VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Aplikace teorie her v deskových hrách

Application of Game Theory in Board Games

2016

Miroslav Ježík

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Miroslav Ježík**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Aplikace teorie her v deskových hrách
Application of Game Theory in Board Games

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem diplomové práce je návrh, implementace a testování strategií ve hře pro více hráčů. Jako modelový příklad hry slouží dáma pro tři hráče.

Diplomová práce bude obsahovat:
1. Stručný úvod do teorie her v rozsahu nezbytném pro následný návrh strategií.
2. Stručný popis vybrané modelové hry.
3. Návrh strategií pro modelovou hru.
4. Implementace strategií.
5. Experimenty se simulovanými partiemi, jejich vyhodnocení a rozbor.

Seznam doporučené odborné literatury:

[1] M. J. Osborne: An Introduction to Game Theory, Oxford University Press, 2009
[2] M. Chvoj: Pokročilá teorie her ve světě kolem nás, Grada Publishing, 2013
Dále dle pokynů vedoucího práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Mgr. Jiří Dvorský, Ph.D.**

Datum zadání: 01.09.2015
Datum odevzdání: 29.04.2016

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 29. dubna 2016 .....

Děkuji svému vedoucímu doc. Mgr. Jiřímu Dvorskému, Ph.D. za veškeré rady a připomínky, které mi velmi pomohly při vypracování této práce.

Abstrakt

Práce se zabývá problematikou teorie her a implementací umělých inteligencí s aplikací strategií za účelem zlepšení schopnosti provádět tahy. Zvolená modelová hra je Dáma pro tři hráče. Jednotlivé strategie jsou nejprve rozebrány z pohledu teorie her a jsou také definovány všechny potřebné vzorce ke správné funkci těchto strategií. Následně jsou implementovány algoritmy k výpočtu tahů umělé inteligence za pomocí klasického algoritmu, který prohledává celý stavový prostor a jako druhý je implementován genetický algoritmus. Algoritmy a strategie jsou na závěr srovnávány z pohledu úspěšnosti v simulovaných partiích a také z pohledu rychlosti výpočtů.

Klíčová slova: Genetický algoritmus, umělá inteligence, strategie, teorie her, partie, herní kámen, simulace, kooperace, preference, výplata

Abstract

Master's thesis is focused on game theory and implementation of artificial intelligences, together with application of strategies, that are supposed to improve ability of those artificial intelligences to make proper turns in games. Game, that was chosen here is Checkers for three players. All strategies are first described from perspective of game theory and after there were defined all required formulas, that provide proper functionality of these strategies. There are also two algorithms for calculations of turns implemented. First is standard algorithm, that calculates turns after searching through whole tree of turn sequences and second is genetic algorithm. All these algorithms and strategies are afterwards compared from perspective of success in simulated games and aswell from perspective of speed, in which those turns were calculated.

Key Words: Genetic algorithm, artificial intelligence, strategy, game theory, game, game piece, simulation, cooperation, preference, payoff

Obsah

Seznam použitých zkratek a symbolů

K	–	Kooperace
PK	–	Preference korunovace
PTP	–	Preference tahu pěšákem
PU	–	Preference útoku
PHHK	–	Přepočet hodnoty herního kamene

Seznam obrázků

Seznam tabulek

1 Úvod

Práce je zaměřena na problematiku teorie her, jakožto matematického nástroje k řešení konfliktních situací. Konkrétně zde budou rozebrány vybrané problémy z teorie her, kdy budou zmíněny hry pro dva a více hráčů, kde je v některých případech povolena komunikace mezi hráči, což vede k možnosti jejich kooperace. Dojde tedy k nastudování užitečných oblastí teorie her a následné aplikaci na volbu jednotlivých strategií, které budou vytvářeny formou modulů, které je možno přidat umělé inteligenci za účelem zlepšené schopnosti reagovat na jisté krajní situace, na než standardně implementovaná umělá inteligence neumí reagovat. Zvolená modelová hra se nazývá Dáma, která je také známá pod anglickým názvem Checkers. Dáma jako taková má také mnoho různých variant, které se liší tvarem herního plánu, počtem herních kamenů, či možným pohybem herních kamenů po šachovnici, přičemž se konkrétně zaměříme na variantu, nazývající se „Dáma pro tři hráče“.

Strategie budou nejprve navrženy z teoretického hlediska, kdy dojde k zavedení potřebných vztahů, které ovlivní výpočty výplat umělých inteligencí. Rozdíly v přepočtech povedou k následným rozdílům v reakcích umělých inteligencí, což budeme testovat experimentálním způsobem a budeme zkoumat, jak jednotlivé umělé inteligence, na které byly aplikovány různé přídavně strategie reagovaly a jak se tím zlepšila jejich schopnost porazit své soupeře. První implementovanou formou umělé inteligence bude algoritmus podobný Minimaxu, či Negamaxu, kde probíhá kompletní prohledávání stavových prostorů. Jako další bude implementován genetický algoritmus, který využívá populaci k prohledávání stavového prostoru, kdy tato populace prochází evolucí za účelem zjištění co možná nejlepšího výsledku. Jelikož tyto umělé inteligence pracují na rozdílném principu, bude zde provedeno srovnání z pohledu implementace a budou zde zmíněny také různá implementační úskalí, na která lze při implementaci narazit.

Na závěr proběhne celkové srovnání umělých inteligencí, které bude opět vycházet z dat, které byly získány za pomocí simulací. Srovnání budou prováděna na základě typu algoritmu, využívaného k výpočtům tahů, případně přídavných strategií, které zde byly zahrnuty. Také se budou srovnávat, jak jednotlivé konfigurace parametrů umělých inteligencí, jako například hloubka prohledávání, či případně jiné parametry, specifické pro genetický algoritmus, ovlivnily výslednou schopnost reagovat na tahy soupeřů a také jak byla ovlivněna rychlost počítání těchto tahů. Například speciálně u genetického algoritmu není nastavení jednotlivých parametrů zcela instinktivní, takže není jednoduché zvolit správnou konfiguraci, přičemž toto nastavení může velmi významným způsobem ovlivnit celkovou schopnost provádět rozumné tahy.

2 Teorie her

Teorie her je matematickým nástrojem k řešení jakékoliv konfliktní situace, přičemž není důležité o jakou situaci se jedná. Může tedy jít o souboj dvou zvířat o potravu, či řešení různých problémů v souvislosti s vedením firmy nebo volba strategie v karetní hře. Důležité ovšem je, že se vždy jedná o konflikt, při kterém se vyplácí promyslet strategii, která by vedla pokud možno k nejlepšímu možnému výsledku. Ačkoliv se v různých oblastech můžeme setkat s různými pojmy, tak se vždy pouze jedná o jinou formu pojmů hra, hráči, strategie a výplata. Jako hra je označován problém, pro který jsou dána jasná pravidla. Hru poté hrají hráči, což jsou jednotlivé objekty, kterých se problém týká. Jednotliví hráči poté zvolí strategie, které jsou dle pravidel hry vyhodnoceny a hráčům je poté přiřazena výplata. Hráči se dále mohou dělit jako inteligentní a neinteligentní. Jako inteligentní je označován hráč, který provádí racionální rozhodnutí k dosažení co nejlepšího výsledku za použití všech dostupných informací, přičemž neinteligentní hráč volí strategie zcela náhodně. Pokud se hráč chová jako inteligentní s pravděpodobností p a s pravděpodobností $1 - p$ jako neinteligentní, tak se jedná o p-inteligentního hráče. [1].

Při snaze analyzovat jednotlivé problémy si ale musíme uvědomit, že ačkoliv teorie her může být mocným nástrojem, tak z důvodu komplexnosti některých konfliktů je problematické určit úplnou množinu strategií. V některých případech je velmi obtížné určit všechny hráče, či přesně definovat výplaty, ale hlavním cílem by vždy mělo být vytvoření modelu, který nám dá lepší pohled na danou situaci. Dalším problémem teorie her je, že i když se snažíme racionálně rozmyslet řešení daného problému, tak předpokládáme také racionální rozhodování ostatních hráčů, což ve skutečnosti většinou neplatí. Za zmínku také stojí fakt, že teorie her nemá jasně definovány předpisy, jak se zachovat při hře dvou hráčů, kteří mají naprosto odlišné cíle a nebo případně jakým způsobem reagovat na některé situace ve hře více hráčů. I přesto, že v teorii her narážíme na tyto překážky a nemusí být poskytnuto již z podstaty složitosti problému jednoznačné řešení, tak i v takovém případě lze vždy získat zajímavé výsledky a doporučení, které by bylo výhodné aplikovat [2].

Kategorizace teorie her

Rozdělení je pouze orientační, jelikož obecně není jednotné a různé zdroje popisují toto rozdělení rozdílným způsobem. Zvolené rozdělení vychází z literatury Pokročilá teorie her ve světě kolem nás od Martina Chvoje a budu se na něj ve zbytku práce případně odkazovat [1].

Kooperativní a nekooperativní hry

Hra je kooperativní pokud je hráčům umožněno uzavírat vynutitelné dohody.

Jednokolové a vícekolové hry

V případě jednokolových her hráči volí strategie na základě faktu, že po získání výplaty je hra rovnou ukončena. Při vícekolových hrách musí hráči při volbě strategie uvažovat možnosti několik kol napřed, aby pokud možno maximalizovali jednotlivé výplaty, přičemž pokud se jedná o poslední kolo vícekolové hry, hráči volí strategie jako kdyby se jednalo o jednokolovou hru. Zvláštním případem vícekolové hry je evoluční hra, kdy hráči kopírují strategie úspěšnějších hráčů.

Symetrické a asymetrické hry

V symetrických hrách mají hráči rovnocenné postavení a tudíž volí ze shodné množiny strategií a někdy se také předpokládá, že výplatní funkce je pro všechny hráče stejná.

Hry s nulovým a nenulovým součtem

Hry s nulovým součtem mají vlastnost, že součet všech výplat hráčů je roven nule nebo předem určené konstantě. Při hře s nenulovým součtem není výsledný součet předem určen.

Hry s úplnou (dokonalou) a částečnou informací

Ve hrách s úplnou informací hráč zná všechny průběhy hry. V případe her s částečnou informací hráč nemusí znát celou množinu všech strategií, počet hráčů a možné výplaty. V mnoha hrách s částečnou informací musí tedy hráči vynakládat prostředky k získávání dodatečných informací.

Nekonečně dlouhé hry

Hry, které nemají konečný počet kol a jejich význam je hlavně teoretický.

Konečné, diskrétní a spojité hry

Rozdělení zde spočívá ve velikosti množiny strategií. Pokud je množina strategií konečná, jedná se o konečnou hru. V případě nekonečné spočetné množiny můžeme hovořit o diskrétní hře a pokud je množina strategií nespočetná, tak se hra označuje jako spojitá.

Simultánní hry a metahry (sekvenční hry)

V simultánních hrách všichni hráči hrají současně a nemají tedy aktuální informace o momentálně zvolených strategiích svých protihráčů. Naproti tomu v metahrách hrají hráči postupně a tedy mají informace o volbě strategie soupeře a mohou tedy na ni případně reagovat.

2.1 Hry dvou hráčů

Jak již název napovídá, jedná se o konflikty mezi dvěma objekty, respektive hráči. Ačkoliv není rozdělení jednoznačné, většina problémů se v této kategorii dělí podle toho, zda se jedná o hru s nulovým, či nenulovým součtem, kdy typickým příkladem problému s nenulovým součtem se nazývá Vězňovo dilema [2].

Hry s nulovým součtem

Jak již bylo zmíněno v kategorizaci teorie her, hry s nulovým součtem mají vždy součet výplat každé kombinace strategií roven nule [2]. Nyní si uvedeme následující příklad:

Tabulka 1: Zápis jednotlivých výplat

		Marek	
	Strategie	A	B
Pavel	A	(4, -4)	(-3, 3)
	B	(-6, 6)	(8, -8)

Dvojice hodnot v této tabulce odpovídají výplatám hráčů, přičemž první hodnota z dvojice patří Pavlovi a druhá hodnota Markovi, kdy každý má možnost vybrat si ze dvou strategií, což vede celkově ke čtyřem možným výsledkům. Jak je vidět, součet hodnot jednotlivých dvojic výplat pro hráče je roven vždy nule, proto je také možno použít následující zápis [2]

Tabulka 2: Zjednodušený zápis jednotlivých výplat

		Marek	
	Strategie	A	B
Pavel	A	4	-3
	B	-6	8

V tomto zápisu se Marek samozřejmě snaží vybrat co nejmenší hodnotu, zatímco Pavel chce získat hodnotu nejvyšší. Na první pohled je zřejmé, že hra je trochu nakloněna na stranu Pavla, který má možnost získat vyšší výplatu každé kolo, než Marek a pokud bychom předpokládali, že hra bude mít dostatečný počet kol a hráči by volili strategie rovnoměrně, získá celkově Pavel více bodů. Pokud by při této hře oba hráči volili strategie současně, přičemž by ani jeden z hráčů nevěděl, kterou strategii druhý zvolí, tak lze již intuitivně říci, že Pavel by nejraději získal 8 bodů, kdy tuto příležitost má, pokud zvolí strategii B. Pavel ovšem musí věřit, že Marek nezvolí strategii A, jelikož pro tohoto hráče, je strategie A také možností, jak získat co největší bodové ohodnocení. Ve hrách s nulovým součtem lze určit nejvhodnější kombinaci strategií, pokud nalezneme sedlový bod.

Sedlový bod je taková kombinace strategií hráčů, kdy je hodnota na řádku minimální a současně je hodnota ve sloupci maximální. To znamená, že pokud by kterýkoli hráč hrál danou strategii, která vede do sedlového bodu, tak je vždy současně nejlepší odpověď protihráče taková,

která také vede do sedlového bodu, jelikož volbou kterékoli jiné strategie by si tento protihráč nemohl pomoci. Mimo jiné sedlový bod nemusí v dané hře vždy existovat, či případně může být sedlových bodů více [7].

V původním příkladu sedlový bod není, a tak by se dalo říci, že záleží na intuici jednotlivých hráčů, jinak řečeno, že se oba hráči snaží předpovědět tah toho druhého a nelze s jistotou předem říci, zda bude zvolená strategie správná. Lze ještě na příkladu ukázat, že Pavel má teoreticky vyšší šancí na vítězství, čím více kol bude probíhat, jelikož má možnost získat průměrně více bodů za kolo, než Marek. Kdyby hra byla pouze jednokolová, tak je jasné, že mají oba hráči stejnou šanci na úspěch a dalo by se hru považovat za férovou. Také by se dalo ukázat, že pokud by oba hráči předem znali konečný počet kol, mohli by strategie také volit na základě již získaných výplat. Za předpokladu, že by hráči měli společné body, přičemž záporná hodnota na konci hry by znamenala výhru pro Marka a kladná hodnota výhru pro Pavla a do konce hry by zbývaly tři kola, tak pokud by hodnota skóre byla více, než devět a volil by Pavel do konce hry strategii A, tak již předem tuto hru nelze prohrát. Pro Marka by mohlo platit to samé, ovšem pro celkové skóre by muselo platit, že hodnota by byla menší, než dvanáct.

V případě předem neurčeného počtu kol si hráči nemohou maximalizovat své zisky pomocí volby ryzích strategií, ale místo toho musí volit smíšené strategie, kdy jednotlivé strategie volíme s určitou pravděpodobností. Hráč si tedy v takovém případě volí pravděpodobnostní rozdělení, které mu zajistí maximální možný zisk pro případ, že by protihráč zvolil také ideální pravděpodobnostní rozdělení [7].

Jinými slovy, pokud by jeden z hráčů hrál ideální smíšenou strategii, tak si zaručí určitý zisk pro případ, že soupeř bude také reagovat ideální smíšenou strategií. V případě, že by jeden z hráčů zvolil jako odpověď jiné pravděpodobnostní rozdělení, tak si tím může jedině uškodit [7].

Jako další příklad bude uvedena hra mezi Rose a Colinem, kdy matice výplat je určena následujícím způsobem:

Tabulka 3: Matice výplat mezi Colinem a Rose [2]

		Colin			
	Strategie	A	B	C	D
Rose	A	12	-1	1	0
	B	5	1	7	-20
	C	3	2	4	3
	D	-16	0	0	16

Jako v minulém příkladu, se Rose snaží vybírat nejvyšší hodnoty, zatímco Colin vybírá nejnižší. Při předpokladu, že hráči opět volí strategie současně, tak v případě Rose by bylo rozhodně nejlepší získat hodnotu 16, zatímco Colin usiluje o získání výplaty -20. Pokud se ovšem podíváme pozorně, zjistíme, že pokud Rose zvolí strategii D za účelem možného získání výplaty s hodnotou 16, může se v dalších dvou případech stát, že také nezíská nic a nebo může

dokonce stejnou hodnotu ztratit. Jako v minulém případě je zde vidět, že hra není opět zcela férová, ačkoliv v minulém případě měl Marek nějakou šanci zvítězit. Zde je jasné, že pokud Rose bude stále volit strategii C, tak v každém kole musí nutně získat alespoň dva body a jelikož je tato hodnota také maximem ve sloupci, tak se jedná o sedlový bod. Pokud nyní pomineme skutečnost, že kombinace strategií C a B vede do sedlového bodu, tak se dá s jistotou říci, že by Colin neměl za žádných okolností zvolit strategii C, jelikož když se pozorně podíváme, tak strategie B je striktně lepší než strategie C. V takovém případě říkáme, že strategie B *dominuje* strategii C jelikož ať už Rose zvolí jakoukoli strategii, tak při volbě Colinově strategie B je výplata vždy lepší, nebo alespoň stejná. Pokud se tedy hráč při volbě strategií rozhoduje racionálně, tak by v pozici Colina nikdy strategii C nezvolil [2].

Hry s nenulovým součtem

V případě her s nenulovým součtem, jak již název napovídá, není součet výplat hráčů roven nule. Je tedy nutné pro popsání jednotlivých výplat definovat hodnoty těchto výplat pro oba hráče. Jelikož součet výplat není roven nule, tak tyto hodnoty mohou být libovolné a cíle jednotlivých hráčů mohou tedy být různé. Hráči poté volí strategii, aby dosáhli čistě svého cíle, což se nemusí nutně zcela lišit od cíle druhého hráče. Díky tomu může u těchto typů her docházet k jisté kooperaci, pokud by byla tato kooperace výhodná pro oba hráče. Tato kooperace by ovšem vyžadovala jistou komunikaci mezi hráči, což záleží na typu zkoumané hry. Pokud hráči volí strategie současně, přičemž ke komunikaci mezi nimi nedochází, tak se volba strategií podobá minulým příkladům u her s nulovým součtem [2].

