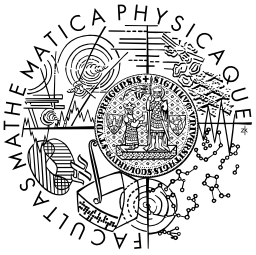


MATEMATICKO-FYZIKÁLNÍ
FAKULTA
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

David Košťál

Implementace hry Duke

Informatický ústav Univerzity Karlovy (208. • 32-IUUK)

Vedoucí bakalářské práce: RNDr. Ondřej Pangrác, Ph.D.
Studijní program: Informatika (B1801)
Studijní obor: ISDI (1801R049)

Praha 2021

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ....

Podpis autora

Děkuji vedoucímu práce RNDr. Ondřeji Pangrácovi, Ph.D. Dále děkuji všem, kteří mě naučili něco nového. Děkuji také Týnce. Děláš můj svět barevnější.

Název práce: Implementace hry Duke

Autor: David Košťál

Ústav: Informatický ústav Univerzity Karlovy (208. • 32-IUUK)

Vedoucí bakalářské práce: RNDr. Ondřej Pangrác, Ph.D., katedra

Abstrakt: Práce se zabývá implementací hry The Duke včetně počítačového protivníka. Vzhledem k tomu, že The Duke nebyla podrobena rozsáhlému výzkumu, nejsou určeny koeficienty optimální strategie a to například hodnoty figurek. Z tohoto důvodu se počítačový protivník sám vyvíjí pomocí hry počítač proti počítači. The Duke má vysoký faktor větvení a velký stavový prostor, bylo tedy nutné udělat řadu optimalizací. Pro zjištění optimálního tahu se používá algoritmus minimax s alfa-beta ořezáváním a ukládáním již uvažovaných stavů. Pro vylepšování strategií je využit princip genetických algoritmů.

Klíčová slova: Duke Desková Hra Implementace

Title: Duke board game implementation

Author: David Košťál

Institute: Computer Science Institute of Charles University (208. • 32-IUUK)

Supervisor: RNDr. Ondřej Pangrác, Ph.D., department

Abstract: This work is about implementation of board game called The Duke including player controlled by computer. The Duke was not yet analyzed on deeper level, so coefficients (for example values of figures) are not yet determined. For this reason player controlled by computer evolves and improves itself by playing against itself. The Duke has a high branching factor and big state space and because of this a few optimilizations had to be made. Algorithm minimax with alpha-beta prunning is used for determining the best possible move. Also considered states are stored. The principle of genetic algorithms is used for evolution of strategies.

Keywords: Duke Board Game Implementation

Obsah

Úvod

Hry tvoří důležitou část našeho života. Od mládí se díky nim můžeme mnoho naučit a to například prostorovou představivost, logiku a strategii (Bavelier a Davidson, 2013). Dále také mohou podpořit naší soutěživost. Díky dobrému pocitu z výhry nám dávají velikou motivaci vytvrvat v hraní a učení se nejlepších strategií. Díky této motivaci je snažší věnovat se delší čas jedné aktivitě, což vede ke zdokonalení se v ní. Nejedná se tedy pouze o zábavu, ale hry mohou být důležitý učící nástroj. A to nejen při zkoumání strategie hry. I při běžném učení můžeme přidat herní prvky, abychom dosáhli větší motivace a lepších výsledků. Tomuto principu se říká gamifikace (Huotari a Hamari, 2012).

Když jsem byl malý měl jsem matematickou počítačovou hru "Alík - veselá matematika", kterou jsem hrál velmi často a díky tomu jsem se nejen v matematice velmi zlepšil, ale i jsem si ji oblíbil. Hry nám tedy mohou i pomoci si vytvořit vazbu na jejich téma, která může trvat dlouho po dokončení hraní.

Více než dříve se nyní během pandemie ukazuje význam počítačů a možnosti hrát hry na počítači. Navíc počítačový protihráč je vždy připraven hrát a může být na vyšší úrovni než lidský protihráč. I nejlepší profesionální hráči jsou stále ve více hrách poraženi umělou inteligencí. Například v šachách hranými na čas, počítač poprvé porazil mistra světa v roce 1996 když počítač Deep Blue porazil tehdějšího mistra světa v šachách Garryho Kasparova (Newborn, 2012), který je i aktuálně velmi uznávaným hráčem. I díky daleko většímu výkonu moderních počítačů jsou dnešní počítačoví protihráči na daleko lepší úrovni. Počítač poprvé porazil nejlépe hodnoceného hráče ve hře Go až v roce 2016, tedy přes devatenáct let později než šachového. Tuto dlouhou prodlevu má na svědomí především obrovský stavový prostor hry Go (Granter a kol., 2017).

Tato práce se zabývá deskovou hrou The Duke. Tato hra připomíná hru šachy. Bere si z ní některé principy jako například vyhození figurky či některé způsoby pohybů. Ve hře The Duke je však prvek náhody, kterým je hra odlišena od šachů. Není tedy možné efektivně plánovat na velký počet tahů dopředu, protože se potřebujeme přizpůsobit vlivu náhody. Další rozdíl je, že se jedná se o poměrně novou hru, která ještě nebyla přiliš podrobena rozboru strategií. Hra je pro dva hráče. Hraje se s šachovnicí 6x6 políček a cílem je vyhodit protihráčovu figurku Duke [1], která je obdobou šachového krále. Hra má ruzná rozšíření a volitelná pravidla, která nejsou rozsahem této práce. Tato práce se soustředí na základní verzi a její implementaci, včetně implementace rozhodovacího algoritmu pro hru proti počítači.

V dalších kapitolách je blíže představena hra, jsou rozebrána její pravidla a její historie. Dále je představena implementace počítačového protivníka, rozebrána uskalí implementace a jejich řešení.

[1] Duke je anglický výraz pro vévodu. Nicméně u názvů figurek se běžně nepoužívá překlad. Z tohoto důvodu v této práci budou používané skloňované anglické názvy.

1. The Duke

1.1 Popis hry

The Duke je strategická hra s malým prvkem náhody, která se projevuje při tažení nových figurek.

Narozdíl od šachů se nezačíná se všemi figurkami na plánku, ale pouze se třemi pro každého hráče. První hráč umístí svojí figurku Duke na jedno ze dvou prostředních políček v řádku, které je nejblíž k němu a dále dvě figurky Footman, aby sousedili se svým Dukem a to hranou. Sousedství pouze rohem nelze. Druhý hráč udělá to samé na protilehlé straně.

Cílem hry je zajmout [1] protihráčova Duka. Každý hráč dostane sáček se zbylými figurkami, ze kterých během hry může vytáhnout náhodnou novou figurku a umístit ji vedle svého Duka.

1.2 Historie

Hra byla vytvořeno v roce 2013. Její tvůrci jsou Jeremy Holcomb a Stephen McLaughlin. Hru vydala společnost Catalyst Game Labs. Jednoznačně vychází z principů šachů, které používá jako základ a přidává nové způsoby pohybů.

The Duke byl v roce 2013 nominován do soutěže the Dice Tower Gaming Awards v kategorii 'Nejlepší hra pro dva hráče'. Dále vyhrál v roce 2014 soutěž Mensa Select.

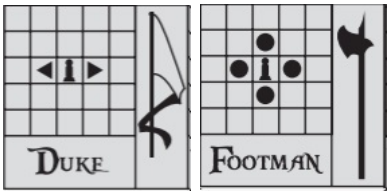
1.3 Pravidla

Součástí the Duke je šachovnice 6x6 políček, na které se odehrává veškerá akce. Dále dva sáčky s figurkami. Figurky jsou reprezentovány jako destičky, které mají na sobě z obou stran tabulku pohybů, které se zpravidla liší. Uprostřed těchto tabulek je symbol figurky, který značí kde se aktuálně figurka nachází na šachovnici a ostatní políčka pohybové tabulky je nutné brát relativně vzhledem k aktuální pozici figurky na šachovnici. Seznam figurek se nachází v sekci 1.4. Na začátku hry si každý hráč vezme jeden sáček a vytáhne z něj svého Duka a dva Footmany. Zbylé figurky nechá v sáčku. Následně se určí začínající hráč. Ten umístí své počáteční figurky, a to tak jak bylo popsáno v sekci 1.1. Poté to samé udělá i druhý hráč.

1.3.1 Orientace

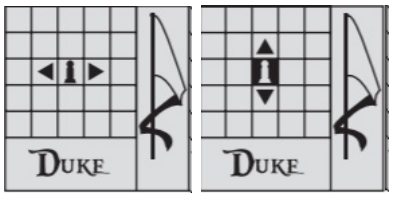
Každá figurka musí být umístěna svým jménem ke hráči, který tuto figurku ovládá a musí zůstat takto natočená celou hru. Pokaždé když se umisťuje nová figurka do hry, tak je otočena startovní stranou vzhůru. Startovní strana se pozná podle toho, že symbol figurky je černý na světlém pozadí. Naopak nestartovní

[1] Zajmutí figurky je obdoba šachového vyhození figurky. Po zajmutí je figurka odebrána ze hry a není ji možné znovu přidat



Obrázek 1.1: Každá figurka má z každé stran své jméno a 5x5 mřížku s vyznačením své aktuální polohy a možnými pohyby

strana se pozná tak, že symbol figurky je světlý na černém pozadí. Viz obrázek 1.2



Obrázek 1.2: Vlevo příklad otočení startovní stranou vzhůru, vpravo příklad otočení nestartovní stranou vzhůru.

1.3.2 Průběh hry

Ten hráč, který umístil své počáteční figurky jako první, začíná. V průběhu hry se hráči pravidelně střídají po každém tahu, dokud jeden nedosáhne vítězství nebo hra neskončí remízou. V každém tahu je na výběr ze dvou možností. První je pohnout některou ze svých figurek. Druhá je přidat náhodně novou figurku ze svého sáčku. Je nutné si vybrat právě jednu z těchto možností, není možné tah vynechat.

1.3.3 Přidání nové figurky

Hráč se může rozhodnout přidat novou figurku pouze pokud je volné některé z políček, která sousedí s jeho Dukem. Uvažují se pouze ta políčka, která s Dukem sousedí hranou, nestačí sousedství pouze rohem. Přidání probíhá vytažením náhodné nové figurky ze sáčku, a následným povinným umístěním na libovolné volné políčko z dříve specifikovaných. Důležité je dodržet podmínky orientace vypsané v sekci 1.3.1.

