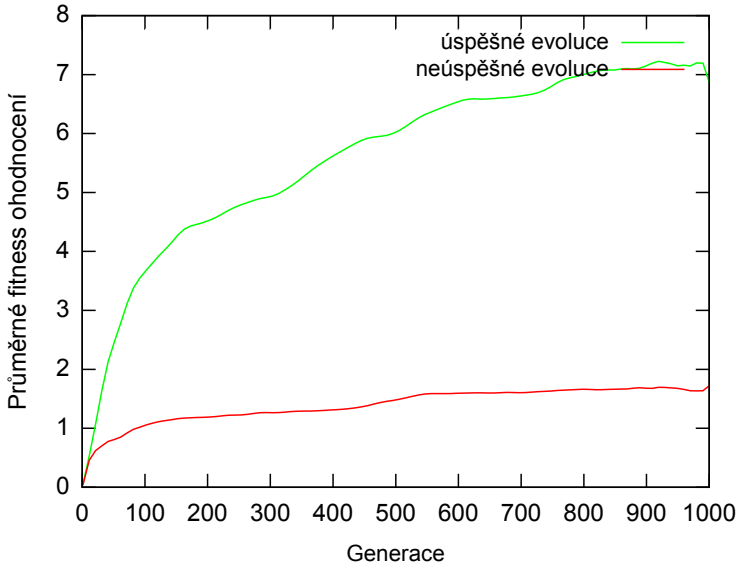
Z tabulky 6.2 vyplývá, že velikost populace významně ovlivňuje počet generací potřebných pro nalezení řešení. Výsledná délka řešení je ovlivněna v menším měřítku. Instance A je dostatečně jednoduchá, aby řešení bylo nalezeno spolehlivě ve všech zkoumaných případech. Pro instance B a C můžeme velikost populace 12 označit jako nevhodnou, protože v polovině případů není řešení vůbec nalezeno.

Prozkoumáme blíže neúspěchy v hledání řešení u instancí B a C při velikosti populace 12. V grafech na obrázku 6.1 vidíme srovnání vývoje průměrného fitness ohodnocení populací úspěšných a neúspěšných běhů programu u obou instancí. U instance B konvergují oba průběhy funkce. Můžeme proto předpokládat, že navýšení počtu generací dovede k úspěšnému řešení i běhy programu, které byly v experimentu neúspěšné. U instance C vidíme průběh jiný. Průměrně od 100. generace přestává funkce růst, vývoj uvázl v lokálním optimu. U této instance tedy nelze řešit nízkou úspěšnost evoluce navýšením počtu generací.

Vidíme rozdílné dopady velikosti populace na instance B a C. Příčinou je podstata jednotlivých instancí. Instance B neobsahuje velké riziko globálního uváznutí a menší velikost populace vede pouze k prodlužování vývoje. Instance C obsahuje vysoké riziko globálního uváznutí a pro nalezení úspěšného řešení potřebuje evoluce nad touto instancí větší rozmanitost populace.



(a) instance B



(b) instance C

Obrázek 6.1: Srovnání aproximací průměrného fitness ohodnocení jedinců během generací nad instancemi B a C u úspěšných a neúspěšných běhů programu.

6.2 Typ selekce

Tento experiment obsahuje srovnání selekce typu turnaj a ruleta. Velikost turnaje je stanovena na 6 jedinců v jedné skupině. Při velikosti populace 24 tedy dostáváme celkem 4 skupiny a z každé jsou vybráni 3 nejlepší jedinci jako rodiče. Pro selekci typu ruleta jsou použity statistiky z předchozího experimentu.

Typ selekce	**A**			**B**			**C**		
	U	**N**	**D**	**U**	**N**	**D**	**U**	**N**	**D**
turnaj	100%	11	66	80%	283	390	70%	298	327
ruleta	100%	18	65	78%	371	377	75%	282	332

Tabulka 6.3: Statistiky evoluce pro typy selekce turnaj a ruleta.

V tabulce 6.3 vidíme, že u instancí A a B se selekce typu turnaj jeví jako efektivnější. Naopak u instance C vidíme mírný pokles úspěšnosti. Při daném nastavení selekce typu turnaj je jisté, že minimálně 3 jedinci s nejvyšším fitness ohodnocením budou vybráni jako rodiče. Můžeme proto mluvit o jisté míře elitismu (v rámci výběru jedinců pro křížení). Právě tento elitismus zabraňuje evoluci dostat se mimo lokální optimum při globálním uváznutí. U instancí A a B, kde riziko globálního uváznutí je nízké, vysoký selekční tlak u výběru rodičů urychluje konvergenci k úspěšnému řešení bez uváznutí v lokálním optimu a má tak pozitivní vliv na evoluci.