Tabulka 4: Příklad hry s nenulovým součtem [2]

		Colin	
	Strategie	A	B
Rose	A	(2, 3)	(3, 2)
	B	(1, 0)	(0, 1)

V tomto příkladu je pro Rose lepší zvolit strategii A z důvodu, že v každém případě při volbě strategie A získá vyšší výplatu, než při volbě strategie B (volba strategie A je opět dominantní nad volbou strategie B). Pokud by tento postup Colin u Rose předpokládal, zvolí také strategii A. Při této volbě strategií situace dává výhodu Colinovi, jelikož získá tři body, zatímco Rose pouze 2. I v tomto případě se ale dá hovořit o rovnovážném výsledku, a je tedy pro hráče výhodné dané strategie použít [2].

Vězňovo dilema

Typickým příkladem hry dvou hráčů s nenulovým součtem je také Vězňovo dilema. Tento problém je jedním z nejznámějších problémů teorie her. Jedná se o model konfliktu, kdy jsou dva lidé zadrženi za určitý zločin a poté jsou vyslýcháni. Pokud se při výslechu oba odmítnou při-

znat, tak stráví oba tři roky ve vězení. Jestli se naopak oba přiznají, tak stráví pět let ve vězení. Poslední možností je se přiznat a zároveň udat svého komplice, zatímco se on snaží zapírat. V takovém případě dostane tento komplic deset let vězení a vězni co ho udal je trest snížen pouze na jeden rok. Tyto možnosti lze také zapsat následující maticí [2]:

Tabulka 5: Vězňovo dilema [1]

		Vězeň 2	
	Strategie	Zapírat	Přiznat se
Vězeň 1	Zapírat	(-3, -3)	(-10, -1)
	Přiznat se	(-1, -10)	(-5, -5)

Základním předpokladem tohoto problému je, že mezi sebou vězni nemohou nijak komunikovat, takže nikdo neví, jak se ten druhý rozhodl. Je zřejmé, že pokud by byli vězni vzájemně loajální, tak by pro ně bylo výhodné zapírat. Problém je, že je zde velice lákavá nabídka se přiznat a druhého vězně udat, čímž je trest daného vězně snížen na jediný rok ve vězení. V případě jednokolové hry možnost „přiznat se“ dominuje nad možností zapírat, takže se rovnovážný bod nachází v kombinaci rozhodnutí, kdy se oba vězni přiznají. V případě vícekolové hry, kdy se mohou vězni rozhodovat v závislosti na informaci o minulých kolech existuje více možných způsobů, jak postupovat. V roce 1981 byl Robertem Axelrodem uspořádán turnaj, kdy měly být srovnávány možné algoritmy rozhodování. Jako příklad může být uveden algoritmus, který zapírá, dokud se komplic v jednom kole nepřizná a poté se už jen přiznává. V jiném případě může algoritmus kopírovat určitou sekvenci rozhodnutí, nebo se může rozhodovat náhodně, či případně může kopírovat předchozí tahy druhého vězně [1].

Kooperativní hry

Podmnožinou her s nenulovým součtem jsou také kooperativní hry, kdy jsou hráči schopni mezi sebou komunikovat a uzavírat jisté dohody. Oproti nekooperativním hrám, kde by hráč musel věřit protihráči, že neporuší dohodu, tak v kooperativních hrách existuje mechanismus, který ho donutí tuto dohodu dodržet. Tyto hry se následně dělí na hry s přenosnou a nepřenosnou výhrou, kdy u her s přenosnou výhrou dovolíme při vytvoření dohody přesunout část výplaty druhému hráči, který na danou dohodu přistoupil. V takovém případě se vždy vyhledává nejlepší možný součet výplat, který je poté přerozdělen mezi hráče v závislosti na vznesených hrozbách, které by hráči využili, pokud by se nedomluvili na dohodě [7].

Uvažujme tedy následující příklad:

Tabulka 6: Příklad kooperativní hry s přenosnou výhrou [7]

		Hráč 2	
	Strategie	A	B
Hráč 1	A	(5, 3)	(0, -4)
	B	(0, 0)	(3, 6)

Jelikož jsme schopni libovolně přenášet výplatu, tak je logickým řešením zvolit kombinaci strategií, při které je součet výplat hráčů nejvyšší. V tomto případě se jedná o kombinaci strategií v pravém dolním rohu. Následnou otázkou je, jak přerozdělit dané výplaty. Dalo by se říci, že by bylo fér, kdyby oba hráči získali právě polovinu, tedy $4\frac{1}{2}$. Na to ovšem může první hráč pohrozit, že pokud nezíská výplatu minimálně 5, tak bude hrát strategii A. Bohužel pro druhého hráče hrozba volby strategie B není možná, jelikož by ho to poškodilo daleko více (-4), než hráče prvního (0). Při tomto modelu domluvy mohou tedy hráči vzájemně vznášet své hrozby, dokud nedojde ke vzájemné dohodě, na kterou oba přistoupí [7].

2.2 Hry více hráčů

Jedná se o hry, které jsou také nazývány jako hry N-hráčů, kde spadají všechny konflikty, ve kterých jsou alespoň tři hráči. Pokud bychom chtěli zobrazit hru tří hráčů, kdy každý hráč má možnost vybrat si z dvou možností, tak by celkový počet různých situací byl $2^3 = 8$. Obecně je v těchto hrách tedy n^p možných situací, kde n je počet možností každého hráče a p je počet hráčů. V případě tří hráčů by se tedy při zobrazení jednalo o třírozměrné pole. Stejně jako v hrách dvou hráčů s nulovým součtem zde existují rovnovážné výsledky, ke kterým by strategie jednotlivých hráčů měly směřovat, pokud daní hráči volí své strategie logicky. Rozdíl je v tom, že rovnovážných výsledků může být více a každý hráč může preferovat jiný. Pokud se tedy každý hráč pokusí volbou strategie směřovat ke svému nejvýhodnějšímu rovnovážnému výsledku, tak nakonec může dojít k situaci, kdy výsledná výplata nemusí být rovnovážná. V takovém případě opět nemusí být zcela jednoznačné, jakou strategii by měl hráč zvolit. Dalším problémem je možnost kooperace, při které počítáme výplatu koalice jako výplatu jednotlivce. Pokud bychom brali v potaz hru tří hráčů, tak takovou hru lze opět popsat klasickou dvourozměrnou maticí. Zkoumáme-li takovouto hru, je nám jasné, že hráči, kteří tvoří koalici se snaží využít svého spojenectví, aby co nejvíce poškodili třetího hráče. V takové situaci musí třetí hráč volit tzv. obezřetnou strategii, která zajistí co nejlepší možný výsledek proti dané koalici hráčů [2].

Truel

Jedná se o speciální situaci, kdy je na rozdíl od duelu do konfliktu zapojen třetí hráč. Jako známý příklad truelu je uváděn souboj tří střelců A, B a C, kteří mají rozdílnou šanci na zásah, kdy A je nejlepší střelec a C nejhorší, přičemž pořadí, ve kterém střelci střílejí je C, B, A. Zároveň mají také střelci nekonečný počet nábojů. Při zahájení souboje je pro střelce C nejvýhodnější vystřelit do vzduchu, jelikož pokud bude na tahu B, bude se chtít rozhodne zbavit nejsilnějšího soupeře A. Pokud střelec B mine, bude střelec A opět mířit na největší hrozbu, kterou je střelec B. V takovém případě má střelec C největší šanci zůstat naživu. Kromě tohoto případu se dá také uvažovat nad různými obměnami s různými pravidly. Pokud by se předpokládalo, že střelci vždy zasáhnou, přičemž mají pouze jednu ránu, bylo by rozhodování střelců stejné, až na skutečnost, že by zůstali dva živí střelci. Také je možné uvažovat nad případem, kdy má každý střelec opět

rozdílnou šanci na zásah, ale není mu povoleno střílet do vzduchu. V takovém případě musí střelec vždy uvážit své šance na přežití, pokud vystřelí na určitého soupeře a zvolit si lepší možnost. V tomto případě bude první střelec vždy mířit na silnějšího soupeře, s kterým by poté měl složitější souboj v duelu. Pokud mine, jeho situace se nemění, ale pokud zasáhne, tak zůstane v duelu se slabším střelcem. V případě možnosti uzavírání spojenectví mezi střelci je možné uvést případ, kdy je pořadí střelců A, B, C a pravděpodobnost zásahu je $P(a) > P(b) > P(c)$, přičemž mají všichni střelci vysokou šanci na zásah (větší, než 60%). Pokud střílí nejprve A, je pro něj nejlepší vystřelit na druhého nejsilnějšího střelce, což je B. Pokud zasáhne, má C vysokou šanci na přežití. Pokud A mine, střelec B bude střílet na A, přičemž má střelec C stále ze všech střelců největší šanci na přežití, zatímco pravděpodobnost na přežití střelce B je nejmenší. V takovém případě je pro střelce A a B výhodné uzavřít spojenectví a střílet nejprve na C, čímž se šance A a B na přežití zlepší [5].

3 Dáma pro tři hráče

Jedná se o verzi dámy, které budu věnovat největší pozornost, jelikož jsem si ji zvolil k implementaci v programovacím jazyce C#. Důvodem volby je hlavně netradičnost, ať už v počtu hráčů, tak také ve tvaru herního plánu, který je pro standardní dámu netypický. Následující přesné znění pravidel, které se skládá z pravidel české dámy a dámy pro tři hráče, jsem opět převzal z knihy „Velká kniha deskových her od Miloše Zapletala“ [3].

Pravidla

Počet hráčů: 3.

Potřeby: Herní plán má tvar rovnostranného trojúhelníku a je rozdělen na 144 trojúhelníkových polí. Z toho je 78 polí tmavých, 66 bílých. Hracích kamenů je 21 bílých, 21 černých a 21 červených.

Výchozí situace: Hráči sedí tak, aby měli před sebou jeden roh herního plánu. Kameny jsou rozestaveny na tmavých polích, v každém rohu jiná barva (viz obrázek 1).

Úkol hráčů: Vyřadit co nejvíc kamenů obou soupeřů ze hry.

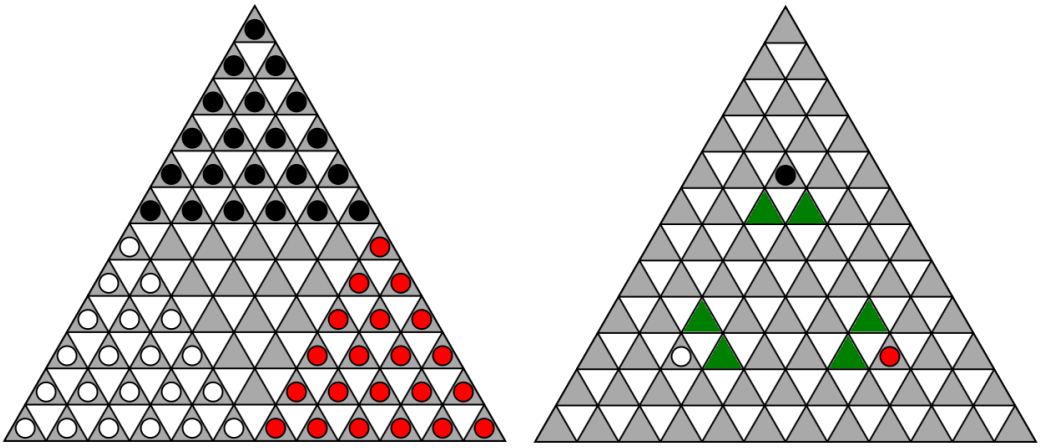
Tahy: Začíná bílý, druhý je na řadě černý, třetí červený. Dál se pravidelně střídají v tomto pořadí.

Pohyb kamenů: Všechny kameny postupují po tmavých polích úhlopříčně vpřed. Dovoleny jsou posuny o jedno pole nebo přes pole obsazené soupeřovým kamenem.

Přeskoky: Skákání je povinné. Kdo úmyslně nebo přehlédnutím poruší toto pravidlo a místo přeskoku jen posune některý svůj kámen, může být na toto opomenutí upozorněn a soupeř má právo mu vzít kámen, který pravidlo o povinném skákání nerespektoval. Obvykle se přitom vyslovuje formule: "Zapomněl jsi skákat!"Hráči musí provést možný vícenásobný přeskok v plném rozsahu. Jestliže ho nedokončí (například místo možných tří přeskoků udělají jen dva), může jim soupeř odebrat kámen, který se provinil proti tomuto pravidlu.

Povyšování: Kámen, který dojde do dlouhé řady polí na opačném konci herního plánu než je roh, z něhož vyšel, mění se v dámu a má její obvyklá práva.

Pohyb a boj dámy: I dáma se pohybuje jen po tmavých polích, ale na rozdíl od prostých kamenů má právo postupovat všemi směry. Dáma se smí přemístit v jednom úhlopříčném směru přes jakýkoli počet volných polí a zastavit se na kterémkoli z nich. Stojí-li dámě v cestě kámen



Obrázek 1: Výchozí pozice herních kamenů a ukázka tahů pro jednotlivé hráče

nebo dáma jiné barvy, za nimiž je alespoň jedno pole volné, musí je přeskočit. Přeskočený kámen je ihned odklizen z desky. Tento skok lze provést přes jakýkoli počet volných polí. Ani dáma však nesmí přejít přes kámen stejné barvy, ten je pro ni nepřekonatelnou překážkou. Má-li dáma možnost přeskočit víc než jeden kámen, je povinna provést skok v plném rozsahu. Mezi přeskakovanými kameny musí být vždy alespoň jedno pole volné. Dva kameny stojící za sebou ani dáma nemůže přeskočit. Zato má právo po každém dopadu změnit směr pohybu. Povinné skákání platí i pro dámu. Porušení tohoto pravidla se trestá stejně jako u obyčejných kamenů. Dáma, která skok vůbec neprovede nebo ho neuskuteční v plném rozsahu, je soupeřem odklizena z desky.

Zakončení hry: Když jeden z hráčů ztratí všechny kameny, ostatní dva pokračují v boji do konečného rozhodnutí.

3.1 Strategie v dámě pro tři hráče

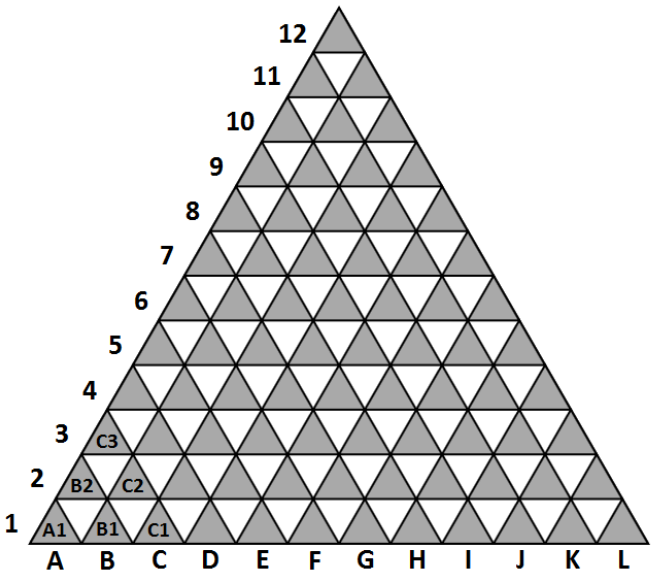
Jak je vidět, dáma pro tři hráče je specifická nejen tvarem herního plánu, pohybem herních kamenů, či jejich počtem, ale také faktem, že jsou zde tři hráči. Tato skutečnost otevírá nové možnosti, co se týče volby strategií, jelikož musíme předpokládat, že proběhnou dva tahy, namísto jednoho po tom, co provedeme tah vlastní. Také musíme brát v potaz to, že každý hráč se snaží získat vítězství sám pro sebe a nespoléhají na druhého. Opravdu ale tedy nemůže mezi hráči existovat nějaký druh kooperace? Přes to, že se každý snaží zvítězit, tak musíme počítat se situacemi, kdy je pro oba hráče výhodné uzavřít dočasné spojenectví, i kdyby to mělo trvat pouze jeden tah. Mimo jiné se tato skutečnost také promítá také v následné tvorbě umělé inteligence, kdy nelze využít klasické algoritmy a implementace musí být vytvořena jiným, specifickým způsobem.

Ačkoliv v implementační části budeme využívat konkrétní hodnoty pro jednotlivé události, tak zde pro rozbor strategií budeme předpokládat hodnotu herního kamene $\alpha \in \mathbb{N}$, hodnotu královny $\beta \in \mathbb{N}$ a hodnotu tahu, při kterém dochází ke korunovaci jako $\kappa \in \mathbb{N}$. Následně si definujeme množinu herních kamenů P, kdy dolními indexy budou označovány typy herních kamenů, resp. P_p jako množinu pěšáků, množinu královen P_q a množinu vyřazených herních kamenů P_d. Horními indexy budou dále označování konkrétní hráči (P^w, P^b, P^r) nebo případně libovolný hráč P^x. Počáteční stav všech hráčů na začátku hry definujeme jako $|P_p| = 21, |P_q| = 0, |P_d| = 0$ a také musí během hry pro každého hráče logicky platit $|P_p| + |P_q| + |P_d| = 21$. Množinu herních polí na šachovnici označíme F, kdy platí, že $|F| = 78$ a také budeme rozlišovat herní pozice, kde je možné udělat korunovaci pro jednotlivé hráče (F_c^w, F_c^b, F_c^r), kdy $|F_c| = 12$. Tahy budeme následně definovat jako uspořádanou trojici $t = \{(p, f, d)|p \in P^x, f \in F, d \subset P, \forall x \in d|x \notin P^x\}$, kdy se jedná vždy o herní kámen, který provedl tah, pozice na které skončil a množina eliminovaných herních kamenů. Tyto tahy budeme následně řetězit do sekvencí určených vektorem $\vec{t_x} = (t_1^x, t_2^x, ..., t_n^x)$, přičemž posloupnost tahů určených jednotlivým hráčům budeme určovat dolním indexem $(\vec{t_w}, \vec{t_b}, \vec{t_r})$ a prvky uspořádaných trojic následujícím způsobem $t_n^x = (p_n^x, f_n^x, d_n^x)$. Výplatu hráčů definujeme jako $V^x \in \mathbb{N}$, kdy horním indexem budeme opět značit jednotlivé hráče, zatímco dolní index bude využíván v případě posloupnosti několika tahů, kdy budeme potřebovat výplatu pro jednotlivé tahy. V případě vyjádření výplat všech hráčů budeme využívat zápis uspořádané trojice $V = (V^w, V^b, V^r)$.

Pokud budou zmíněny určitá pole herního plánu, tak budeme využívat číslování, které je naznačeno na obr. 2. Jak je z nákresu vidět, nebudeme vůbec předpokládat bílá pole na šachovnici, kde se herní kameny ve hře dostat nemohou a budeme číslovat pouze ty, které jsou hodnoceny jako možné pozice herních kamenů.