1.3.4 Pohyb

Druhá možnost je pohnout s vlastní figurkou na plánu. Po dokončení pohybu je tažená figurka otočena a tím dojde ke změně jejích možných pohybů. Figurka má na každé straně pohybovou mřížku, kde je vyznačena její aktuální pozice a dále má vyznačena políčka s kterými může interagovat, relativně vzhledem k její aktuální pozici. Různé pohybové ikony v mřížce určují způsob interakce figurky s daným políčkem. Samozřejmě není možné se tímto způsobem dostat mimo hrací plán.

1.3.5 Pohybové ikony

Hráč si v rámci jednoho tahu může vybrat jedinou figurku (počáteční políčko), jediné cílové políčko a jediný způsob interakce bez ohledu na počet možností. Seznam pohybových ikon vyskytujících se na pohybové mřížce figurky:

1. Move

   Pokud je na cílovém políčku tato ikona, může se na něj figurka přesunout a to pouze v případě, že z počátečního políčka sem vede přímá cesta a v cestě není žádná přátelská ani nepřátelská figurka. Pokud je přátelská figurka na cílovém políčku tah nelze provést. Naopak pokud na něm je nepřátelská figurka tah lze provést a nepřátelská figurka je zajata.

2. Jump

   Figurku je možné přesunout na políčko vyznačené symbolem jump pokud se na ní nenachází přátelská figurka. Pokud se na něm nacházela nepřátelská figurka pak je zajmuta. Nic se nestane "přeskočeným" figurkám.

3. Slide

   Umožňuje figurce se posunout o libovolný nenulový počet políček v daném směru, nesmí se však přejít přes políčko s přátelskou figurkou. Na políčko s nepřátelskou figurkou je možné vstoupit, ale v takovém případě je nutné zde tah ukončit a nepřátelská figurka zajata.

4. Jump slide

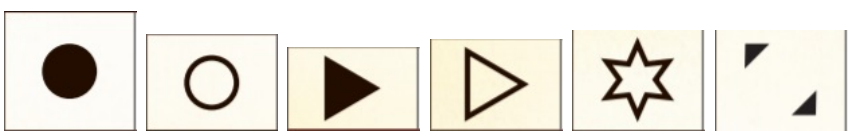
   Značí, že figurka nejdříve povinně skočí na políčko vyznačené tímto symbolem a poté se může posunout o libovolný počet políček v daném směru dle pravidel specifikovaných v bodě 3. Není možné pomocí tohoto pohybu zajmout figurku na jejímž políčku nebyl pohyb ukončen.

5. Strike

   Umožňuje zajmout nepřátelskou jednotku na vyznačeném políčku. Figurka se v tomto případě nehýbe a zůstává na stejném místě. Jako vždy dojde k otočení na druhou stranu.

6. Command

   Figurka s tímto symbolem na své pohybové mřížce může přesunout libovolnou přátelskou figurku z políčka označeného tímto symbolem na jiné takové políčko, pokud na cílovém políčku není přátelská figurka. Tímto pohybem může dojít k zajetí nepřítele. Otáčí se pouze figurka, která měla tento pohyb na své mřížce, nikoli ta která se ve skutečnosti pohnula.

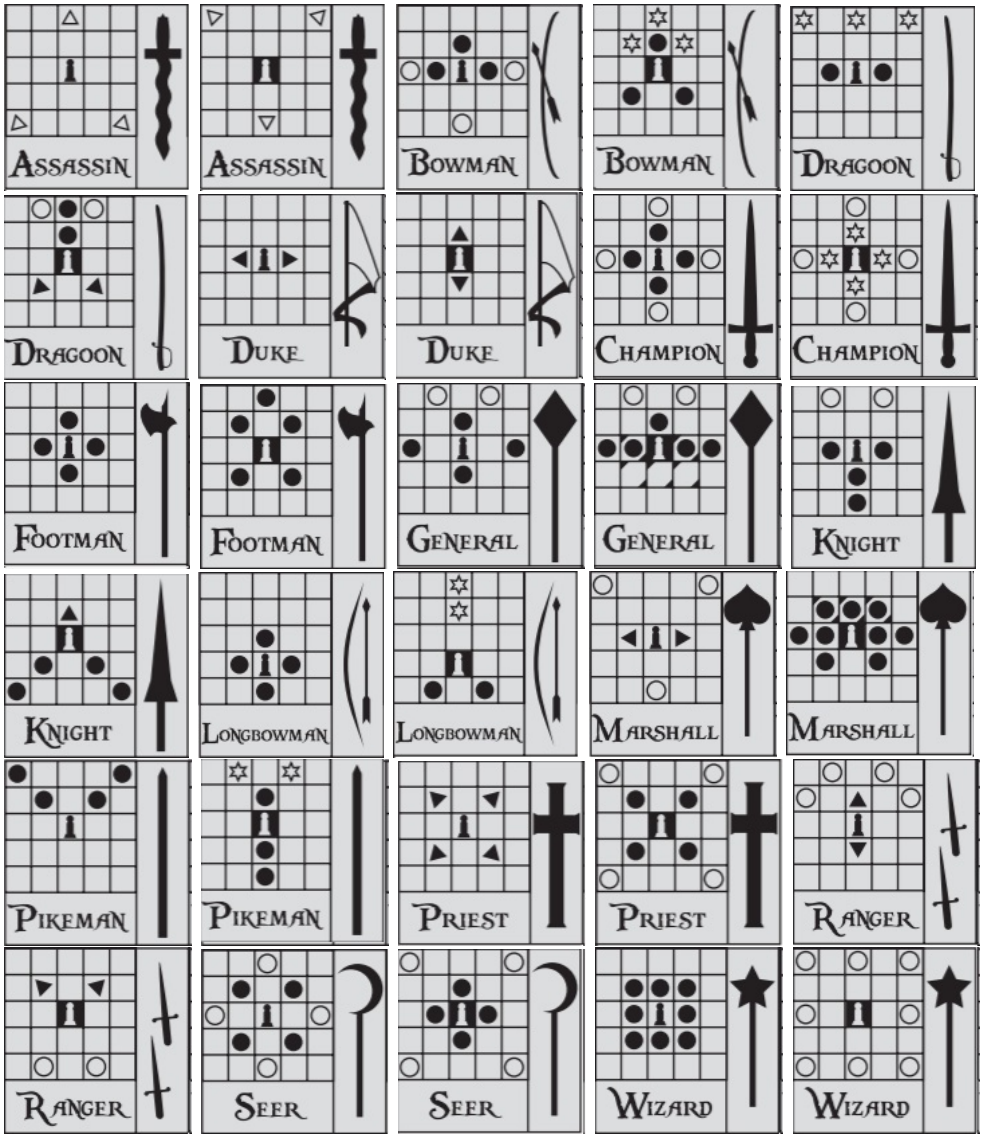


Obrázek 1.3: Pohybové ikony. Zleva: Move, Jump, Slide, Jump slide, Shoot a Command. Command může být na stejném políčku jako jiný symbol.

Hráč vyhraje pokud zajme protihráčova Duka. Pokud hráč nemá žádnou možnost na tah, hra končí remízou. Z praktického důvodu jsem přidal pravidlo, že pokud 30 tahů nedojde k přidání či zajmutí figurky hra také končí remízou.

1.4 Seznam figurek

Každý hráč má k dispozici stejnou sadu figurek, se stejnými pohyby, liší se pouze v barvě pozadí figurky. Každou figurku má k dispozici hráč pouze jednou, kromě Footmanů a Pikemanů. Každý hráč má tři Footmany a tři Pikemany. V základní hře se nachází patnáct různých figurek, pro každého hráče.



Obrázek 1.4: Přehled figurek, které má každý hráč k dispozici. Pikemana má třikrát, Footmana také. Každou z ostatních figurek má právě jednou. Každá figurka má dvě strany s různými tabulkami pohybů.

2. Hra proti počítači

V rámci této práce je vytvořen i počítačový protivník, proti kterému může uživatel hrát.

Při implementování rozhodovacího algoritmu jsem narazil na limity výkonu mého počítače. Už při plánování na tři tahy dopředu program počítal nejlepší tah velmi dlouho. Je to způsobeno složitostí hry The Duke a to především vysokým faktorem větvení (Branching factor) a velikým stavovým prostorem. V sekci 2.6 jsou popsány metody, které program zrychlily.

2.1 Stavový prostor

Stavový prostor udává kolik je při hře možné dosáhnout ruzných pozic hry, včetně těch špatně dosažitelných. Pozice hry se liší dle rozestavení podmnožiny figurek a jejich otočení. Dále také dle figurek, které mají hráči k dispozici ve svých sáčcích a které již byly vyhozeny.

2.1.1 Algoritmus pro výpočet stavového prostoru

V obrázku 2.1. uvádím pseudokód programu, který jsem navrhl tak, aby spočítal odhad všech možných pozic. Počítá počet možných stavů začínajíc po přídání počátečních figurek dokud neskončí hra, tedy dokud jsou oba Dukové na šachovnici. Program spočítá stavový prostor pomocí ukládání si mezivýsledků v tabulce. Při implementování jsem narazil na stack overflow z důvodu příliš veliké tabulky. Uvádím tedy pouze výsledné hodnoty pro zjednodušené situace. Nicméně níže uvedený pseudokód platí i pro skutečnou situaci. Pokud by každý hráč měl pouze jednoho Footmana, pak by počet stavů byl $3{,}59 * 10^{52}$. Pokud bychom uvažovali, že každý hráč má každou figurku pouze jednou, pak by počet stavů byl $1.3768 * 10^{49}$.

2.1.2 Odůvodnění algoritmu

První řádek kódu říká, že spočítáme počet možností jak umístit obě figurky "Duke" a vynásobíme to počtem možností jak umístit zbylých 24 různých figurek a 3 pikemany prvního hráče, 3 pikemany druhého hráče, 3 footmany prvního hráče a 3 footmany druhého hráče do 34 zbývajících políček.

Pokud s je rovno 0, pak jsme už prošli všechny políčka. Pak zbývá vyřešit kolika možnostmi můžeme rozdělit zbylé figurky do dvou skupin (vyhozené a ty které jsou ještě v sáčku). Každý hráč začíná s dvěma footmany, nemohou být tedy v sáčcích.