6.3 Počet bodů mutace

Předmětem tohoto experimentu je sledování vlivu počtu bodů mutace na vývoj evoluce u jednotlivých instancí. V tabulce 6.4 vidíme statistiky pro jednobodovou, dvoubodovou a tříbodovou mutaci.

Počet bodů mutace	**A**			**B**			**C**		
	U	**N**	**D**	**U**	**N**	**D**	**U**	**N**	**D**
1	100%	18	65	78%	371	377	75%	282	332
2	100%	17	46	78%	494	306	78%	368	264
3	100%	18	56	60%	490	355	80%	398	309

Tabulka 6.4: Statistiky evoluce pro typy selekce turnaj a ruleta.

Z tabulky 6.4 můžeme vyvodit, že pro instance s vyšším rizikem globálního uváznutí je vhodnější vícebodová mutace. Příčinnou je nedokonalá detekce globálního uváznutí. Selekcí pak mohou být vybráni rodiče, kteří takové uváznutí obsahují. Křížením těchto rodičů může vzniknout jedinec dědící tu část chromozomu, která uváznutí způsobuje. Při vícebodové mutaci je větší pravděpodobnost, že tato část chromozomu bude změněna. Je tedy potlačen vliv operace křížení, čímž se zároveň prodlužuje doba, potřebná k nalezení úspěšného řešení.

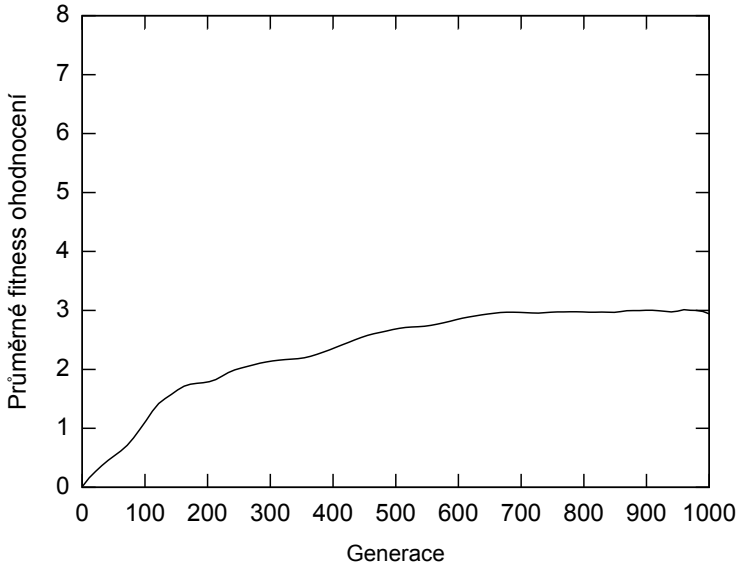
6.4 Problematické typy instancí

V této podkapitole se zaměříme na instance hry, které aplikace není schopna vyřešit, nebo je řeší s velice malou úspěšností. Tyto instance obsahují problematiku alespoň z jedné z následujících dvou skupin:

- vysoké riziko globálního uváznutí, které vytváří množství podproblémů při řešení
- během úspěšného řešení je třeba odsouvat krabice z míst cílových

První skupina obsahuje problémy, které definují míru obtížnosti i pro člověka. Do této skupiny řadíme druhou úroveň původní sady instancí (obrázek 4), kterou aplikace není schopna vyřešit. Aproximaci průměrného fitness ohodnocení jedinců v populaci během evoluce nad touto instancí vidíme v grafu na obrázku 6.2. Přibližně od sedmisté generace vývoj stagnuje. Během evoluce se podařilo umístit na cílová místa vždy maximálně 4 krabice z 10. Umístit krabice další nebylo možné kvůli globálnímu uváznutí, které však často vzniklo již při umístění první krabice. Rozpoznat uváznutí v tak raném stádiu však pro aplikaci není možné. Způsob fitness ohodnocování společně s principy genetických algoritmů nedovolují vrátit se zpět k tomuto problému, protože rozdíl v průměrném ohodnocení v době, kdy jsou umístěny 4 krabice a v době, kdy došlo k uváznutí, je příliš vysoký.

