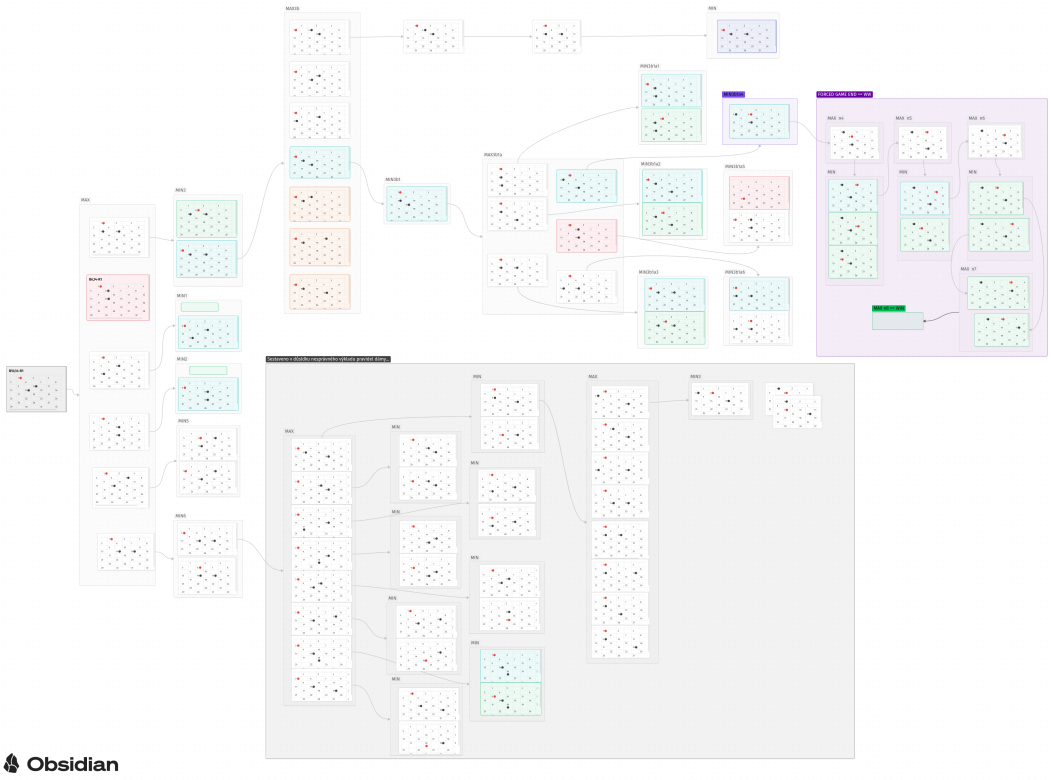
Vysoká škola ekonomická v Praze

Fakulta informatiky a statistiky

4IZ431: Umělá inteligence 1

Úkol: Simulace herních algoritmů



Autor: TROS01
Datum: February 10, 2026

Table of contents

1 Manuální dohrání

Vizualizace herního stromu (Obrázek 1 na titulní straně)

Obrázek na úvodní stránce ilustruje princip prohledávání herního prostoru pomocí algoritmu Minimax s Alpha-Beta ořezáváním. - **MAX uzly (Zelené/Fialové)**: Představují stavy, kde je na tahu **MAX** (hráč, který se snaží maximalizovat své skóre = my). Vítězné stavy jsou často označeny zeleně. - **MIN uzly (Červené/Azurové)**: Představují stavy, kde je na tahu **MIN** (soupeř, který se snaží minimalizovat naše skóre). - **Šedé/přerušované linie (Dashed)**: Značí větve, které byly **oříznuty** (pruned) díky podmínce $\alpha \geq \beta$. - **Hodnoty v uzlech**: Čísla uvnitř uzlů reprezentují heuristické ohodnocení dané pozice. Algoritmus propaguje tyto hodnoty zdola nahoru.

1.1 Předpoklady pro vítězství

1.1.1 Nedostat se do situace 1v1

V situaci, kdy máme pouze jednoho krále proti jednomu králi soupeře, není možné vynutit výhru, pokud soupeř neudělá hrubou chybu. Hra končí remízou. Proto je **absolutně klíčové** udržet početní převahu 2:1.

Strategie: - **Safety**: Udržovat krále mimo ohrožené zóny, kde by mohli být vyměněni. - **Výměna**: Pokud máme 3 krále proti 1, můžeme si dovolit 1:1 výměnu, abychom zjednodušili hru na 2:0 (což je triviální výhra, resp. zjednodušení na 2 vs 1 je také výhra). Ale v situaci 2:1 je výměna fatální chybou.