Přepočet hodnoty herního kamene

Ačkoliv je hodnota herního kamene α a β, tak stále musíme předpokládat, že pokud při získání soupeřova herního kamene ztratíme vlastní herní kámen, tak se oproti klasické variantě dámy



Obrázek 2: Číslování herního plánu

nejedná o rovnocennou výměnu, ale ve skutečnosti z toho získá vždy třetí hráč, který neměl ztrátu žádnou. Reálně je ve výchozí pozici dvojnásobný počet herních kamenů soupeřů oproti vlastním herním kamenům, takže musíme předpokládat, že pokud na začátku ztratíme herní kámen, tak v závislosti na typu herního kamene ztratíme α nebo β, kdežto pokud získáme herní kámen soupeře, tak například za pěšáka získáme pouze hodnotu $\frac{\alpha}{2}$. I z logického hlediska dává smysl, že vyrovnaná situace nastane, pokud v sérii tahů ztratí všichni hráči právě stejný počet herních kamenů stejného typu.

I když můžeme při získání soupeřova herního kamene stále uvažovat poloviční hodnotu, je lepší se na problém podívat z pohledu aktuální situace, jelikož pokud jsme v přesile, tak si můžeme dovolit více obětovat herní kameny k rychlejší porážce soupeře. Při aplikaci strategie by se tedy výpočet výplaty za získaný herní kámen měl počítat jako (1), kde jsou počty herních kamenů aktuálního hráče označeny indexem $x1$, přičemž je vzorec uveden pro případ s pěšákem, kdy v případě královny změníme pouze hodnotu herního kamene.

$$V = \frac{\alpha \cdot (|P_p^{x1}| + |P_q^{x1}|)}{|P_p^{x2}| + |P_q^{x2}| + |P_p^{x3}| + |P_q^{x3}|} \tag{1}$$

V následujícím textu budeme ovšem pro zjednodušení uvažovat výchozí výplatu za získání soupeřova herního kamene jako poloviční.

Preference korunovace herního kamene

Samozřejmě, i v klasické dámě se hráči snaží maximalizovat zisk královen, ačkoliv je nutné zdůraznit výhodu hráče, který jako první získá královnu. Již v klasické dámě je to velká výhoda, avšak v této variantě je dáma schopna se pohybovat neomezeně celkem čtyřmi směry, kdy jsou

tyto směry závislé na barvě herního kamene. Je ovšem možné tuto královnu následně využít k blokování hran herního plánu k zabránění korunovace soupeřových herních kamenů, takže stejně tak, jak je výhodné co nejdříve získat dámu, tak je také nutné pokud možno zabránit ostatním hráčům její získání. Z pohledu hodnoty herního kamene se při korunovaci v principu provádí navýšení této hodnoty z hodnoty α na hodnotu β. Proto, ať jsou tyto konstanty zvoleny libovolně, mělo by platit, že $\alpha + \kappa = \beta$.

Kontrola hrany herního plánu

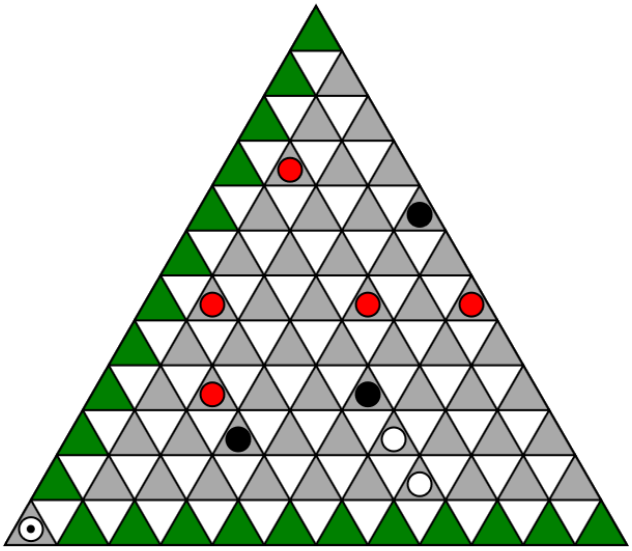
Jelikož se v této variantě dámy využívá trojúhelníkový herní plán a herní kameny se snaží pohybovat vpřed celkem třemi směry, existují celkem tři konečné hrany, kde dochází ke korunovaci jednotlivých hráčů. Je nutno si uvědomit, že po korunovaci herního kamene se může královna následně neomezeně pohybovat po dvou hranách herního plánu, na kterých soupeři provádějí své korunovace, takže stejně jako v klasické dámě máme nejdelší diagonálu, tak zde jsou nejdelšími cestami tyto hrany. Pokud tedy získáme královnu, tak je možné ji využít k možnosti blokování soupeře, aby nebyl schopen korunovat svého pěšáka, kdy pokud se pokusí o korunovaci, tak dojde v dalším tahu o její okamžitou eliminaci. Pokud bychom navíc chtěli využít maximální potenciál této strategie, zvolíme roh herního plánu, kde má královna možnost sledovat hrany, kde dochází ke korunovaci obou soupeřů.

Přes to, že je výhodné kontrolovat právě hrany herního plánu, není nutně potřebné se s královnou pohybovat pouze po hranách, či ji ihned po korunovaci umístit do rohu, kde by měla výhled na obě hrany, kde dochází ke korunovacím soupeřů, jelikož v tu chvíli nemusí hrozit bezprostřední nebezpečí korunovace. Předpokládejme tedy, že hrajeme za bílého hráče a budeme strategii využívat proti černému hráči. V tom případě označíme sekvenci tahů černého hráče $\vec{t_b} = (t_1^b, t_2^b, ..., t_n^b)$, a bílého hráče $\vec{t_w} = (t_1^w, t_2^w, ..., t_m^w)$. Pokud následně platí předpoklad (2), který říká, že má černý hráč existující posloupnost tahů, která mu v n tazích zajistí korunovaci a zároveň neexistuje posloupnost tahů pro nás jakožto hráče, kdy bychom byli schopni této korunovaci zabránit eliminací daného herního kamene. Pokud tedy platí tato podmínka a platí vztah (3), který říká, že pokud existuje posloupnost tahů soupeře, který mu v n tazích zajistí korunovaci, a existuje způsob, jak v $n-1$ tazích dostat královnu na jedno z polí, na kterých dochází ke korunovaci daného hráče, tak je vhodné tuto strategii využít.

$$\exists \vec{t_b} \nexists \vec{t_w} | t_n^b \in \vec{t_b}, t_m^w \in \vec{t_w}, f_n^b \in F_c^b, f_m^w \notin F_c^b, p_n^b \in P_p^b, p_n^b \notin d_m^w, m < n \tag{2}$$

$$\exists \vec{t_b} \exists \vec{t_w} | t_n^b \in \vec{t_b}, f_n^b \in F_c^b, p_n^b \in P_p^b \implies t_m^w \in \vec{t_w}, f_m^w \in F_c^b, p_m^w \in P_q^w, m = n - 1 \tag{3}$$

Důvod k tomuto je takový, že se snažíme pokud možno maximálně využít potenciál královny eliminací herních kamenů po herním plánu a využíváme tuto kontrolu hrany až v případě, kdy hrozí korunovace. Ačkoliv jsme předpokládali nyní pouze dva hráče, tak strategie funguje obdobně i pro případ třetího, kdy pokud by hrozila případná korunovace od obou těchto hráčů,



Obrázek 3: Kontrola hrany herního plánu

tak v konkrétním případě je vhodné zvolit pozici f v tahu t jako $f \in F_c^b \cap F_c^r$, což je právě jedna rohová pozice v herním plánu.

Ačkoliv je kontrolování této hrany za účelem zamezení možnosti korunovace důležité, tak k implementaci tato strategie nebude využita, jelikož to již vychází ze základní implementace umělé inteligence jako takové. Pokud tedy umělá inteligence nedokáže zamezit postupu soupeřova herního kamene na pole, kdy by dalším tahem získal královnu, tak se na danou hranu přesune, jelikož je to po prozkoumání aktuálního stavového prostoru jediná možnost, jak dané korunovaci zabránit. Tato skutečnost je také závislá na hloubce prohledávání tohoto stavového prostoru, kdy v první řadě záleží na tom, zda dokáže dostatečně rychle předpovědět soupeřův postup a v pravou chvíli reagovat s přesunem královny na požadované pole.

Preference tahu herním kamenem

V tomto případě je herním kamenem nazýván tzv. „pěšák“, který ještě neprošel korunovací. Z teoretického hlediska by tato strategie nebyla vůbec potřebná, kdyby umělá inteligence dokázala předpovídat situace 11 tahů dopředu. V takové situaci bychom se ovšem ocitli v hloubce 33, co se týče prohledávání stavového prostoru, takže pokud bychom předpokládali, že průměrné větvení tahů bude 10, tak bychom museli projít 10^{33} možností, což je z hlediska časové náročnosti nepřijatelné. Navíc pokud by se na šachovnici vyskytovaly královny, tak každá královna má v nejlepším případě až 22 možných pozic, kam se přesunout, takže by se výsledný počet možnosti opět ještě navýšil.

Pokud by tedy umělá inteligence předpovídala dva tahy předem, tak tedy máme hloubku prohledávání 6, kdy klasický algoritmus musí projít řádově 10^6 až 10^9 pozic v závislosti na aktuálním větvení tahů. I při této složitosti problému se bude s pěšákem posouvat ke korunovaci

pouze v případě, že od ní maximálně dva tahy vzdálen. V ostatních případech volí náhodně jeden z nejlepších možných tahů, takže se stává, že pokud má umělá inteligence na výběr, zda táhnout s pěšákem, či s královnou, tak se často rozhodne pro královnu, jelikož z celkové množiny tahů mívá největší počet možných tahů právě královna. Ve výsledku tedy umělá inteligence přesune královnu na jinou bezpečnou pozici, místo pokusu o získání další královny. Popsaná situace se ovšem vztahuje pouze na případy, kdy v následujících dvou tazích není možné provést žádný přeskok, aniž by při tom nedošlo k významným ztrátám.

Při implementaci strategie tedy tahům pěšáků přidáváme hodnotu $v \in \mathbb{N} | v << \alpha$. V zásadě zde ale nesmí docházet k situaci, kdy by tato preference narušila provedení tahu, který je spojen například s přeskokem, takže musí tato hodnota být opravdu nízká.

Preference útoku na silnějšího soupeře

Oproti standardní verzi dámy, kde je naší jedinou preferencí získat při jednotlivých tazích co nejlepší pozici pro naše herní kameny, bychom se v dámě pro tři hráče měli také rozhodovat, jakým způsobem tahy provádět tak, abychom při útoku stále neoslabovali pouze jednoho hráče, zatímco druhý hráč bude téměř beze ztrát. Situace vychází z teorie truelu, kdy na modelovém příkladu bylo vysvětleno, proč bychom měli vždy preferovat silnějšího střelce a stejně je tomu také zde. Nesnažíme se co nejdříve eliminovat slabšího hráče, jelikož se dříve nebo později dostaneme do hry jeden na jednoho proti silnějšímu. V takovém případě se hra dostane v podstatě na úroveň klasické dámy, kdy ovšem bude jeden hráč rovnou v nevýhodě.

Definujeme-li si hodnotu $S | S \in \mathbb{N}, S > 0$ jako sílu hráče a budeme předpokládat množinu herních kamenů libovolného soupeře P^x. Síla soupeře je poté rovna hodnotě viz. (4).

$$S = \sum_{i=1}^{|P_p^x|} \alpha_i + \sum_{j=1}^{|P_q^x|} \beta_j \tag{4}$$

I když tedy jednoznačně rozeznáme sílu soupeřů, rozhodně bychom se měli při rozhodování také zamyslet, zda bude pro nás v konkrétní situaci výhodné na aktuálně silnějšího soupeře útočit. Například nám v dané situaci může útok na slabšího soupeře zajistit korunovaci herního kamene, či několikanásobný přeskok, takže ačkoliv bychom preferovali útok na silnějšího, nemělo by se to rozhodně stát nutným pravidlem.

Ve výpočtu tahů umělé inteligence by se tedy neměla nacházet podmínka útoku na silnějšího, ale pouze nějaké doporučení. Předpokládejme tedy indexy 1, 2 u proměnných jako označení soupeřů. Toto doporučení by tedy obecně mělo být definováno vztahem (5).

$$S^1 > S^2 \implies \alpha^1 > \alpha^2, \beta^1 > \beta^2 \tag{5}$$

Tento vztah se dá slovy popsat jako podmínka, kdy pokud je jeden hráč silnější, než ten druhý, měla by být hodnota jeho herních kamenů vyšší, než hodnota herních kamenů druhého hráče. V takovém případě je ale ještě nutné odpovědět na otázku, jakým způsobem se budou

přesně tyto hodnoty herních kamenů měnit. Na jednu stranu potřebujeme tedy změnit hodnotu dostatečným způsobem, aby docházelo k preferenci útoku, ale zároveň, aby nedocházelo k bezhlavým útokům na silnějšího, zatímco slabší soupeř bezpečně korunuje své pěšáky. Nejvhodnější by bylo takovýto problém řešit optimalizačním algoritmem, který by nám vyhodnotil, jak by měly hodnoty událostí soupeře narůstat v závislosti na jeho síle. V našem případě budeme ale definovat novou hodnotu události silnějšího soupeře jako $\omega \in \mathbb{N}$, kdy pokud máme libovolnou událost o hodnotě $\theta \in \mathbb{N}$, tak za předpokladu, že $S^1 > S^2$ počítáme novou hodnotu události viz. vzorec 6.

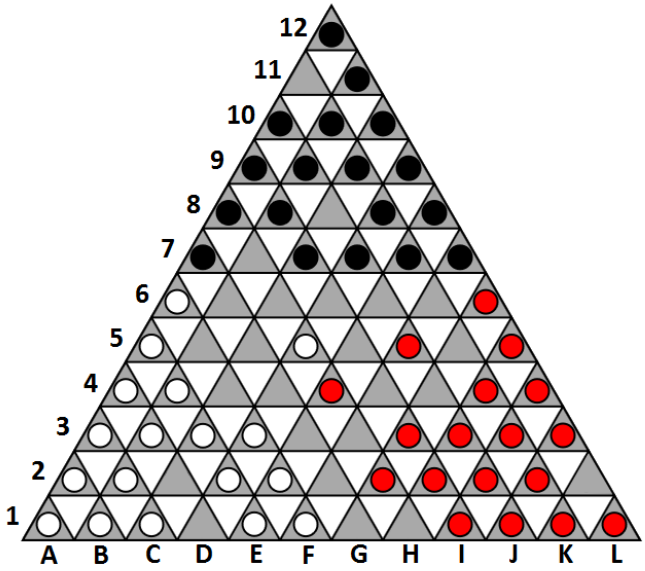
$$\omega = \theta + \frac{S^1}{S^2} \cdot \frac{1}{10}\theta \tag{6}$$

K následnému zamezení velkého růstu hodnot musí zároveň platit podmínka $\omega \leq \frac{6}{5}\theta$.

Dočasné spojenectví

Dočasné spojenectví částečně vychází z problematiky truelu a je pravděpodobně jednou z nejdůležitějších strategií, kdy tato strategie silně závisí na schopnosti předpovídat tahy hráčů a také na faktu, že hráči neuzavřeli trvalé spojenectví, které v tomto případě nemá smysl předpokládat, jelikož při uzavření trvalého spojenectví je proti dvojnásobné přesile takřka nemožné zvítězit. Jelikož se tedy v dámě pro tři hráče snaží každý hráč získat vítězství pro sebe, neobáváme se, že by jeden hráč úmyslně napomáhal tomu druhému, ale čeho bychom se obávat měli, jsou právě tato dočasná spojenectví, která většinou trvají pouze jeden tah, ačkoliv nic nebrání v kooperaci i na více, než jen jeden tah. Jak nyní ale rozeznáme, kdy hráči kooperovat mohou a kdy ne?

Pro následující rozbor situací budeme pro zjednodušení předpokládat tahy $t^x = (p^x, f^x, d^x)$, pro které platí $p^x \in F$ a také $\forall y | y \in d_x, y \in F$. Znamená to pouze, že nebudeme odkazovat na herní kámen, ale na pole herního plánu, které nám také jednoznačně určí daný herní kámen, jelikož bychom zde jinak museli zavést určité číslování herních kamenů, čímž bychom je jednoznačně odlišili. Předpokládejme situaci viz. obrázek 4, kdy na tahu je bílý hráč. Jednotlivé počty herních kamenů hráčů jsou ve stavu $|P_p^w| = |P_p^b| = |P_p^r| = 19$, $|P_q^w| = |P_q^b| = |P_q^r| = 0$. Každý hráč proto vidí sílu svých soupeřů jako $S = 19\alpha$, proto pokud bychom brali hodnoty α^1, α^2 vždy jako hodnoty soupeřů, tak platí $\alpha^1 = \alpha^2$. Jelikož jsou síly naprosto vyrovnané, není nutné uvažovat preferenci útoku. V klasické dámě bychom v podobné situaci reagovali asi tak, že bychom pokračovali v předsouvání herních kamenů tak, abychom si vytvořili co nejvíce prostoru na herním poli a případně bychom tvořili tandemy na herním plánu abychom zamezili možnosti přeskoku. V každém takovémto případě bychom uvažovali výplatu $V = 0$. V tomto případě to ovšem není možné a musíme reagovat jinak. Pokud bychom provedli tah $t^w = (H5, I6, \emptyset)$, tak je zřejmé, že herní kámen ztratíme. Černý hráč má poté dvě možnosti jak provést vynucený tah. První způsob je provést tah $t^b = (I7, G3, \{I6, H4\})$, kdy by následoval tah červeného hráče $t^r = (H2, H3, \emptyset)$, při kterém by se připravil na přeskok $t^r = (H3, H5, \{H4\})$, po tom, co bychom byli nuceni provést přeskok $t^w = (F2, H4, \{G3\})$. Po této výměně by byla výsledná



Obrázek 4: Příklad kooperace

výplata hráčů $V = (-\alpha, \frac{\alpha}{2}, \frac{\alpha}{2})$. Druhá možnost pro černého hráče je provést několikanásobný přeskok $t^b = (J7, H1, \{I6, H4, H2\})$ a zároveň současně získal královnu. Červený hráč by následně provedl přeskok této královny $t^r = (I1, G1, \{H1\})$, který by byl okamžitě následován naším tahem $t^w = (F1, H1, \{G1\})$. V tomto případě by byla výplata $V = (\alpha, \alpha, -2\alpha)$, jelikož i přes korunovaci víme, že $\kappa - \beta = -\alpha$, takže černý hráč z celkových 2α ztratí právě hodnotu α. Můžeme tedy konstatovat, že pokud bude černý hráč uvažovat racionálně, tak zvolí druhou strategii, ze které získá lepší výplatu o $\frac{\alpha}{2}$ vyšší a zároveň zařídí, že červený hráč odejde s hodnotou výplaty $V^r = -2\alpha$, přičemž současně my, jakožto bílý hráč také získáme výplatu $V^w = \alpha$. Mimo jiné lze říci, že pokud by se hráči ocitli v tomto postavení herních kamenů, tak se jedná o chybu červeného hráče, že nedokázal tuto kooperaci předpovídat.