Skupina druhá může zahrnovat instance relativně jednoduché. Na obrázku 5 vidíme příklad instance z této skupiny, kterou není aplikace schopna vyřešit. Podstata příčiny je stejná jako u první skupiny. Samotné odsouvání krabic z míst cílových vede ke snižování fitness ohodnocení. Takové jedince evoluce zavrhuje. Jedinec by musel v rámci jedné generace počet odsunutých krabic nahradit počtem správně umístěných krabic na cílových místech. Pravděpodobnost, že se toto stane, klesá s narůstajícím počtem odsunutých krabic. K umístění krabic na cílová místa je třeba postupného vývoje, který jedinec v tomto případě nemá k dispozici.



Obrázek 6.2: Aproximace průměrného fitness ohodnocení jedinců v populaci během evoluce nad druhou úrovní z původní sady instancí. Vstupní data jsou průměrem výsledků ze čtyřiceti běhů programu.

6.5 Shrnutí experimentů

Experimenty jsme zjistili, že vliv zkoumaných parametrů programu je odlišný v závislosti na typu instance. Hlavní příčinou je rozdíl v riziku globálního uváznutí u těchto instancí.

K velikosti populace lze obecně říci, ze s rostoucí velikostí stoupá také úspěšnost aplikace, jak lze očekávat. Zároveň však stoupají časové nároky na běh programu. Kompromisem pro všechny tři použité typy instancí je velikost populace 24. U instance A však postačuje i poloviční velikost. U instancí, které neobsahují vysoké riziko globálního uváznutí lze malou velikost populace nahradit zvýšením počtu iterací.

Nejvýznamnější rozdíly v dopadu na evoluci u jednotlivých instancí vidíme u experimentu s typy selekcí. Selekce typu turnaj není vhodná pro instance obsahující vysoké riziko globálního uváznutí, ale urychluje konvergenci u jednodušších instancí. Selekce typu ruleta konverguje pomaleji, ale je schopná vyváznout z lokálního optima. Selekcí typu turnaj tedy řešení bude nalezeno s větší pravděpodobností, ale na úkor časových nároků aplikace. Selekce typu ruleta je méně úspěšná v hledání řešení, ale pokud je nějaké nalezeno, časové nároky aplikace budou pravděpodobně menší.

Experiment s počtem bodů mutace ukazuje, že nejvhodnější je mutace dvoubodová. U instancí s vysokým rizikem globálního uváznutí může mírně zvýšit úspěšnost i mutace vícebodová, ale vzhledem k tomu, že mutace v této práci vnáší poměrně výrazné změny do fenotypu jedince, důsledky více než dvoubodové mutace jsou zanedbatelné.

Kapitola 7

Závěr

Cílem této práce bylo vytvořit automatické řešení hry Sokoban s pomocí genetických algoritmů. Rozsáhlému stavovému prostory hry bylo třeba přizpůsobit reprezentaci chromozomu. Agent na ploše reaguje na aktuální stav hrací plochy a generuje tak průchod plochou, kde každý gen definuje jeden proveditelný tah. Pravidla, která agent musí dodržovat při výběru směru tahu, blíže definují stavový prostor, ve kterém úspěšné řešení leží. Problematiku rozsáhlého stavového prostoru se tak podařilo vyřešit do míry, kdy je aplikace schopna řešit i instance s rozsáhlou hrací plochou s řešením čítajícím stovky tahů. Příkladem je první úroveň originální sady instancí hry Sokoban.

Zásadní skutečností, která významně ovlivňuje úspěšnost aplikace, zůstává přítomnost dynamického globálního uváznutí. Použitá detekce jedinců, kteří potenciálně takovéto uváznutí obsahují, je schopná rozeznat uváznutí až v jeho konečném stádiu a je proto často nedostatečná. Tento problém je i příčinou, proč aplikace není schopna řešit jinou než první úroveň původní sady instancí hry. Řešením může být využití i jiných algoritmů prohledávajících stavový prostor s heuristickými prvky, například A* nebo IDA*, které jsou schopné rozpoznat uváznutí v ranějším stádiu .