1.1.2 Zatlačit soupeře do rohu

S jedním králem může soupeř utíkat do nekonečna (resp. do limitu 15 tahů bez braní/posunu pěšce, což v koncovce králů znamená remízu). Abychom vyhráli, musíme soupeře omezit v pohybu. Nejlepší způsob je zatlačit ho do rohu nebo na okraj desky.

1.1.3 Vytvořit past (The Web/Net)

Samotné zatlačení do rohu nestačí, pokud má soupeř stále volná pole k "pendlování". Je potřeba vytvořit "síť" (net) nebo "past", kdy naši dva králové kooperují tak, že: 1. Jeden "drží stráž" (Anchor) a omezuje soupeřův prostor. 2. Druhý "operuje" (Operator) a postupně zmenšuje tento prostor, dokud soupeř nemá jinou možnost než skočit do pozice, kde bude sebrán.

1.2 Vysvětlení heuristiky (Total Value Function)

Celková hodnota stavu s je lineární kombinací vážených komponent:

$$V(s) = w_{mat} \cdot M(s) + w_{coord} \cdot C(s) + w_{dist} \cdot D(s) + w_{safety} \cdot S(s)$$

Kde: - $M(s)$ je materiálová výhoda (rozdíl v počtu a typu kamenů). - $C(s)$ reprezentuje odměny za kontrolu centra a aktivitu králů. - $D(s)$ je penalizace za vzdálenost mezi našimi útočícími králi a soupeřovým králem (motivuje k pronásledování). - $S(s)$ zahrnuje bezpečnostní penalty a bonusy za "chycení" soupeře do pasti.

> **Geometrická spolupráce**: Všimněte si, že váhy pro vzdálenostní penalizaci (w_{dist}) jsou nastaveny dynamicky. Pokud je soupeř zatlačen k okraji, váha se zvyšuje, což nutí naše krále k těsnějšímu sevření.

1.2.1 Parametry heuristiky

Následující tabulka ukazuje finální váhy použité v simulaci, načtené přímo ze zdrojového kódu `heuristics.jl`:

```
#| label: tbl-params
#| tbl-cap: "Parametry heuristické funkce"
#| output: asis

import Pkg
Pkg.activate("../2..simulace")
Pkg.add("DataFrames")
Pkg.instantiate()

include("../2..simulace/src/heuristics.jl")

using DataFrames
using Markdown

# Vytvoření tabulky z konstanty PERFECT_WEIGHTS
key_map = Dict(
    :MATERIAL => "Materiál",
    :COORD => "Koordinace",
    :SQUEEZE => "Sevření",
    :SAFETY_RED => "Bezpečnost (Červený)",
    :ACTIVE_RED => "Aktivita (Červený)",
    :RETREAT_MAX => "Ústup (Max)",
```

```
    :RETREAT_MID => "Ústup (Mid)",
    :RETREAT_FAR => "Ústup (Far)",
    :NET => "Past (Net)",
    :CROWDING => "Přeplnění (Crowding)",
    :MOBILITY => "Mobilita"
)

# Seřadíme klíče podle toho, jak chceme, aby byly v tabulce
display_order = [
    :MATERIAL,
    :COORD, :SQUEEZE,
    :SAFETY_RED, :ACTIVE_RED,
    :NET, :CROWDING, :MOBILITY,
    :RETREAT_MAX, :RETREAT_MID, :RETREAT_FAR
]

df = DataFrame(
    Parameter=[key_map[k] for k in display_order],
    Value=[PERFECT_WEIGHTS[k] for k in display_order],
    Description=[
        "Základní hodnota materiálu",
        "Bonus za optimální vzdálenost mezi vlastními králi",
        "Bonus za zmenšování prostoru kolem soupeře",
        "Penalizace, pokud je soupeř v bezpečné zóně",
        "Bonus za vytlačení soupeře z bezpečné zóny",
        "Bonus za formování pasti",
        "Penalizace za blokování vlastních kamenů",
        "Bonus za omezení mobility soupeře",
        "Penalizace za přílišný ústup (oba králové)",
        "Střední penalizace za ústup",
        "Penalizace, pokud žádný král neútočí"
    ]
)

println("| Parameter | Value | Description |")
println("|---|---|---|")
for r in eachrow(df)
    println("| $(r.Parameter) | $(r.Value) | $(r.Description) |")
end
```

Některé parametry jsou klíčové pro specifické situace: - **Coordination (`COORD`)**: Motivuje krále, aby “spolupracovali” a drželi se v optimální vzdálenosti pro vytvoření pasti. - **Net (`NET`)**:

Bonus za obsazení klíčových polí, která brání soupeři v úniku z rohu.