Jak si lze ovšem povšimnout, tak kooperace v předchozím případě nebyla zcela dobrovolná, jelikož se v podstatě jednalo o vynucené přeskoky. Uveďme si tedy jinou situaci, viz. obrázek 5, kdy je právě na tahu bílý hráč a síla jednotlivých hráčů je $S^w = 2\alpha$, $S^b = \alpha$ a $S^r = 7\alpha$. Pokud bychom brali hru opět z perspektivy bílého hráče, tak jelikož nejsou síly vyrovnané tak by pro nás hodnota herních kamenů červeného hráče byla navýšena, kdy by se konkrétně jednalo o hodnotu $\alpha^r = \frac{5}{6}\alpha$. Pro tuto situaci ovšem s navýšenou hodnotou počítat nebudeme a budeme brát v úvahu klasickou hodnotu herního kamene $\alpha^r = \alpha$. Pokud se přesuneme v možnostech jak hrát jednotlivé tahy, tak naše priorita by měla být v co nejrychlejším získání královny. Na první pohled se proto zdá, že nejvhodnějším tahem tahem je $t^w = (I2, J2, \emptyset)$, jelikož ačkoliv po povinném přeskoku černého hráče $t^b = (F5, D3, \{E4\})$ získá červený přeskokem $t^r = (E3, C3, \{D3\})$ královnu, tak stále máme k dispozici na hraně herního plánu herní kámen, který ji okamžitě eliminuje přeskokem $t^w = (B2, D4, \{C3\})$. V této situaci byl již černý hráč eliminován a dochází k duelu mezi námi a červeným hráčem. Nejrozumnější sekvence tahů pro oba hráče, kdy první tah náleží

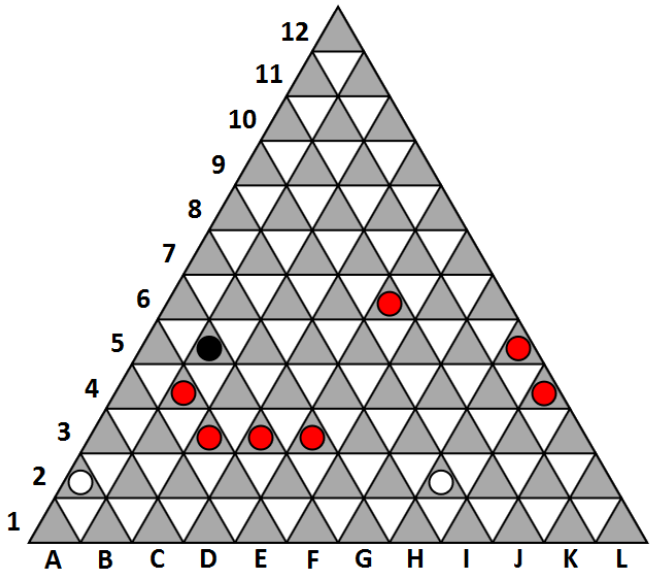
červenému hráči je $\vec{t} = ((F3, E3, \emptyset), (J2, K2, \emptyset), (E3, D3, \emptyset), (K2, L2, \emptyset), (D3, C3, \emptyset))$. Ačkoliv jsme tímto dokončili korunovaci a jsme aktuálně na tahu, tak náš soupeř také korunoval jeden herní kámen, kdy lze hodnotit jeho sílu jako $S^r = 4\alpha + \beta$. Hlavním problémem aktuálně je, že ovládáme pouze jednu královnu a z důvodu přítomnosti soupeřovy královny nelze ovládat hranu herního plánu, kde u něj dochází ke korunovacím, takže může soupeř potenciálně dostat na herní plán až pět královen, jelikož pokud bychom se rozhodli v přesun na hranu, mohl by soupeř jednoduše nalíčit past pomocí vynucení povinného přeskoku a tím nás porazit. V každém případě, pokud bude červený hráč volit tahy obezřetně, tak dokáže s přehledem zvítězit.

I když se nám při minulém rozhodnutí podařilo provést korunovaci, tak jsme se stále ocitli v situaci, která vede k porážce, takže se vrátíme do výchozí pozice a promyslíme možnost kooperace. V tomto případě můžeme černému hráči umožnit přímou korunovaci tahem $t^w = (B2, C2, \emptyset)$. To by ovšem znamenalo, že by se černý hráč mohl pokusit o eliminaci našeho posledního pěšáka pomocí tahu $t^b = (B1, K10, \emptyset)$, kdy by vyčkával, až provedeme libovolný tah $t^w(Jx, Ky, \emptyset)$, kde by provedl přeskok a následně se přesunul na pozici $K11$ odkud by již mohl vyčkávat na herní kameny soupeře a postupně je eliminovat. To by ve výsledku vedlo k vítězství černého hráče, pokud se červený hráč nerozhodne vynutit přeskok černého hráče tahem $t^r = (E3, D3, \emptyset)$ ve chvíli, kdy procházíme přes Kx pozice, čímž by nám pomohl získat královnu. Pro něj by to ovšem nemělo žádný užitek, jelikož by následně černý hráč mohl kontrolovat postup jeho herních kamenů, které nemají jinou možnost, než se pokusit o korunovaci. Navíc by měl ještě proti sobě dalšího soupeře, který by ovládal královnu a mohl by tedy také eliminovat jeho herní kameny, což není rozhodně jeho preferovaným cílem. Partie by v takovém případě s největší pravděpodobností skončila remízou mezi námi a černým hráčem.

Třetí variantou zahrání prvního tahu je pouhá ochrana kamene černého hráče tahem $t^w = (B2, C3, \emptyset)$, čímž po přeskoku černého znemožníme červenému eliminovat jeho herní kámen, který získá zároveň možnost korunovace v následujících dvou tazích. V tom případě je červený hráč také schopen provést během dvou tahů korunovaci, zatímco my potřebujeme ke korunovaci tahy tři. Přesto, že potřebujeme jeden tah navíc, tak protože jsou na pozicích $L4$ a $L5$ umístěny herní kameny červeného, tak se není schopen dostat na hranu herního plánu, kde by byl schopen zastavit naši korunovaci. Konkrétně tedy proběhne posloupnost tahů 7, kde je na tahu jako první červený hráč.

$$\begin{aligned}\vec{t} = &((E3, E4, \emptyset), (I2, J2, \emptyset), (D3, C2, \emptyset), (E4, E5, \emptyset),\\ &(J2, K2, \emptyset), (C2, B1, \emptyset), (F3, F4, \emptyset), (K2, L2, \emptyset), ...)\end{aligned} \tag{7}$$

Jako předposlední tah posloupnosti 7 se červený hráč snaží posunout další herní kámen blíže ke korunovaci, i když je černý hráč v tuto chvíli již schopen kontrolovat hranu herního plánu (herní pole $A1$ až $L12$), takže se ve výsledku bude muset následně stáhnout i z vlastní královnou, pokud dojde k tahu černého hráče $t^b = (B1, B2, \emptyset)$. I přes skutečnost, že je nyní síla



Obrázek 5: Příklad kooperace

hráčů $S^w = \alpha + \beta$, $S^b = \beta$, $S^r = 5\alpha + \beta$ a červený hráč má tedy stále početní převahu, tak zde již existuje reálná šance na získání patové situace pro všechny hráče současně.

4 Implementace

Při aktuální implementaci strategií a umělých inteligencí jsem využil již existující aplikaci, kterou jsem vytvořil při tvorbě své bakalářské práce. I když rozložení tříd a většina názvů metod z bakalářské práce byla zachována, tak došlo k přepisům logiky výpočtu přeskoků, za účelem rychlostní optimalizace těchto výpočtů, takže se již k většině implementace těchto částí aplikace nebudu vracet a zaměřím se na implementaci umělých inteligencí, které obsahuje namespace NewAI.

4.1 MoveRules

Jedná se o třídu, kde jsou implementována pravidla pro výpočty přeskoků. Tato pravidla byla implementována při tvorbě mé bakalářské práce. Důvodem zmínění jsou rozsáhlé změny, které byly oproti bakalářské práci provedeny. Tyto změny se týkají výpočtů tahů obou typů herních kamenů (pěšáka i královny), kdy důvodem jejich úpravy byla hlavně rychlostní optimalizace.

Jelikož jsem v bakalářské práci využíval pro uložení herního plánu pole, tak jsem při výpočtu tahu pěšáka procházel všechny pole herního plánu a vybíral ty, které odpovídaly požadovaným směrovým vektorům, které mají hráči definovány vzhledem k aktuální pozici herního kamene. Důvodem tohoto způsobu implementace byla v zásadě jeho jednoduchost, kdy nebyly vyžadovány vysoké nároky na rychlost. Uložení herního plánu bylo změněno na dvourozměrné pole, kdy jednotlivé indexy současně odpovídají souřadnicím a při vyhledávání se tedy rovnou odkazujeme na tyto indexy. Musíme si jen hlídat hranice, abychom se při výpočtu nepokoušeli dostat za hranice herního plánu.

Stejná změna byla aplikována na výpočty královny. Při té příležitosti byl celkově refaktorován zastaralý kód, aby bylo dosaženo dalšího zefektivnění výpočtu. Mimo to došlo také ke snížení pohyblivosti královny, kdy v bakalářské práci měla povolen pohyb všemi šesti směry. Tento pohyb měl za následek, že královna měla občas možnost přeskočit i 10 herních kamenů současně, čímž dokázal hráč, vlastnící královnu porazit soupeře i během několika tahů, pokud byly jejich herní kameny v danou situaci nekryty. Další problém byl, že výpočet počítal všechny možné cesty přeskoků, kterých bývaly i tisíce, v případě, že bylo k dispozici dostatek herních kamenů na přeskok a také tedy dostatečné množství cest, kterými mohl být několikanásobný přeskok veden.

4.2 AIObjects

Jedná se o namespace doménových modelů, využívaných při výpočtech umělé inteligence.

BestTurn obsahuje atributy BestTurnGP a BestTurnPath, které uchovávají informaci o nejlepším tahu, kdy je uložen herní kámen, který je schopen tah zahrát a také cesta tahu, kdy se

při několikanásobném přeskoku může jednat o posloupnost pozic. V případě, že existuje více takových tahů se stejným bodovým hodnocením, vybere se náhodně jeden z těchto tahů.

EventValues vlastní informaci o hodnotách herních kamenů a herních událostí, které při partii mohou nastat.

PlayerValue pouze zapouzdřuje objekt typu IPlayer (aktuálního hráče) společně s hodnotou tahu. Tento model byl vytvořen pouze za účelem zachování hodnoty tahu společně s informací o daném hráči, což je výhodné při rekurzivním prohledávání stavového prostoru všech možných tahů, které klasická umělá inteligence prohledává. V případě aplikace jednotlivých strategií lze také tímto způsobem předat jednotlivé hráče s jejich dosavadními výplatami, které jsou následně pozměněny při aplikaci dané strategie.

TurnPayoff je model, obsahující výplaty jednotlivých hráčů. Jedná se tedy o vektor výplat, který je uchovává buďto vzhledem k aktuálnímu tahu nebo v určitých případech je v něm uložena informace o výplatách za určitou sekvenci tahů.

SimulatedMove je model, který je využíván při simulacích tahů umělé inteligence. Veškeré změny, které na herním plánu jsou při těchto výpočtech provedeny, musí být poté navráceny. Prohledávání stavového prostoru možných tahů implementováno rekurzivním algoritmem, což principiálně funguje jako zásobník (LIFO), do kterého jsou simulované tahy nejprve ukládány a následně navráceny zpět algoritmu k invertování daného tahu. Tímto je na herním plánu zaručeno, že po výpočtu umělé inteligence bude plán vždy v původním stavu.

Model tedy obsahuje informaci o původní pozici herního kamene a jeho kompletní cestu přeskoků. Dále zaznamenává vektor výplat, které byly při tomto tahu rozděleny hráčům a list herních kamenů, které nebyly vymazány při přeskoku soupeřem, nýbrž při nedodržení povinného přeskoku. V tomto případě tedy vždy platí, že list takto smazaných herních kamenů obsahuje prvky pouze v případě, že byl proveden tah bez přeskoku. Jako poslední je využíván ukazatel IsPromoted, který označuje tahy, při kterých došlo ke korunovaci, aby bylo možné následně provést zpětnou záměnu při navrácení simulovaného tahu. Genetický algoritmus ještě využívá konkrétní objekt královny PromotedGamePiece. Důvod k zachovávání objektu je, že se uchovává jistá populace posloupností tahů, která si pro provádění tahů uchovává konkrétní objekty herních kamenů, které tahy provádějí. Pokud se tedy uchová konkrétní objekt královny, která byla korunována během této sekvence, tak by se při nové simulaci vytvořil jiný objekt typu Queen, jehož reference by nebyla shodná s tou, která je uchována v SimulatedMove objektu. Tím bychom se mohli vystavit situaci, že by nebylo zpětně možné při inverzi tahů dohledat královnu, která má být zaměněna za klasického pěšáka.

Individual se využívá výhradně při výpočtech tahů za pomocí genetického algoritmu, kdy objekty této třídy představují jedince, který obsahuje unikátní ID, sekvenci tahů hráčů a výplatu. Během procesu výpočtu je mu také přiřazeno rozmezí hodnot při ruletové selekci.

TreeNodeGA, TreeStructureGA jsou třídy, které zajišťují proces ohodnocení jednotlivých jedinců, kdy je sestavován částečný strom, přičemž uzly jsou tvořeny TreeNodeGA. TreeStructureGA následně obsahuje tuto strukturu stromu a také obsahuje veškerou funkcionalitu potřebnou k jeho sestavení, ohodnocení a následně navrácení populace, která je setříděna od nejlepšího jedince po nejhoršího. Tento proces je více popsán v části, která se zabývá samotným procesem výpočtu genetického algoritmu.

4.3 AIFunctions

V tomto namespace je pouze jediná třída AIFunctions, která obsahuje pomocné metody pro oba algoritmy umělé inteligence. Třída je implementována návrhovým vzorem Singleton, takže k ní mají oba algoritmy přístup. Původně byla třída navržena pouze ke sdílení těchto metod pro oba algoritmy. Během simulací strategií pomocí standardního algoritmu se ovšem ukázalo, že by genetický algoritmus vyžadoval drobné úpravy na těchto metodách, proto byly následně vytvořeny jejich kopie, které implementovaly požadované úpravy.

4.4 Strategies

K implementaci strategií byl využit návrhový vzor Strategy, kdy jednotlivé strategie implementují rozhraní IStrategy, které obsahuje hlavičku procedury ApplyStrategy. Tato procedura přijímá jako parametry herní kámen a jeho provedený tah, a také trojice výplat jednotlivých hráčů. Takže pokud přidáme určité strategie dané umělé inteligenci, tak tyto strategie fungují jako přidaná hodnota k již vypočteným výplatám hráčů, což znamená, že se na konci hodnotící funkce umělé inteligence procházejí a aplikují všechny aktivované strategie. Také je nutné podotknout, že každý hráč, který zná určitou strategii, tak předpokládá, že tuto strategii používají také ostatní. Nepředpokládá se tedy, že každá umělá inteligence zná množiny strategií obou soupeřů a také se nepředpokládá, že by umělá inteligence vždy předpokládala, že je jediná, která využívá danou strategii [10].

4.5 Implementace umělých inteligencí

Implementace umělých inteligencí je opět realizována pomocí návrhového vzoru Strategy, kdy je implementováno rozhraní IArtificialIntelligence, obsahující metody MakeTurnAI a SetMaxDepth. Konkrétně metoda SetMaxDepth je využívána pouze klasickým algoritmem umělé inteligence, kdy je potřeba za běhu programu navýšit, či případně snížit hloubku prohledávání [10].

4.5.1 StdAI

Třída obsahuje kompletní implementaci umělé inteligence, která využívá při výpočtech kompletní prohledávání stavového prostoru. Pokud bychom chtěli algoritmus přirovnat k některému ze standardních algoritmů typu Minimax/Negamax, tak má algoritmus blíže k Negamaxu. Problémem při implementaci je, že nelze jednoduše tyto algoritmy implementovat z toho důvodu, že je ve hře třetí hráč. V klasickém Minimaxu platí, že pokud bychom vypočítali ideální tahy v partii až do konce, tak v případě, že soupeř nezahraje ideální tah, tak nám to jen pomůže. Bohužel v této variantě dámy toto pravidlo neplatí a špatný tah jednoho soupeře může nahrát soupeři druhému, který situace jen využije. Příkladem mohou být opakované útoky jednoho hráče, kdy pokud nás daný hráč donutí provádět výměny, tak z toho těží pouze třetí hráč [8].

MakeTurnAI je metoda, která spouští výpočet tahu umělé inteligence, kdy nejprve dojde k výpočtu nejlepšího tahu a následně k jeho provedení.

CalculateTurn, CalculateFirstEnemyTurn, CalculateSecondEnemyTurn jsou metody zajišťující procházení stavového prostoru, kdy se metody při prohledávání volají v daném pořadí, dokud nedojde k zanoření do požadované hloubky prohledávání, která odpovídá počtu tahů, které je umělá inteligence schopna předpovědět. K porovnávání tahů v dané hloubce se využívá vektoru tahů, která odpovídá výplatám, které hráči získali od začátku sekvence tahů, až k tomuto bodu. V tuto chvíli se také ukládá aktuální vektor výplat pro daný tah. Důvodem je to, že při prohledávání ukládáme celkový vektor výplat, který je potřeba při vynoření do nižší hloubky navrátit do stavu, který odpovídá dané hloubce.

Evaluate se využívá pro ohodnocení tahu, kdy přijímá herní kámen, který provedl tah a také samotnou cestu tahu. Dále také přijímá trojici proměnných, které uchovávají informace o hráčích a jejich výplatu. V metodě se následně kontroluje, zda nedošlo k přeskoku herního kamene soupeře, ke ztrátě vlastního herního kamene, v případě, že nebyl dodržen povinný přeskok a také, jestli nebyla při tahu provedena korunovace na dámu. V poslední řadě se aplikují veškeré strategie, které daný hráč zná.

MakeTurn, RevertTurn obě tyto metody přijímají jako parametry tah a herní kámen, který tento tah provádí. Metody se využívají k simulaci aktuální situace na herním plánu, kdy pokud je při výpočtu simulován určitý tah, je nutné tento tah opět vrátit zpět, abychom zajistili, že nedojde k pozměnění situace na herním plánu v průběhu výpočtu.

4.5.2 GeneticAlgorithm

GeneticAlgorithm je druhá třída, implementující rozhraní IArtificialIntelligence a jedná se o druhý algoritmus, využívaný k výpočtům tahů. Jak název vypovídá, je k výpočtům využíván

genetický algoritmus, jehož obecný popis je k nalezení například v publikaci Evoluční výpočetní techniky - principy a aplikace [9].

Evaluate, EvaluatePath, EvaluateAllPaths jsou metody, využívané při ohodnocování jednotlivých tahů, sekvencí tahů a poslední je metoda, která projde celou populaci jedinců a ohodnotí jejich sekvence tahů.

Generate population je metoda, využívaná k počátečnímu vygenerování populace, kdy jsou jednotlivé tahy generovány zcela náhodně a následně jsou shlukovány do sekvencí, které následně tvoří jedince populace. Při generování této populace je samozřejmě nutné stále zachovávat původní postavení jednotlivých herních kamenů, jelikož je generované tahy vždy nutné rovnou nasimulovat, aby bylo možné vytvářet validní sekvence. Po vygenerování sekvence do požadované hloubky jsou všechny provedené tahy vždy navráceny.