Jinak můžeme políčko buď nechat prázdné, nebo ho zaplnit libovolnou zbývající figurkou. Figurka může být otočena dvěmi způsoby. Pokud políčko nenecháme prázdné tak tím vyřešíme jednu figurku. V každém případě nám zbývá vyřešit o políčko méně.

```
Vrať 36*2*35*2*A(24,3,3,3,3,34)

Nastav A(f,p1,p2,f1,f2,s) = 0
Pokud s je 0 pak:
        fx1 = max(f1-2,0)
        fx2 = max(f2-2,0)
        A(f,p1,p2,f1,f2,s) = 2^f * (p1+1) * (p2+1) * (fx1+1) * (fx2+1)
Jinak:
    Zvětši A(f,p1,p2,f1,f2,s) o A(f,p1,p2,f1,f2,s-1)
    Pokud f je alespoň 1 pak:
        Zvětši A(f,p1,p2,f1,f2,s) o 2*f*A(f-1,p1,p2,f1,f2,s-1)
    Pokud p1 je alespoň 1 pak:
        Zvětši A(f,p1,p2,f1,f2,s) o 2*A(f,p1-1,p2,f1,f2,s-1)
    Pokud p2 je alespoň 1 pak:
        Zvětši A(f,p1,p2,f1,f2,s) o 2*A(f,p1,p2-1,f1,f2,s-1)
    Pokud f1 je alespoň 1 pak:
        Zvětši A(f,p1,p2,f1,f2,s) o 2*A(f,p1,p2,f1-1,f2,s-1)
    Pokud f2 je alespoň 1 pak:
        Zvětši A(f,p1,p2,f1,f2,s) o 2*A(f,p1,p2,f1,f2-1,s-1)
    Pokud f,p1,p2,f1,f2 jsou nula pak:
        A(f,p1,p2,f1,f2,s) = 1
```

Obrázek 2.1: Pseudokód pro výpočet velikosti stavového prostoru.

2.2 Faktor větvení (Branching factor)

Branching factor neboli faktor větvení určuje jak moc se větví herní strom. Pokud se tedy nacházíme v nějakém vrcholu herního stromu branching factor určuje kolik má průměrně daný vrchol potomků. Obdobně to značí z kolika možných tahů má průměrně hráč na výběr. Přizpůsobil jsem program, aby počítal průměrný počet možností. Do výpočtu jsem zahrnul pouze počet možností v opravdu nastalých situacích, nikoliv počet možností ve větší hloubce minimaxu. Chtěl jsem tak předejít zkreslení výsledku, protože minimax uvažuje i neobvyklé situace, do kterých by se hráči běžně nedostali. Program během svých her navštívil 113777 situací a měl 4060059 možných tahů. Tedy měl průměrně 35,7 možných tahů. Jedná se o přibližný odhad za použití empirických metod. Větší přesnosti lze dosáhnout pomocí analyzování více her. Empirické metody také použil docent David Barnes z University of Kent při analýze šachů. Analyzoval 2,5 milionů her a výsledný průměr byl 31,1 možných tahů (Barnes, 2019). Dle literatury je průměrný faktor větvení u šachů 30 (Bernstein a de V. Roberts, 1958) či 35 (Laramée, 2000).

Veliký stavový prostor znemožňuje efektivní persistentní zapamatování si stavů a optimálních tahů pro ně. Počet těchto záznamů by byl příliš veliký. Navíc v každé hře je možné navštívit různé situace. Příliš mnoho záznamů by mohlo program zpomalit. Ukládání těchto informací po dobu více her také komplikuje to, že se mohou měnit parametry strategií a optimalní tah pro jednu strategii může být velmi špatným tahem pro jinou viz sekce 2.5. Nedochází tedy k ukládání uvažovaných situací mezi partiemi a tyto situace jsou ukládány pouze během

jedné hry viz podsekce 2.6.2. Vysoký branching factor je zase překážkou pro rychlé prohledávání stavového stromu z aktuálního vrcholu (situace) do větší hloubky. Z tohoto důvodu program prohledává stavový strom do hloubky 4 od aktuálního vrcholu. Tento strom je procházen za použití algoritmu minimax. Vzhledem k tomu, že by stále docházelo k vyhodnocení přibližně 1624000 situací, je aplikováno alfa beta ořezávání (alpha beta pruning), které dokáže ovlivnit minimax a oříznout prohledávanou část stromu, anižby došlo k ovlivnění výsledku.

2.2.1 Přidávání figurky

Branching factor je velmi ovlivňen možností přidat během hry náhodnou novou figurku. Například pokud vedle své figurky Duke má hráč dvě volná políčka a zbývá mu 13 různých figurek tak je potřeba zvážit 26 různých tahů, jenom abychom zjistili jestli chceme přidávat novou figurku, při uvažování na jeden tah dopředu. Program však vyhodnocuje všechny pozice [1] dosažitelné během čtyř tahů. I kdybychom tahy omezili pouze na přidávání nových figurek, tak v situaci kdy oba hráči mají tři volná políčka vedle svého duka a 13 různých figurek, existuje $39 * 39 * 24 * 24$ tedy 876 tisíc situací, které je nutné vyhodnotit. A to neuvažujeme kombinaci s pohybem figurky. A stále se nedá na výsledek spolehnout, protože si nevybíráme figurku, kterou chceme, ale náhodnou vytáhneme. Z tohoto důvodu jsem se rozhodl při zjišťování jestli nejlepší tah je přidat figurku vyzkoušet pouze několik náhodných figurek jakožto zástupce.

Ohodnocení stavů po přidání těchto figurek (zástupců) je možné několika způsoby. První možností je vzít průměr z ohodnocení dosažených stavů. Tato možnost může zapříčinit, že program bude mírně odvážnější a bude více spoléhat na náhodu. Druhá možnost je skeptický přístup. Tedy pokud je protihráčův tah předpokládáme, že se mu podaří vytáhnout pro něj nejlepší figurku. Pokud je náš tah předpokládáme, že vytáhneme nejhorší figurku. Tento způsob povede k opatrnějšímu přístupu, kde bude méně spoléháno na náhodu.

2.3 Minimax

Při hledání nejlepšího tahu je použita teorie minimaxu a s ní i spojený algoritmus. Tento koncept předpokládá, že každý hráč zahraje nejlepší tah ze všech možných. Snaží se maximalizovat minimální možný zisk, který hráč může ze svého tahu získat, pokud se protivník[2] bude snažit možný zisk mu co nejvíce zmenšit.

2.3.1 Teorie

Než rozebereme minimax více, potřebujeme se seznámit s pojmem zero-sum game, neboli hra s nulovým součtem. Nulový součet je situace v herní teorii, kde zisk jednoho hráče je roven ztrátě jiného (Khotsianovich a Barisevich, 2015). Hra s nulovým součtem musí být pro minimálně dva hráče, ale počet hráčů není shora

[1] Ve skutečnosti díky alfa beta prořezávání algoritmu program nemusí vyhodnotit všechny takové pozice, ale pro ilustraci situace zde uvažujeme všechny.

[2] Minimax lze obecně aplikovat i pro hry o libovolném počtu hráčů. Vzhledem k tomu že The Duke je právě pro dva hráče, budeme se soustředit výhradně na verzi s dvěma hráči.

nijak omezený. Například Bitcoin, stejně jako všechny ostatní kryptoměny, podléhá tomuto principu. Na obchodování s ním lze vydělat pouze pokud někdo jiný na tomto obchodu prodělá. Dalším příkladem jsou šachy. Hráč může získat materiální výhodu pouze pokud druhý hráč o materiál přijde (například je vyhozena jeho figurka) [3]. The Duke je také zero-sum game. Jeden hráč může vyhrát pouze pokud druhý prohraje.

Věta 1 (Minimax pro hry s nulovým součtem pro dva hráče)**.** *V každé situaci existuje hodnota V a strategie pro každého hráče, takové že platí následující:*

1. *První hráč má garantovaný zisk V, bez ohledu na strategii druhého hráče.*

2. *Druhý hráč má garantovaný zisk -V, bez ohledu na strategii prvního hráče.*

Koncept dostal své jméno minimax, protože každý hráč minimalizuje maximální možný zisk pro protivníka a tím, vzhledem k tomu, že se jedná o hru s nulovým součtem, i minimalizuje maximální svoji ztrátu.

2.3.2 Algoritmus

Algoritmus pro minimax je poměrně jednoduchý. Nacházíme se v určité situaci a chceme najít tah, tak aby, nehledě na to co zahraje protihráč, jsme se po několika tazích dostali do nejlepší možné pozice. Prvně rozeberme situaci kdy chceme vybrat nejlepší tah na základě uvažování na jeden tah dopředu.V takovém případě stačí vyzkoušet každý možný tah, ohodnotit situace do kterých jsme se takto dostali a vybrat z nich tu nejlepší. Pokud uvažujeme na více tahů dopředu pak využíváme algoritmus minimax založený na DFS [4]. Prohledáváme tedy herní strom do uvažované hloubky a to tak, že pokud algoritmus již dosáhl požadované hloubky (bylo odehráno požadované množství tahů), ohodnotíme evaluační funkcí aktuální herní situaci a hodnotu vrátíme. Jinak zjistíme jaké má aktuální stav potomky a rekurzivně se zavoláme na každém z nich. Tím dostaneme seznam možných tahů a jejich ohodnocení. Poté vybereme nejlepší možnost pro toho kdo je zrovna na tahu. Pokud se jedná o náš tah pak vybereme maximální hodnotu a pokud hraje protihráč pak minimální. Získanou hodnotu znovu vrátíme.

2.3.3 Alfa Beta ořezávání

Jak bylo uvedeno na začátku Kapitoly 2, vzhledem k rozsáhlému stavovému prostoru a vysokému faktoru větvení je hra počítače výpočetně velmi náročná. Z důvodu uvažování na čtyři tahy dopředu se dostáváme do velikého množství odlišných situací. Nicméně ne všechny je nutné zvážit, protože minimax předpokládá, že každý hráč zahraje nejlepší možný tah. Hráči si navzájem škodí a jeden druhému se snaží zamezit dostat se do příliš výhodné situace. A právě alfa beta

[3] Materiální výhoda v šachách lze získat i pomocí povýšení pěšáka na posledním políčku na jinou figurku. Pěšák má totiž ostře menší hodnotu než všechny ostatní figurky. Nicméně i toto znamená ztrátu pro protihráče, protože se jeho situace zhorší.