Navržený způsob ohodnocování jedinců sleduje základní cíle hry, v některých případech však může být zavádějící. Instance hry, ve kterých je během řešení třeba odstraňovat krabice z míst cílových, jsou pro evoluci obtížně řešitelné. Reálná kvalita jedince s nižším počtem krabic na místech cílových v těchto případech může být vyšší než kvalita jedince s větším počtem takto umístěných krabic. Tuto skutečnost však fitness funkce nerozezná. Tento problém by bylo možné řešit sledováním vlivu určitých stavů hrací plochy na vývoj jedinců, jejichž řešení tyto stavy obsahují.

Během experimentů bylo zjištěno, že vliv velikosti populace, typu selekce rodičů a počtu bodů mutace se liší na základě typu instance. Důležitá je především informace, jak vysoké riziko globálního uváznutí instance obsahuje. Pokud bychom dokázali tento typ uváznutí včas rozpoznat, bylo by možné určit prospěšnost zkoumaných parametrů v obecném měřítku.

Práce ukázala, že ve srovnání s principy algoritmu IDA*, které byly použity pro automatické řešení ve studii [10], je řešení složitějších instancí hry Sokoban na základě genetických algoritmů méně úspěšné. Pro jednodušší instance (i s delším řešením) jsou však genetické algoritmy spolehlivé. Zajímavé by bylo srovnat časové nároky programu a délku nalezeného řešení právě pro takové instance. Vzhledem k odhaleným problémům i schopnostem, které řešení hry Sokoban pomocí genetických algoritmů obsahuje, můžeme předpokládat, že přínosem by bylo spojení tohoto principu s jinými metodami. Například zmíněným algoritmem IDA*, jehož schopnosti v rámci řešení hry Sokoban byly v minulosti ověřeny.

Literatura

[1] Ashlock, D.: *Evolutionary Computation for Modeling and Optimization*. Springer, 2004.

[2] Ashlock, D.; Joenks, M.: ISAc lists, a different representation for program induction. In *Genetic Programming 98, proceedings of the third annual genetic programming conference*, San Francisco: Kaufmann, 1998, s. 3 – 9.

[3] Ashlock, D.; Schonfeld, J.: Evolution for automatic assessment of the difficulty of Sokoban boards. In *IEEE Congress on Evolutionary Computation*, 2010, s. 1–8.

[4] Ashlock, D.; Willson, S.; Leahy, N.: Coevolution and tartarus. In *Proceedings of the 2004 Congress on Evolutionary Computation*, ročník 2, 2004, s. 1618 – 1624.

[5] Coldridge, J.; Amos, M.: Genetic algorithms and the art of zen. In *IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA)*, 2010, s. 1417 – 1423.

[6] Culberson, J.: Sokoban is PSPACE-complete. In *International Conference on Fun with Algorithms*, 1998, s. 65 – 76.

[7] Darwin, C.: *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. London: Murray, 1859.

[8] Dor, D.; Zwick, U.: SOKOBAN and other motion planning problems. *Computational Geometry: Theory and Applications*, ročník 13, č. 4, 1999: s. 215 – 228.

[9] Junghanns, A.; Schaeffer, J.: Sokoban – Evaluating Standard Single-Agent Search Techniques in the Presence of Deadlock. In *Advances in Artificial Intelligence*, Springer Verlag, 1998, s. 1–15.

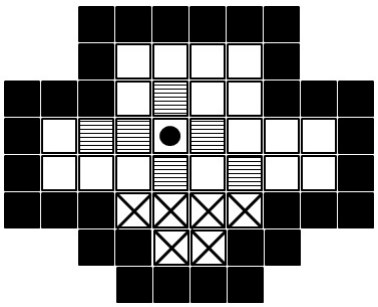
[10] Junghanns, A.; Schaeffer, J.: Sokoban – Enhancing General Single-Agent Search Methods Using Domain Knowledge. *Artificial Intelligence*, ročník 129, 2001: s. 219–251.

[11] Murase, Y.; Matsubara, H.: Automatic making of sokoban problems. In *Pacific Rim International Conference on Artifical Intelligence*, 1996, s. 592 – 600.

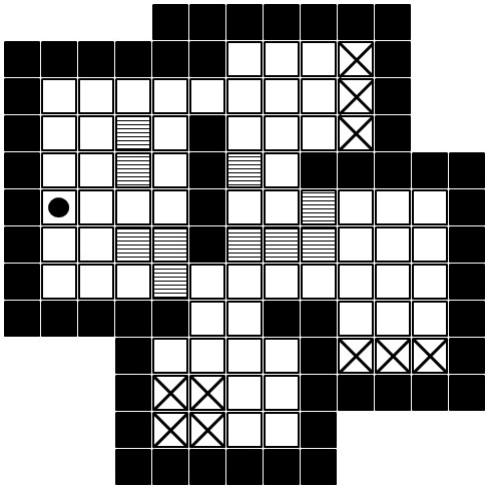
[12] Reznick, D.; Ricklefs, R.: Darwin's bridge between microevolution and macroevolution. *NATURE*, ročník 457, 2009: s. 837–842.