FindAvailableGamePieces navrací list indexů herních kamenů, které má hráč aktuálně k dispozici. Metoda také využívá parametru, který v závislosti na nastavené pravděpodobnosti úmyslně vyhledá pouze herní kameny, které jsou schopny provést přeskok. Pokud žádné nalezeny nejsou, jsou navráceny všechny dostupné kameny.

EvaluatePopulationRoulette, CountRoulette, RollRoulette zajišťují selekci pomocí rulety, kdy nejprve zajistíme přiřazení hodnot rulety mezi jednotlivé jedince v závislosti na jejich pořadí a následně jsme schopni po roztočení rulety vyhledat daného jedince z populace pomocí algoritmu půlení intervalu.

PopulationRecombination provádí křížení jedinců. Křížení je prováděno do chvíle, kdy je buď vytvořen počet nových jedinců, který je roven velikosti populace. V druhém případě je křížení zastaveno v případě, kdy počet iterací dosáhl velikosti populace. Druhá podmínka existuje pouze pro případ, že by algoritmus nebyl schopen nacházet nové validní jedince. Křížení jedinci jsou rozděleni vždy na jednom místě v sekvenci tahů a jsou rovnou vždy testovány všechny kombinace.

CheckValidIndividual je metoda, která jako parametr přijímá jedince, kterého je nutné validovat. Nejprve dojde ke kontrole sekvence tahů, kdy pokud v některém z tahů je nalezen problém, tak se metoda pokusí o opravu chybného jedince. Jestliže není jedince možné opravit, tak je kontrola ukončena a metoda navrátí hodnotu false a jedinec je následně smazán.

MutatePopulation prochází celou populaci a s určitou pravděpodobností provede na aktuálním jedinci mutaci. Pokud je mutace prováděna, tak je v místo náhodného prvku sekvence

vygenerován náhodně jiný tah. Poté je vyhodnocena validita nového jedince, kdy je nutné zjistit, zda je celá sekvence validní, aby mohl být tento jedinec zachován. V opačném případě je nový jedinec smazán a zachová se jedinec původní.

5 Principy fungování algoritmů

Sekce je úzce spojená se samotnou implementací těchto algoritmů, ale je více zaměřena na detailní popis teoretických procesů jejich fungování.

5.1 Princip fungování standardního algoritmu

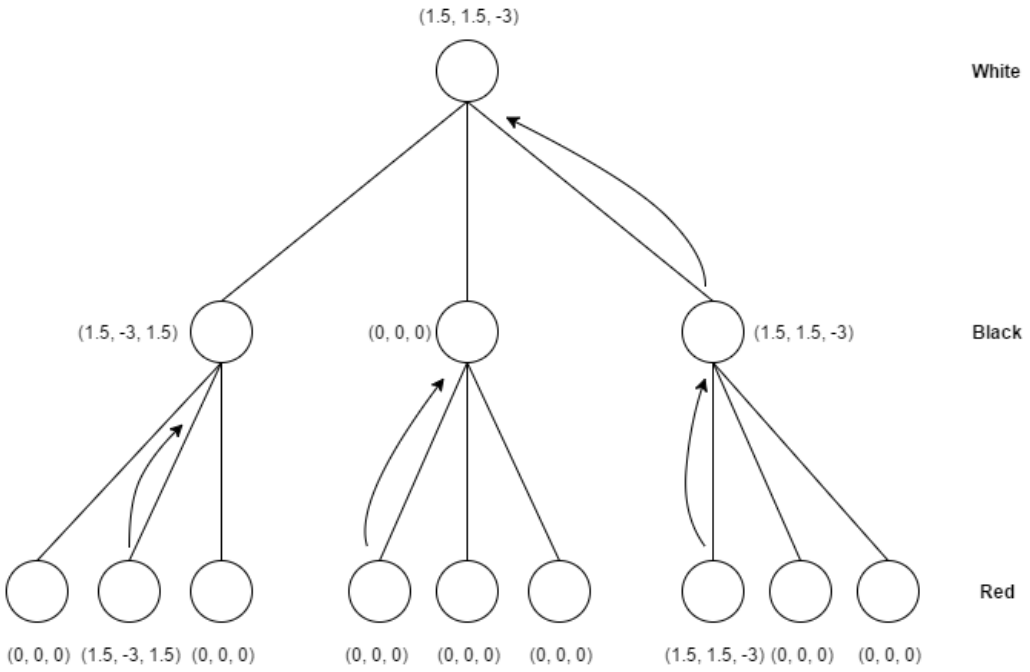
Princip fungování tohoto algoritmu je podobný Minimaxu, či Negamaxu. Prochází se stavový prostor, který musí být pro zjištění nejlepšího tahu kompletně prozkoumán. Jak již bylo řečeno výše, tak se při každém zanoření do hloubky provedou metody CalculateTurn, CalculateFirstEnemyTurn a CalculateSecondEnemyTurn. Tyto metody jsou do sebe následně rekurzivně zanořeny a postupně navracejí jednotlivé ohodnocení, z kterých následně umělá inteligence volí [8].

Na obrázku 6 je ukázán průběh volby nejlepších tahů. Rozdíl oproti Minimaxu, či Negamaxu je ten, že nemáme pouze jednu hodnotu, na základě které bychom určili, zda tah je, či není výhodný. Místo toho máme trojici hodnot, kdy každá hodnota je přiřažena jednomu z hráčů. Konkrétně je první hodnota přiřazena aktuálnímu hráči, druhá hodnota prvnímu soupeři, který je na tahu jako další a třetí hodnota druhému soupeři. Každý hráč si následně pak volí tahy na základě nejlepší hodnoty v parametru, kde je obsažena jeho hodnota [8].

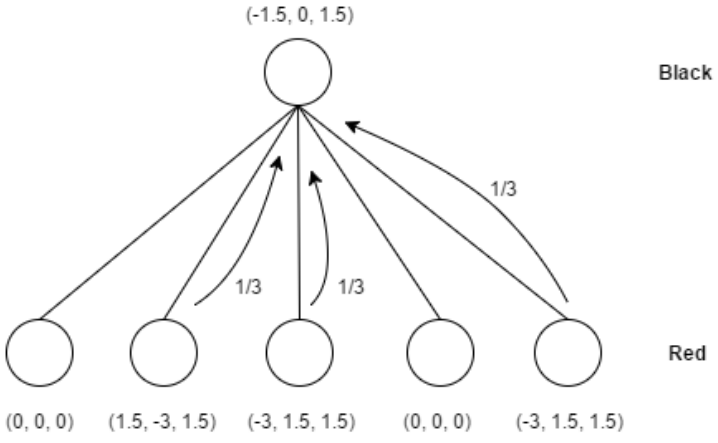
V případě, pokud má aktuální hráč, který výpočet provádí na výběr z více možných tahů, tak si náhodně zvolí jeden z nich. V případě soupeřů, jak je ukázáno na obr. 7, si soupeř také vybere jeden tah náhodně, ačkoliv problém nastává ve chvíli, kdy jeden z těchto tahů může poškodit nás, zatímco jiný z těchto tahů může poškodit soupeře. V tomto případě není jiná možnost, než tyto hodnoty průměrovat, takže pokud má soupeř na výběr dva možné tahy, jak zaútočit na nás, ale současně má jeden možný tah, jak zaútočit na druhého soupeře, tak je pravděpodobnost 2/3, že se rozhodne zaútočit na nás. Soupeři je totiž ve výsledku jedno, na koho zaútočí, pokud jsou tyto tahy stejně hodnotné a pokud neimplementuje strategii, která by tuto hodnotu mohla ovlivnit. Kdybychom tuto situaci chtěli vyjádřit z pohledu teorie her, tak hráč volí smíšenou strategii, kdy rozdělí pravděpodobnosti mezi jednotlivé nejhodnotnější tahy. Pokud ale nemá důvod preferovat útok na konkrétního hráče, tak zkrátka tuto pravděpodobnot může rozdělit rovnoměrně mezi všechny tahy.

5.2 Princip fungování genetického algoritmu

Jak již bylo dříve zmíněno, tak algoritmus je konkrétní implementací genetického algoritmu, k jehož obecným principům se již nebudu vracet. Tato kapitola se tedy bude zabývat detailním popisem této implementace.



Obrázek 6: Průběh počítání tahů



Obrázek 7: Průběh výpočtu při tazích stejné hodnoty

Vygenerování populace

je provedeno náhodným výběrem herních kamenů a tahů. Nejprve se tedy vyhledají všechny herní kameny, které jsou aktuálně k dispozici. Proto je důležité stále udržovat herní plán aktualizován během vytváření tahů a správně simulovat a následně navracet tahy, abychom byli schopni opravdu najít pouze ty herní kameny, které jsou v určitém tahu sekvence stále naživu. Při získání indexů těchto kamenů v poli následně jeden náhodně vybereme a provedeme výpočet možných tahů. Jestli je nalezen alespoň jeden vhodný tah, tak se vytvoří nový jedinec. Pokud není nalezen žádný tah, jelikož je herní kámen zablokován, tak je tento herní kámen vyřazen ze seznamu dostupných herních kamenů, abychom zabránili zbytečným pokusům a výpočet tahů u stejného herního kamene, který je zablokován a vybereme náhodně další herní kámen. Takto pokračujeme, dokud nenajdeme tah a nebo dokud není seznam s možnými herními kameny prázdný. V takovém případě je jasné, že hráč buď nemá k dispozici žádný herní kámen a nebo jsou všechny jeho herní kameny zablokovány a nemůže tedy provést tah. V každém případě je v takové situaci hráč eliminován a hra pro něj končí. Pro generování sekvence je situace, kdy hráč již není ve hře znázorněna způsobem, že je v sekvenci tahů vygenerován tah, který má všechny atributy, které jsou nullable nastaveny na null a všechny bool hodnoty na false. Druhý přístup by mohl být takový, že by se tento tah v sekvenci vynechal a sekvence by takto byla kratší. Tento přístup by ovšem byl výrazně komplikovanější z pohledu implementace, kdy bychom museli kontrolovat, kdy hráč již není ve hře, abychom při simulaci tahů jeho tahy přeskakovali a následně by bylo složité provádět jakékoliv křížení mezi jedinci, kteří by měli takto různě dlouhé sekvence tahů.

Následně jsou zde využity dva konfigurovatelné parametry, kdy jsme schopni nastavit pravděpodobnost, že bude generátor upřednostňovat možné přeskoky herních kamenů, přičemž jsou oba parametry konfigurovatelné hodnotami 0 až 1 (0% až 100%). První parametr je využíván při volbě herních kamenů, kdy se provede metoda, která nastaví všem herním kamenům, které jsou schopny přeskoku hodnotu ViableJump na true. Následně již algoritmus náhodně volí jeden z těchto herních kamenů, kdy je jasné, že každý z těchto herních kamenů má možný tah a tím je minimálně tento přeskok. Druhý parametr zajistí, že herní kámen, který jsme aktuálně zvolili, opravdu náhodně zvolí tah, který je schopen přeskoku. Jako množina možných tahů je tedy vybrána pouze množina možných přeskoků. Jestliže je množina těchto přeskoků prázdná, tak místo toho volíme z množiny všech možných tahů.

Je jasné, že nastavením těchto parametrů výrazně snižujeme velikost prohledávaného prostoru, ve kterém může být určitá podmnožina tahů, které by byly vhodné pro náš tah. V dámě je ovšem výskyt těchto tahů velmi vzácný a situací, kdy opravdu chceme obětovat herní kámen kvůli provedení jisté strategie, je málo a většina tahů, kdy přeskok neprovedeme má pro nás spíše destruktivní účinky. Můžeme tedy takto výrazně zefektivnit vyhledávání tahů při zachování rychlosti daného výpočtu. Druhou výhodou je, že tímto přístupem získáme v populaci více sekvencí, které obsahují přeskoky. Důvodem je, že pokud v počáteční populaci nezískáme do-

statek jedinců, kteří tyto přeskoky obsahují, tak následným křížením těchto jedinců získáváme jen další, které je opět obsahovat nebudou. Jediná cesta k získání těchto přeskoků by následně byla mutace jedinců. Problémem mutace ovšem je, že je pouze malá šance jejího provedení, abychom zajistili konvergenci výpočtu k určitému řešení a také je opět pouze malá šance, že bude mutována právě část sekvence, kde je přeskok možný a ani v takovém případě ještě není volba daného tahu jistá.

Ohodnocení sekvencí tahů jedinců

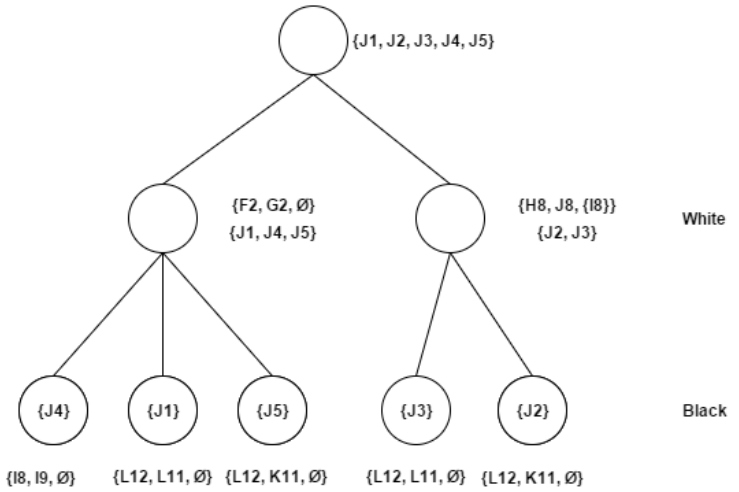
Tato fáze prochází celou populaci, která je postupně ohodnocena. Nejedná se zde o získání hodnoty fitness, která by určila pořadí jedinců v populaci, ale o získání výplat jednotlivých hráčů pro danou sekvenci tahů. Je tedy nutné projít jednotlivé tahy a rozdělit výplaty hodnotící funkcí, která na základě číselných hodnot herních kamenů a případných strategií tyto výplaty rozdělí a navrátí vektor výplat.

Seřazení populace dle významnosti

Klasicky evoluční algoritmy využívají fitness funkce, které každému jedinci přiřadí jednoznačně číselnou hodnotu, na základě které vyhodnotíme kvalitu jedince. Většinou se navíc využívá tzv. normalizované fitness, kdy dochází k převedení do intervalu hodnot 0 až 1 [9].

Na první pohled by se dalo říci, že se jako fitness dá jednoduše využít hodnota, která je navrácena hodnotící funkcí a seřadit populaci dle prvního parametru vektoru výplat. Problémem je, že sekvence tahů jedinců jsou zcela náhodné a tedy není pouze důležité určit, zda je tah na základě této sekvence nejlepší, ale taky zda je možné danou sekvenci předpokládat. Mohou totiž nastat situace, kdy hráč během simulované sekvence přeskočí několik herních kamenů soupeřů a neztratí při tom žádný. To ale bude pravděpodobně jen znamenat, že se podařilo algoritmu vygenerovat takovou sekvenci, kdy soupeři úmyslně nahrávali danému hráči na přeskoky. Takový jedinec by následně mohl výrazně zkreslovat výsledek, jelikož by takto inteligentní hráči nikdy nehráli.

Je tedy zjevné, že je potřeba kromě výplaty zkoumat také závislosti jednotlivých sekvencí navzájem. Přesně tento problém řeší klasický algoritmus, který prochází celý stavový prostor, kdy je vždy vybrána nejvhodnější možnost, kterou si daný hráč může vybrat, jelikož předpokládáme, že soupeři reagují inteligentně. Dojde tedy k sestavení částečného stavového prostoru, který bude obsahovat uzly na základě aktuální populace, kdy každý uzel znamená určitý tah hráče, který je v dané hloubce stromu na tahu. Samotný uzel kromě určitého tahu také samozřejmě obsahuje reference na všechny potomky daného uzlu a také obsahuje seznam jedinců, kteří obsahují sekvence, které procházejí tímto uzlem. Důvod je ten, že když pro určitý uzel stromu získáme určitou výplatu na základě nejlepších odpovědí, tak je tato výplata reálná pro všechny tyto jedince, jelikož je pro nás při prohledávání tohoto stavového prostoru důležité najít možnosti,



Obrázek 8: Příklad stromu tvořeného jedinci

které obsahují nejlepší odpovědi soupeřů, což modeluje reálnou situaci, kdy zjistíme jak by to pro nás jako hráče mohlo dopadnout nejhůř.

Nejprve si tedy vytvoříme kořen stromu, který znázorňuje nejlepší tah. Následně procházíme populaci a tvoříme uzly tak, že nejprve zkontrolujeme, zda daný tah neexistuje mezi uzly, které jsou potomky aktuálního uzlu. Pokud není tah nalezen, je vytvořen a přidán mezi potomky, přičemž je také přidána reference na aktuálního jedince do seznamu jedinců a je proveden přesun do tohoto uzlu. Pokud je tah mezi uzly nalezen, tak pouze přidáme referenci na jedince a přesuneme se do daného uzlu. Takto pokračujeme dokud není celý strom sestaven. Příklad takového stromu je uveden na obr. 8.

Po sestavení tohoto stromu je nutno daný strom ohodnotit. Hodnocení pro jednotlivé uzly provedeme rekurzivním algoritmem, který prochází jednotlivé hloubky stromu a navrací hodnocení nejlepších možných tahů pro hráče, který je v tu chvíli právě na tahu. V podstatě se jedná o stejný algoritmus, který využívá Minimax [8]. Jediným rozdílem je, že tento algoritmus prochází pouze částečný stavový prostor, kdy ačkoliv najde nejlepší odpovědi soupeřů, tak ještě není jisté, zda neexistuje lepší odpověď na daný tah v dané hloubce. Ve výsledku poté v každém uzlu zjistíme, jaké hodnocení tam hráč může získat a zároveň i víme, pro které jedince a tím i sekvence tahů toto hodnocení v daném uzlu nastává. Takže i když má určitá sekvence jisté hodnocení pro danou sekvenci, tak se dá říci, že reálně je schopna získat po provedení tohoto tahu maximálně hodnotu, která byla určena po ohodnocení tohoto stromu.