[4] DFS je zkratka pro depth first search, tedy algoritmus pro prohledávání stromu do hloubky. Prohledávání používá zásobník. Pokud jsme ve vrcholu V pak dáme na vršek zásobníku všechny jeho dětské uzly. Díky tomu, že algoritmus dále pokračuje svrchu zásobníku dosáhneme prohledávání nejdříve do hloubky. Algoritmus se používá například pro topologické řazení, detekci cyklů či řešení sudoku.

ořezávání vylepší minimax, aby bral v potaz tuto myšlenku. Díky tomu je dramaticky zmenšen počet prohledávaných situací, aniž by došlo k ovlivnění výsledku. Tím je možné dojít k výsledku rychleji či uvažovat na více tahů dopředu.

```
funkce minimax(vrchol,hloubka,jeMaximalizováno,alfa,beta){
   pokud je vrchol list nebo hloubka je 0:
        vrať evaluaci vrcholu

   pokud jeMaximalizováno:
        nejlepšíHodnota = -nekonečno
        pro každého přímého potomka :
                hodnota = minimax(potomek,hloubka-1,nepravda,afa,beta)
                nejlepšíHodnota = max(nejlepšíHodnota,hodnota)
                alfa = max(alfa, nejlepšíHodnota)
                pokud beta <= alpha:
                        zastav
                vrať nejlepšíHodnota
   jinak:
        nejlepšíHodnota = +nekonečno
        pro každého přímého potomka :
                hodnota = minimax(potomek,hloubka-1,pravda,afa,beta)
                nejlepšíHodnota = min(nejlepšíHodnota,hodnota)
                alfa = min(alfa, nejlepšíHodnota)
                pokud beta <= alpha:
                        zastav
                vrať nejlepšíHodnota
}
//Volání funkce
minimax(kořen,chtěnáHloubka,pravda,-nekonečno,+nekonečno)
```

Obrázek 2.2: Pseudokód minimaxu s alfa beta ořezáváním

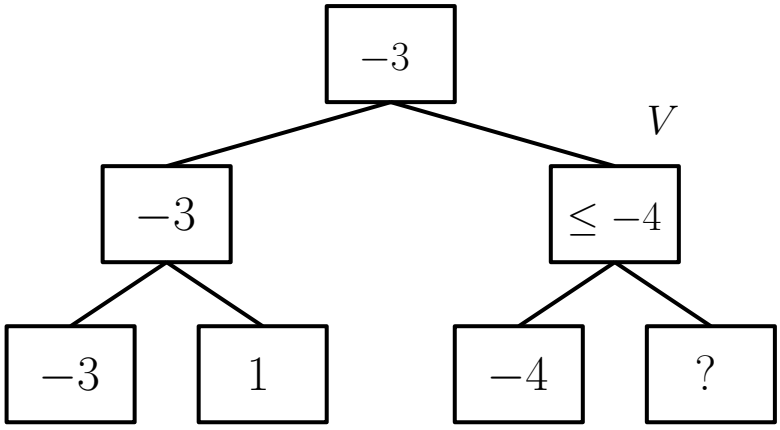
Algoritmus používá dva parametry. První je α (Alfa), což je nejlepší hodnota pro prvního hráče, pokud bychom minimax začali používat až v aktuálním vrcholu nebo v některém z jeho předků. Tedy pokud by hráči začali hrát nejlepší tahy až v této situaci, nebo některé která ji předcházela. Obdobně β (Beta) je nejlepší hodnota pro druhého hráče. Obrázek 2.3 ukazuje algoritmus na příkladu.

2.4 Evaluační funkce

Jak bylo zmíněno v sekci 2.3, vhodný tah je vybrán na základě ohodnocení situací do kterých vede. Tato sekce je zaměřená na sestavení vhodné ohodnocující funkce. V následujících podsekcích jsou představeny části této funkce a na závěr je sestaven její vzorec.

2.4.1 Ohodnocení figurek

Základem ohodnocující funkce musí být ohodnocení figurek. Ohodnocující funkce bez ohledu na figurky by vedla k nehlídání si figurek a jejich ztrátě. Na



Obrázek 2.3: Předpokládejme, že jsme spustili minimax s alfa beta prožezáváním a při výpočtu jsme se dostali do vrcholu V. α je rovna -3, β se rovná -4. $\beta <= \alpha$ a tedy algoritmus nepokračuje dalšími potomky vrcholu V, protože hráč 1 zamezí hráči 2 se do této situace dostat. Pokud by se do ní hráč 2 dostal, potom by si mohl vybrat -4 a byl by na tom určitě lépe než v levé větvi bez ohledu na zbylé možnosti.

rozdíl od šachů není pro hru The Duke určena hodnota každé jednotky [5] . Pokud bude mít každá figurka stejnou hodnotu, může docházet k chybám, že schopnější obětujeme za daleko slabší a tím si zhoršíme naší situaci. Počítačový protivník si tedy udržuje vektor s hodnotami jednotlivých figurek.

Navíc musí být brány v potaz figurky zbývající v sáčcích. Pokud bychom s nimi algoritmus nepočítal, pak by se evaluace situace po přidání a následném zajetí figurky protihráčem nezměnila a program by byl otevřený takovým situacím. Naopak, pokud by figurky měly stejnou hodnotu v sáčku jako i na šachovnici, poté by program neměl dostatečnou motivaci pro přidání nové figurky. Každý počítačový protivník má v rámci parametrů koeficient mezi hodnotou figurky v sáčku a na plánku.

Jak je již zmíněno v sekci o branching factoru (Sekce 2.2), když zvažujeme přidání figurky, vybereme náhodně několik zástupců a pomocí nich zjistíme jestli je přidávání nové figurky dobrý tah. Uvažujeme dvě možnosti jak zpracovat získaná ohodnocení po přidání zástupců. Buď z těchto hodnot uděláme průměr nebo předpokládáme nejhorší a tedy pokud jsme přidávali my tak vybereme nejhorší možnost a pokud protihráč tak nejlepší. Každý počítačový protivník má v parametrech uloženou informaci o používané variantě.

2.4.2 Počet možných tahů

Další část funkce je počet možných tahů. Tento jednoduchý princip dává základní představu o kontrole polí na hrací desce. Pokud má hráč k dispozici velké množství tahů, tak lze předpokládat, že velké množství políček i ovládá. Ovládáním políčka jednoho hráče je myšleno, že se jeho figurka může na daném poli nacházet, a to aniž by hrozilo, že hráč o danou figurku příjde bez odpovídající

[5] V šachách jsou hodnoty následující: dáma 9,věž 5,střelec 3,jezdec 3 a pěšák 1. Král má teoreticky nekonečnou hodnotu a zpravidla se ve výčtu hodnot vynechává. Někdy se také uvádí, že střelec má hodnotu 3,25 a tedy je cennější než jezdec.

náhrady. Pro zjištění skutečného počtu ovládaných políček by bylo potřeba zjistit jaké všechny figurky lze na dané políčko přesunout. Přesunutí jedné figurky může uvolnit cestu jiné, která na ohrožené pole dříve nemohla. Bylo by tedy potřeba uvažovat několik tahů dopředu, čímž by se ještě více zhoršila časová náročnost. Počítačový protivník má mezi parametry i koeficient, který určuje váhu počtu možných tahů.

2.4.3 Počet možných tahů Dukem

Poslední část evaluační funkce je počet tahů, které může udělat Duke. Jednoduše se totiž může stát, že Duke nebude mít žádný možný pohyb a tím se stane velmi zranitelným. Bez uvažování počtu těchto pohybů by byl hráč ovládaný počítačem zranitelnější, protože je velmi často výhodné přidat nové figurky vedle svého Duke. Mezi parametry počítačového protivníka se nachází i koeficient, který určuje váhu počtu možných tahů Dukem.

2.4.4 Podoba evaluační funkce

Evaluační funkce je tedy lineární kombinací výše zmíněných parametrů. Vzorec pro její výpočet je určen v Definici 1.

I	Množina aktivních figurek prvního hráče
J	Množina aktivních figurek druhého hráče
I_B	Množina záložních figurek prvního hráče
J_B	Množina záložních figurek druhého hráče
M	Množina všech možných tahů prvního hráče
D	Množina možných tahů Dukem prvního hráče
$\vec{w}$	Vektor ohodnocení jednotlivých figurek z pohledu prvního hráče.
C_A	Poměr hodnot aktivní a záložní figurky z pohledu prvního hráče.
C_M	Váha celkového počtu pohybů z pohledu prvního hráče.
C_D	Váha počtu pohybů Dukem z pohledu prvního hráče.

Definice 1. *Předpokládejme, že je první hráč na tahu a že při hledání nejlepšího tahu jsme se minimaxem po odehrání požadovaného počtu tahů dostali do situace S pro kterou platí informace uvedené v tabulce výše v Sekci 2.4.4. Poté je ohodnocení situace S z pohledu prvního hráče následující:*

$$E_{S,1} = \sum_{i \in I} w_i - \sum_{j \in J} w_j + \sum_{i \in I_B} w_i / C_A - \sum_{j \in J_B} w_j / C_A + |M| * C_M + |D| * C_D \tag{2.1}$$

2.4.5 Získané parametry

Nechal jsem program spočítat 100 generací parametrů (Viz sekce 2.5). Nejlepším jedincem ve sté generaci má následující nastavení parametrů. Všechny řešení ve sté generaci mají stejnou hodnotu prvních tří parametrů a liší se pouze ve čtvrtém a hodnotě figurek.

Poměr hodnot aktivní a záložní figurky (aktivní má vyšší hodnotu)	1,25
Váha počtu pohybů Dukem z pohledu prvního hráče	0,5
Minimax při přidávání využívá skeptický přístup	Ano
Váha celkového počtu pohybů	0,6875

Assassin 16	Bowman 20	Dragoon 28
Duke 1000	Footman 22	General 30
Champion 23	Knight 28	Longbowman 1
Marshall 9	Pikeman 18	Priest 19
Ranger 6	Seer 9	Wizard 16

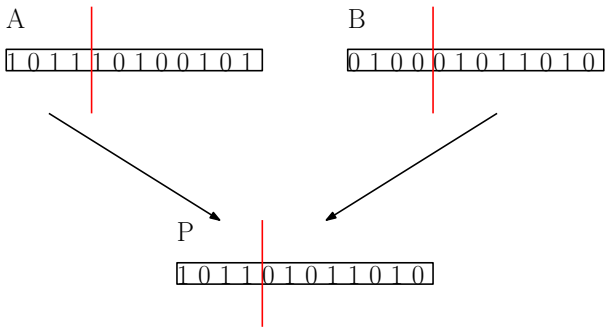
2.5 Genetický algoritmus s roulette wheel selection

2.5.1 Obecný princip

Evaluační funkce závisí na několika koeficientech, které jsou diskutovány v Sekci 2.4. Pro správné fungování programu, je potřeba dosadit co nejlepší hodnoty do koeficientů. Vzhledem k tomu, že hra The Duke není podrobena takovému rozboru jako například šachy, tyto parametry nejsou ještě určeny. Vyvíjený program obsahuje implementaci genetického algoritmu, který umožňuje vyvíjet a vylepšovat strategie (Razali a kol., 2011).