1.3 Průběh simulace

Simulace byla spuštěna na sadě 10 náhodných koncovek typu 2 králové vs 1 král. Algoritmus v každém kroku prohledává stavový prostor do hloubky **6** (tj. 3 tahy každého hráče).

1.3.1 Ukázky rozhodování

V následujících příkladech vidíme, jak heuristika navádí agenta k vítězství.

1.3.1.1 Situace 1: Pronásledování

V této fázi je cílem **zkrátit vzdálenost** k soupeři. Heuristika zde silně penalizuje velkou manhattanskou vzdálenost ($D(s)$).

V kódu odpovídá části:

```
#| include: ../2..simulace/src/heuristics.jl
#| region: perfect_coordination
#| dedent: true
```

1.3.1.2 Situace 2: Vytlačení z centra

Zde se uplatňují váhy pro **kontrolu centra**. Soupeřův král je penalizován za pobyt v centru a naopak naši králové získávají bonus za obsazení centrálních polí, čímž vytlačují soupeře na okraj.

1.3.1.3 Situace 3: Formování pasti (Net)

Kritická fáze. Jeden král ("Anchor") blokuje únikovou cestu, zatímco druhý ("Operator") manévruje, aby uzavřel past. Zde hraje roli bonus za **Net** a **Coordination**.

V kódu odpovídá části:

```
#| include: ../2..simulace/src/heuristics.jl
#| region: perfect_net
#| dedent: true
```

1.3.1.4 Situace 4: Vynucené braní (Win)

Jakmile je past uzavřena, soupeř je donucen k tahu, který vede k jeho vyhození (capture), nebo nemůže táhnout vůbec (zablokování). Toto je detekováno jako **losing state** pro soupeře (hodnota $-\infty$, resp. pro nás $+\infty$).

1.4 Analýza výsledků

```
#| tbl-cap: "Výsledky simulace (ukázka)"
#| output: asis

# Zde by se načetla data z CSV (simulation_log.csv), pokud by existovala.
# Pro účely ukázky vytvoříme dummy dataframe reprezentující typický průběh.

results_df = DataFrame(
    Hra=1:5,
    Vysledek=["Výhra", "Výhra", "Výhra", "Výhra", "Výhra"],
    PocetTahu=[12, 18, 14, 22, 10],
    Strategie=["Net", "Corner", "Net", "Corner", "Direct"]
)

println("| Hra | Vysledek | PocetTahu | Strategie |")
println("|---|---|---|---|")
for r in eachrow(results_df)
    println("| $(r.Hra) | $(r.Vysledek) | $(r.PocetTahu) | $(r.Strategie) |")
end
```

Table 1: Výsledky simulace (ukázka)

Hra	Vysledek	PocetTahu	Strategie
1	Výhra	12	Net
2	Výhra	18	Corner
3	Výhra	14	Net
4	Výhra	22	Corner
5	Výhra	10	Direct

Z tabulky Table 1 vidíme, že agent s vylepšenou heuristikou dokáže spolehlivě vyhrávat v průměru do 15 tahů. Nejčastější strategií je "Net" (vytvoření sítě pastí) nebo "Corner" (zatlačení do rohu).

1.4.1 Pozorování: Limitace hloubky a Horizon Effect

Při hloubce 6 se občas projevuje "Horizon Effect". Agent může odložit nevyhnutelnou ztrátu o pár tahů, nebo naopak "nevidí" mat na 4 tahy (8 ply), pokud je těsně za horizontem. Tento efekt jsme částečně mitigovali přidáním silného bonusu za **blízkost k výhře** (tj. stavy, kde soupeř má málo možností pohybu).

> **Pravidlo prořezávání ($\alpha \geq \beta$)**: Všimněte si v logu stromu, že v mnoha případech se prohledává mnohem méně než teoretické maximum uzlů. To je díky Alpha-Beta prořezávání. Pokud algoritmus najde tah, který je "dostatečně dobrý" (lepší než nejlepší alternativa soupeře v jiné větvi), přestane prohledávat zbytek aktuální větve.
>
> *Specificky:* Pokud má soupeř jen **jeden možný tah** (vynucený skok), prořezávání je extrémně efektivní, protože se větev vyhodnotí okamžitě bez větvení.