Posledním krokem je zde seřazení populace na základě těchto hodnocení. Není zde tedy využívána normalizovaná fitness, jelikož není jasné, které hodnoty by měly být minimální, či maximální. [9] Jako klasickou fitness bychom mohli předpokládat hodnotu výplaty pro hráče, který výpočet provádí. Nemůžeme ovšem seřadit populaci tak, že bychom v hloubce 1 vzali jednotlivé podmnožiny populace a seřadili je jejich sloučením v závislosti na ohodnocení v těchto uzlech, jelikož nejsou tyto sekvence jedinců nijak seřazeny. Prvním přístupem by zde mohlo být

procházení stromu, kdy vyhledáme jedince, který obsahuje ohodnocení, které připadlo kořenu stromu, které se tímto nachází v pozici nejlepšího řešení. Takto bychom museli po nalezení jedince tohoto jedince odstranit ze stromu, provést přepočet dané větve, a na základě tohoto přepočtu opět pokračovat v hledání aktuálně nejlepšího jedince, dokud není strom prázdný. Problémem tohoto přístupu je, že neustálé přepočítávání a procházení stromu je časově velmi náročné. Řešením, které je v mé práci využíváno je seřazení potomků jednotlivých uzlů v závislosti na parametru výplaty hráče, který je na tahu. Na základě tohoto seřazení následně odebíráme prvky, kdy začneme u nejlepšího jedince a systematicky sbíráme jedince, kteří jsou listy stromu. Pokud bychom předpokládali, že seřazení je sestupné, tak v prohledávání postupujeme z levé strany a postupujeme doprava.

Tímto přístupem si také zajistíme existenci nejlepšího jedince, který je na prvním místě. Následují za ním ovšem jedinci, které jsou ve stejném podstromu jako daný jedinec. Je složité na první pohled říci, který z přístupů by byl pro genetický algoritmus lepší, jelikož první přístup opravdu obsahuje nejlepší jedince, kteří jsou seřazeni dle významnosti daného stromu. Výhoda druhého přístupu ovšem je, že se zachová celý podstrom nejlepšího tahu, takže při křížení s největší pravděpodobností získáme jedince, kteří spadají do daného podstromu. Takto jsme schopni detailněji prohledat daný podstrom pro případ, že nalezneme řešení, které by zjistilo lepší odpovědi soupeřů a tím snížilo významnost dané větve. Tím by se na první místo dostala jiná větev spolu s novým nejlepším jedincem a začala se opět prohledávat.

Ruletová selekce

Ruletová selekce je druh výběru jedinců, kteří jsou zvoleni, aby byli kříženi. Tento přístup využívá rozdělení jedinců na jednotkovou kružnici, kdy má každý jedinec přiřazenu určitou část na základě kvality tohoto jedince [9]. Zde není seřazení dle fitness možné, jelikož i když známe pořadí, tak nelze jednoznačně určit, o kolik je jistý jedinec lepší, než jiný. Proto pro rozdělení do rulety využíváme indexy pořadí.

V první řadě je nutné vypočítat celkovou velikost hodnot, které budeme přerozdělovat, takže počítáme sumu převrácených hodnot indexů pořadí, abychom docílili výsledku, že první jedinec bude zabírat největší část rulety. Výsledná hodnota je tedy vypočítána viz. vzorec (8).

$$T_v = \sum_{i=1}^{n}(\frac{1}{i}) + I_v, I_v \in \mathbb{R} \tag{8}$$

Kromě sumy indexů je ve vzorci konstanta I_v, neboli *influence value*. Tato konstanta má za úkol snížit rozdíly mezi jedinci v ruletě a rozdělit ruletu rovnoměrněji mezi všechny jedince. V mé práci je nastavena na hodnotu $I_v = 0.01$, což po vytvoření rulety znamená, že 50% rulety nepřipadá prvním 50 jedincům z 1500, ale například prvním 500 jedincům z 1500. Důvodem existence této konstanty je, že při křížení vzniká velký počet vadných jedinců. Proto jsou vždy vyzkoušeny všechny kombinace křížení prvního jedince s druhým. Po vyzkoušení všech možností je tato kombinace jedinců uložena a při novém náhodném vybrání již není kombinována. Kon-

stanta tedy zredukuje šanci, že jsou určití jedinci vybíráni stále znovu, kdy by algoritmus mohl nacházet nové řešení až po mnoha pokusech.

Dalším krokem je následný přepočet jedinců do rulety. Výpočet pro prvky populace je prováděn vzorcem viz. (9).

$$R_i = \sum_{j=0}^{i-1} R_j + \frac{\frac{1}{i+1} + I_v}{T_v} \tag{9}$$

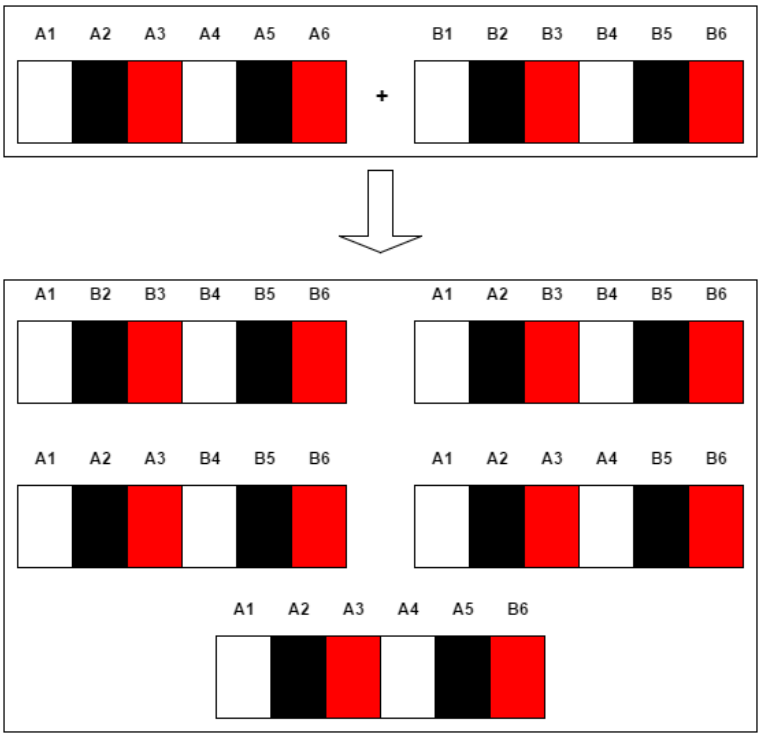
Jelikož zde využíváme indexy pole, ve kterém je populace uložena, tak musíme indexovat již od 0. V takovém případě ovšem není možné využívat pouze hodnotu i, jelikož by u prvního prvku pole probíhalo dělení nulou. Proto musíme index vždy navýšit o 1. Minimální hodnota rulety je poté rovna sumě hodnot rulet všech předchozích jedinců a maximální hodnota rulety je rovna součtu této sumy a přepočtu rulety pro další prvek, kdy opět bereme v úvahu převrácenou hodnotu indexu pořadí spolu s I_v a dělíme celkovou sumou hodnot rulety.

Následná selekce pak probíhá náhodným vybráním hodnoty v intervalu 0 až 1, kdy pomocí metody půlení intervalu nacházíme prvek, do jehož rozmezí hodnot rulety spadá náhodně vygenerované číslo. Pokud by se při výpočtech nevyužívala hodnota I_v, tak by metoda půlení intervalu neměla žádný význam, jelikož by byla největší šance, že najdeme prvek právě na začátku seřazené populace a nikoliv na konci, kdy by byla šance vybrání posledního prvku téměř mizivá.

Křížení

Poté co máme jedince rozděleny do rulety, jsme tedy schopni náhodně vybírat jedince, kteří by se měli zkřížit a účelem vzniku nových jedinců. Křížení je prováděno jednobodově, což znamená, že jsou křížení jedinci rozděleni v jednom místě, kdy z prvního jedince vybereme první sekvenci tahů, která končí řezem, na což naváže druhá sekvence, převzatá z druhého jedince [9]. Výhoda jednobodového křížení je, že si zajistíme validitu první sekvence a následně musíme validovat pouze druhou sekvenci, zda správně navazuje na první. Vícebodovým křížením bychom s největší pravděpodobností výrazně zvýšili počet chybných jedinců, které nelze ani opravit, takže by náš algoritmus nebyl schopen nacházet nová řešení a prohledávat tak stavový prostor. Ukázka křížení je zobrazena na obr. 9.

Jak již bylo zmíněno, tak jsou při křížení vyzkoušeny všechny možnosti. Jednobodové řezy jsou tedy prováděny na všech možných místech a je zde tedy potenciál na získání $n - 1$ nových jedinců, kde je n bráno jako délka řetězce. Následně je kombinace těchto jedinců uložena do seznamu abychom se nesnažili křížit stále stejné jedince a vytvářeli tak velmi redundantní populaci s minimální diverzitou. Je nutno dodat, že pokud vybereme jako první jedince pořadovým indexem 10 a druhého s pořadovým indexem 22, tak tím vyloučíme pouze kombinaci, kdy je první jedinec právě na první pozici a druhý právě na druhé. Jinými slovy se stále může stát, že



Obrázek 9: Křížení jedinců

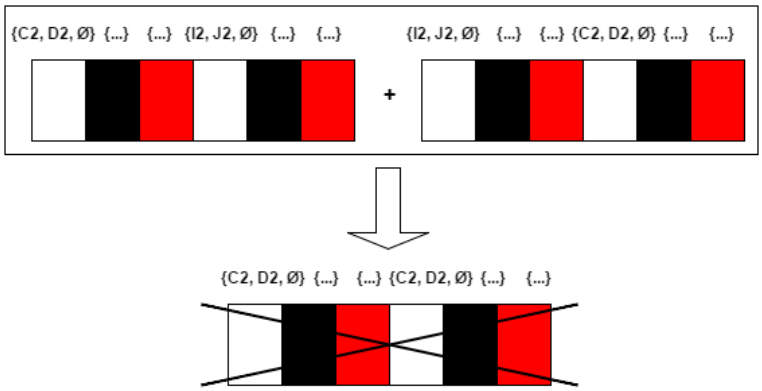
vybereme jako prvního jedince s pořadovým indexem 22 a druhého s indexem 10 a zde je možné křížení opět provést.

Podmínka, která ukončuje průběh křížení je nastavena na počet iterací, který je roven velikosti populace. Algoritmus může skončit také dříve a to tehdy, kdy se velikost populace zdvojnásobí.

Mutace

Mutace je proces, kdy s určitou pravděpodobností pozměníme řetězec jedince. Jelikož se standardně genetický algoritmus implementuje tak, že jedinci mají binární podobu, tak jde pouze o invertování bitu na určité pozici [9]. V našem případě je jedinec tvořen sekvencí tahů, takže místo invertování provedeme vygenerování nového náhodného tahu a vložíme ho na určené místo v sekvenci. Mutace je nastavena na 2% pravděpodobnost a pokud je prováděna, tak má pouze jeden pokus na provedení a pokud nalezne špatného jedince, tak zůstane v populaci jedinec původní. Ve výsledku je tedy šance na provedení mutace o něco nižší, pokud vezmeme v potaz pravděpodobnost na získání validního jedince.

Pokud se tedy rozhodneme mutaci provést, tak nejprve náhodně zvolíme místo v sekvenci, které bude mutováno. Následně se nasimuluje sekvence tahů až po tohoto jedince, jelikož kdybychom se rozhodli náhodný tah provést přímo, tak je velmi nízká šance, že by navázání bylo úspěšné, pokud se ovšem nejedná o první tah sekvence. Po nasimulování této sekvence náhodně



Obrázek 10: Ukázka vadného jedince po křížení

vygenerujeme tah provedeme zkopírování jedince, kdy u kopie nahradíme tah novým náhodným tahem. Pokud bude nově nalezený jedinec validní, tak nahradí jedince původního.

Jelikož algoritmus využívá *elitismu*, kdy je nejlepší řešení během probíhání algoritmu vždy zachováváno, tak mutace na prvním jedinci populace nikdy nemůže nastat. Tím zabráníme poškození nejlepšího řešení, které by mohlo být mutací ztraceno, pokud by byla na takovémto jedinci provedena například při vytvoření poslední generace dané populace [9].

Validace jedinců

Validace jedinců nutně probíhá při fázích křížení a mutace. Pokud bychom využívali v algoritmu jedince, kteří jsou tvořeni pouze číselně, tak křížením, či mutací čísla, které je zapsáno binárně vznikne pouze jiné číslo. Problém ovšem nastává ve chvíli, kdy využíváme sekvence tahů. Je přirozené, že sekvence tahů jsou schopny tvořit jedince, jelikož se také jedná o řetězec. Jsme je tak schopni křížit a také mutovat, pokud určitý prvek sekvence nahradíme jiným. Tyto nově vytvořené sekvence ovšem nemusí vůbec existovat ve stavovém prostoru. Jako příklad lze uvést případ křížení na obr. 10.

Je proto jasné, že musíme určitým způsobem tyto jedince kontrolovat a nevalidní jedince nesmíme do populace vůbec zařadit, jelikož by nám mohli velmi negativně ovlivnit výsledek.

V první řadě je nutné kontrolovat, zda daný tah je vůbec možné provést, takže je nutné zjistit, zda se na výchozí pozici nachází herní kámen daného typu a také, zda patří danému hráči. Následně je nutno říci, zda se na cílové pozici nenachází jakýkoli herní kámen. Pokud se jedná o přeskok, musíme také prověřit, zda na požadovaných souřadnicích existuje soupeřův herní kámen. V případě několikanásobného přeskoku je nutné kontrolovat celou cestu těchto přeskoků včetně všech herních kamenů, které by se na dané cestě měly nacházet. Další zásadní věc, kterou je nutné validovat, je eliminace hráče, která je v sekvenci tahů označena tak, že od určitého bodu sekvence jsou všechny následující tahy hráče inicializovány instancí, která má nastaveny všechny atributy na null a false. V případě, že máme k dispozici tah, který je takto

inicializován, tak musíme zjistit, zda daný hráč opravdu nemá k dispozici jediný možný herní kámen, či případně tah.

Kromě kontroly, zda jsou tyto tahy možné je také nutné provést potřebné opravy na jedincích, kteří použitelní být mohou, ale dali by se označit za „poškozené“. Poškozený jedinec totiž obsahuje reference na špatné objekty. Musí se proto také kontrolovat, zda herní kámen, který tah v jiné sekvenci prováděl není zrovna na jiné pozici. Na první pohled by se dalo říci, že takový tah by nemusel být přípustný, jelikož určitý herní kámen, který tah prováděl se nenachází na určeném místě, ale jelikož je tam jiný herní kámen, který může tah provést, tak jen zaměníme referenci. Stejná věc se týká u přeskoků herních kamenů, kde nás nezajímají konkrétní herní kameny a dokonce ani typ herního kamene. Při původním tahu se na pozici mohl nacházet pěšák, ale pokud se tam v této situaci nachází královna a stále jde provést přeskok, tak výslednému provedení tahu nic nebrání. Poslední typ oprav je prováděn pouze v případě, kdy se nejedná o přeskok. Při klasickém tahu totiž dochází ke kontrole, zda neměl daný hráč libovolným herním kamenem skákat a jelikož může být situaci na šachovnici jiná, tak se může lišit počet i typ herních kamenů, které při takovémto tahu mají být smazány.

Zmíněné opravy jsou nezbytné pro fungování algoritmu, jelikož křížení různorodých sekvencí přináší často mnoho problémů, týkajících se jejich následné validity. Pokud bychom neprováděli žádné opravy a pouze kontrolovali reference, tak by pouze malé procento výsledných jedinců bylo validní a nedocházelo by k potřebné evoluci.

Nastavení parametrů

Genetický algoritmus má celkem 5 volitelných parametrů na základě kterých funguje, jejichž nastavením těchto parametrů výrazně ovlivňujeme funkčnost umělé inteligence. Kromě těchto parametrů je umělá inteligence také plně konfigurovatelná přidanými strategiemi, které fungují stejným způsobem jako pro klasický algoritmus.

Hloubka prohledávaného prostoru je prvním parametrem, kterým nastavíme počet tahů, které je schopna umělá inteligence předpovědět. V závislosti na nastavení počtu předpovídaných tahů, je následná hloubka vyhledávání trojnásobná, jelikož jsou ve hře tři hráči a předpověď tahu vždy trvá, dokud nedohraje druhý soupeř. Nastavíme-li tedy parametr na hodnotu 3, tak jsou jedinci tvořeni sekvencemi devíti tahů. Je nutné si uvědomit, že čím vyšší je hloubka, tím více se snižuje efektivita algoritmu, pokud bychom zanechali stejnou velikost populace a počet generací. Při hloubce 3 je prohledávací prostor na začátku partie 10^9 a pokud dojde ke korunovacím tak se může v závislosti na větvení tahů dostat až na 10^{12}, jelikož pokud hráč vlastní dvě královny, tak může mít k dispozici řádově desítky tahů. Projít takový počet možností by klasickému algoritmu časově trvalo v nejlepším případě asi dvě hodiny a v nejhorším případě by se mohlo jednat řádově o týdny až měsíce. Vše by také bylo samozřejmě ovlivněno použitým hardwarem.

Velikost populace je parametr, který určuje, jak velká populace je během vykonání algoritmu uchovávána. Po provedení křížení jsou nejhorší jedinci vždy vyloučeni z populace a opět je zachována populace, která obsahuje stejný počet prvků, jako při prvotní inicializaci. Nastavení dostatečně velké populace je důležité pro vytvoření úvodní diverzity a k možnosti vytvoření dostatečného počtu nových jedinců během křížení. Velikost populace by měla být u genetického algoritmu nastavována v řádech tisíců [9].

Počet generací šlechtění populace ovlivňuje počet iterací, kdy probíhá křížení, mutace, ohodnocení populace a následná redukce nové populace na původní počet. Tímto procesem zpřesňujeme výsledné řešení, kdy postupně šlechtíme populaci a vylepšujeme množinu dostupných řešení. Je nemožné přesně určit, kolik generací je potřebných k určení nejlepšího řešení, či případně k řešení, které se mu blíží. Jelikož zde nepracujeme s jedinci, kteří jsou tvoření číselně, tak je celkově problematické tento stavový prostor prohledávat. Vyšší počet generací ale obecně zvyšuje šanci k získání lepšího řešení [9].

Nucený přeskok je konfigurovatelný pomocí dvou posledních parametrů. Tyto parametry jsou typu double a definují se v rozmezí 0 až 1. Jedná se o parametry, které ovlivňují pravděpodobnost, při které algoritmus úmyslně zvolí u herního kamene přeskok, či případně úmyslně vyhledá herní kámen, který je schopen skákat. Pokud není schopen žádný takový přeskok najít, volí při generování tahu herní kámen a výsledný tah zcela náhodně. Jak již bylo dříve zmíněno v sekci o generování populace, tak nastavení tohoto parametru je schopno výrazně snížit prohledávaný prostor a generovat více jedinců, kteří by mohli být schopnými kandidáty na výsledné řešení.

6 Srovnání algoritmů umělé inteligence

Tato část textu bude věnována srovnání algoritmů umělé inteligence a také srovnání účinků, při zakomponování strategií do umělé inteligence pomocí experimentů.

Nastavení základních hodnot pro hodnotící funkce umělých inteligencí

K výpočtům hodnot tahů jsou parametry hodnotící funkce nastaveny dle tabulky 7. Hodnota, kterou vždy získáme při přeskoku určitého herního kamene je poloviční oproti ztrátě daného herního kamene za předpokladu, že není využívána strategie, která by tyto hodnoty mohla ovlivnit. Při ztrátě tedy počítáme s těmito hodnotami a při přeskoku cizího herního kamene počítáme s hodnotou poloviční. Uvedená hodnota korunovace je také počítána pro případ, že není využívána strategie preference korunovace.