[13] *University of Alberta* [online]. 2004 [cit. 2012-5-10], Our Program – Rolling Stone, Dostupné z WWW: ⟨`http://webdocs.cs.ualberta.ca/~games/Sokoban/program.html`⟩.

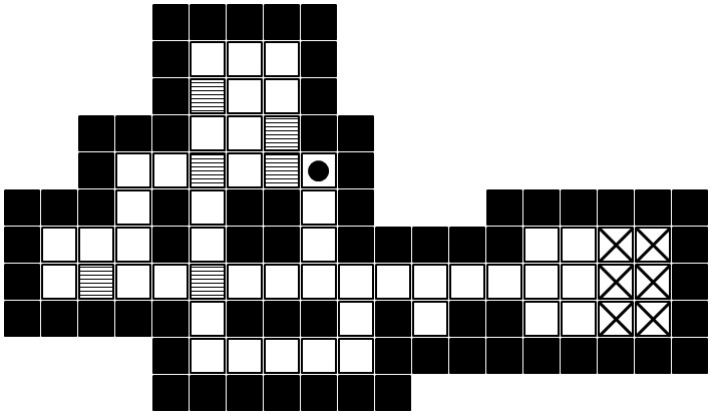
Použité instance



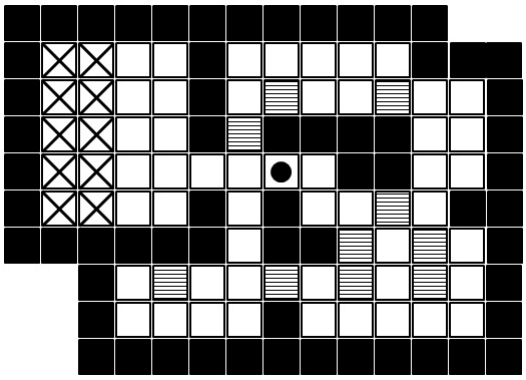
Obrázek 1: Malá instance hry s krátkým řešním.



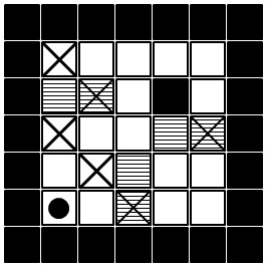
Obrázek 2: Nepříliš obtížná instance hry, ale hrací plocha je rozsáhlá a výsledné řešení dlouhé.



Obrázek 3: První úroveň originální hry Sokoban.



Obrázek 4: Druhá úroveň původní sady instancí.



Obrázek 5: Instance, během jejíhož řešení je třeba odsouvat krabice z cílových míst.

Konfigrační soubor

Parametr	**Popis**	**Povinný**	**Hodnoty**
mating pool size	počet jedinců vybraných pro křížení	ANO	celočíselné>1
tournament size	velikost skupiny v turnaji při selekci typu turnaj	ANO	celočíselné>0
goal length	počáteční délka chromozomu a nárůst chromozomu na jednu krabici na cílovém místě v rámci populace	ANO	celočíselné>0
population size	velikost populace	ANO	celočíselné>0
mutation points	počet bodů pro mutaci	NE	celočíselné>0, výchozí 1
selection type	typ selekce	NE	ROULETTE, TOURNAMENT–výchozí
replacement type	typ nahrazování jedinců	NE	ROULETTE, WORST–výchozí
max populations	maximální počet populací	ANO	celočíselné>0
min length	minimální délka chromozomu	NE	celočíselné>0, výchozí 1
gnuplot	zda mají být z výstupních hodnot tisknuty grafy pomocí programu Gnuplot	NE	YES, NO–výchozí
output file	název výstupních souborů	ANO	textový řetězec

Tabulka 1: Nastavitelné parametry programu a jejich přípustné hodnoty.

Velikost populace se může po nastavení změnit vlivem počtu rodičů při selekci typu TOURNAMENT tak, aby počet jedinců v populaci byl dělitelný velikostí skupiny. Tímto způsobem se turnaje zúčastní každý jedinec v populaci.