Tento algoritmus vychází z principu evoluce. Parametry přepíšeme do řetězce 0 a 1. Výsledný řetězec budeme nazývat chromozom. Chromozom obsahuje veškerou informaci řešení. Populace je skupina chromozomů. Zpravidla má populace fixní velikost a tedy stejný počet chromozomů po celou dobu běhu.

Před diskutováním algoritmu je také třeba zavést další dva pojmy a to křížení a mutace. Ke křížení jsou potřeba dva chromozomy, kterým se říká rodičové či předci. Dále je náhodně vygenerován index. Tím jsou oba rodičové rozděleny na dvě části. Výsledný chromozom vznikne složením první části jednoho rodiče s druhou částí druhého rodiče. Ukázka křížení je na obrázku 2.4.



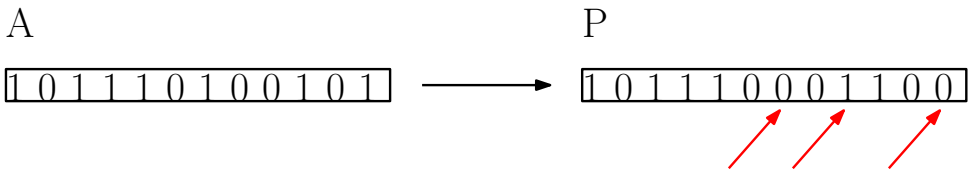
Obrázek 2.4: Příklad křížení. Potomek P vznikl křížením rodiče A a rodiče B. Pokud aplikujeme křížení vícekrát na stejnou dvojici rodičů můžeme dostat různé potomky.

Během mutace procházíme celý chromozom a každý bit má určitou pravděpodobnost, že bude znegován. Nula může být změněna na jedničku a obráceně. Pokud je tato pravděpodobnost příliš malá, poté se změnší počet změn oproti

rodiči a nedojde k velkému přesunu v prostoru možností. Tím dochází ke konzervativnějšímu přístupu k prohledávání prostoru možných chromozomů, což může v některých případech zpomalit konvergenci. Na druhou stranu se jedná o spolehlivější přístup, který se více drží principu, že lepší řešení mají lepší potomky. Méně se spolehá na náhodu, a to díky tomu, že jsou prohledávána řešení bližší rodičovi.

Naproti tomu, pokud bude pravděpodobnost mutace větší, bude docházet k více změnám. Tím dochází k větším přesunům v prostoru možných chromozomů. Tento přístup doufá, že řešení náhodou objevíme co nejdříve. Snižuje však riziko, že algoritmus nalezne pouze lokální extrém.

Pomocí mutace může zpravidla vzniknout libovolný chromozom bez ohledu na předka[6]. Nicméně, čím větší počet změn oproti předkovi, tím je šance výskytu menší. Mutace je ukázána na obrázku 2.5



Obrázek 2.5: Příklad mutace. Potomek P vznikl aplikací mutace na předka A. Pokud aplikujeme mutaci vícekrát na stejného předka můžeme dostat rozdílné výsledky.

2.5.2 Populace

Velikost populace ovlivňuje rychlost a přesnost výsledku (Rylander a Gotshall, 2002). Menší populace znamená rychlejší konvergenci, tedy dosažení stavu, kdy se populace již nemění, protože nově vytvořená řešení jsou horší než každý člen populace. Porovnáváním řešení se zabývá podsekce 2.5.4. Větší populace zapříčíní pomalejší konvergenci, ale výsledek je přesnější, tedy algoritmus má větší šanci nalézt skutečně nejlepší populaci a nikoliv pouze lokální extrém. Lokální extrém je situace kdy populace neobsahuje optimální řešení, ale křížení a mutace nevytváří lepší řešení.

Členy populace lze na začátku zvolit naprosto náhodně. Poté existují dvě možnosti vývoje. První možností je vzít jeden chromozom a ten zmutovat, porovnat s původním a nechat pouze lepší variantu. Program kontroluje, že došlo k mutaci alespoň jednoho bitu chromozomu.

Druhá možnost je vybrat dva rodiče a pomocí křížení vytvořit jejich potomka. Na něj také následně aplikujeme mutaci s menší pravděpodobností pro jednotlivé bity a navíc upustíme od požadavku, že musí dojít alespoň k jedné mutaci. Pro zachování fixní velikosti populace je poté nutné vybrat horšího rodiče a odstranit jej z populace. Pokud za oba rodiče zvolíme stejný chromozom, nedojde během křížení k žádné změně a algoritmus se této variantě vyhýbá.

[6]Libovolný chromozom může vzniknout za předpokladu, že mutace jednotlivých bitů chromozomu má pravděpodobnost v intervalu (0,1). Pokud by pravděpodobnost byla nula pak je potomek vždy stejný jako rodič a pokud by pravděpodobnost byla jedna poté je potomek negací předka.

V programu jsou používány obě tyto varianty a v každé generaci je náhodně jedna z nich vybrána. Křížení s mutací má pravděpodobnost $\frac{1}{3}$ a samotná mutace má pravděpodobnost $\frac{2}{3}$. Tyto pravděpodobnosti byly vybrány tak aby převládala samotná mutace, ale aby křížení s mutací bylo stále možné. Po provedení mutace a porovnání potomka s předkem je jisté, že odebraný rodič nebyl lepší než potomek. Na druhou stranu tento přístup je náchylnější k zastavení výpočtu na lokálních extrémech. Pokud provedeme křížení a mutaci, pak existuje riziko, že odebraný rodič byl lepší než vzniklý potomek, ale snížila se pravděpodobnost zastavení výpočtu na lokálním extrému. Ideální je tedy kombinace těchto přístupů s převahou samotné mutace.

2.5.3 Výběr rodičů

Mechanismus selekce určuje chromozomy, vybrané pro reprodukci. Hlavní princip je, že lepší chromozom má větší šanci být vybrán a je tedy předkem častěji. Křížení a mutace zařizují prohledávání prostoru možností, zatímco díky selekci jsou odříznuta potenciálně špatná řešení. Jedná se tedy o heurestiku, která říká, že lepší rodičové mají větší šanci na to mít dobrého potomka (Razali a kol., 2011). Vzhledem k tomu, že se jedná pouze o heurestiku, je potřeba dát prostor také horším řešením, i když méně často než těm lepším.

Jeden z možných způsobů výběru se nazývá roulette wheel selection, neboli výběr pomocí ruletového kola. Spočívá v tom, že rozdělíme kolo na tolik částí kolik máme chromozomů. Velikost každé části bude rovna ohodnocení daného řešení. Toto ohodnocení je definováno v Podsekci 2.5.4. Poté zatočíme kolem a tím vybereme rodiče. To tedy nutně znamená, že lepší řešení mají větší šanci být vybrána, ale i ta horší stále mají tuto možnost.

2.5.4 Hodnocení chromozomů a jejich porovnávání

V sekci 2.5.2 je zmíněné, že algoritmus rozlišuje úrovně jednotlivých řešení. Existují tedy horší a lepší chromozomy.

Při výběru předků se používá fitnes funkce, udávající o jak kvalitní řešení se jedná. Každý chromozom má svoji hodnotu fitnes funkce. Začíná s hodnotou 20 a pokaždé, když v porovnání porazí jiné řešení, je jeho hodnota zvýšena o jedna. Čím má chromozom větší hodnotu fitnes funkce, tím má větší šanci, že bude vybrán jako předek. Při křížení vznikne nový chromozom, čímž se zvýší velikost populace. Protože chceme zachovat fixní velikosti populace, musíme vybrat horšího rodiče a toho z populace odstraníme. V případě mutace znovu vznikne nový chromozom a tím se zvýší velikost populace. Poté porovnáváme rodiče s potomkem a horší řešení odstraníme.

Porovnávání dvou řešení se neřidí dle fitness hodnoty, ale necháme tato řešení hrát několik partií proti sobě a poté v populaci ponecháme pouze ten chromozom, který vyhrál vícekrát. Jeho hodnotu fitnes funkce navíc zvýšíme o jedna, aby měl příště větší šanci být vybrán. Implementovaný program odstraní z populace chromozom, který z pěti her vyhraje méněkrát. V případě stejného počtu výher při křížení je jeden předek odstraněn náhodně. Při remíze během porovnávání předka a potomka vzniklého křížením je zachován předek. Pokud jedna strategie již nemůže druhé v počtu výher dostihnou, není pokračováno dalšími partiemi. Po

prvních dvou hrách se prohodí pořadí hráčů, aby ten který začíná neměl výhodu.

Záznam každé hry v rámci evoluce je uložen v složce Logs. V té se nachází soubory s popisem obou strategií, záznam jednotlivých tahů a výsledek. Ukázka logu je na obrázku 2.6.

```
————————————First————————————
10010000101000110000110010011010011111110000110010011 10110000101010101110110100101000
18, 2, 17, 1000, 16, 25, 6, 19, 30, 3, 4, 27, 1, 10, 23,
Coefficient duke moves 0.1875
Coefficient moves 0.875
Coefficient active passive 3.25
Adding take average 1
————————————Second————————————
10010000101000110000110010011010111111110000110010011 10110000101010101110110100111000
18, 2, 17, 1000, 16, 25, 6, 23, 30, 3, 4, 27, 1, 10, 23,
Coefficient duke moves 0.25
Coefficient moves 0.875
Coefficient active passive 3.25
Adding take average 1
————————————Game————————————
Add Duke[3,0] Add Footman[3,1]
Add Footman[2,0] Add Duke[2,5]
Add Footman[2,4] Add Footman[1,5]
Move [3,0] to [5,0] Move [2,5] to [3,5]
...
Move [0,4] to [0,2] Add Knight[1,2]
Add Wizard[0,1] Move [1,2] to [0,2]
————————-Message————————
Second player won!
————————-End of message————————-
```

Obrázek 2.6: Ukázka záznamu hry. Nejdříve jsou popsány obě strategie, které proti sobě hrají. Na prvním řádku popisu je vypsán chromozom, na druhém jsou uvedeny hodnoty jednotlivých figurek. Jejich pořadí je stejné jako na obrázku 1.4. V sekci "Game"je popsán průběh hry. Na každém řádku je tah prvního hráče, následován tahem druhého hráče.