Tabulka 7: Nastavení hodnotící funkce

Typ	Hodnocení
Hodnota pěšáka	2540
Hodnota královny	6650
Korunovace	810

Takovéto hodnocení bylo získáno za použití Saatyho metody stanovení vah kriterií, kdy je výpočet v tabulce 8. V tabulce je hodnota váhy k_1 přiřazená hernímu kamenu, váha k_2 královně a k_3 je korunovace. Výsledné hodnoty byly vynásobeny hodnotou 10 000, kdy byla snaha o počty ve vyšších číslech z důvodu rizika nepřesnosti datového typu Double na vyšší počet desetinných míst.

Tabulka 8: Výpočet vah pomocí Saatyho metody

	k_1	k_2	k_3	$s_i = \prod_{j=1}^{3} s_{ij}$	$R_i = (s_i)^{\frac{1}{3}}$	$v_i = R_i / \sum_{i=1}^{3} R_i$
k_1	1	$\frac{1}{5}$	3	$\frac{5}{3}$	$1,19$	$0,254$
k_2	5	1	6	30	$3,11$	$0,665$
k_3	$\frac{1}{3}$	$\frac{1}{6}$	1	$\frac{1}{18}$	$0,38$	$0,081$

Nastavení parametrů strategií

Některé ze strategií využívají ke své funkčnosti dalších přídavných hodnot. Jako první lze zmínit hodnotu strategie, kdy je preferován tah pěšákem, pokud existují stejně hodnotné tahy královny a pěšáka, kdy se snažíme docílit rychlejší korunovace. Hodnota této preference je nastavena na $S_v = 10$. Jedná se o přídavnou hodnotu, která má za úkol navýšit hodnotu tahu, ale současně

by neměla ovlivnit jiné události, které jsou v tahu zahrnuty, takže všeobecně by mělo platit, že $S_v << \alpha$.

V případě strategie, kdy provádíme preferenci útoku na silnějšího hráče je klasicky prováděna pomocí vzorce, který byl definován v části návrhu strategií a maximální koeficient preference je nastaven také na hodnotu 0.2, což také vychází z podmínky. Jak přesně nastavit hranici není zcela intuitivní, ale všeobecně by mělo platit, že by tato přídavná hodnota měla být výrazně nižší, než hodnota herního kamene. Důvodem je, že nechceme obětovat vlastní herní kameny jen za cenu toho, že zaútočíme na soupeře, který je aktuálně silnější.

Strategie preference korunovace a přepočtu hodnoty herního kamene jsou také implementovány podle vzorců, které jsou v sekci návrhu strategií a nejsou zde žádné omezující podmínky.

Částečná kooperace také nemá žádné přídavné hodnoty, se kterými by počítala, ale jen využívá ty stávající k tomu, aby přerozdělovala správným způsobem výplaty mezi hráče, kdy pokud soupeř přeskočí herní kámen druhého soupeře, tak bychom stále měli získat stejnou hodnotu výplaty, jako soupeř, který přeskok sám provedl. Tím docílíme schopnosti předpovídat možnou kooperaci soupeřů, jelikož si počítáme situace, kdy by kooperace byla výhodná pro oba dva, aby na ni byli ochotni přistoupit, jelikož se zde nejedná o typickou formu spojenectví.

6.1 Experimenty

Experimentální srovnání bude nejprve zaměřeno na jednotlivé strategie a následně proběhne srovnání klasického algoritmu s genetickým algoritmem.

Strategie

Veškeré strategie byly testovány za pomocí standardního algoritmu, který prohledává všechny možnosti do hloubky dvou tahů, takže celková hloubka prohledávaného stromu je šest. V tomto případě dochází také k dočasnému nastavení predikce tahů na tři tahy předem, což nastává pouze pokud je jeden z hráčů vyřazen a současně nejsou na šachovnici více, než dvě královny. Pokud by tedy během duelu dvou zbylých hráčů došlo ke korunovaci přídavných královen, tak se hloubka prohledávání opět sníží na dva tahy předem. Důvodem je, že v této variantě dámy má královna mnoho možností, jak provést tah, takže dochází k velkému větvení, což vede k velkému množství možností, které je potřeba projít.

Celkově bylo implementováno 5 strategií, kdy vždy probíhaly partie stylem, že první dva hráči měli společnou strategii, zatímco třetí hráč využíval strategii jinou. Tímto způsobem jsme odehráli partie, kdy hrála každá strategie proti každé. Samozřejmě, že jelikož byly některé ze strategií v partii dvakrát, tak měly ve výsledku větší šanci na výhru. Celkem bylo tedy odehráno 1000 partií, kdy každá dvojice strategií odehrála 100 partií. Z těchto partií byly poté zpracovány tak, že vítěz získal bod a při remíze hráčů se tento bod rozdělil rovnoměrně mezi všechny hráče, kteří remizovali. Za každou sérii partií bylo tedy možné získat 100 bodů.

Data z těchto simulací jsou zpracována v tabulce 9.

Tabulka 9: Přehledová tabulka dat

Strategie	Počet hráčů	Výhry	Remízy	Prohry	Body
Kooperace	8	250	118	432	308,7
Preference korunovace	7	153	42	505	173,9
Preference tahu pěšákem	6	159	85	356	211,5
Preference útoku	5	115	52	333	141
Přepočet hodnoty her. kamene	4	149	26	225	162

Problémem je, že nelze tyto data přímo srovnávat a získat tak žebříček strategií, které by byly seřazeny dle počtu bodů, jelikož tyto umělé inteligence zastávaly svou přítomnost v různých počtech. Lze ale přesně říci některé souvislosti, kdy je na první pohled jasné, že si například celkově Přepočet hodnoty herního kamene vedl lépe, než preference korunovace a preference útoku. V případě preference útoku byla strategie schopna získat při zastání u čtyř hráčů větší bodové hodnocení než tato preference útoku s pomocí aplikace na pět hráčů. Současně se strategie také blížila počtu bodů preference korunovace, která byla aplikována až na sedm hráčů. Stejný rozdíl existuje také v počtu bodů preference tahů pěšákem a preference korunovace. Celkově se tedy přibližně nejlépe umístila strategie přepočtu hodnoty herního kamene spolu s kooperací. Třetí místo poté zabere preference tahu pěšákem a na posledních dvou místech zůstane preference útoku s preferencí korunovace.

Jednotlivé počty bodů v sériích partií jsou následně zobrazeny v tabulce 10.

Tabulka 10: Přehledová tabulka bodů v partiích

	PK	PTP	PU	PHHK
K/K	(40,9/41,3)/17,9	(39,5/37)/23,5	(45,5/37)/17,5	(29/38,5)/32,5
PK/PK	-	(31/27)/42	(29,5/25,5)/45	(23/20)/57
PTP/PTP	-	-	(36,5/39)/24,5	(27,5/33)/39,5
PU/PU	-	-	-	(23,5/30,5)/33

Jak bylo již výše zmíněno, nejlépe si vedly strategie kooperace a přepočet hodnoty herního kamene, jelikož ve hrách s ostatními strategiemi vždy zvítězili a v případě, kdy umělé inteligence s těmito strategiemi hrály proti sobě, tak byl souboj velmi vyrovnaný. Nejhůře dopadla preference korunovace, kdy při aplikaci této strategie nebyla umělá inteligence schopna efektivně porážet žádnou z ostatních strategií. Hlavním důvodem tohoto neúspěchu strategie je způsob, jakým strategie funguje. Původní myšlenka byla taková, že by se umělá inteligence snažila získat královnu i za cenu ztráty některých herních kamenů. Bohužel jsou tyto ztráty většinou významné

a výsledná korunovace není až takovou výhodou. Druhým důvodem je, že umělá inteligence, která využívá určitou strategii nezná množinu strategií ostatních hráčů, ale předpokládá, že ostatní implementují stejnou množinu strategií. V takovém případě se stává, že umělá inteligence předpokládá, že soupeř se bude snažit provést korunovaci i za cenu ztráty herního kamene, což ve výsledku neplatí, pokud tuto strategii skutečně neimplementuje. Tento fakt způsobuje situace, kdy umělá inteligence úmyslně předsune vlastní herní kámen před soupeřův a věří, že se soupeř pokusí o korunovaci i za cenu ztráty tohoto herního kamene, což vede k ještě větším ztrátám ze strany tohoto hráče.

V případě ostatních strategií byly výsledky částečně závislé na povaze strategie. Některé strategie jsou aplikovány po celou dobu hry (PK, PTP, PHHK), zatímco jiné fungují jen v případě, že jsou ve hře tři hráči (K, PU). Jiné dělení je možno provést pomocí fáze hry, kdy se strategie jako kooperace, či preference útoku snaží vytvořit v partii výhodu hned ze začátku, zatímco strategie preference korunovace a preference tahu pěšákem jsou soustředěny na koncovky. Například preference tahu pěšákem začne být určitým způsobem účinná až ve chvíli, kdy daný hráč vlastní alespoň jednu královnu. Přepočet hodnoty herního kamene by se dal označit jako speciální případ, jelikož strategie kombinuje defenzivní a ofenzivní způsob hry v závislosti na tom, jak se partie aktuálně vyvíjí. Dokáže tedy na začátku hry hrát defenzivně v případě, že ztratila při výměnách se soupeři herní kameny a současně dokáže obětovat herní kameny, pokud to povede k rychlejší porážce soupeře ke konci hry, pokud se hráč nachází ve značné početní převaze.

Při posledním experimentu byly jisté kombinace strategií postaveny je proti umělé inteligenci, která nevyužívala strategie žádné. Výsledek je zobrazen v tabulce 11.

Tabulka 11: Experiment - Bílý - {K,PTP,PU}, Černý - {K, PTP, PHHK}, Červený - ∅

Hráč	Strategie	Výhry	Remízy	Prohry	Body
Bílý	{K, PTP, PU}	59	26	115	72
Černý	{K, PTP, PHHK}	65	19	116	74,5
Červený	∅	42	23	135	53,5

V tomto případě bylo při experimentu odehráno 200 partií a byly vybrány dvě trojice strategií, kdy v ani jedné trojici nebyla zahrnuta preference korunovace z důvodu zjištění problému s účinností strategie. Z výsledků simulací je zjevné, že aplikace strategií měla významný dopad na schopnost těchto hráčů reagovat na určité situace a získat tak výhodu nad soupeři.

6.2 Srovnání StdAI a GeneticAlgorithm

Než dojde na úvodní srovnání těchto algoritmů, byl připraven test, který simuluje využití parametrů pro úmyslné získání tahu s přeskokem u genetického algoritmu, namísto náhodného

vybrání tahu. Nastavení parametru počtu předpovídaných tahu byl nastaven na 3, velikost populace na 5000 a počet generací na 60. Výsledek je vidět v tabulce 12.

Tabulka 12: Experiment - parametry povinného přeskoku genetického algoritmu

Hráč	Hodnoty parametrů	Výhry	Remízy	Prohry	Body
Bílý	{0, 0}	5	19	76	14
Černý	{0,5, 0,5}	16	36	48	33,5
Červený	{1, 1}	34	38	28	52,5

Je tedy zřejmé, že využitím parametrů povinného přeskoku významně zlepšíme schopnost genetického algoritmu počítat tahy ve srovnatelném čase.

Nyní přejdeme k samotnému srovnání algoritmů. Pro nastavení parametrů genetického algoritmu budeme používat zápis pomocí množiny parametrů $G_a = \{t, p, g, f_1, f_2\}$, kdy první parametr znamená počet předpovídaných tahů, druhý parametr velikost populace, třetí parametr počet generací a parametry f_1, f_2 jsou parametry pro nastavení vyhledávání povinných přeskoků.

Jako první byla vybrána simulace, kdy dva standardní algoritmy, které prohledávají stavový prostor dva tahy napřed, nasadíme proti genetickému algoritmu s nastavením $G_a = \{3, 5000, 70, 1, 1\}$. Výsledek je zobrazen v tabulce 13.

Tabulka 13: Experiment - srovnání algoritmů umělé inteligence

Hráč	Typ umělé inteligence	Výhry	Remízy	Prohry	Body
Bílý	StdAI	31	31	40	46,5
Černý	StdAI	30	20	52	40
Červený	$G_a = \{3, 5000, 70, 1, 1\}$	6	15	79	13,5

Ačkoliv bylo trvání výpočtu tahů srovnatelné, tak nebyl genetický algoritmus schopen dostatečně správně reagovat na tahy svých soupeřů, i když počítají možnosti jen dva tahy předem.

Byl proto vytvořen další experiment, kdy byly parametry genetického algoritmu nastaveny jako $G_a = \{3, 10000, 100, 1, 1\}$. V tomto případě již genetický algoritmus na výpočet jednoho tahu využívá řádově minuty. Výsledek experimentu je v tabulce 14.

Z důvodu trvání těchto simulací bylo provedeno pouze 70 partií. Oproti minulému experimentu je vidět mírné zlepšení, kdy ačkoliv nedosáhl výrazně více bodů, tak vzhledem k sníženému počtu simulovaných partií si vedl o něco lépe v poměru získaných bodů oproti soupeřům.

Ve výsledku ovšem navýšení nebylo žádným způsobem dramatické. Hlavním důvodem v tuto chvíli je, že klasický algoritmus by pro hloubku, ve které pracuje genetický algoritmus počítal tah

Tabulka 14: Experiment - srovnání algoritmů umělé inteligence

Hráč	Typ umělé inteligence	Výhry	Remízy	Prohry	Body
Bílý	StdAI	15	28	27	29
Černý	StdAI	18	21	31	28,5
Červený	$G_a = \{3, 10000, 100, 1, 1\}$	6	13	51	12,5

řádově v dnech. Z tohoto důvodu je zřejmé, že výpočet tahu, který trvá pouhé minuty, nebude zcela přesný a neposkytne nám dostatečně vhodný výsledek. Pro srovnání proto byl vytvořen experiment, kdy proti sobě hrály tři genetické algoritmy, které měly stejnou populaci i počet generací, jen se lišily v počtu tahů, které byly schopny předpovědět, čímž dojde pouze k rozšíření velikosti prohledávaného prostoru. Výsledek experimentu je v tabulce 15.

Tabulka 15: Experiment - srovnání algoritmů umělé inteligence

Hráč	Nastavení genetického algoritmu	Výhry	Remízy	Prohry	Body
Bílý	{3, 5000, 60, 1, 1}	37	32	21	53
Černý	{4, 5000, 60, 1, 1}	11	24	55	23
Červený	{5, 5000, 60, 1, 1}	4	20	66	14

Jak bylo předpokládáno, pokud zvyšujeme velikost prohledávaného prostoru, přičemž zanecháváme stejnou velikost populace a počet generací, tak dochází pouze k větším nepřesnostem, kdy nedochází k nalezení podstatných reakcí při provedení tahu, které vedou k postupné porážce. Aby tedy algoritmus fungoval odpovídajícím způsobem, je nutné přizpůsobovat nastavení parametrů vzhledem k hloubce prohledávaného prostoru.

Abychom tedy docílili správného prohledávání, musíme algoritmu dát na výpočty mnohem více času. Takovéto simulace by byly ovšem časově velmi náročné a daly by se realizovat pouze vytvořením dlouhodobého experimentu. Proto byla ještě otestována schopnost genetického algoritmu reagovat při nastavení prvního parametru na hodnotu 2. Podstatně se tím sníží velikost prohledávaného prostoru a měl by tedy algoritmus reagovat mnohem lépe. Opět byla tedy provedena simulace dvou standardních algoritmů proti genetickému algoritmu. Tentokrát ovšem genetický algoritmus předpovídal maximálně dva tahy předem, kdy konkréně byly parametry nastaveny jako $G_a = \{2, 2500, 30, 1, 1\}$. Výsledky jsou zpracovány viz. tabulka 16.

I při takto nízké populaci a počtu generací byl algoritmus schopen reagovat na tahy standardních algoritmů lépe, než předcházející varianty, kterým výpočty trvaly mnohem déle. Řádově při tomto nastavení počítal genetický algoritmus tahy přibližně 2 sekundy při prvních tazích, což je poloviční až třetinový čas, oproti standardním algoritmům.

Tabulka 16: Experiment - srovnání algoritmů umělé inteligence

Hráč	Typ umělé inteligence	Výhry	Remízy	Prohry	Body
Bílý	StdAI	25	31	44	40,5
Černý	StdAI	27	29	44	41,5
Červený	$G_a = \{2, 2500, 30, 1, 1\}$	11	14	75	18

Je ovšem zřejmé, že genetický algoritmus nikdy nedokáže klasický algoritmus porážet, pokud bude předpovídat stejné množství tahů. Důvodem je, že standardní algoritmus má vždy garantováno, že nalezne nejlepší tah, zatímco u genetického algoritmu tato garance není zajištěna, přičemž kdykoliv genetický algoritmus nenajde ideální tah, tak se může postupně ocitat v nevýhodě z které vždy vytěží algoritmus standardní.

Další faktory, které mohou tento výsledek ovlivnit, je volba jiného druhu implementace tohoto genetického algoritmu, kdy s využitím jiného přístupu bychom mohli získat lepší výsledky, co se týče rychlosti a tím provádět delší šlechtění populace. Případně by mohla být zdokonalena technika seřazení populace, či jiné faktory, které mohou algoritmus negativně ovlivnit. Jisté je ovšem jen to, že aby byl genetický algoritmus schopen porážet své soupeře, tak musí operovat ve větší hloubce prohledávání při srovnatelné rychlosti výpočtu.

6.3 Rychlostní srovnání

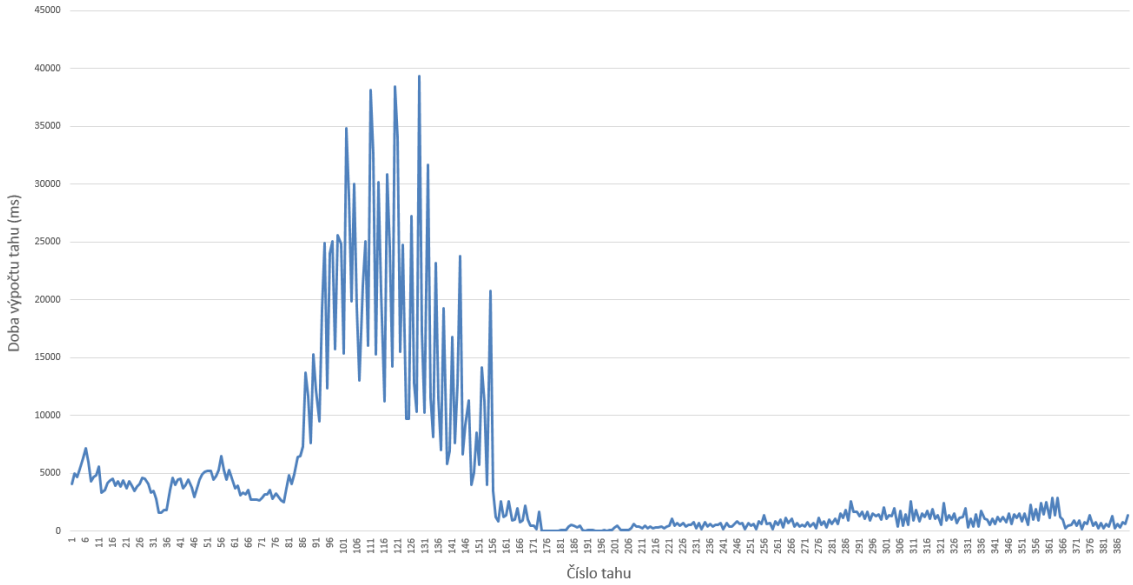
Veškerá rychlostní srovnání byla vytvářena bez přítomnosti strategií, i když by jejich přítomnost měla na rychlost výpočtu pouze minimální význam. Veškeré rychlostní testy byly prováděny na procesoru Intel Pentium G3258 Dual-Core, 4.20GHz.