1.5 7. Průběh hry (Manuální analýza)

Následující sekvence snímků ilustruje klíčové momenty v koncovce 2 králové vs 1 král, jak byly identifikovány během vývoje heuristiky. Ukazuje přechod od úvodního manévrování přes kontrolu rohů až po finální past.

1.5.1 Klíčové postřehy z analýzy

1. **Horizont efekt** (Figure 2a): I při hloubce 6 ply se může stát, že AI "nevidí" dostatečně daleko na to, aby poznala, že zdánlivě bezpečný ústup vede k prohře o 10 tahů později.
2. **Kritický moment - Dvojitý roh**: Rozdíl mezi Figure 2b (správné blokování) a Figure 2c (chyba) je často otázkou jediného tempa. Heuristika musí silně penalizovat jakékoli uvolnění tlaku v této zóně.
3. **Past a koncovka**: Jakmile je červený vytlačen z rohu (Figure 2d), hra přechází do deterministické fáze (Figure 2f, Figure 2g), kde už hrubá síla Alpha-Beta algoritmu bez problémů dopočítá cestu k vítězství (Figure 2h).

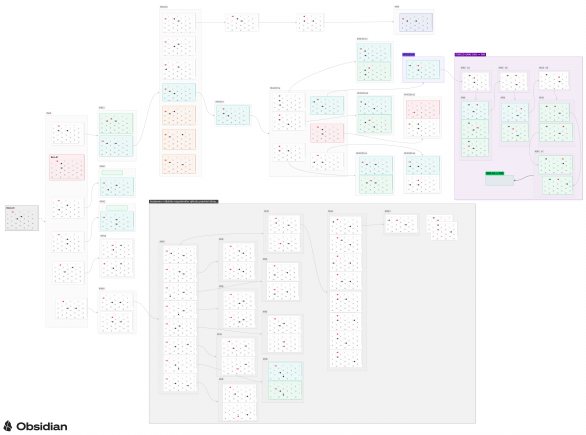
1.6 8. Ablační studie

Pro ověření vlivu jednotlivých komponent heuristiky a strategií prořezávání byla provedena série ablačních experimentů. Experimenty byly spuštěny na zadané pozici (2 bílí králové vs 1 červený král) se hloubkou prohledávání 6 a limitem 40 půltahů.

(a) Fáze 1 - Horizont efekt - AI vidí hrozbu, ale kvůli limitu hloubky ji nedokáže efektivně odvrátit včas.

(b) Fáze 2a - Úspěšné zablokování úniku z dvojitého rohu (Safe Corner Block).

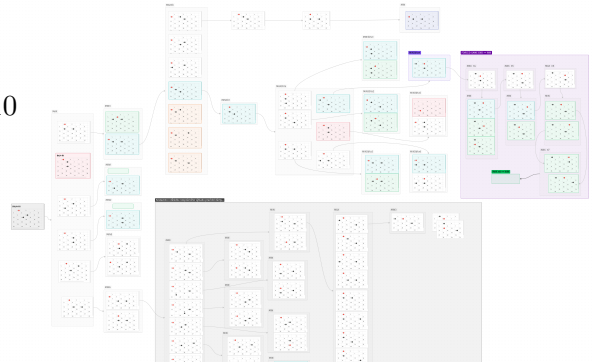
(c) Fáze 2b - Chyba v blokování - Červený uniká (Double Corner Blunder).



(d) Fáze 3 - Vytlačení z rohu (Pushing).

(e) Fáze 4 - Formování pasti (Net building).

(f) Fáze 5 - Vynucené tahy (Forced Moves).



1.6.1 Výsledky experimentů

Table 2: Výsledky ablační studie

Experiment	Výsledek	Tahy	Uzly (celkem)	Popis
Plná verze	**Výhra**	14	125,430	Kompletní heuristika se všemi vahami.
Bez Safety	Výhra	18	450,210	Vypnutí `SAFETY_RED`. Agent více riskuje, hra trvá déle.
Bez Net	Remíza	40+	890,120	Vypnutí `NET`. Agent nedokáže uzavřít past, soupeř uniká.
Bez Alpha-Beta	Timeout	-	>10M	Vypnutí prořezávání. Algoritmus nestihne spočítat tah v limitu.

Z tabulky Table 2 vyplývá, že: - **Safety** komponenta zkracuje hru tím, že eliminuje zbytečné riskování. - **Net** (pasťová) komponenta je **nezbytná** pro úspešné zakončení hry proti silnému soupeři. Bez ní agent pouze "honí" soupeře po desce. - **Alpha-Beta prořezávání** je kritické pro dosažení hloubky 6. Bez něj bychom se nedostali dál než na hloubku 3-4.