2.6 Optimalizace

Počítačový protivník pro hru The Duke je výpočetně velmi náročný. Z tohoto důvodu jsem program vylepšil o několik optimalizací a to například o alfa beta ořezávání (viz podsekce 2.3.3). Mezi další vytvořené optimalizace patří předpočítávání možných pohybů, ukládání si již uvažovaných stavů a průběžný výpočet součtu hodnot figurek. Dále jsem také obohatil minimax, aby nehledal lepší tah, pokud hráč již může vyhrát. Alfa beta ořezávání funguje na principu, že když je tah moc dobrý, tak jiný není potřeba hledat. Informaci jestli je tah dobrý získává porovnáním s jinými možnostmi a tedy nezná informace ohledně hry, konkrétně výherní ohodnocení.

2.6.1 Předpočítávání možných pohybů

Při hře počítače jsou nejdříve nalezeny všechny možné tahy a poté je z nich na základě minimaxu vybrán tah nejlepší. První verze algoritmu zvážila všechny dvojice políček a zjistila jestli aktuální hráč může pohnout figurkou z prvního políčka na druhé[7]. Tento přístup pochopitelně nebyl velmi efektivní, vzhledem k

[7]Zkouší se také jestli je možné na dané políčko vystřelit, tedy zajmout nepřátelskou figurku bez pohybu. V následujícím textu je tato možnost považována za pohyb

tomu, že v každé situaci bylo potřeba prozkoumat 36∗36 neboli 1296 potenciálních tahů. Pro ukázku na příkladu uvažujme, že počítač při uvažování na čtyři tahy dopředu zváží 160 tisíc situací. Poté je celkově potřeba ověřit či vyvrátit 200 milionů pohybů.

V druhé verzi algoritmu si každý hráč ukládá seznam svých figurek a jejich souřadnice. Jako startovní políčka uvažuje pouze ta z tohoto seznamu. Pokud má hráč na šachovnici šest figurek pak je potřeba v každé situaci zvážit $6 * 36$ potenciálních tahů. Při prohledávání do hloubky čtyř tahů se celkově jedná o 34 milionů potenciálních pohybů.

Ve třetí verzi algoritmu jsou předpočítány existující pohyby před začátkem hry. Pro každou figurku, její pozici a otočení je vytvořen seznam políček kam by se mohla pohnout, pokud by byla na šachovnici sama a nebránila by ji v pohybu žádná jiná figurka. Při hře je pro každou hráčovu figurku nalezen odpovídající seznam a pouze políčka ze seznamu jsou uvažována jako cílová. Program spočítal celkový počet všech možných dvojic startovních a cílových políček a vyšlo, že existuje 5016 možných tahů. Je $6 * 6$ políček, 15 figurek a každá má 2 možnosti otočení. Průměrně je seznam pro figurku, její pozici a otočení dlouhý $5016/(15 * 6 * 6 * 2)$ neboli 4,64 položky. Při uvažování na čtyři tahy dopředu je nutno zvážit $160000 * 4{,}64 * 6$ tedy 4,454 milionů pohybů.

2.6.2 Ukládání uvažovaných stavů

Jak bylo zmíněno v sekci 2.2 uvažované stavy neukládáme persistentně mezi partiemi, ale v rámci jedné hry jsou tyto stavy ukládány. K tomuto účelu je používaná C++ struktura map, do které se přidávají dvojice klíčů a hodnot a která umožňuje hledání prvků dle klíču s časovou náročností $O(\log n)$, kde n je počet prvků v mapě. Jako klíč používáme hash aktuální situace, který obsahuje aktuálního hráče, zápis figurek na šachovnici (jejich symbol, souřadnice a otočení) a figurek v sáčcích hráčů. Jako hodnotu používáme uvažovanou hloubku, ohodnocení situace a nejlepší tah. Když se dostaneme s minimaxem do nějaké situace tak nejdříve vytvoříme hash, zjistíme jestli se již nachází v uvažovaných stavech a jestli odpovídá hloubka. Uložená hloubka musí být alespoň rovna aktuální. Hloubka určuje kolik tahů ještě chceme uvažovat. Nechceme použít uloženou hodnotu pokud by to znamenalo zmenšení hloubky výpočtu. Naopak programu pouze prospěje pokud se hloubka zvýší. Vzhledem k tomu, že druhý hráč může mít jinou strategii, je vhodné mít uvažované stavy oddělené. Každý hráč má tedy vlastní strukturu map.

2.6.3 Průběžný výpočet součtu hodnot figurek

Jak je psáno v sekci 2.4, tak součástí evaluační funkce je hodnota figurek. Každý počítačový protivník má vektor hodnot jednotlivých figurek.

Původně program odehrál všechny možné čtveřice tahů a poté v nastalých situacích jednoduše spočítal součet hodnot figurek. Sečetl hodnotu všech svých figurek a odečetl hodnotu figurek protihráče. Tento přístup je neefektivní. Hodnota stejné figurky je nezávisle využita během výpočtu ve více různých situacích. A dokonce některé figurky se mohou vyskytovat ve všech dosažených situacích.

Řešením tohoto problému je předem si spočítat aktuální součet hodnot figurek a poté tento součet průběžně upravovat. Pokud je tedy na tahu hráč, který je ovládaný počítačem, pak nejdříve spočítá součet hodnot figurek a následně spustí algoritmus minimax. Během minimaxu může počítač či jeho protihráč zajmout figurku či přidat novou. Po každé této změně je součet hodnot figurek upraven a tím je jednoduše dosažen součet hodnot figurek pro aktuální situaci. Při rekurzivním volání minimaxu je tedy předáván upravený součet hodnot figurek. Hlavní výhodou je, že při přidání figurky je součet upraven pouze jednou a zmíněnou figurku není potřeba řešit ve všech navazujících situacích.

3. Uživatelská příručka

3.1 Hlavní menu

Práce je složena ze dvou programů. Prvním programem je konzolová aplikace pro hru dvou počítačových protivníků a zlepšování hry počítače. Po jeho spuštění se automaticky začne hrát a to až do doby kdy uživatel program vypne. Po každém tahu se vypíší informace o aktivních i záložních figurkách obou hráčů a aktuální podoba šachovnice. Na prázdných políčkách je symbol ___, na ostatních je písmeno označující figurku. Pokud je písmeno velké, pak je figurka ve startovní pozici. Číslice za písmenem označující figurku značí, kterému hráči daná figurka patří.

Druhý program je aplikace s grafickým uživatelským rozhraním pro hru uživatele proti uživateli nebo uživatele proti počítači. Dále se budu věnovat aplikaci s grafickým uživatelským rozhraním. Po jejím spuštění se zobrazí menu kde má uživatel možnost spustit partii nebo vypnout program.

3.2 Výběr hráčů

Po kliknutí na tlačítko "Play"je hlavní menu schováno. Uživateli je zobrazeno menu, kde lze pro každého hráče nastavit jestli bude ovládán počítačem nebo uživatelem. Aktuální volba je zvýrazněna červeným obdélníkem. Změna volby probíhá kliknutím na symbol uživatele či počítače. Hra je následně spuštěna kliknutím na tlačítko "Start".

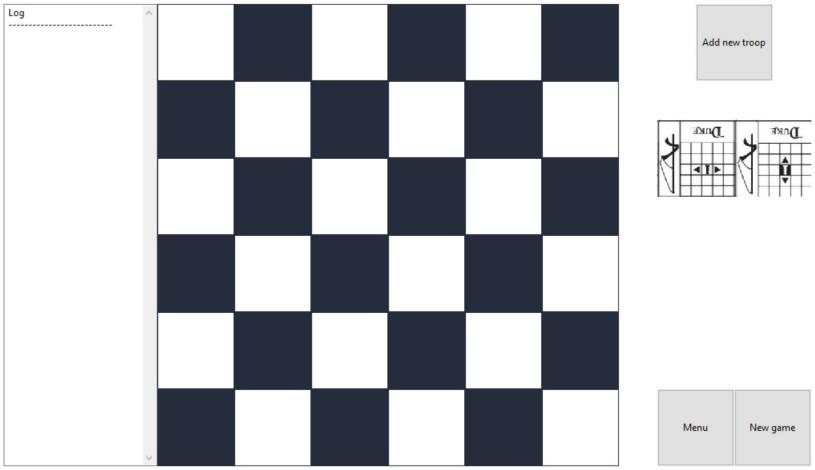
3.3 Hra

Během hry je obrazovka rozdělena na několik částí viz obrázek 3.1. Uprostřed obrazovky se nachází šachovnice, na které se během hry pohybují figurky. Vlevo od šachovnice je textové okno, které slouží pro záznam tahů. Napravo od šachovnice jsou tlačítka "Add new troop", "New game"a "Menu".

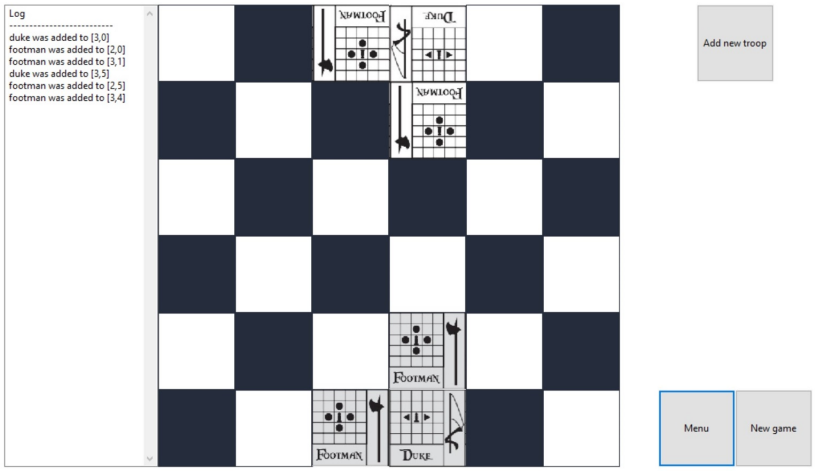
3.3.1 Počáteční umístění figurek

Hra začíná umístěním počátečních figurek. Počítač tyto figurky umístí automaticky. Uživateli jsou zobrazeny obě strany aktuálně přidávané figurky a to pod tlačítkem "Add new troop". Samotné přidávání probíhá kliknutím na políčko šachovnice. Je důležité umístit svého Duka na jedno ze dvou prostředních políček na své straně šachovnice. První hráč hraje za bílé figurky a začíná na horní části hrací desky. Druhý hráč hraje za šedivé figurky a začíná na spodní části hrací desky. Příklad správného umístění počátečních figurek je uveden na obrázku 3.2.