StdAI

Je jasné, že podobně jako u minimaxu musíme projít všechny uzly stromu, abychom získali nejlepší možný tah. Počet procházených uzlů je možné snížit alfa-beta ořezáváním, ale bohužel v této variantě dámy to není možné, z důvodu existence strategie částečné kooperace [8].

Alfa-beta ořezávání využívá možnosti, kdy nemusíme procházet určité uzly stromu, pokud víme, že náš výsledek již nemohou ovlivnit. Jelikož při kooperaci nám může soupeř navýšit skóre při přeskoku soupeřova herního kamene, tak předem nevíme, zda není možné, abychom v určité větvi nezískali lepší skóre.

Musíme tedy počítat s tím, že je složitost problému exponenciální, kdy nám exponent narůstá v závislosti na prohledávané hloubce stavového prostoru. Druhým faktorem je také aktuální větvení, kdy nám může významně narůst čas při výpočtu, pokud jsou na šachovnici královny, které mají mnoho možností, jak provést tah.



Obrázek 11: Doba trvání výpočtu tahů

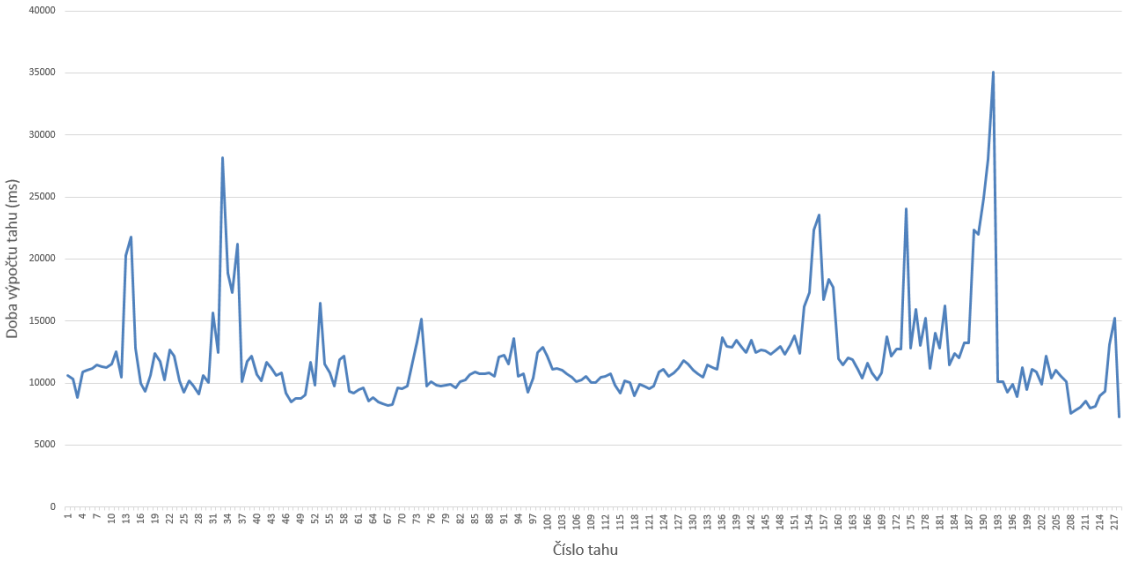
Byla tedy provedena simulace viz. 11, kdy jak již bylo výše zmíněno, nebyly aplikovány strategie pro žádného hráče.

Všichni hráči následně uvažovali situace, které by mohly nastat dva tahy předem, kdy vždy také předpovídali reakce svých soupeřů, což znamená, že hloubka vyhledávání byla šest. Při simulacích strategií byly použity metody, kdy se hloubka za určitých podmínek mohla navýšit na tři tahy předem. To nastávalo, když byli proti sobě už jen dva hráči a také za předpokladu, že na herním plánu nejsou více, než dvě královny. V tomto případě se ovšem staticky zanechalo předvídání dvou tahů, takže jakmile byl jeden z hráčů poražen, tak se hloubka vyhledávání snížila na čtyři.

Graf tedy zobrazuje délku trvání výpočtu každého tahu během partie, kdy je na ose x zobrazován aktuální tah, zatímco na osu y se zobrazuje doba trvání v milisekundách. Na začátku se tedy doba výpočtu pohybovala kolem 4 - 5 sekund, při průměrném faktoru větvení 10. V 87. tahu, kdy došlo k první korunovaci se výpočet prodloužil na 13 sekund, kdy následně výpočet tahů fluktuoval kolem 10 - 40 sekund, což následně ustalo během 155. tahu, kdy došlo k eliminaci prvního ze soupeřů. Tímto se snížila hloubka prohledávání, takže i přes existenci královen trval výpočet tahu v intervalu 50 - 2000 milisekund. Odehrání celé partie takto trvalo 1768061 milisekund, což je v přepočtu 29 minut a 28 sekund.

GeneticAlgorithm

Oproti klasickému algoritmu genetický algoritmus nemusí procházet celý stavový prostor, ale délka jeho výpočtu závisí na nastavení parametrů. Genetický algoritmus také nezajišťuje, že bude vždy nalezen nejlepší možný tah, ale v závislosti na počtu generací a velikosti populace jsme schopni tento výsledek zpřesnit. Byla tedy provedena simulace, kdy měly genetické algoritmy



Obrázek 12: Doba trvání výpočtu tahů

nastaveny hloubku prohledávání nastaveny na 3 tahy předem, velikost populace byla 5000 a počet generací byl 60. Také byly nastaveny parametry nucených přeskoků na 1. Výsledek simulace je vidět na obr. 12.

Výpočet je dle očekávání o něco stabilnější, kdy nám aktuální větvení stabilně nenavýší čas potřebný k výpočtům tahů. Objevují se ovšem jisté výkyvy, které jsou s největší pravděpodobností způsobeny dopředným vyhledáváním herních kamenů, které jsou schopny provést povinný přeskok společně s možností, kdy je mnoho jedinců, kteří byli získáni pomocí křížení, nalezeno poškozených. V takovém případě se musí algoritmus pokoušet o křížení během tvorby nových generací delší dobu, což podstatně ovlivní výsledný čas výpočtu.

7 Závěr

V práci byla okrajově zmíněna problematika teorie her, kterou jsme využili k návrhu strategií, které byly následně implementovány. Implementováno bylo celkem pět strategií, které se nazývají preference korunovace, preference tahu pěšákem, přepočet hodnoty herního kamene, preference útoku a kooperace. Následně byly implementovány dva různé druhy umělé inteligence, kdy se jednalo o různé druhy přístupu k výpočtům tahů, kde první přístup funguje na bázi Minimaxu a využívá kompletního prohledávání stavového prostoru do námi určené hloubky. Druhý způsob byl realizován pomocí genetického algoritmu, u kterého je možné pomocí nastavení volitelných parametrů regulovat dobu výpočtu tahu a tím také ovlivnit kvalitu výsledku.

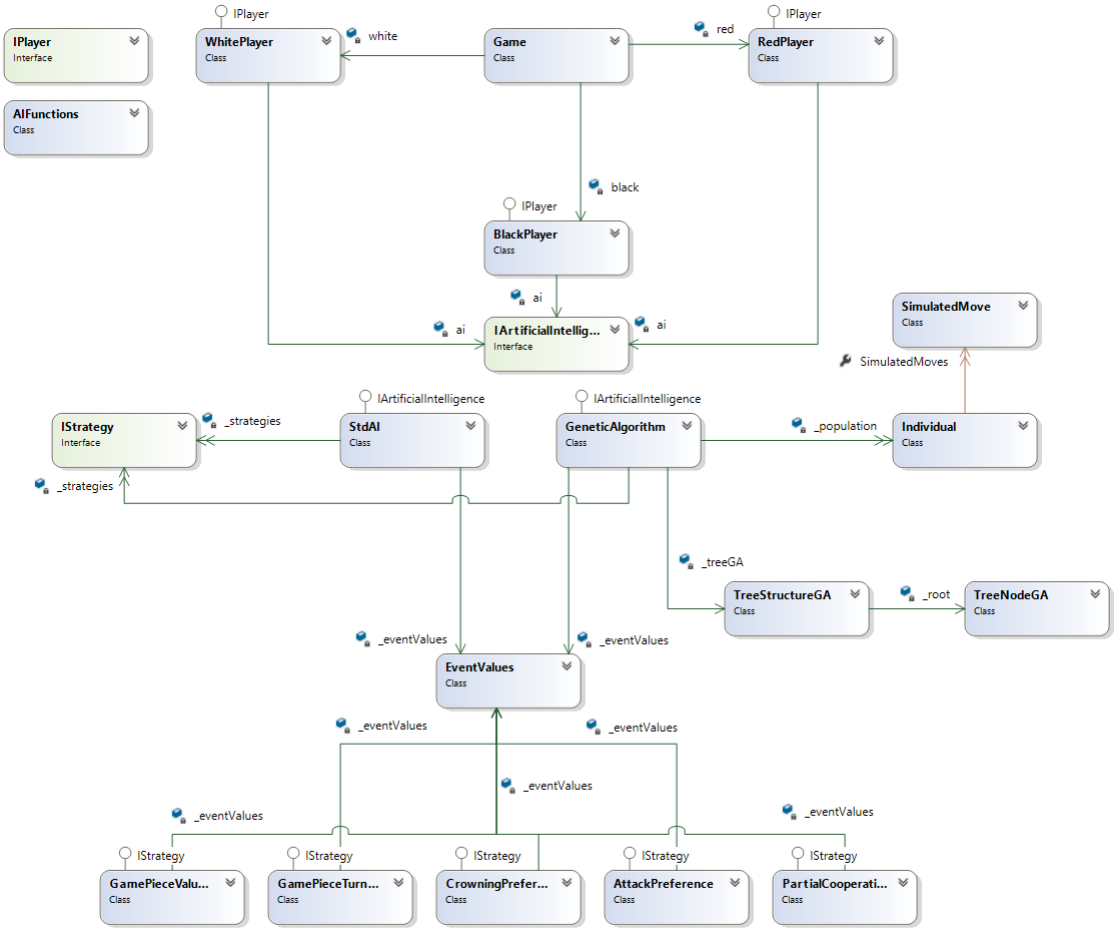
Strategie byly následně aplikovány na standardní algoritmus a byla srovnávána úspěšnost jednotlivých hráčů využívajících tyto strategie. Nejúspěšnější se ukázaly strategie kooperace a přepočtu hodnoty herního kamene, kdy jsme si schopni pomocí kooperace vytvořit úvodní náskok nad svými soupeři. Oproti tomu přepočet hodnoty herního kamene přepíná umělou inteligenci mezi fázemi útoku a obrany, kdy se v případě, že je v početní nevýhodě, snaží minimalizovat své ztráty, zatímco pokud má početní převahu, tak nemá problém obětovat některé ze svých herních kamenů, za účelem rychlejší eliminace svých soupeřů. Přibližně nejhorší se ukázala strategie preference korunovace, kdy docházelo ke zvýšení hodnoty korunovace, takže byla umělá inteligence schopna za cenu korunovace obětovat některé ze svých herních kamenů. Tyto oběti ovšem měly většinou fatální následky v dalším průběhu hry. Dalším problémem zde bylo nastavení této strategie, kdy umělá inteligence předpokládala, že ostatní soupeři tuto strategii také implementují, a tak předpokládala, že se soupeři budou také snažit obětovat herní kámen za korunovaci, k čemuž ve výsledku nedošlo.

Jako poslední proběhlo srovnání standardního algoritmu výpočtu tahů a genetického algoritmu, za pomocí experimentů. V experimentech byly proti sobě nejprve postaveny dva standardní algoritmy proti jednomu genetickému algoritmu, kdy genetický algoritmus operoval ve větší hloubce prohledávání, než standardní algoritmus. Následně také proběhlo srovnání genetického algoritmu, který operoval na stejné hloubce jako algoritmy standardní. Ve všech případech docházelo ovšem k snížené schopnosti genetického algoritmu vítězit, než u algoritmu standardního. Jak bylo již v práci zmíněno, standardní algoritmus má zaručeno, že vždy dojde k nalezení nejlepšího řešení pro danou hloubku prohledávání, což u genetického algoritmu zaručeno není. To může vést k postupným ztrátám a nakonec až k porážce. Dalšími důvody neúspěchu genetického algoritmu může být nevhodný přístup ke křížení jedinců a jejich seřazení v populaci dle významnosti, či případně může docházet ke snížení rozmanitosti jedinců během křížení, což může následně vést k nalezení lokálního maxima.

Je nutné zmínit, že pro genetický algoritmus je celkově problematické vyhledávat řešení v tomto typu stavového prostoru, jelikož byl originálně navržen pro jedince v binárním zápisu. Zde dochází k vysoké chybovosti nových jedinců, získaných křížením, což vede k nutnosti oprav některých jedinců, či případně úplnému vyřazení jedinců z populace. Hlavním problémem zde

beze sporu jsou situace, kdy se genetický algoritmus zastaví na určitém „nejlepším“ tahu, který je vyhodnocen jako nejlepší, jelikož algoritmus nebyl schopen najít určitý tah, který by dokázal snížit jeho významnost.

A Třídní diagram



Obrázek 13: Třídní diagram

B Experimenty

Tabulka 17: Experiment - Kooperace/Kooperace/Preference korunovace

Hráč	Strategie	Výhry	Remízy	Prohry	Body
Bílý	Kooperace	33	16	51	40,9
Černý	Kooperace	35	13	52	41,2
Červený	Preference korunovace	16	4	80	17,9

Tabulka 18: Experiment - Kooperace/Kooperace/Preference útoku

Hráč	Strategie	Výhry	Remízy	Prohry	Body
Bílý	Kooperace	37	17	46	45,5
Černý	Kooperace	29	16	55	37
Červený	Preference útoku	12	11	77	17,5

Tabulka 19: Experiment - Kooperace/Kooperace/Preference tahu pěšákem

Hráč	Strategie	Výhry	Remízy	Prohry	Body
Bílý	Kooperace	28	23	49	39,5
Černý	Kooperace	27	20	53	37
Červený	Preference tahu pěšákem	19	9	72	23,5

Tabulka 20: Experiment - Kooperace/Kooperace/Přepočet hodnoty herního kamene

Hráč	Strategie	Výhry	Remízy	Prohry	Body
Bílý	Kooperace	26	6	68	29
Černý	Kooperace	35	7	58	38,5
Červený	Přepočet hodnoty herního kamene	29	7	64	32,5

Tabulka 21: Experiment - Preference korunovace/Preference korunovace/Preference útoku

Hráč	Strategie	Výhry	Remízy	Prohry	Body
Bílý	Preference korunovace	28	7	65	29,5
Černý	Preference korunovace	22	3	75	25,5
Červený	Preference útoku	41	8	51	45

Tabulka 22: Experiment - Preference korunovace/Preference korunovace/Přepočet hodnoty herního kamene

Hráč	Strategie	Výhry	Remízy	Prohry	Body
Bílý	Preference korunovace	19	8	73	23
Černý	Preference korunovace	17	6	77	20
Červený	Přepočet hodnoty herního kamene	53	8	39	57

Tabulka 23: Experiment - Preference korunovace/Preference korunovace/Preference tahu pěšákem

Hráč	Strategie	Výhry	Remízy	Prohry	Body
Bílý	Preference korunovace	27	8	65	31
Černý	Preference korunovace	24	6	70	27
Červený	Preference tahu pěšákem	37	10	53	42

Tabulka 24: Experiment - Preference tahu pěšákem/Preference tahu pěšákem/Preference útoku

Hráč	Strategie	Výhry	Remízy	Prohry	Body
Bílý	Preference tahu pěšákem	28	17	55	36,5
Černý	Preference tahu pěšákem	30	18	52	39
Červený	Preference útoku	18	13	69	24,5

Tabulka 25: Experiment - Preference tahu pěšákem/Preference tahu pěšákem/Přepočet hodnoty herního kamene

Hráč	Strategie	Výhry	Remízy	Prohry	Body
Bílý	Preference tahu pěšákem	20	15	65	27,5
Černý	Preference tahu pěšákem	25	16	59	33
Červený	Přepočet hodnoty herního kamene	36	7	57	39,5

Tabulka 26: Experiment - Preference útoku/Preference útoku/Přepočet hodnoty herního kamene

Hráč	Strategie	Výhry	Remízy	Prohry	Body
Bílý	Preference útoku	19	9	72	23,5
Černý	Preference útoku	25	11	64	30,5
Červený	Přepočet hodnoty herního kamene	31	4	65	33

Literatura

[1] Chvoj, Martin. *Pokročilá teorie her ve světě kolem nás,* Praha: Grada Publishing, a.s., 2013. ISBN 978-80-247-4620-3.

[2] D. Straffin, Philip. *Game Theory and Strategy,* The Mathematical Association of America, 1993. ISBN-13 098-0883856379.

[3] Zapletal, Miloš. *Velká kniha deskových her,* Brno: Mladá fronta, 1991.

[4] Osborne, Martin J. *An Introduction to Game Theory,* Oxford University Press, 2009.

[5] Kilgour, D. Marc a Steven J. Brams. *Mathematics Magazine Vol. 70, No. 5,* Mathematical Association of America, 1997.

[6] Poundstone, William. *Prisoner's Dilemma,* New York: Doubleday, 1992. ISBN 0-385-41567-2.

[7] S. Ferguson, Thomas. *Game Theory, Second Edition* Mathematics Department, UCLA, 2014. [online]. [cit. 2016-04-02]. Dostupné z: `https://www.math.ucla.edu/~tom/Game_Theory/Contents.html`.

[8] Russell, Stuart J. a Peter Norvig. *Artificial Intelligence. A modern Approach, Third Edition,* Prentice Hall, 1995. ISBN 0-13-103805-2.

[9] Zelinka, Ivan aj. *Evoluční výpočetní techniky - principy a aplikace*, BEN-Technická literatura, 2008. ISBN 80-7300-218-3.

[10] Shvets, Alexander, Gerhard Frey a Marina Pavlova. *SourceMaking.* [online]. [cit. 2016-04-02]. Dostupné z: `http://www.oodesign.com/`