Po přidání počátečních figurek začíná hra. Ve svém tahu má hráč na výběr ze tří možností. Může pohnout s figurkou, použít Command symbol nebo přidat novou figurku.



Obrázek 3.1: Začátek hry. Uprostřed šachovnice, nalevo od ní log, napravo tlačítka pro přidání nové figurky, spuštění nové hry či vrácení do menu. Pod tlačítkem pro přidání nové figurky je nakreslena aktuálně umisťovaná figurka.



Obrázek 3.2: Jeden z možných stavů hry po přidání počátečních figurek.

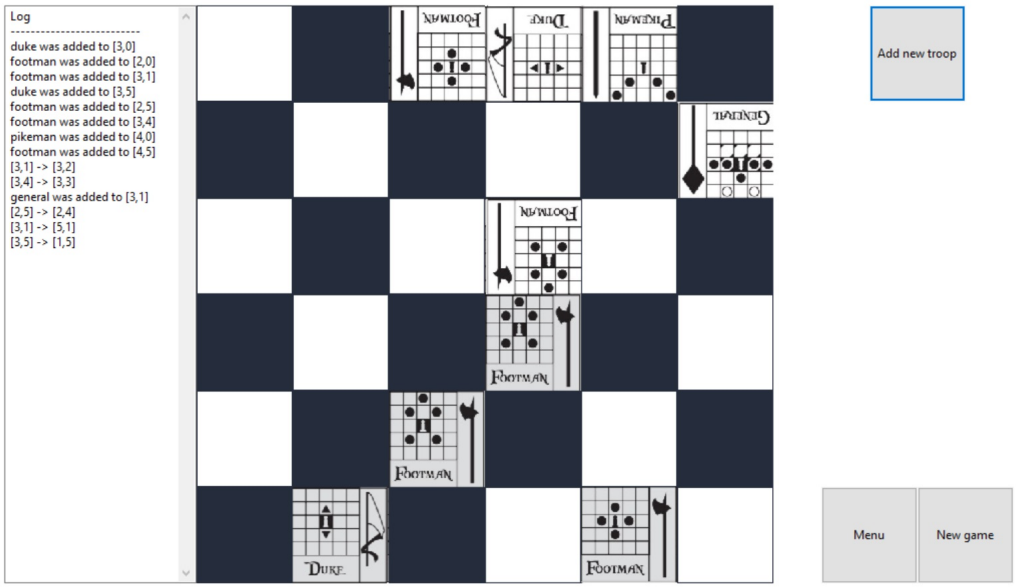
3.3.2 Pohyb s figurkou

Uživatel může s figurkou pohnout pomocí stisknutní levého tlačítka myši. Tím je vybrané startovní políčko. Následným uvolněním tohoto tlačítka je vybráno cílové políčko. Pohyb figurky tedy se skládá ze tří částí. V první části je stisknuto levé tlačítko myši, v druhé části je myš přesunuta nad cílové políčko a ve třetí části je levé tlačítko myši uvolněno. Pokud lze tah provést je provedena aktualizace šachovnice a tah je připsán do logu. Pokud nelze provést je o tom uživatel informován v logu.

3.3.3 Command

Místo pohybu může uživatel použít Command symbol některé ze svých figurek. Command se také skládá ze tří částí. První částí je kliknutí na figurku s Command symbolem. Následuje kliknutí na počáteční políčko a na konec je vyžadováno kliknutí na cílové políčko. Pokud lze tah provést je provedena aktualizace

šachovnice a tah je připsán do logu. Pokud nelze provést je o tom uživatel znovu informován v logu. Princip je ukázán na obrázku 3.3.



Obrázek 3.3: Bílý je na tahu a je ovládán uživatelem. Nyní může být pikeman na souřadnicích [4,0] přesunut do rohu [5,0] pomocí tří kliknutí - nejdříve na generála na souřadnicích [5,1], poté na pikemana a nakonec na cílové políčko. Souřadnice jsou počítány od nuly, zleva doprava a odshora dolů.

3.3.4 Přidání nové figurky

Pro přidání nové figurky je nutné stisknout tlačítko "Add new troop". Po jeho aktivování je vylosována náhodná záložní figurka aktivního hráče a obě její strany jsou zobrazeny pod zmíněným tlačítkem. Po následném kliknutí do šachovnice je otestováno jestli lze figurku přidat a následně přidána. Dané políčko musí být prázdné a sousedit s Dukem aktuálního hráče.

3.3.5 Konec hry

V případě, že jedna strana vyhrála, je o tom uživatel informován. Prohlížení vítězné situace je možné po kliknutí na informační zprávu. Kliknutím na tlačítko "New game"je spuštěna nová hra se stejným nastavením jako předchozí. Pro změnu nastavení je nutné se vrátit do menu pomocí tlačítka "Menu". V menu se také nachází tlačítko "End"pro ukončení programu.

3.3.6 Inspekce figurky

Během hry je možné kliknout na figurku pravým tlačítkem pro její inspekci. Po dobu stisknutí pravého tlačítka je figurka zobrazena z obou stran nad tlačítkami "Menu"a "New game". Pro inspekci je tedy nutné držet tlačítko myší stisknuté.

4. Vývojová dokumentace

4.1 Výběr programovacího jazyka

Pro implementování hry jsem vybral programovací jazyk C++. C++ je efektivní a rychlý jazyk, který má rozsáhlé použití. Lze použít na GUI aplikace ale i na náročné výpočty, což bylo pro implementaci hry The Duke zásadní. Vybral jsem si ho také proto, že se jedná o objektově orientovaný jazyk. Umožňuje vytvářet třídy a používat dědičnost, polymorfismus a enkapsulaci, díky kterým bylo snažší psát spolehlivější a znovupoužitelný kód. Jazyk používá velká aktivní komunita a bylo tedy snadnější najít řešení nastalých problémů.

4.2 Struktura

Vyhledem k rychlosti tahů nebylo vhodné vytvářet grafické uživatelské rozhraní pro hru počítače proti počítači. Byl tedy vytvořen program s grafickým rozhraním pro hru uživatele proti uživateli a uživatele proti počítači. Pro hru počítače proti počítači je použita konzolová aplikace, díky které se hra počítače zlepšuje. Oba programy sdílí stejný backend a líší se pouze uživatelským rozhraním. Sdílený backend je ve složce Common. Změny strategií vzniklé hrou počítače proti počítači jsou automaticky promítnuty při dalším spuštěním grafické aplikace.

Backend má poměrně jednoduché rozdělení na třídy objektů. Hlavní třída game_t reprezentuje samotnou hru. Stará se o správný průběh partie. Dokud hra běží, tak zjistí od aktuálního hráče tah, zkontroluje, že tah lze udělat a poté ho provede. Třída má uloženou šachovnici a to jako dvojrozměrné pole s unikátními ukazateli na figurky.

Další třída je player_t reprezentující hráče. Má přehled o svých balíčcích figurek (packs_t), tedy o aktivních a záložních figurkách. Implementuje metody na úpravu těchto dvou seznamů a díky tomu je game_t může odpovídajícím způsobem měnit.

Dále máme třídy pro jednotlivé figurky, které dědí od společného abstraktního předka. Každá figurka si ukládá tabulku pohybů. Díky tomu při kontrole pohybu není nutné znát název figurky. Stačí zjistit jestli se tah nachází v tabulce možných pohybů figurky, která je zrovna na počátečních souřadnicích. Každý hráč má také označení jestli je ovládán počítačem či uživatelem. Pro případ, že by byl ovládán počítačem má své parametry viz sekce 2.4.

Důležitá třída je i strategy_manager. Ta se stará o ukládání aktuální generace parametrů a jejich evoluci pomocí křížení a mutaci. Evoluce, křížení a mutace jsou popsány v sekci 2.5. V rámci vyvíjení strategií jsou zaznamenávány hry počítače proti počítači a to pomocí třídy logger. Ta se také stará o výpočet průměrného faktoru větvení viz sekce 2.2.

Podrobnější programátorská dokumentace se nachází v přiložené složce Documentation. Tato dokumentace byla vytvořena za použití nástroje doxygen.

Závěr

Cílem této práce bylo implementovat hru The Duke. Výsledný program umožňuje hru dvou uživatelů na stejném počítači či hru uživatele proti počítači. Jedná se o aplikaci s grafickým uživatelským rozhraním, díky čemuž je aplikace uživatelsky příjemná.

Počítačový protivník využívá algoritmus minimax s alfa beta prořezáváním. Narozdíl od šachů hra The Duke dříve nebyla podrobena řádnému zkoumání, kvůli čemu bylo vytváření umělé inteligence složitější. Z tohoto důvodu byl implementován genetický algoritmus s roulette wheel selection, díky kterému se strategie počítače zlepšuje hrou dvou různých verzí proti sobě. Pro tento účel vznikla konzolová aplikace, která umožňuje další vylepšení hry počítače. Pro zdokonalení strategie bylo programem vypočteno sto generací parametrů.

V rámci práce byl také empiricky určen faktor větvení. Výsledná hodnota ukázala, že ve hře The Duke má hráč průměrně 35,7 možných tahů. Tato hra je tedy mírně složitější než šachy, ve kterých je průměrně $30-35$ možných tahů. Dále byl navržen přibližný výpočet velikosti stavového prostoru. Znovu se ukázalo, že The Duke je složitější než šachy, protože má větší stavový prostor.

Vzhledem k vysokému branching factoru a velikému stavovému prostoru trvalo hledání nejlepšího tahu netriviální dobu. Z tohoto důvodu bylo implementováno několik optimalizací neovlivňující výsledek. Mezi ně patří například předpočítávání možných pohybů, alfa beta ořezávání, ukládání již uvažovaných stavů či průběžný výpočet hodnoty figurek. Dále byly vytvořeny optimalizace, které ovlivňují výpočet v přijatelné míře. Mezi tyto optimalizace patří omezení počtu uvažovaných tahů na čtyři tahy dopředu a při přidávání nové figurky prozkoumání pouze několika zástupců. Díky všem těmto optimalizacím byla dosažena svižná hra počítače.

Program by mohl být vylepšen o další optimalizace. Jedna z nich využívá následujících faktů. Program přemýšlí na čtyři tahy dopředu. Poslední tah je zahrán protihráčem. Čtvrtým tahem se naše situace může jenom zhoršit. Pokud je situace po třech tazích horší než na začátku, nemusí dávat smysl zkoumat poslední tahy, protože tato větev minimaxu s velkou pravděpodobností stejně nebude vybrána. Díky tomu můžeme vrátit ohodnocení v menší hloubce a prohledat méně stavů. Tento přístup selhává v případě, že všechny větve minimaxu vedou do horší situace než byla počáteční. V takovém případě víme, že všechny možnosti vedou ke zhoršení situace, ale nerozeznáváme k jak moc velkému zhoršení dojde. Z tohoto důvodu tato optimalizace nebyla implementována.

Seznam použité literatury

Barnes, D. (2019). What is the average number of legal moves per turn? *Chess Stack Exchange*.

Bavelier, D. a Davidson, R. J. (2013). Games to do you good. *Nature*, **494** (7438), 425–426.

Bernstein, A. a de V. Roberts, M. (1958). Computer v. chess-player. *Scientific American*, **198**(6), 96–107.

Granter, S. R., Beck, A. H. a Papke Jr, D. J. (2017). AlphaGo, deep learning, and the future of the human microscopist. *Archives of pathology & laboratory medicine*, **141**(5), 619–621.

Huotari, K. a Hamari, J. (2012). Defining gamification: a service marketing perspective. In *Proceeding of the 16th international academic MindTrek conference*, pages 17–22.

Khotsianovich, A. a Barisevich, A. (2015). Basilar principles of negotiations: game theory.

Laramée, F. D. (2000). Chess programming part iv: Basic search. *GameDev. net*.

Newborn, M. (2012). *Kasparov versus Deep Blue: Computer chess comes of age*. Springer Science & Business Media.

Razali, N. M., Geraghty, J. a kol. (2011). Genetic algorithm performance with different selection strategies in solving tsp. In *Proceedings of the world congress on engineering*, volume 2, pages 1–6. International Association of Engineers Hong Kong.

Rylander, S. G. B. a Gotshall, B. (2002). Optimal population size and the genetic algorithm. *Population*, **100**(400), 900.

A. Přílohy

A.1 Přiložené soubory

Příloha obsahuje následující soubory.

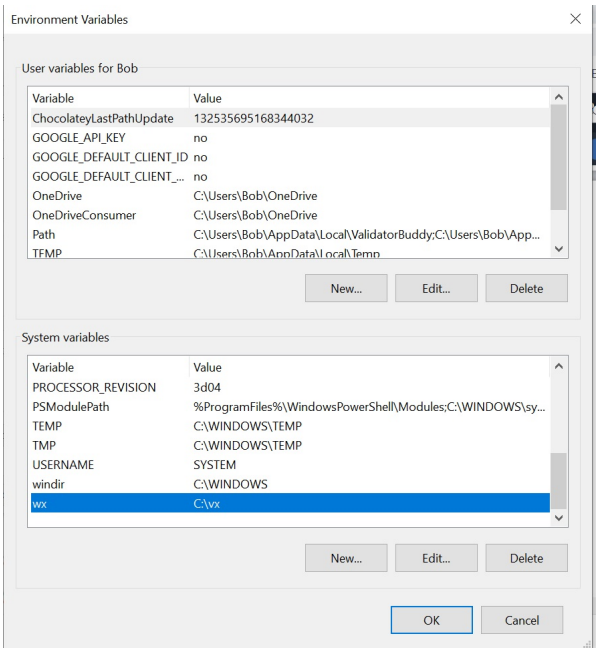
- Složku Documentation s programátorskou dokumentací vytvořenou nástrojem doxygen. Hlavní soubor je index.html .
- Složku Release, která obsahuje konzolovou aplikaci TheDuke_console.exe pro hru počítače proti počítači. Tato složka dále obsahuje soubor TheDuke_GUI.exe neboli aplikaci s grafickým uživatelským rozhraním pro hru uživatele proti uživateli či uživatele proti počítači.
- Složku Common kde se nachází složka s generacemi chromozomů a další složka s logy z jejich vývoje. Dále se zde také nachází zdrojové kódy pro sdílený back-end zmíněných programů.
- Složku TheDuke s zbylými zdrojovými soubory pro konzolovou aplikaci.
- Složku WxWidget s zbylými zdrojovými soubory pro aplikaci s grafickým uživatelským rozhraním.
- Složku Images s potřebnými obrázky jako například grafikou jednotlivých figurek.
- Text této práce vytvořené pomocí latexu ve formátu PDF/A.

A.2 Instalace frameworku WxWidgets

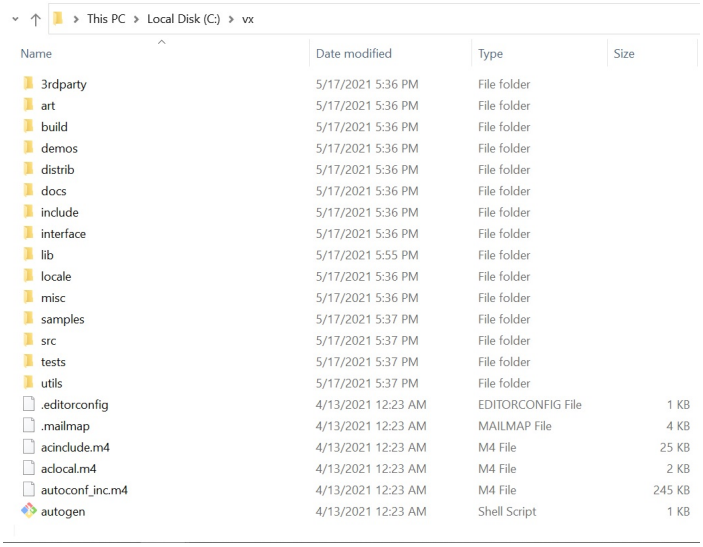
Pro grafické uživatelské rozhraní je používaný framework WxWidgets. Pro běžné používání programu není potřeba nic měnit. Pro změnu kódu a následné spuštění je nutné mít tento framework správně připojený. K tomu je třeba splnit následující kroky.

- Nejdříve je potřeba stáhnout zazipovaný zdrojový kód na

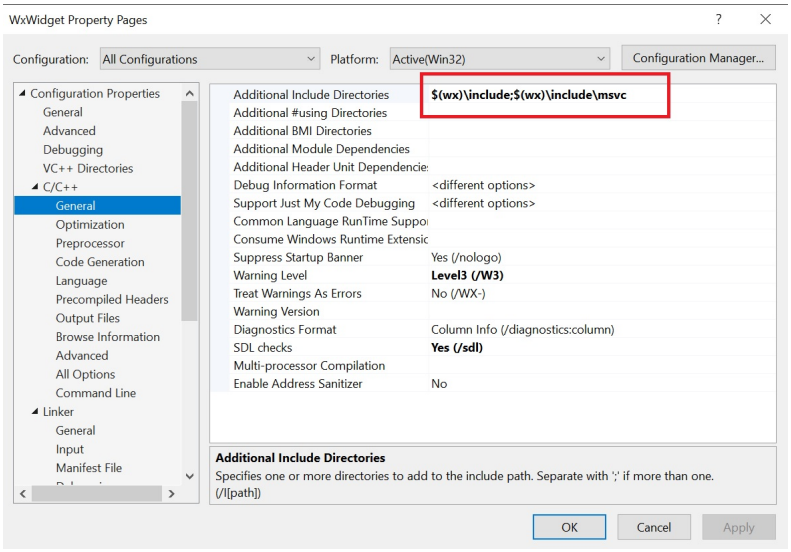
  https://www.wxwidgets.org/downloads/.
- Dalším krokem je framework rozbalit. Po rozbalení se v složce build/msw nachází soubor wx_vc16, na který je potřeba aplikovat batch build. Pokud batch build nahlásil při sestavování chyby, je třeba ho aplikovat znovu.
- Tím jsme dosáhli sestavení frameworku. Dále je důležité nastavit properties projektu, aby věděl, kde má framework hledat. K tomuto účelu lze použít systémovou proměnnou (viz obrázky A.2 a A.1). Je důležité, aby projekt měl odkaz na složky include a include/msvc. Dále je podstatné, aby linker měl odkaz na složku lib/vc_lib. Viz obrázky A.3 a A.4.
- Důležité je také, aby byla vybrána správná procesorová architektura. Program byl vyzkoušený s vybranou možností x86.



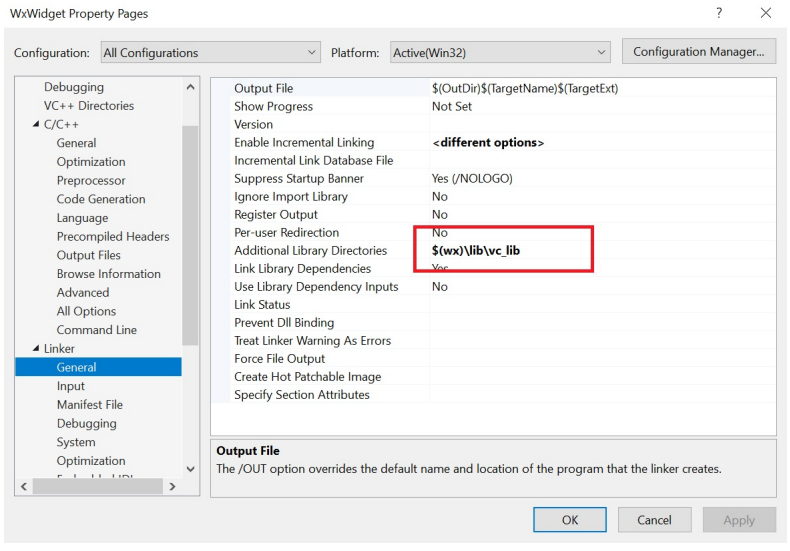
Obrázek A.1: Ukázka podoby systémové proměnné.



Obrázek A.2: Doplnění pro systémovou proměnnou - lokace a podoba složky s frameworkem.



Obrázek A.3: Správné nastavení include v programu visual studio.



Obrázek A.4: Správné nastavení library v programu visual studio.